Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних наук
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

магістр
(назва освітнього ступеня)

на тему: Розробка архітектури та прототипу системи для автоматизованого підбору комплектуючих ПК на основі технологій ШІ та мікросервісів

Виконав: студент VI курсу, групи СНнм-61
спеціальності 122 Комп'ютерні науки
(шифр і назва спеціальності)

_____ Тизунь А.С.
(підпис) (прізвище та ініціали)

Керівник _____ Боднарчук І.О.
(підпис) (прізвище та ініціали)

Нормоконтроль _____ _____
(підпис) (прізвище та ініціали)

Завідувач кафедри _____ Боднарчук І.О.
(підпис) (прізвище та ініціали)

Рецензент _____ _____
(підпис) (прізвище та ініціали)

Тернопіль
2026

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра комп'ютерних наук
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Боднарчук І.О.
(підпис) (прізвище та ініціали)
«_13_» _квітня_ 2026 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Магістр
(назва освітнього ступеня)

за спеціальністю 122 Комп'ютерні науки
(шифр і назва спеціальності)

Студенту Тизунь Андрій Сергійович
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка архітектури та прототипу системи для автоматизованого підбору комплектуючих ПК на основі технологій ШІ та мікросервісів

Керівник роботи Боднарчук Ігор Орестович, к.т.н., доцент кафедри КН
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «_10_» _березня_ 2026 року № _4/9-150__

2. Термін подання студентом завершеної роботи 25 травня 2026 р.

3. Вихідні дані до роботи Розробка архітектури та прототипу системи для автоматизованого підбору комплектуючих ПК на основі технологій ШІ та мікросервісів. Мова програмування Java, JavaScript, Python. Середовище розробки IntelliJ IDEA. СУБД – PostgreSQL

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ. 1 Аналіз предметної області та методів інтелектуального підбору. 2 Проектування архітектури та теоретичне обґрунтування системи. 3 Програмна реалізація та експериментальне дослідження системи. 4 Охорона праці та безпека в надзвичайних ситуаціях. Висновки. Додатки

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
1 Титульна сторінка. 2 Вступ. 3 Вступ (продовження). 4 Мета, задачі, та практична цінність. 5 Мета, задачі, та практична цінність (продовження). 6 Що таке машинне навчання? 7 Види машинного навчання. 8 Огляд аналогів – PCPartPicker. 9 Огляд аналогів – PCBuilder. 10 Огляд аналогів – Corsair PC Builder. 11 Дерево цілей. 12 Діаграма прецедентів 13 Структурна схема. 14 Схема бази даних. 15 Використані технології – Backend. 16 Використані технології – Frontend. 17 Результати роботи – Авторизація. 18 Результати роботи – Підбір комплектуючих. 19 Результати роботи – Конфігурація. 20 Висновки. 21 Завершальний слайд.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Охорона праці	Гурик О.Я., доцент каф. МТ	27.04.2026	10.05.2026
Безпека в надзвичайних ситуаціях	Теслюк В.М., проректор з адміністративно-господарської роботи та будівництва	27.04.2026	10.05.2026

7. Дата видачі завдання 13 квітня 2026 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	13.04.2026	*Виконано*
2.	Підбір та опрацювання наукових публікацій, збір даних по темі роботи	13.04.2026-20.04.2026	*Виконано*
3.	Виконання дослідження згідно теми кваліфікаційної роботи	21.04.2026-03.05.2026	*Виконано*
4.	Оформлення розділу «Аналіз предметної області та методів інтелектуального підбору»	04.05.2026-10.05.2026	*Виконано*
5.	Оформлення розділу «Проектування архітектури та теоретичне обґрунтування системи»	04.05.2026-10.05.2026	*Виконано*
6.	Оформлення розділу «Програмна реалізація та експериментальне дослідження системи»	04.05.2026-10.05.2026	*Виконано*
7.	Виконання завдання до підрозділу «Охорона праці»	27.04.2026-10.05.2026	*Виконано*
8.	Виконання завдання до підрозділу «Безпека в надзвичайних ситуаціях»	27.04.2026-10.05.2026	*Виконано*
9.	Оформлення кваліфікаційної роботи	11.05.2026-13.05.2026	*Виконано*
10.	Нормоконтроль	14.05.2026	*Виконано*
11.	Перевірка на плагіат	15.05.2026	*Виконано*
12.	Попередній захист кваліфікаційної роботи	18.05.2026	*Виконано*
13.	Захист кваліфікаційної роботи	28.05.2026	

Студент _____ Тизунь А. С.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Боднарчук І.О.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Розробка архітектури та прототипу системи для автоматизованого підбору комплектуючих ПК на основі технологій ШІ та мікросервісів // Кваліфікаційна робота освітнього ступеня «Магістр» // Тизунь Андрій Сергійович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра комп'ютерних наук, група СНнм-61 // Тернопіль, 2026 // С. 68, рис. – 20, табл. – 12, кресл. – 0, додат. – 1, бібліогр. – 50.

Ключові слова: машинне навчання, мівкросервіс, персональний комп'ютер, веб-інтерфейс.

Кваліфікаційна робота присвячена розробці архітектури та прототипу системи для автоматизованого підбору комплектуючих ПК на основі технологій ШІ та мікросервісів. В першому розділі кваліфікаційної роботи описано аналіз предметної області та методів інтелектуального підбору методів інтелектуального підбору. В другому розділі спрпоектовано архітектури та теоретичне обґрунтування системи. В третьому розділі кваліфікаційної роботи описана програмна реалізація та експериментальне дослідження системи.

ANNOTATION

Development of the Architecture and Prototype of a System for Automated PC Component Selection Based on AI Technologies and Microservices// The educational level "Master" qualification work // Tyzun Andrii Serhiyovych // Ternopil Ivan Pulyuy National Technical University, Faculty of Computer Information Systems and Software Engineering, Department of Computer Science, SNnm-61 group // Ternopil, 2026 // P. 68, fig. – 20, tables – 12, posters – 0, annexes – 1, ref. – 50.

Key words: machine learning, microservice, personal computer, web interface.

Thesis is devoted to the development of an architecture and a prototype of a system for automated selection of PC components based on AI and microservices technologies. The first section of the qualification work describes the analysis of the subject area and methods of intelligent selection methods of intelligent selection. The second section describes the architecture and theoretical justification of the system. The third section of the qualification work describes the software implementation and experimental research of the system.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ML – Machine Learning (Машинне навчання)

ПК – Персональний комп’ютер

DL – Deep Learning (Глибоке Навчання)

DB - Database (База даних)

API – Application Programming Interface (прикладний програмний інтерфейс)

UI - User Interface (Інтерфейс користувача)

IDE - Integrated Development Environment (Інтегроване середовище розробки)

UML - Unified Modeling Language (Уніфікована мова моделювання)

OS – Operating System (Операційна система)

JSON - JavaScript Object Notation (текстовий фрагмент обміну даними, заснований на JavaScript)

REST - Representational State Transfer (підхід до архітектури мережевих протоколів, які надають доступ до інформаційних ресурсів)\

HTTPS - Hyper Text Transfer Protocol (протокол передачі текстових даних)

ЗМІСТ

ВСТУП

Актуальність теми. Стрімкий розвиток інформаційних технологій та постійна диференціація ринку комп'ютерного апаратного забезпечення призвели до значного ускладнення процесу підбору оптимальних конфігурацій персональних комп'ютерів (ПК). Сучасний ринок пропонує тисячі одиниць комплектуючих, кожна з яких має специфічні параметри сумісності, енергоспоживання та продуктивності. Для користувачів, які не володіють глибокими технічними знаннями, процес самостійної збірки ЕОМ перетворюється на складну багатофакторну задачу, де висока ймовірність виникнення помилок – від фізичної несумісності деталей до неефективного розподілу бюджету (виникнення «вузьких місць») [1].

Традиційні алгоритмічні конфігуратори, що базуються на жорстких правилах, часто не здатні надати гнучку рекомендацію, орієнтовану на реальні сценарії використання (монтаж відео, 3D-моделювання, специфічні ігрові рушії). Водночас впровадження інтелектуальних методів аналізу даних стримується відсутністю відкритих, структурованих та актуальних наборів даних (датасетів) про збалансовані конфігурації [2].

Отже, розробка інтелектуальної системи, яка використовує сучасні методи машинного навчання для аналізу великих масивів технічних характеристик та базується на гнучкій мікросервісній архітектурі, є актуальним науково-прикладним завданням. Особливої ваги набуває вирішення проблеми «холодного старту» моделі через генерацію напівсинтетичних даних, що дозволяє системі надавати експертні поради навіть за обмеженої кількості реальних прикладів збірок на ринку [3].

Мета дослідження – розробка архітектурних рішень та програмного прототипу системи інтелектуального підбору комплектуючих ПК, яка на основі технологій машинного навчання (Random Forest) та мікросервісного підходу забезпечує формування оптимальних конфігурацій відповідно до неспеціалізованих вимог користувача в умовах дефіциту навчальних даних [4].

Для досягнення поставленої мети визначено такі завдання:

1. Провести системний аналіз предметної області та існуючих методів автоматизації підбору комп’ютерного обладнання.

2. Виконати порівняльний аналіз моделей машинного навчання для задач регресії та класифікації в контексті прогнозування параметрів ЕОМ.

3. Розробити математичну модель та алгоритми генерації напівсинтетичних наборів даних для навчання інтелектуального модуля.

4. Спроектувати мікросервісну архітектуру системи, що забезпечує незалежне масштабування модулів обробки даних та клієнтського інтерфейсу.

5. Реалізувати програмний прототип системи з використанням стека технологій Java Spring Boot, Python та React JS.

6. Провести експериментальне дослідження розробленого прототипу та оцінити точність роботи моделі підбору на різних сценаріях використання.

Об’єкт дослідження – процеси автоматизованого проектування та підбору конфігурацій персональних комп’ютерів.

Предмет дослідження – архітектурні рішення мікросервісних систем, методи машинного навчання та алгоритми підготовки специфічних наборів даних для інтелектуальних систем підтримки прийняття рішень.

Наукова новизна отриманих результатів полягає у вдосконаленні методики інтелектуального підбору комплектуючих шляхом адаптації обмежених технічних даних у напівсинтетичні набори для навчання ансамблевих моделей машинного навчання [5]. Це дозволяє досягти стабільної точності рекомендацій у динамічному ринковому середовищі без необхідності постійного ручного оновлення баз експертних правил [6].

Практичне значення роботи визначається створенням робочого прототипу системи, який може бути використаний як база для комерційних конфігураторів або як допоміжний інструмент для ритейлерів комп'ютерної техніки. Запропонована мікросервісна структура дозволяє легко інтегрувати систему з зовнішніми маркетплейсами та сервісами моніторингу цін.

Апробація результатів. Була проведена в межах двох праць конференцій

Публікації. «Інтелектуальна система автоматизованого підбору комплектуючих пк для непрофесійних користувачів на основі технологій машинного навчання» та «Методика формування синтетичних навчальних вибірок для моделей машинного навчання в задачах конфігурування персонального комп'ютера» в межах всеукраїнської студентської наукової конференції «Експериментальні та теоретичні дослідження в контексті сучасної науки» (20.03.2026, м. Вінниця, Україна) (Див. додаток А).

Структура й обсяг кваліфікаційної роботи. Кваліфікаційна робота складається зі вступу, чотирьох розділів, висновків, списку літератури з 50 найменувань та 1 додатку. Загальний обсяг кваліфікаційної роботи складає 68 сторінки, з них 51 сторінки основного тексту, який містить 20 рисунків та 12 таблиць.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА МЕТОДІВ ІНТЕЛЕКТУАЛЬНОГО ПІДБОРУ

1.1 Аналіз сучасного стану ринку комп'ютерного апаратного забезпечення та тенденцій його розвитку

Сучасний ринок комп'ютерних комплектуючих у період 2024-2026 років перебуває у фазі фундаментальної трансформації, що зумовлена завершенням епохи екстенсивного нарощування тактових частот та переходом до епохи спеціалізованих гетерогенних обчислень. На сьогодні персональний комп'ютер перестав бути статичним набором стандартних вузлів, перетворившись на складну екосистему, де ефективність роботи залежить не від потужності окремого компонента, а від синергії апаратного забезпечення та алгоритмів штучного інтелекту (ШІ).

Ключовим трендом 2025-2026 років стало виокремлення сегменту «AI PC» як того, де очікується значне зростання. Згідно з прогнозами провідних аналітичних агентств, до кінця 2026 року частка комп'ютерів з інтегрованими модулями прискорення нейронних мереж (NPU – Neural Processing Unit) складе понад 55% від загального обсягу поставок (див. рисунок 1.1) [7].

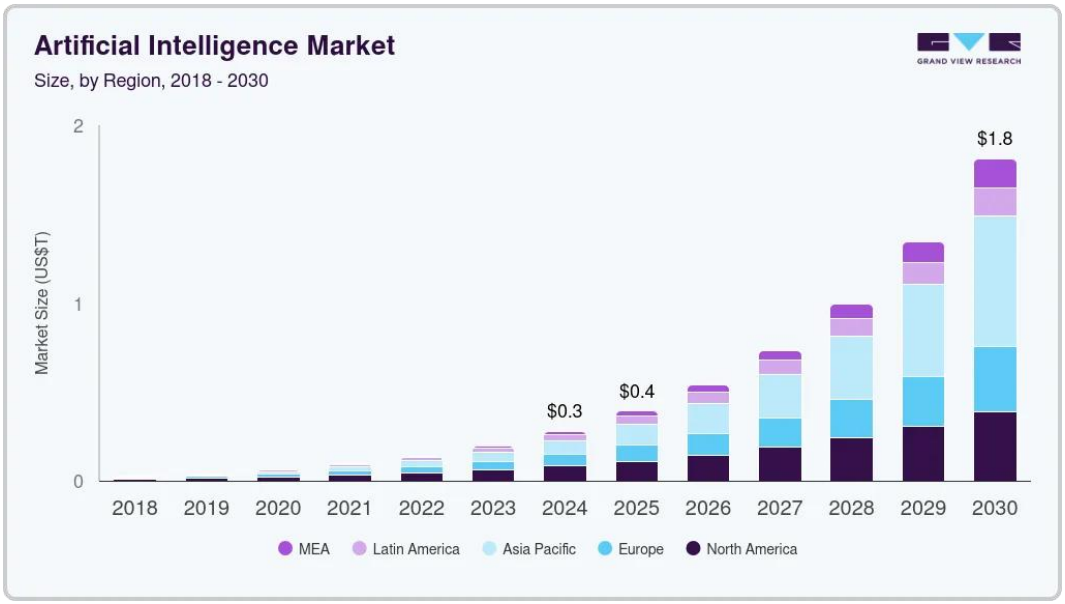

Рисунок 1.1 – Прогноз росту частки AI-орієнтованих систем на світовому ринку

Це суттєво змінює принципи підбору комплектуючих: якщо раніше основними критеріями були частота CPU та обсяг VRAM, то сьогодні критичного значення набуває здатність системи виконувати локальні обчислення малих мовних моделей (SLM) без звернення до хмарних сервісів [8].

Процес підбору ускладнюється стрімкою зміною стандартів передачі даних та живлення, які наведені в таблиці 1.1.

1. Процесорні архітектури: Вихід платформ наступного покоління (наприклад, Intel Nova Lake та AMD Zen 6) супроводжується впровадженням гібридних ядер, де завдання розподіляються між продуктивними (P-cores), енергоефективними (E-cores) та низькоспоживаючими (LP E-cores) ядрами. Це потребує від систем підбору врахування специфіки планувальників задач операційних систем.

2. Пам'ять та збереження даних: У 2026 році стандарт DDR5 остаточно витіснив DDR4, проте ринок стикнувся з явищем значного здорожчання модулів через дефіцит чіпів DRAM, які масово переспрямовуються на потреби дата-центрів та ШІ-прискорювачів (HBM пам'ять). Це створює ситуацію, коли вартість оперативної пам'яті може складати до 25-30% бюджету всієї системи.

3. Інтерфейси: Масове впровадження PCIe 5.0 та анонси PCIe 6.0 створюють надлишкову пропускну здатність для відеокарт, але є критичними для NVMe-накопичувачів нового покоління, які потребують спеціалізованих систем охолодження, що часто конфліктують з габаритами материнських плат або процесорних кулерів.

Таблиця 1.1 – Порівняльна характеристика ключових технологічних стандартів

Технологічний сегмент	**Основні стандарти (2024–2026)**	**Ключові характеристики та інновації**	**Вплив на процес підбору комплектуючих**
Центральні процесори (CPU)	Intel LGA1851 (Arrow Lake, Nova Lake), AMD AM5 (Zen 5, Zen 6)	Інтеграція NPU (Neural Processing Unit) з продуктивністю 40+ TOPS. Гібридні архітектури (P, E та LP-E ядра).	Необхідність врахування наявності ШІ-прискорювача для відповідності вимогам "AI PC".
Оперативна пам'ять (RAM)	DDR5-6400 – DDR5-9000+	Перехід на стандарти CUDIMM та CAMM2. Використання технологій NitroPath для стабілізації сигналу на частотах понад 8000 МТ/с.	Жорстка прив’язка частоти до можливостей контролера пам’яті (IMC) конкретного покоління CPU.
Графічні прискорювачі (GPU)	NVIDIA Blackwell (RTX 50), AMD RDNA 4 (RX 8000)	Підтримка PCIe 5.1. Перехід на пам'ять GDDR7 (швидкість до 32 Гбіт/с). Впровадження тензорних ядер 5-го покоління.	Збільшення вимог до пропускної здатності шини та об'єму відеопам'яті (VRAM) для локальних LLM.

Продовження таблиці 1.1

Технологічний сегмент	**Основні стандарти (2024–2026)**	**Ключові характеристики та інновації**	**Вплив на процес підбору комплектуючих**
Інтерфейси передачі даних	PCIe 5.0 (Mainstream), PCIe 6.0 (Early adoption)	Подвоєння пропускної здатності (до 128 ГБ/с для x16 Gen 6). Стандарт NVMe Gen 5 як базовий для професійних систем.	Критичність систем охолодження для SSD та врахування обмежень ліній PCIe процесора.
Електроживлення (PSU)	ATX 3.1, PCIe 5.1	Впровадження роз'єму 12V-2x6 (безпечніший аналог 12VHPWR). Підтримка пікових навантажень до 200% від номіналу.	Обов'язкова перевірка відповідності роз'ємів БЖ та відеокарти без використання перехідників.
Материнські плати	Chipsets: Intel Z890, AMD X870E / B850	Впровадження AI BIOS Advisors. Стандартна підтримка USB4 (40 Гбіт/с) та Wi-Fi 7 (802.11be).	Автоматизація налаштувань енергоспоживання (Smart TDP) залежно від типу СО.

Окрім технічних складнощів, ринок піддається значним ціновим коливанням. За даними звітів 2026 року, середня вартість ігрового та професійного ПК зросла на 20% порівняно з 2024 роком. Це зумовлено не лише дефіцитом напівпровідників, а й ускладненням ланцюгів постачання рідкоземельних металів. У таких умовах автоматизований підбір стає

інструментом економічної оптимізації, дозволяючи знайти «золоту середину» між продуктивністю та вартістю в умовах обмеженого бюджету [9].

Зі збільшенням асортименту (лише у сегменті відеокарт середнього класу представлено понад 50 варіацій від різних виробників з різними системами живлення та охолодження) звичайний користувач опиняється в ситуації інформаційного перевантаження. Статистика показує, що значна частина користувачів, які збирають ПК самостійно без консультації фахівця, допускають критичні помилки:

- Невідповідність потужності блоку живлення (PSU) піковим навантаженням сучасних GPU;
- Вибір несумісних за висотою модулів RAM та масивних повітряних кулерів;
- Використання застарілих корпусів з недостатньою вентиляцією для компонентів з високим рівнем TDP.

Таким чином, аналіз ринку підтверджує нагальну потребу в інтелектуальному інструменті, який би зміг агрегувати величезні масиви даних про характеристики комплектуючих, враховувати новітні тренди (як-от наявність NPU чи підтримку локальних ШІ-моделей) та пропонувати збалансовані рішення для користувачів, мінімізуючи як технічні, так і фінансові ризики [10].

1.2 Порівняльний аналіз існуючих програмних рішень та онлайн-сервісів конфігурування ПК

На сучасному етапі розвитку цифрових сервісів існує значна кількість інструментів, покликаних полегшити процес вибору комп'ютерного обладнання. Проте аналіз показує, що більшість із них мають суттєві обмеження у методах обробки даних та взаємодії з непідготовленим користувачем. Для проведення системного порівняння доцільно класифікувати існуючі рішення на три ключові групи [11].

Глобальні агрегатори та конфігуратори, такі як PCPartPicker, є світовим стандартом у галузі самостійної збірки ПК.

Приклад інтерфейсу наведено на рисунку 1.2. Система базується на великій базі даних технічних характеристик та автоматизованих алгоритмах перевірки сумісності.

- Переваги: Глибока перевірка фізичних параметрів (наприклад, сумісність довжини відеокарти з габаритами корпусу), моніторинг цін на десятках торгових майданчиків, активна спільнота.
- Недоліки: Система працює за принципом «від компонента до цілого». Користувач повинен заздалегідь знати, який саме процесор і відеокарту він хоче. Вона не пропонує інтелектуальної допомоги у виборі концепції системи для людини, яка не розуміє різниці між техпроцесами чи архітектурами.

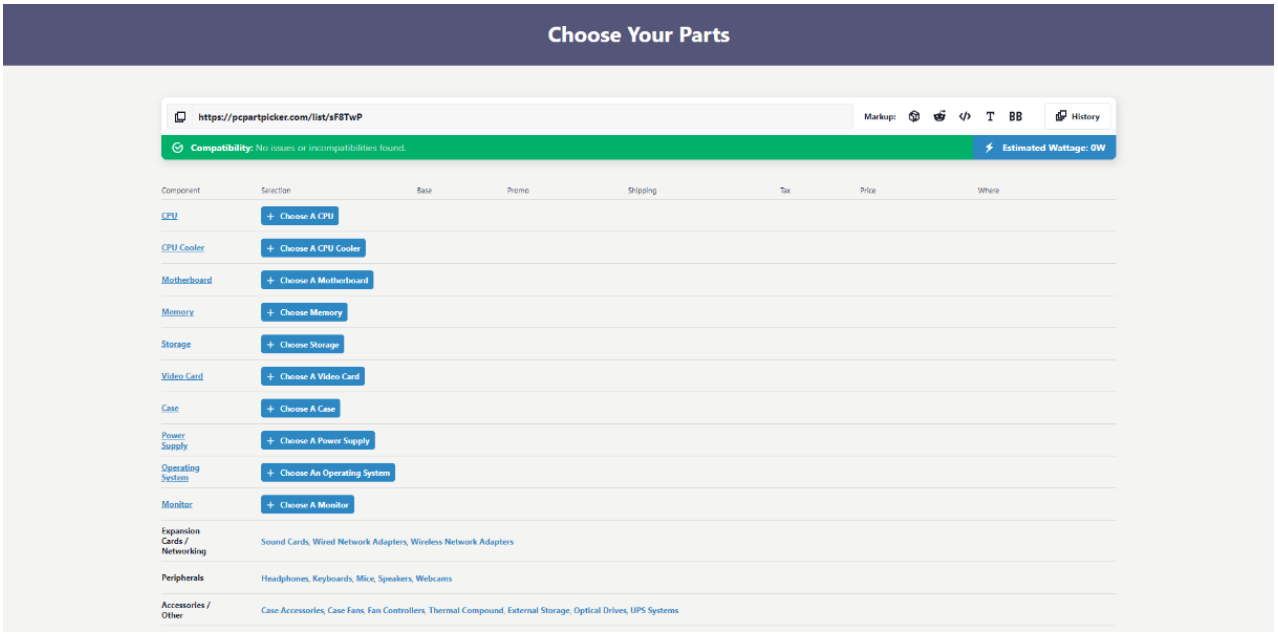

Рисунок 1.2 – Інтерфейс підбору комплектуючих PCPartPicker

Конфігуратори великих ритейлерів та виробників, наприклад Newegg, Rozetka, HP, Dell та Corsair, як наведено на рисунку 1.3. Ці інструменти інтегровані безпосередньо в інтерфейси магазинів.

- Переваги: Прямий зв'язок з актуальними залишками на складі, гарантія сумісності в межах пропонованого асортименту.

- Недоліки: Маркетингова упередженість. Алгоритми часто налаштовані на просування товарів з вищою маржинальністю або тих, які тривалий час зберігаються на складі, а не на реальну оптимізацію за критерієм «ціна/продуктивність». Їхня логіка є закритою (Proprietary logic), що унеможливлює незалежну оцінку.

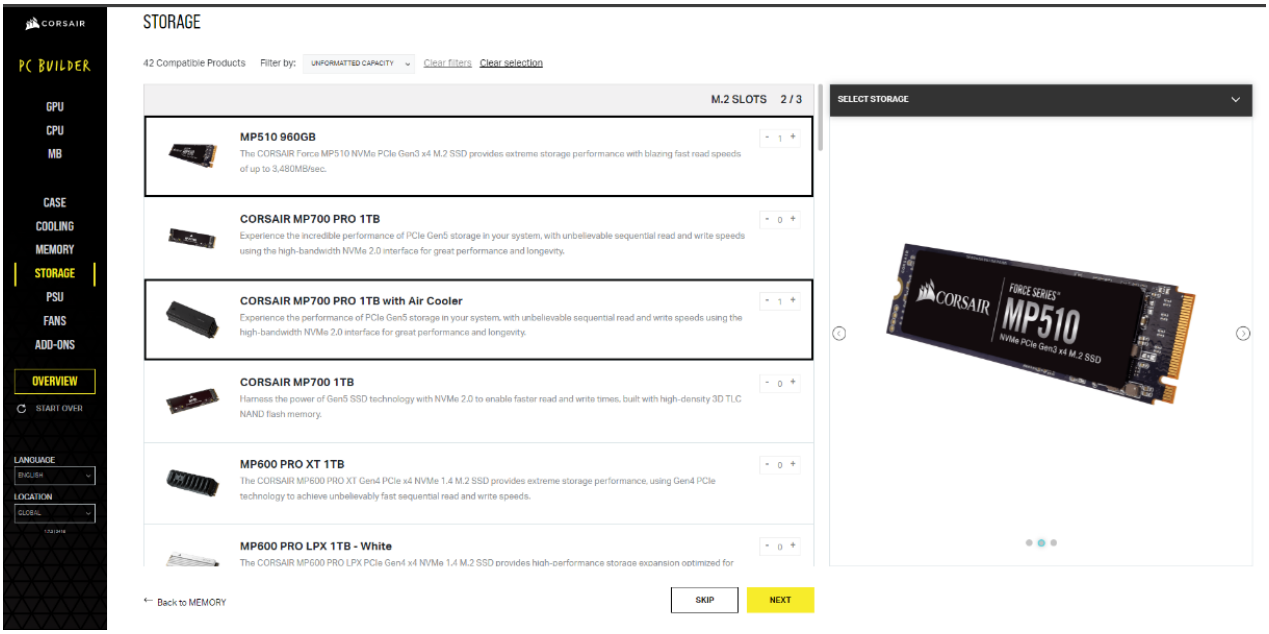

Рисунок 1.3 – Інтерфейс підбору комплектуючих Corsair PC Builder

Спеціалізовані бенчмарк-сервіси, наприклад UserBenchmark, наведений на рисунку 1.4, та Technical.city.

- Переваги: Величезні масиви статистичних даних від реальних користувачів.
- Недоліки: Відсутність комплексного підходу. Вони порівнюють «процесор А проти процесора Б», але не дають відповіді на питання, як ця різниця вплине на продуктивність у конкретній робочій станції з урахуванням інших компонентів.

Ці ресурси зосереджені на порівнянні продуктивності окремих вузлів.

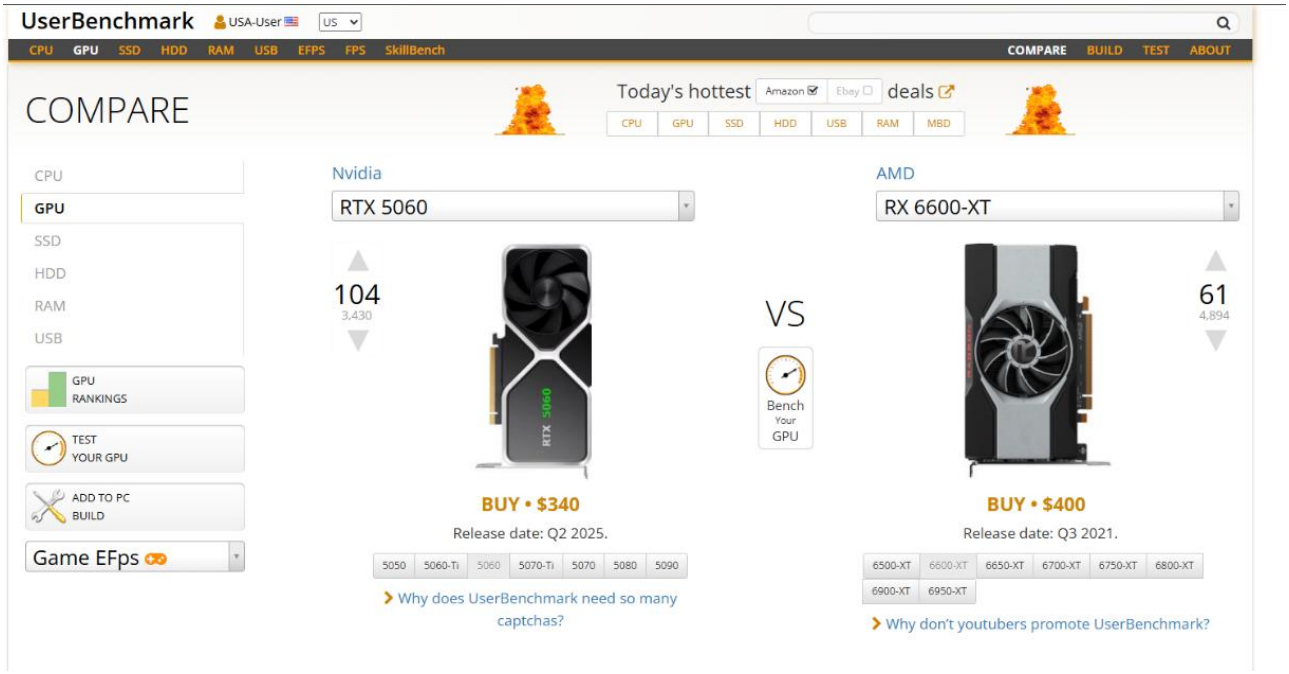

Рисунок 1.4 – Інтерфейс порівняння комплектуючих UserBenchmark

Для детального аналізу функціональної повноти існуючих рішень у таблиці 1.2 наведено порівняльну характеристику за ключовими для магістерського дослідження параметрами.

Таблиця 1.2 – Порівняльна характеристика існуючих систем підбору комплектуючих ПК

Критерії порівняння	**PCPartPicker (Глобальний агрегатор)**	**Newegg / Rozetka (Конфігуратори ритейлерів)**	**UserBenchmark / Technical.city**	**Розроблювана система (МКР)**
Метод перевірки сумісності	Детермінований (жорсткі логічні фільтри)	Базові параметри (сокет, тип пам'яті)	Порівняння окремих вузлів (без збірки)	Інтелектуальний (ML-моделювання + валідація)

Продовження таблиці 1.2

Критерії порівняння	**PCPartPicker (Глобальний агрегатор)**	**Newegg / Rozetka (Конфігуратори ритейлерів)**	**UserBenchmark / Technical.city**	**Розроблювана система (МКР)**
Використання методів ШІ/ML	Відсутнє	Відсутнє (маркетингові алгоритми)	Статистичний аналіз (бенчмарки)	Ансамблеві методи
Рівень входу для користувача	Високий (необхідні технічні знання)	Середній (вибір із готових категорій)	Середній (орієнтація на цифри тестів)	Низький (природно-мовні запити та сфери застосування)
Архітектура системи	Монолітна	Монолітна (частина e-commerce)	Монолітна	Мікросервісна (Spring Boot + Python + React)
Гнучкість підготовки даних	Статичне оновлення баз даних	Пряма залежність від складу магазину	Залежність від звітів користувачів	Адаптивна (генерація напівсинтетичних даних)
Аналіз «вузьких місць» (Bottleneck)	Відсутній (лише фізична сумісність)	Відсутній	Спрощений статистичний	Математично обґрунтований (вагові коефіцієнти)
Доступність інтеграції (API)	Обмежена (закриті протоколи)	Відсутня (Internal only)	Часткова (Web-based)	Висока (RESTful API мікросервіси)

Аналіз таблиці дозволяє виділити наступні критичні «прогалини» в існуючих інструментах:

1. Детермінованість логіки: Майже всі системи використовують жорсткі IF-THEN правила. Якщо в базі немає прямої вказівки на сумісність, система видає помилку або невідомий результат. Це робить їх вразливими до появи нових, нестандартних компонентів.

2. Відсутність семантичного аналізу потреб: Існуючі системи не розуміють запитів на кшталт «мені потрібен ПК для комфортної роботи в 3DS Max у бюджеті 1500$». Вони вимагають від користувача перекладу цієї вимоги на мову технічних характеристик (ядер, потоків, ГБ VRAM).

3. Монолітність: Комерційні рішення є закритими системами. На ринку практично відсутні відкриті інтелектуальні API, які могли б бути інтегровані в сторонні сервіси (наприклад, у чат-боти або корпоративні системи закупівель) без значних витрат.

Проведений аналіз підтверджує, як зображено на рисунку 1.5, що сучасні програмні рішення зосереджені на вирішенні технічної задачі (перевірка сумісності) для підготовленої аудиторії.

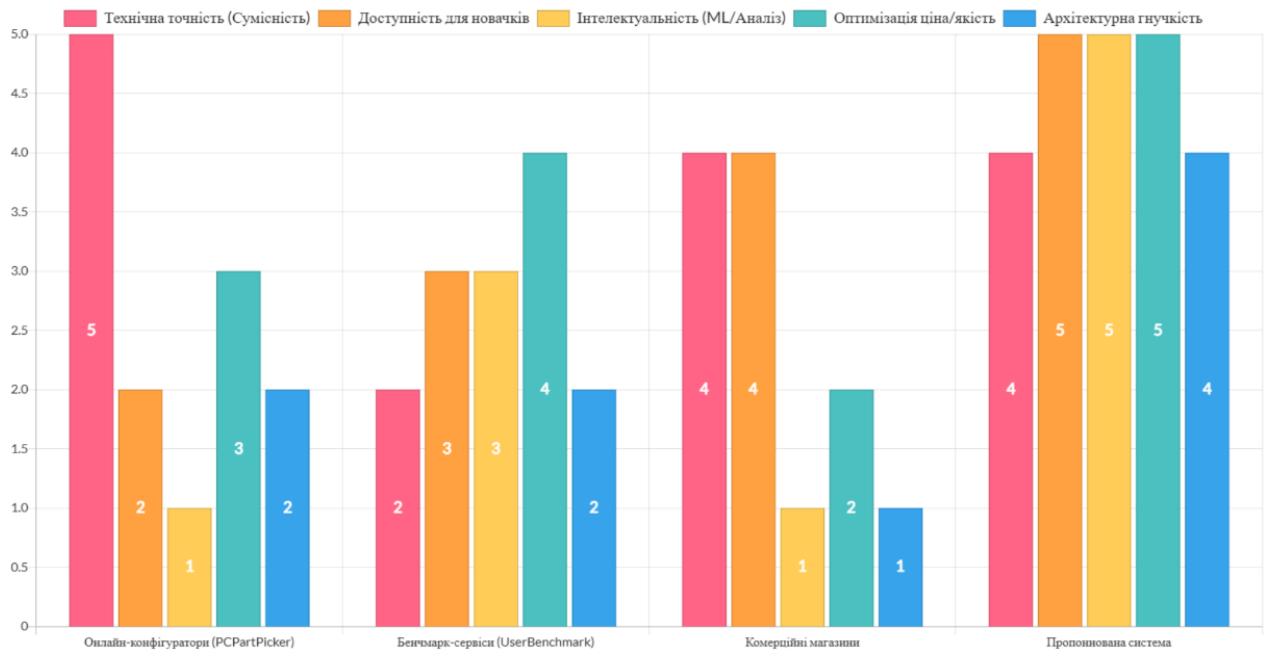

Рисунок 1.5 – Діаграма порівняння охоплення потреб користувача різними типами систем

Сегмент систем, що базуються на машинному навчанні та здатні інтерпретувати високорівневі запити користувачів, залишається практично вільним. Це створює наукове та практичне підґрунтя для розробки системи, що поєднує інтелектуальний аналіз даних та розподілену архітектуру [12].

1.3 Огляд та критичний аналіз методів машинного навчання у задачах класифікації та регресії для систем підтримки прийняття рішень

Розробка інтелектуальних систем підтримки прийняття рішень (СППР) у галузі комп'ютерної інженерії потребує вибору математичного апарату, здатного ефективно оперувати великою кількістю ознак та нелінійними залежностями між ними. Задача підбору комплектуючих може бути сформульована як задача багатофільтрової класифікації (вибір конкретних моделей) або регресії (прогнозування рейтингу продуктивності конфігурації) [13].

Для вирішення подібних задач у сучасній практиці Data Science найчастіше розглядаються такі класи алгоритмів:

Метод k-найближчих сусідів (k-Nearest Neighbors, k-NN). Даний алгоритм базується на припущенні, що схожі об'єкти в просторі ознак належать до одного класу або мають схожі цільові значення.

Переваги: Простота реалізації та відсутність етапу навчання моделі («lazy learning»).

Недоліки: Висока чутливість до масштабу ознак та «прокляття розмірності». У контексті комплектуючих ПК, де ознаки мають різну природу (ціна в одиницях, частота в МГц, TDP у Ватах), k-NN потребує складної попередньої обробки та нормалізації, а обчислювальна складність пошуку сусідів у великих базах даних робить його неефективним для мікросервісів з високим навантаженням.

Штучні нейронні мережі (Artificial Neural Networks, ANN). Глибоке навчання (Deep Learning) демонструє вражаючі результати в задачах обробки зображень та тексту, проте їх застосування для табличних даних (якими є специфікації комплектуючих) має свої особливості.

Переваги: Здатність апроксимувати функції будь-якої складності та виявляти високорівневі абстракції.

Недоліки: Нейронні мережі є «чорними скриньками», що ускладнює інтерпретацію результату (користувач не може зрозуміти, чому система обрала саме цей процесор). Крім того, ANN потребують величезних масивів розмічених даних для запобігання перенавчанню (overfitting), що в умовах дефіциту реальних збірок ПК стає критичним бар'єром.

Градієнтний бустинг над деревами рішень (XGBoost, LightGBM, CatBoost). Бустинг є потужним методом побудови ансамблів, де кожне наступне дерево виправляє помилки попередніх.

Переваги: На сьогодні CatBoost та XGBoost вважаються лідерами у роботі з табличними даними. Вони забезпечують найвищу точність прогнозування.

Недоліки: Бустинг дуже чутливий до шумів у даних та викидів. Оскільки в даній роботі використовуються напівсинтетичні дані (які можуть містити статистичні відхилення), бустинг може занадто «підлаштуватися» під специфіку генератора, втрачаючи здатність до узагальнення на реальних ринкових прикладах.

Випадковий ліс (Random Forest). Алгоритм Random Forest базується на побудові великої кількості незалежних дерев рішень (Decision Trees) з використанням методів беггінгу (Bagging) та випадкових підпросторів ознак.

Переваги: На відміну від бустингу, Random Forest будує дерева паралельно, що робить його стійким до викидів та перенавчання. Алгоритм ефективно обробляє параметри на кшталт «Socket», «Form Factor», «Brand» без необхідності створення сотень додаткових стовпців (One-Hot Encoding). Модель також дозволяє математично обґрунтувати, який параметр (наприклад, обсяг кешу чи частота пам'яті) мав найбільший вплив на фінальну рекомендацію. Завдяки можливості розпаралелювання обчислень, Random Forest оптимально інтегрується в мікросервісну архітектуру на базі Python, забезпечуючи швидкий відгук системи [14].

Для об'єктивного порівняння методів у межах даного дослідження складено таблицю 1.3.

Таблиця 1.3 – Порівняльний аналіз моделей машинного навчання для задач підбору апаратного забезпечення

Критерії порівняння	**Метод k-найближчих сусідів (k-NN)**	**Штучні нейронні мережі (ANN)**	**Градієнтний бустинг (XGBoost/CatBoost)**	**Випадковий ліс (Random Forest)**
Стійкість до перенавчання	Низька (чутливий до шумів)	Середня (потребує регуляризації)	Середня (схильний до перенавчання на малих вибірках)	Висока (завдяки беггінгу та випадковим підпросторам)
Обробка табличних гетерогенних даних	Потребує обов'язкової нормалізації	Складна попередня підготовка	Висока ефективність	Дуже висока (не чутливий до масштабу ознак)
Інтерпретованість результатів	Висока (на основі близькості)	Низька («чорна скринька»)	Середня (складна ієрархія помилок)	Висока (наявність Feature Importance)
Вимоги до об'єму навчальної вибірки	Малі/Середні	Дуже високі	Високі	Середні (ефективний на напівсинтетичних даних)
Швидкість навчання та прогнозу	Миттєве навчання, повільний прогноз	Повільне навчання, швидкий прогноз	Повільне навчання, швидкий прогноз	Висока швидкість (можливість паралелізації)

Продовження таблиці 1.3

Критерії порівняння	**Метод k-найближчих сусідів (k-NN)**	**Штучні нейронні мережі (ANN)**	**Градієнтний бустинг (XGBoost/CatBoost)**	**Випадковий ліс (Random Forest)**
Робота з категоріальними ознаками (Socket, Brand)	Лише через кодування (One-Hot)	Лише через кодування	Вбудована підтримка (у CatBoost)	Вбудована підтримка через розбиття в вузлах

Проведений аналіз підтверджує, що для систем із гетерогенними даними та необхідністю високої стабільності на малих і синтетичних вибірках, ансамблеві методи на основі дерев рішень є найбільш доцільними. Незважаючи на високу точність нейронних мереж, їхня «непрозорість» та вимогливість до даних роблять їх менш придатними для MVP магістерського проекту. Найбільш збалансованим варіантом для реалізації інтелектуального ядра системи визначено алгоритм Random Forest, детальне обґрунтування якого буде наведено в наступному підрозділі [15].

1.4 Обґрунтування вибору алгоритму Random Forest для прогнозування оптимальних конфігурацій

Вибір математичного апарату для інтелектуального ядра системи є критичним етапом, оскільки він визначає здатність прототипу до масштабування та точність його рекомендацій. На основі проведеного в підрозділі 1.3 аналізу, найбільш доцільним методом для вирішення задачі автоматизованого підбору комплектуючих визначено алгоритм Random Forest (Випадковий ліс) [16].

Дане рішення ґрунтується на декількох фундаментальних теоретичних та практичних аспектах, що безпосередньо корелюють зі специфікою предметної області.

Random Forest є ансамблевим методом, що реалізує парадигму беггінгу (Bootstrap aggregating) над деревами рішень. Основна ідея полягає у побудові великої кількості незалежних дерев, кожне з яких навчається на випадковій підвибірці даних з використанням випадкової підмножини ознак.

Математично це дозволяє суттєво знизити дисперсію (variance) моделі без збільшення зміщення (bias). У контексті даної МКР це має вирішальне значення: оскільки для навчання використовуються напівсинтетичні набори даних, модель повинна бути стійкою до можливих «шумів» або статистичних відхилень, які виникають при автоматичній генерації збірок. Схема роботи алгоритму наведена на рисунку 1.6.

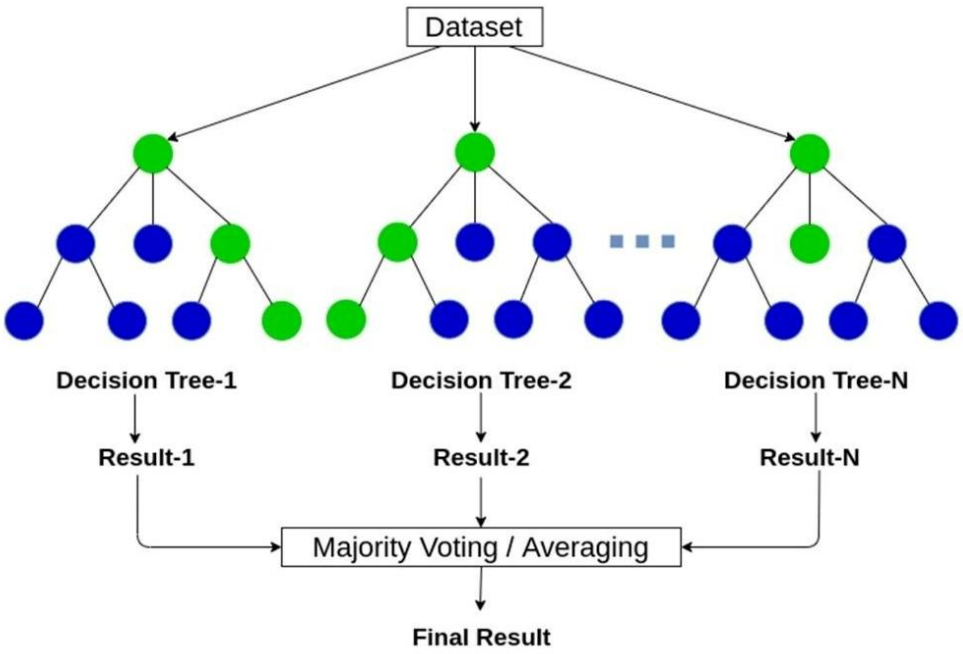

Рисунок 1.6 – Схема роботи алгоритму Random forest

Random Forest у контексті підбору комплектуючих:

Гетерогенність даних. Специфікації ПК включають дані різної природи:

Числові: тактова частота (МГц), обсяг пам'яті (ГБ), тепловиділення (TDP), ціна.

Категоріальні: тип сокета (LGA1700, AM5), форм-фактор (ATX, ITX), бренд виробника. На відміну від градієнтного спуску в нейронних мережах, дерева рішень у складі «лісу» не потребують обов'язкового масштабування або нормалізації ознак, що спрощує архітектуру мікросервісу обробки даних на Python.

Обробка нелінійних залежностей. Взаємодія компонентів ПК часто має нелінійний характер. Наприклад, подвоєння обсягу оперативної пам'яті може дати нульовий приріст продуктивності в офісних задачах, але бути критичним для відеомонтажу. Random Forest природним чином моделює такі порогові ефекти через ієрархічну структуру розбиття у вузлах дерев.

Інтерпретованість та аналіз важливості ознак (Feature Importance). Алгоритм дозволяє обчислити внесок кожної ознаки в процес прийняття рішення за формулою зменшення домішок (Gini impurity) або приросту інформації (Information Gain). Це дозволяє системі в майбутньому пояснювати користувачеві, чому саме цей компонент було визначено як пріоритетний (наприклад, через високий вплив частоти CPU на конкретну прикладну програму) [17].

Таблиця 1.4 – Порівняння теоретичної складності та вимог Random Forest

Параметр оцінки	**Теоретичне значення / Опис**	**Вплив на систему (Spring Boot / Python)**	**Значущість для напівсинтетичних даних**
Алгоритмічна складність навчання	O(n log n d k)	Дозволяє проводити регулярне перенавчання моделі.	Висока: забезпечує швидку ітерацію при налаштуванні генератора даних.

Продовження таблиці 1.4

Параметр оцінки	**Теоретичне значення / Опис**	**Вплив на систему (Spring Boot / Python)**	**Значущість для напівсинтетичних даних**
Алгоритмічна складність прогнозу (Inference)	O(d \ k)	Гарантує відповідь мікросервісу протягом <50 мс, що критично для чуйності React-інтерфейсу.	Прогноз базується на глибоких зв’язках даних.
Стійкість до мультиколінеарності	Висока (автоматичний відбір ознак)	Дозволяє використовувати взаємозалежні ознаки (наприклад, TDP та кількість ядер) без втрати точності.	Критична: напівсинтетичні дані часто мають жорсткі кореляції.
Ризик перенавчання (Overfitting)	Низький (завдяки закону великих чисел для ансамблів)	Не потребує складних методів регуляризації (як-от Dropout у нейромережах) на стороні Python.	Мінімізує «запам’ятовування» специфічних патернів синтетичного генератора.

Продовження таблиці 1.4

Параметр оцінки	**Теоретичне значення / Опис**	**Вплив на систему (Spring Boot / Python)**	**Значущість для напівсинтетичних даних**
Вимоги до пам’яті (RAM)	Пропорційно кількості та глибині дерев	Модель є компактною, що оптимально для Docker-контейнера.	Можливість зберігати версії моделі для різних категорій ПК (Gaming/Office) одночасно.
Інтерпретованість (Feature Importance)	Вбудована (Gini Importance / Mean Decrease Impurity)	Дозволяє передавати на фронт-енд «вагу» факторів, що вплинули на вибір конкретної конфігурації.	Допомагає верифікувати, чи правильно генератор даних розставляє пріоритети.

Використання Random Forest дозволяє реалізувати «легкий» ML-мікросервіс. Оскільки модель після навчання є компактною (сукупність структур дерев), вона не потребує великих обсягів GPU-пам'яті, що дозволяє розгортати систему в хмарних середовищах з обмеженими ресурсами без втрати продуктивності. Це відповідає концепції мікросервісів, описаній у Розділі 2, де кожен сервіс має бути максимально автономним та ефективним.

Таким чином, Random Forest обрано не лише як інструмент з високою прогнозною здатністю, а й як найбільш адаптивний метод для роботи в умовах дефіциту реальних даних. Його здатність до узагальнення на синтетичних вибірках та стійкість до викидів забезпечують надійний фундамент для побудови інтелектуальної складової системи.

1.5 Системний аналіз об’єкта автоматизації: побудова дерева проблем та дерева цілей

Системний розгляд об’єкта автоматизації потребує чіткої структуризації викликів, що виникають у процесі проектування інтелектуальних засобів підбору апаратного забезпечення. Для візуалізації та ієрархічної декомпозиції ключових перешкод у межах дослідження застосовано метод побудови дерева проблем. Цей інструментарій дозволяє трансформувати складну ситуацію у логічну схему, де виділяються корінні причини та їхні наслідки, що є фундаментом для формування ефективної стратегії розробки. Графічне представлення ієрархії виявлених труднощів наведено на рис. 1.7.

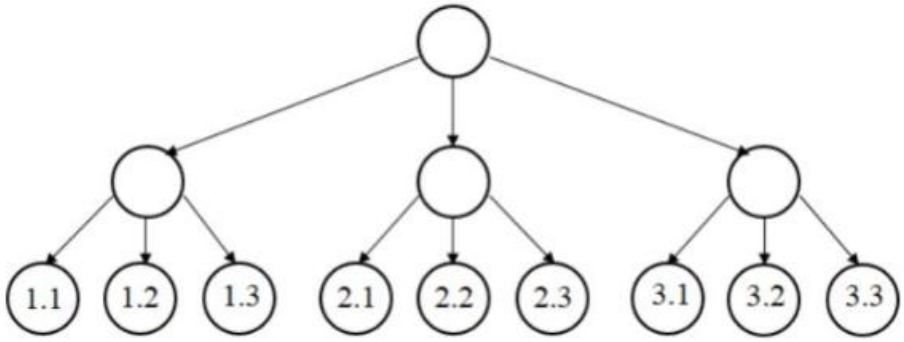

Рисунок 1.7 – Графічне представлення дерева проблем системи

Декомпозиція дерева проблем проводиться шляхом виділення генеральної проблеми, яка формулює основний виклик проекту – необхідність створення ефективного інтелектуального механізму підбору комплектуючих ПК на основі машинного навчання. Дана проблема поділяється на наступні критичні вузли:

- Інформаційний дефіцит: труднощі у виборі репрезентативних та актуальних джерел даних про технічні характеристики та ринкову вартість компонентів.
- Алгоритмічна складність: необхідність розробки компетентної моделі, здатної враховувати нелінійні зв'язки між вузлами системи.
- Комунікаційний бар'єр: відсутність зручного та безпечного інтерфейсу, адаптованого для користувача з будь-яким рівнем технічної підготовки.

- Валідаційний виклик: складність верифікації згенерованих конфігурацій та відсутність механізму зворотного зв'язку для самонавчання моделі.

Аналіз умов функціонування системи. Проектована система розробляється для експлуатації в умовах помірної конкуренції. Ключовою конкурентною перевагою є орієнтація на максимальну доступність для непідготовленого користувача. На відміну від глобальних сервісів на кшталт PCPartPicker, що орієнтовані на експертну аудиторію, дана розробка нівелює поріг входження у процес вибору обладнання. З точки зору технічної доступності, система реалізується як кросплатформний веб-сервіс. Актуальність бази знань та точність рекомендацій забезпечуються шляхом ітеративного донавчання моделі на основі нових ринкових даних та успішних кейсів конфігурування.

Шляхи вирішення виявлених проблем. Для подолання встановлених деструктивних чинників у роботі запропоновано наступні стратегічні підходи:

- Оптимізація джерел даних: використання агрегованої інформації з офіційних специфікацій та незалежних бенчмарк-сервісів. Це гарантує точність вхідних ознак для моделі.
- Інтелектуалізація підбору: впровадження методів машинного навчання, здатних автоматично виявляти патерни сумісності та продуктивності на основі великих масивів даних, що перевершує можливості детермінованих систем.
- Ергономіка інтерфейсу: застосування інтуїтивних дизайн-метафор та розгалуженої системи інтерактивних підказок, що забезпечує комфортну взаємодію навіть при мінімальному досвіді користувача.
- Гібридна верифікація: впровадження окремого модуля перевірки технічної сумісності, який працює паралельно з ML-моделлю, забезпечуючи стовідсоткову фізичну працездатність обраної конфігурації.

Побудова дерева цілей. На основі аналізу проблем розроблено дерево цілей, яке визначає вектор розробки системи від загальної концепції до конкретних функціональних завдань, включаючи вимоги до безпеки та конфіденційності.

Графічне представлення дерева цілей наведно на рисунку 1.8.

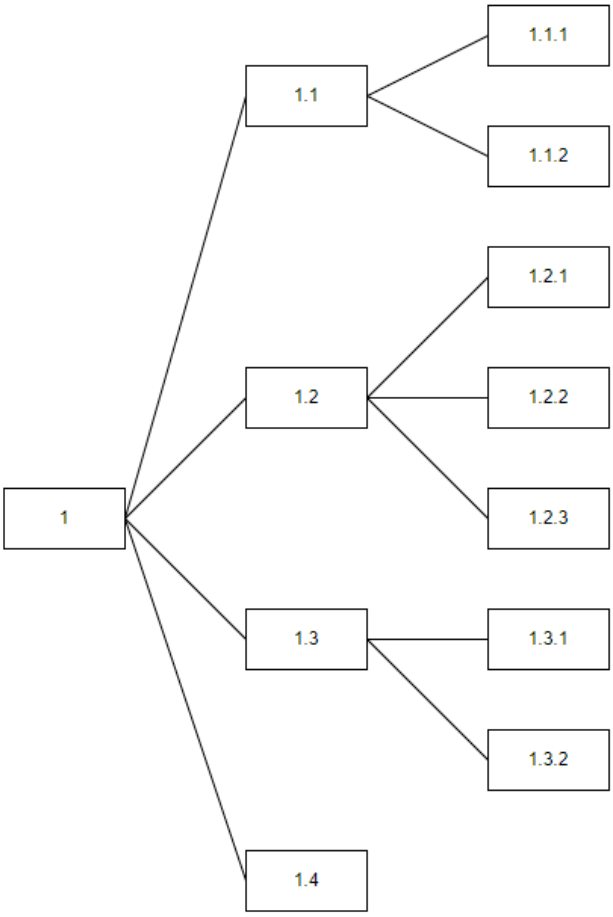

Рисунок 1.8 – Графічне представлення дерева цілей

Генеральною ціллю є реалізація інтелектуальної системи підбору комплектуючих ПК, яка декомпонується на наступні рівні:

4. Підсистема управління доступом: реалізація модулів реєстрації та авторизації з акцентом на безпеку персональних даних.

5. Аналітичне ядро (ML-модуль): збір, препроцесинг даних та розробка прогностичної моделі з подальшою оцінкою її метрик якості.

6. Серверна та інфраструктурна логіка: побудова гнучкого API для обробки клієнтських запитів та впровадження контейнеризації на базі Docker Compose для забезпечення мобільності розгортання.

7. Користувацька підтримка: розробка системи контекстних інструкцій та візуальних гайдів для спрощення процесу експлуатації системи.

1.6 Висновок до першого розділу

У першому розділі проведено комплексне дослідження предметної області, методів інтелектуального аналізу та існуючих технологічних рішень у сфері підбору комп'ютерного апаратного забезпечення. За результатами розділу можна зробити наступні висновки:

1. Проаналізовано стан ринку комплектуючих, який у 2024-2026 роках характеризується критичним ускладненням стандартів сумісності та появою нового сегменту AI-орієнтованих систем (AI PC). Виявлено, що «парадокс вибору» для кінцевого користувача створює реальну потребу в автоматизованих системах підтримки прийняття рішень.

2. Виконано порівняльний аналіз аналогів, який показав, що більшість існуючих онлайн-конфігураторів є детермінованими системами, орієнтованими на професіоналів. Вони не здатні обробляти високорівневі запити новачків та мають закриту монолітну архітектуру, що обмежує їх масштабування.

3. Обґрунтовано вибір математичного апарату. Шляхом порівняння методів k-NN, нейронних мереж та ансамблевих моделей встановлено, що алгоритм Random Forest є оптимальним для даної задачі. Він забезпечує необхідну стійкість до перенавчання на напівсинтетичних даних та ефективну роботу з гетерогенними ознаками комплектуючих.

4. Проведено системний аналіз, у ході якого побудовано дерева проблем та цілей. Визначено, що наукова новизна роботи зосереджена на методиці формування напівсинтетичних навчальних вибірок.

5. Сформульовано вимоги до системи, де ключовим аспектом визначено поєднання мікросервісного підходу на базі Java та Python із фронт-енд рішенням на React JS для забезпечення гнучкості та швидкодії прототипу.

2 ПРОЕКТУВАННЯ АРХІТЕКТУРИ ТА ТЕОРЕТИЧНЕ ОБҐРУНТУВАННЯ СИСТЕМИ

2.1 Розробка концептуальної архітектури системи на основі мікросервісного підходу

Проектування архітектури інтелектуальної системи підбору комплектуючих ПК потребує врахування високих вимог до масштабованості, гнучкості та можливості інтеграції різнорідних технологічних стеків. У межах даної магістерської роботи обрано мікросервісну архітектуру, що дозволяє розділити складну бізнес-логіку на сукупність автономних сервісів, кожен з яких вирішує специфічне коло завдань [18].

Для обґрунтування обраного підходу було проведено порівняння з традиційною монолітною архітектурою. Монолітні системи, де інтерфейс користувача, логіка підбору та обробка даних знаходяться в межах одного виконавчого файлу, мають суттєві недоліки для інтелектуальних систем:

- Складність оновлення ML-моделей: Будь-яка зміна в алгоритмі випадкового лісу вимагала б повного перескладання всієї системи.
- Обмеженість технологічного вибору: Важко поєднати високу продуктивність Java-платформи для обробки запитів та широкі можливості Python-бібліотек для машинного навчання в одному моноліті.
- Ризики відмов: Помилка в модулі аналізу даних могла б призвести до повної зупинки веб-інтерфейсу.

Натомість мікросервісний підхід дозволяє ізолювати ресурсомісткі обчислення ML-моделі від основного API, забезпечуючи стабільність та можливість незалежного горизонтального масштабування.

Запропонована архітектура складається з чотирьох основних рівнів (шарів), які взаємодіють між собою через стандартизовані протоколи :

1. Рівень представлення (Frontend Service): Реалізований на базі бібліотеки React JS. Цей сервіс відповідає за збір вимог користувача (бюджет,

цілі використання) та візуалізацію отриманих результатів. Взаємодія з бек-ендом відбувається за допомогою асинхронних HTTP-запитів.

2. Рівень оркестрації та бізнес-логіки (Backend API Service): Побудований на фреймворку Java Spring Boot. Цей мікросервіс виконує роль «центрального вузла» або API Gateway. Він відповідає за:

- Автентифікацію та валідацію вхідних даних;
- Управління транзакціями в базі даних;
- Формування запитів до ML-модуля;
- Агрегацію фінального звіту для користувача.

3. Аналітичний рівень (ML Inference Service): Спеціалізований мікросервіс на мові Python. Його єдине завдання – приймати векторизовані вимоги користувача, пропускати їх крізь навчену модель Random Forest та повертати прогнозовані ідентифікатори оптимальних комплектуючих.

4. Рівень даних (Data Storage): Реляційна база даних PostgreSQL, що зберігає інформацію про компоненти, їхні технічні характеристики, а також логи тренування моделей.

Основним протоколом обміну даними між мікросервісами обрано REST (Representational State Transfer) з використанням формату JSON. Це забезпечує легкість інтеграції та високу швидкість парсингу даних [19].

- Spring Boot → Python ML: Передається JSON-об'єкт із параметрами користувача.
- Python ML → Spring Boot: Повертається масив рекомендованих ID комплектуючих та імовірнісна оцінка впевненості моделі в обраному варіанті.

Таблиця 2.1 – Розподіл функціональної відповідальності між компонентами архітектури

Компонент системи	**Технологічний стек**	**Ключові функціональні обов'язки**	**Роль у контексті наукового дослідження**
Frontend Service	React JS, Axios, Lucide	Збір вимог користувача; візуалізація підібраних конфігурацій; управління станом інтерфейсу.	Забезпечення низького порогу входження через ергономічний UI/UX та абстрагування складних процесів.
Backend API Gateway	Java 17+, Spring Boot, JPA	Оркестрація мікросервісів; валідація бізнес-правил; управління транзакціями; фінальна верифікація сумісності.	Забезпечення цілісності системи та стабільності взаємодії між різнорідними модулями (Java ↔ Python).
ML Inference Service	Python 3.10+, Scikit-learn, Flask	Препроцесинг вхідних ознак; виконання прогнозів на основі моделі Random Forest; повернення ідентифікаторів оптимальних вузлів.	Реалізація інтелектуального ядра системи та практичне застосування методів машинного навчання.

Продовження таблиці 2.1

Компонент системи	**Технологічний стек**	**Ключові функціональні обов'язки**	**Роль у контексті наукового дослідження**
Database Storage	PostgreSQL 15	Збереження гетерогенних специфікацій компонентів; логування результатів підбору; зберігання даних користувачів.	Створення надійної інформаційної бази для навчання моделі та функціонування підсистеми фільтрації.
Infrastructure Layer	Docker, Docker Compose	Контейнеризація сервісів; управління мережевою взаємодією між контейнерами; ізоляція середовищ виконання.	Забезпечення мобільності системи та відтворюваності експериментального середовища розробки.

Використання мікросервісів у даній роботі не є надлишковим, оскільки воно дозволяє реалізувати принцип Polyglot Programming. Для даної роботи це критично, бо дає змогу зосередитися на науковій новинні (генерація даних та ML) у Python-середовищі, не відволікаючись на складнощі низькорівневої обробки веб-запитів, які краще реалізовані в екосистемі Java Spring Boot [20].

Така структура також закладає фундамент для майбутнього розширення системи: наприклад, додавання мікросервісу моніторингу цін на базі Node.js або сервісу сповіщень без зміни основної логіки підбору.

2.2 Математична модель оцінки сумісності та продуктивності комплектуючих (WIP)

Процес автоматизованого проектування оптимальної конфігурації ЕОМ можна представити як задачу пошуку в багатовимірному просторі ознак, де цільова функція спрямована на максимізацію продуктивності при дотриманні бюджетних та технічних обмежень.

Нехай C – множина всіх доступних на ринку комплектуючих, яка складається з підмножин за типами компонентів:

$$C = \{C\_cpu, C\_gpu, C\_mb, C\_ram, C\_psu, C\_drive, C\_case\} \quad (2.1)$$

де C_cpu – множина процесорів, C_mb – материнських плат і так далі.

Тоді конфігурація ПК K є впорядкованим набором елементів, де з кожної підмножини обирається рівно один (або n для дисків/ОЗП) елемент:

$$K = c\_1, c\_2, c\_n, \text{де } c\_i \in C\_i \quad (2.2)$$

Математична модель повинна гарантувати фізичну та логічну сумісність обраних вузлів. Формально це виражається через предикат сумісності P(c_i, c_j), який набуває значення 1 (істина), якщо компоненти сумісні, та 0 (хибність) у протилежному випадку:

$$f\_comp(K) = Prod\_\{i,j\} P(c\_i, c\_j) = 1 \quad (2.3)$$

Ключові обмеження включають:

- Сокетна сумісність: Socket(c_cpu) = Socket(c_mb).
- Енергетичний баланс: sum_{i=1}^n TDP(c_i) k_{safe} Power(c_psu), де k_safe – коефіцієнт запасу потужності (зазвичай k_safe ~ 1.25).
- Габаритна сумісність: L(c_gpu) L_max(c_case), де L – лінійні розміри.

Для оцінки якості конфігурації вводиться інтегральний показник продуктивності R(K), який залежить від сфери застосування S in {Gaming, Professional, Office}. Кожен компонент має свою «вагу» w_i для конкретного сценарію S:

$$R(K, S) = Sum_{i=1}^n w_i(S) score(c_i) \quad (2.4)$$

де score(c_i) – нормалізований показник продуктивності компонента (наприклад, процентиль у бенчмарках).

Вагові коефіцієнти w_i визначають пріоритетність підсистем. Наприклад, для ігрової системи w_gpu > w_cpu, тоді як для розрахункових станцій (Professional) пріоритет зміщується в бік w_cpu та w_ram.

Таблиця 2.2 – Матриця вагових коефіцієнтів w_i для різних сценаріїв використання ПК

Сценарій використання	**wcpu (Процесор)**	**wgpu (Відеокарта)**	**wram (ОЗП)**	**wdrive (Накопичувач)**
Gaming (Ігрова станція)	0.30	0.50	0.10	0.10
Professional (Відео/3D)	0.45	0.25	0.20	0.10
Office (Офісні задачі)	0.40	0.10	0.20	0.30

Однією з наукових задач є уникнення дисбалансу системи. Математично це виражається через мінімізацію дельти між відносними показниками продуктивності ключових вузлів:

$$Delta(c_i, c_j) = |score(c_i) - score(c_j)| \quad (2.5)$$

Якщо Delta перевищує певне порогове значення T, конфігурація вважається неоптимальною (один компонент суттєво обмежує інший).

Оскільки аналітичний розрахунок усіх можливих комбінацій C є NP-складною задачею через комбінаторний вибух, модель Random Forest виступає як апроксиматор цільової функції. Навчена модель прогнозує вектор оптимальних характеристик V на основі вхідного вектора вимог користувача U:

$$V = RF(U) \tag{2.6}$$

Після чого система виконує пошук у базі даних компонентів, що найбільш точно відповідають вектору V за метрикою Евклідової відстані або косинусної подібності.

Побудована математична модель дозволяє перейти від суб'єктивного вибору до об'єктивного розрахунку оптимальності. Використання вагових коефіцієнтів та аналізу «вузьких місць» забезпечує наукову обґрунтованість підбору, а інтеграція ML-моделі як оптимізатора дозволяє ефективно оперувати великими масивами даних про комплектуючі в режимі реального часу [21].

2.3 Методологія формування та збагачення напівсинтетичних наборів даних для навчання моделі

Однією з головних перешкод при розробці інтелектуальних систем підбору апаратного забезпечення є відсутність репрезентативних, збалансованих та актуальних наборів даних (datasets). Традиційні методи збору даних (наприклад, парсинг форумів або сайтів з оголошеннями) надають вибірки з високим рівнем «шуму» – неоптимальні конфігурації, застарілі компоненти або помилкові зв'язки [22].

Для вирішення цієї проблеми у даній роботі запропоновано методику формування напівсинтетичних наборів даних, яка базується на комбінаторному синтезі конфігурацій із подальшою багатокритеріальною оцінкою їхньої якості.

Процес підготовки даних розділено на три стратегічні етапи (відповідно до алгоритмічної логіки, закладеної в прототипі):

1. Етап 1: Формування бази реальних специфікацій (Seed Data). На початковому етапі виконується збір технічних характеристик окремих комплектуючих із відкритих джерел (API виробників, технічні каталоги). Дані структуруваються у форматі CSV/JSON, де кожен рядок – це унікальний компонент із набором ознак: архітектура, TDP, кількість ядер/потоків, тип пам'яті, габарити тощо. Ці дані є «фундаментом» для подальшого синтезу.

2. Етап 2: Процедурна генерація конфігурацій (Combinatorial Synthesis). На цьому етапі система виконує автоматичне поєднання компонентів. Оскільки кількість можливих комбінацій є астрономічною, генерація обмежена жорсткими фільтрами сумісності (Socket, RAM type, Form factor), описаними в підрозділі 2.2. Алгоритм випадковим чином обирає базові компоненти (CPU, GPU) та доповнює їх сумісними елементами, формуючи «сиру» конфігурацію. Таким чином створюється масив із тисяч варіантів, що охоплюють усі цінові сегменти.

3. Етап 3: Оцінка та маркування (Scoring & Labeling). Для того, щоб модель Random Forest могла навчитися розрізняти «вдалі» та «невдалі» збірки, кожній синтетичній конфігурації присвоюється цільове значення (target) – рейтинг оптимальності. Розрахунок рейтингу базується на нормалізованих показниках продуктивності та вартості.

Для підвищення узагальнюючої здатності моделі до отриманих даних застосовується техніка Data Augmentation. Ми навмисно вносимо у вибірку невеликий відсоток (5-10%) «субоптимальних» конфігурацій (наприклад, з надлишковим блоком живлення або дещо застарілою пам'яттю). Це дозволяє моделі Random Forest навчитися ідентифікувати не лише ідеальні варіанти, а й розпізнавати помилкові патерни, що робить її більш стійкою до варіативності реального ринку [23].

Таблиця 2.3 – Порівняння характеристик початкової та розширеної (напівсинтетичної) вибірок

Характеристика порівняння	**Початкова вибірка (Seed Data)**	**Розширена вибірка (Semi-synthetic Dataset)**
Обсяг даних (кількість записів)	~500–1000 унікальних компонентів.	10 000+ цілісних конфігурацій ПК.
Тип об'єктів аналізу	Атомарні (окремі специфікації CPU, GPU, RAM).	Комплексні (впорядковані набори сумісних вузлів).
Статистичний розподіл	Нерівномірний (зміщення в бік популярних або флагманських моделей).	Рівномірний (збалансоване представлення всіх бюджетних категорій).
Цільова мітка (Target Label)	Відсутня (лише технічні описи).	Наявна (розрахований інтегральний рейтинг оптимальності R).
Технічна сумісність	Не визначена (дані існують незалежно).	100% верифікована за жорсткими логічними предикатами.
Рівень «шуму» та варіативності	Мінімальний (лише заводські параметри).	Контрольований (внесено 5–10% суб'опитимальних збірок для регуляризації).
Репрезентативність категорій	Залежить від повноти парсингу ринку.	Формується штучно для охоплення всіх сценаріїв (Gaming/Office/Pro).
Кореляційна складність	Лінійна (характеристики окремого вузла).	Багатофакторна (врахування ефектів Bottleneck та синергії вузлів).

Запропонований підхід дозволяє подолати проблему дефіциту даних без залучення дорогої праці експертів для ручного складання тисяч варіантів ПК. Згенерований датасет є динамічним: при появі нових моделей відеокарт чи процесорів система автоматично проводить цикл перегенерації та донавчання моделі, що забезпечує актуальність рекомендацій системи в режимі реального часу [24].

Використання напівсинтетичних даних є оптимальним рішенням для інтелектуальних систем у вузькоспеціалізованих технічних доменах. Математично обґрунтоване маркування згенерованих збірок дозволяє перетворити задачу підбору на задачу регресійного аналізу, де модель навчається прогнозувати оптимальність конфігурації на основі складних взаємозв'язків між характеристиками її вузлів.

2.4 Проектування інформаційного забезпечення системи: розробка схеми бази даних PostgreSQL

Ефективність функціонування інтелектуальної системи підбору комплектуючих ПК безпосередньо залежить від якості проектування її інформаційного забезпечення. База даних (БД) у межах даної роботи виконує подвійну функцію: вона виступає як сховище актуальних характеристик компонентів для формування фінальних конфігурацій та як джерело ознак (features) для моделі машинного навчання Random Forest [25].

Для реалізації системи обрано об'єктно-реляційну систему керування базами даних PostgreSQL. Даний вибір зумовлений такими науково-технічними чинниками:

- Підтримка складних типів даних. PostgreSQL ефективно працює з числовими даними подвійної точності, що є критичним для зберігання бенчмарків та показників енергоефективності.
- Надійність та відповідність принципам ACID. Забезпечення цілісності даних при виконанні складних вибірок для генерації синтетичних наборів.

- Продуктивність при агрегації. Висока швидкість обробки JOIN-запитів при перевірці логічної сумісності (наприклад, відповідність сокетів процесора та материнської плати).

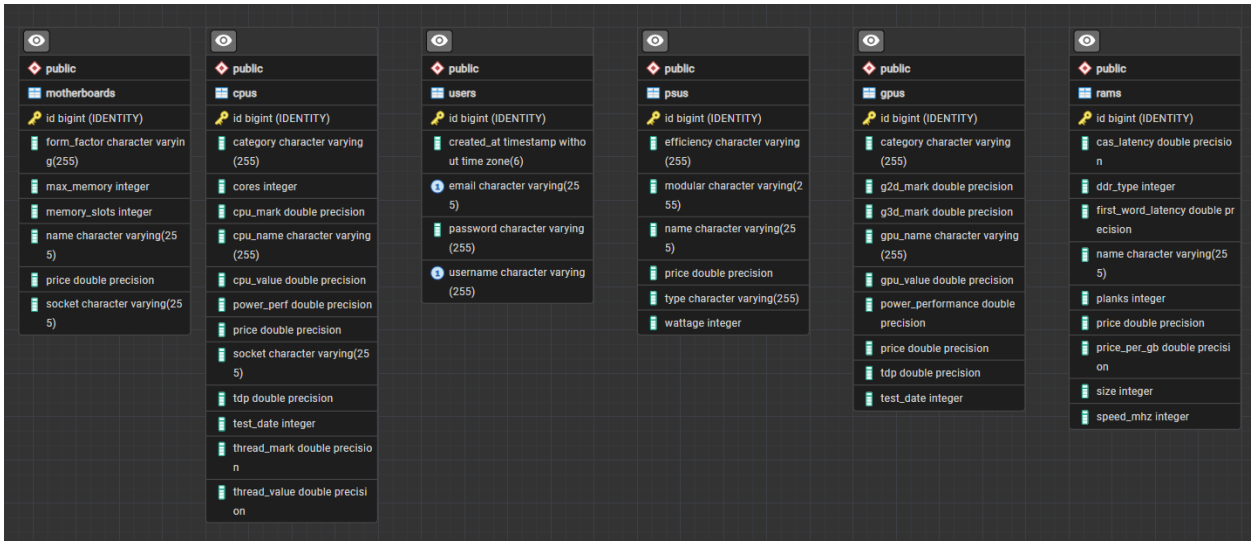

Рисунок 2.2 – Схема бази даних

Схема бази даних розроблена з дотриманням вимог третьої нормальної форми (3NF) для мінімізації надлишковості даних та уникнення аномалій оновлення. Інформаційна модель складається з декількох ключових доменів та наведена на рисунку рисунок 2.2:

- Домен обчислювальних модулів (cpus, gpus): Ці таблиці містять не лише технічні характеристики (кількість ядер, TDP), а й синтетичні показники продуктивності (cpu_mark, thread_mark, g3d_mark). Ці поля є ключовими для математичної моделі оцінки оптимальності конфігурації, описаної в підрозділі 2.2.
    - Атрибути cpu_value та gpu_value: Відображають співвідношення продуктивності до ціни, що дозволяє моделі ML швидше знаходити «економічний оптимум».
- Домен системної сумісності (motherboards, rams, psus): Ці таблиці містять параметри-обмеження:
    - Таблиця motherboards: поля socket та form_factor є основними фільтрами при процедурній генерації даних (Step 2).

- Таблиця psus: поле wattage використовується для верифікації енергетичного балансу системи.
- Таблиця rams: атрибути ddr_type та speed_mhz визначають сумісність із платформою та загальну швидкість обміну даними.

- Домен користувачів (users): Забезпечує персоніфікацію доступу до системи. Зберігання полів created_at та email дозволяє в майбутньому реалізувати систему збережених конфігурацій та історію запитів для кожного користувача.

Таблиця 2.4 – Аналіз атрибутивного складу бази даних та їх роль у ML-моделі

Сутність	**Ключові атрибути**	**Значення для Random Forest**
CPUs	cpu_mark, tdp, socket	Визначає потужність та вимоги до охолодження
GPUs	g3d_mark, g2d_mark, price	Основний фактор графічної продуктивності
Motherboards	socket, form_factor	Визначає жорсткі обмеження сумісності
RAMs	size, speed_mhz, ddr_type	Впливає на загальну швидкість системи
PSUs	wattage, efficiency	Параметр надійності та стабільності системи

Спроектована структура дозволяє легко розширювати систему без зміни існуючої логіки. Наприклад, додавання таблиць для накопичувачів (SSD/HDD) або систем охолодження відбуватиметься за аналогічним принципом із використанням ідентифікаторів id bigint як первинних ключів для швидкого індексування [26].

Зв'язки між таблицями в межах мікросервісів реалізуються на рівні бізнес-логіки в Java Spring Boot за допомогою JPA (Java Persistence API), що дозволяє

уникнути надмірної завантаженості бази даних тригерами та складними каскадними зв'язками, забезпечуючи високу швидкодію системи.

2.5 Моделювання бізнес-процесів системи

Для забезпечення цілісності розробки та розуміння механізмів функціонування системи на різних рівнях абстракції проведено комплексне моделювання процесів. Це дозволяє верифікувати логіку роботи системи ще до етапу повної програмної реалізації [27].

На верхньому рівні абстракції процес підбору комплектуючих представлено за допомогою методології IDEF0, як зображено на рисунку 2.3.

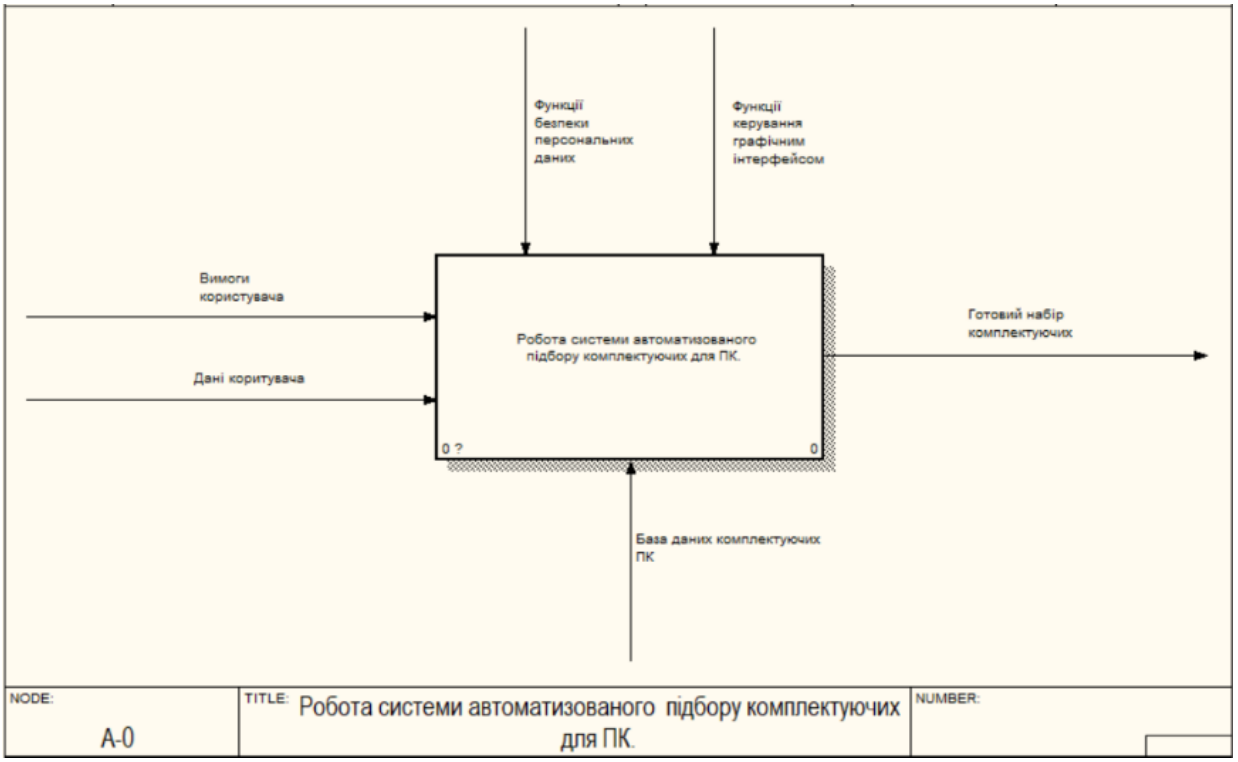

Рисунок 2.3 – Контекстна діаграма функціонування системи в нотації IDEF0

Контекстна діаграма (A-0) відображає систему як єдиний функціональний блок, де:

- Вхідними даними (Input) є запити користувача (бюджет, тип ПК) та його дані автопризації.

- Керуючими впливами (Control) виступають інтерфейс та інструменти бепеки.
- Механізмами (Mechanism) є розроблене програмне забезпечення (мікросервіси Java/Python) та БД.
- Результатом (Output) є сформована та валідована конфігурація ЕОМ.

Декомпозиція головного процесу 1-го рівня дозволяє виділити основні етапи роботи сервісу [28]. Вона зображена на рисунку 2.4.

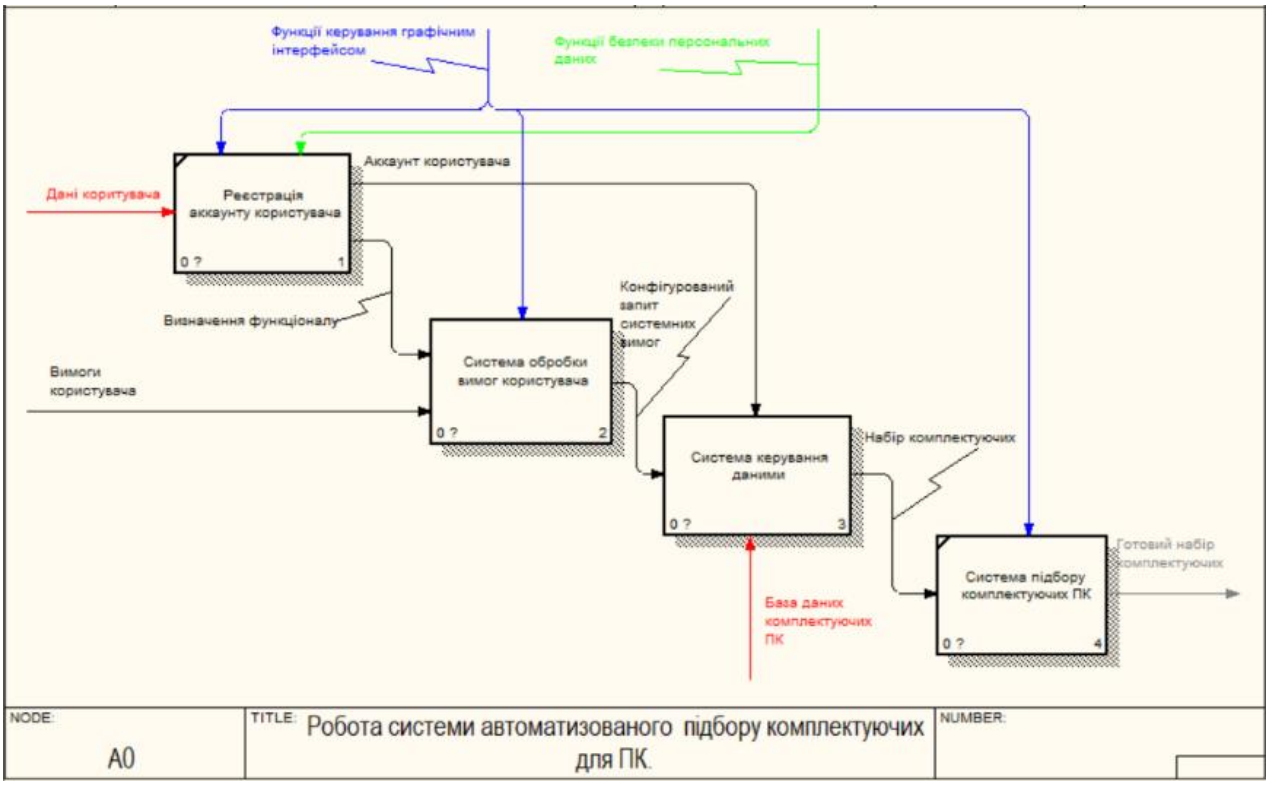

Рисунок 2.4 – Діаграма декомпозиції 1-го рівня в нотації IDEF0

Враховуючи мікросервісну природу системи, особлива увага приділена діаграмам потоків даних (Data Flow Diagrams) , як зображено на рисунку 2.5. Нотація DFD дозволяє візуалізувати рух інформації між зовнішніми сутностями (користувач), процесами обробки та накопичувачами даних (PostgreSQL).

Особливістю DFD у даному проекті є відображення «точок розгалуження», де дані передаються від основного API (Spring Boot) до аналітичного модуля (Python) та повертаються у вигляді прогнозів [29].

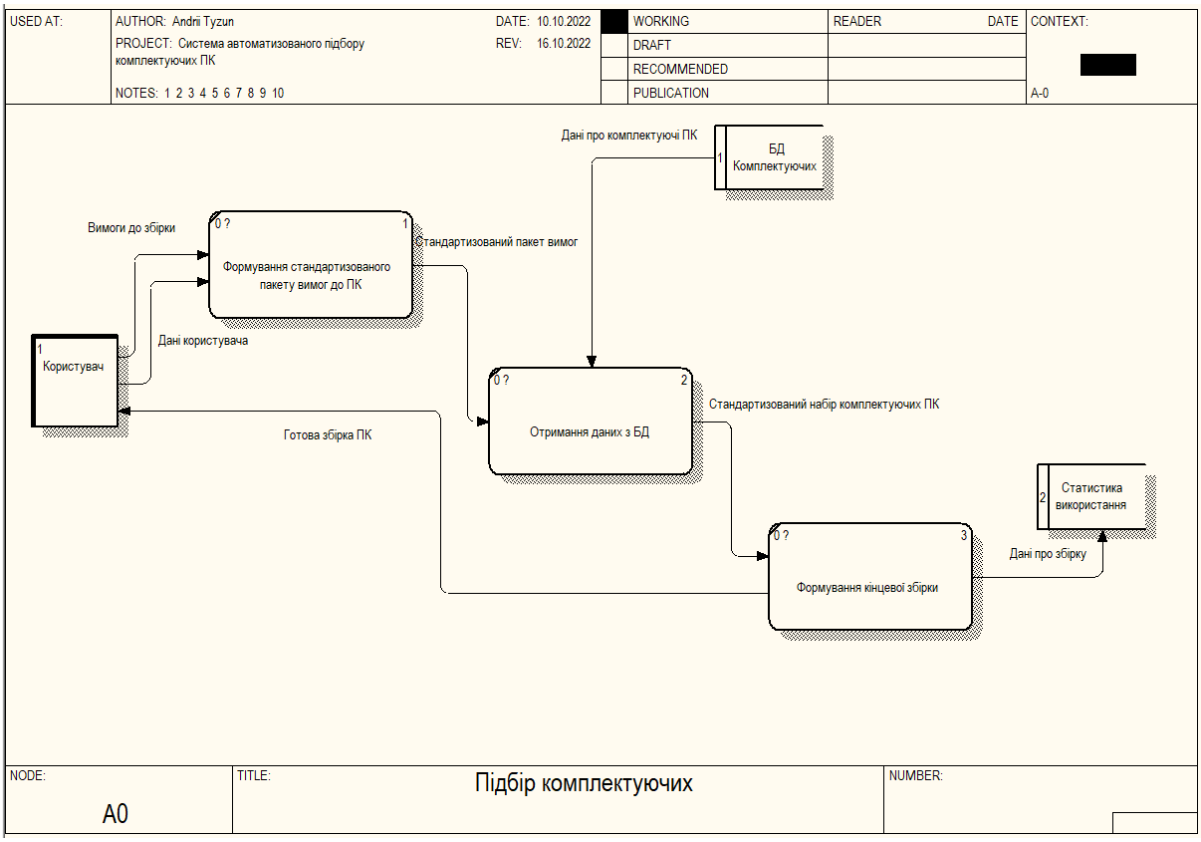

Рисунок 2.5 – Діаграма потоків даних (DFD) процесу інтелектуального підбору

Для деталізації програмного функціоналу було створено UML Use-case діаграму [30], яка зображена на рисунку 2.6.

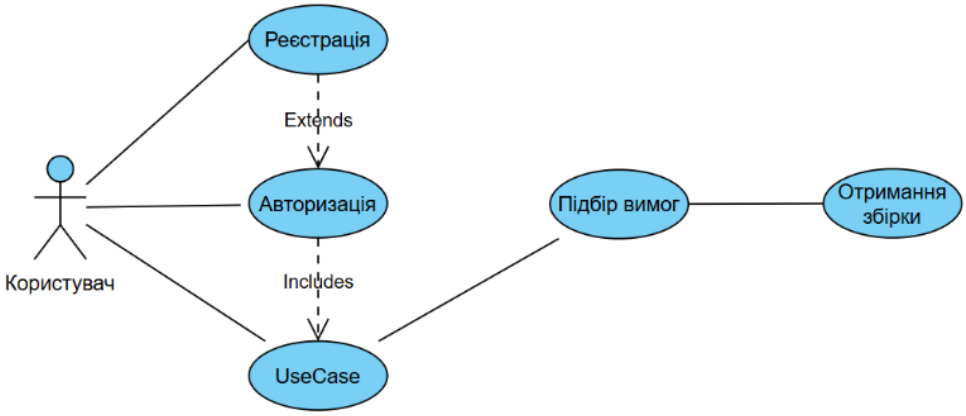

Рисунок 2.6 – UML діаграма прецедентів

Діаграма охоплює 5 основних функцій, які користувач може виконувати в системі. Ці прецеденти включають різні аспекти використання програми, такі як авторизація, реєстрація, вибір вимог і отримання конфігурацій ПК [31].

2.6 Висновок до другого розділу

У другому розділі проведено повний цикл проектування інтелектуальної системи підбору комплектуючих ПК, результати якого стали фундаментом для практичної реалізації.

1. Розроблено концептуальну мікросервісну архітектуру, що базується на розділенні функцій інтерфейсу (React), бізнес-логіки (Java Spring Boot) та інтелектуального аналізу (Python/ML). Це дозволило забезпечити високу масштабованість системи та можливість використання оптимального інструментарію для кожної задачі.
2. Сформовано математичну модель оцінки оптимальності, яка враховує вагові коефіцієнти для різних сценаріїв використання (Gaming, Professional, Office) та включає механізм аналізу «вузьких місць» (bottleneck analysis) для досягнення збалансованості конфігурацій.
3. Запропоновано та обґрунтовано методику формування напівсинтетичних даних, що є ключовим аспектом наукової новизни роботи. Описано триетапний процес від збору сирих даних до автоматизованого маркування та регуляризаційного збагачення датасету для навчання моделі Random Forest.
4. Спроектовано структуру бази даних PostgreSQL, яка оптимізована для зберігання гетерогенних характеристик компонентів та швидкої вибірки параметрів, необхідних для роботи алгоритмів машинного навчання.
5. Здійснено комплексне моделювання бізнес-процесів з використанням нотацій IDEF0, DFD та UML. Побудовані моделі підтвердили несуперечність архітектурних рішень та готовність логічної структури до програмного втілення.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ СИСТЕМИ

3.1 Вибір та обґрунтування інструментальних засобів реалізації

Етап програмної реалізації магістерської роботи потребує ретельного підбору технологічного стека, який забезпечить стабільну взаємодію між автономними мікросервісами та високу точність обчислень інтелектуального модуля. Враховуючи гетерогенність задач (веб-інтерфейс, бізнес-логіка, ML-аналітика), було обрано комбінований стек технологій.

Для реалізації центрального API-шлюзу та сервісу бізнес-логіки обрано платформу Java з використанням фреймворку Spring Boot [32]. Це рішення базується на наступних перевагах:

- Висока продуктивність та багатопотоковість. Java забезпечує стабільну роботу при великій кількості одночасних запитів, що важливо для систем підтримки прийняття рішень.
- Spring Boot Starters. Дозволяє швидко налаштувати систему без надмірного конфігурування XML-файлів. Використання Spring Data JPA спрощує взаємодію з PostgreSQL, а Spring Security закладає фундамент для захисту даних користувачів.
- Підтримка мікросервісів. Екосистема Spring ідеально підходить для створення RESTful сервісів, які легко масштабуються незалежно один від одного.

Незважаючи на те, що основна логіка написана на Java, для сервісу машинного навчання було обрано Python. Це зумовлено наявністю спеціалізованих бібліотек, які є де-факто стандартом у науковому середовищі:

- Scikit-learn: Бібліотека, яка надає найбільш стабільну та оптимізовану реалізацію алгоритму Random Forest. Вона дозволяє легко проводити гіперпараметричне налаштування моделі та оцінку її точності.
- Pandas та NumPy: Необхідні для швидкої обробки та векторизації напівсинтетичних наборів даних, описаних у Розділі 2.

- Flask / FastAPI: Легковажні веб-фреймворки, що дозволяють перетворити ML-скрипт на повноцінний мікросервіс з REST API за лічені хвилини.

Для розробки користувацького інтерфейсу обрано бібліотеку React JS [33]. Основні чинники вибору:

- Компонентний підхід: Дозволяє створити модульний інтерфейс, де кожен елемент (форма вибору CPU, таблиця результатів) є незалежним об'єктом.
- Virtual DOM: Забезпечує високу швидкість рендерингу при динамічній зміні параметрів підбору, що створює комфортний досвід для користувача (UI/UX).
- State Management: Використання хуків (Hooks) дозволяє ефективно керувати станом складної форми підбору конфігурації без перевантаження сторінки.

Для наочності та систематизації обраних засобів у таблиці 3.1 наведено їх порівняльну характеристику.

Таблиця 3.1 – Технологічний стек програмного прототипу

Рівень системи	**Технологія**	**Основна функція**	**Роль у магістерському дослідженні**
Backend / API	Java 17+, Spring Boot	Оркестрація даних, бізнес-правила	Забезпечення надійності та архітектурної цілісності
ML Engine	Python 3.10+, Scikit-learn	Прогнозування та класифікація	Реалізація наукової новизни (Random Forest)
Frontend	React JS, Axios	Взаємодія з користувачем	Візуалізація результатів інтелектуального підбору
Database	PostgreSQL 15	Збереження характеристик	Надійне сховище для гетерогенних даних

Інструментальне середовище розробки. Для підвищення ефективності написання коду та налагодження системи використано наступне середовище:

1. IntelliJ IDEA (Ultimate Edition): Для розробки Java-сервісів завдяки вбудованій підтримці Spring та інструментів профілювання.

2. PyCharm / VS Code: Для написання ML-модулів та проведення експериментів з навчанням моделі.

3. Postman: Для тестування взаємодії між мікросервісами (RESTful API testing).

4. Git / GitHub: Для контролю версій та забезпечення ітеративного процесу розробки.

Обраний стек технологій (Java, Python, React) дозволяє реалізувати повноцінну мікросервісну систему, де кожна мова програмування використовується для тих задач, у яких вона є найбільш ефективною. Це забезпечує не лише працездатність прототипу, а й відповідає вимогам до сучасної професійної розробки складних інтелектуальних систем [34].

3.2 Реалізація серверної частини

Центральним елементом розробленої системи є мікросервіс на базі фреймворку Java Spring Boot, який виконує роль оркестратора та API-шлюзу. Основне завдання цього модуля полягає в агрегації даних із реляційної бази, управлінні бізнес-процесами підбору та забезпеченні безперебійного зв’язку між клієнтським інтерфейсом та аналітичним ML-модулем.

Архітектурна структура проекту. Програмна реалізація сервісу базується на багатошаровій архітектурі (Layered Architecture), що забезпечує чіткий поділ відповідальності (Separation of Concerns) [35]. Основні пакети проекту включають:

- controller: рівень обробки HTTP-запитів, де визначено REST-ендпоінти для взаємодії з React-фронтендом.
- service: рівень бізнес-логіки, де реалізовано алгоритми попередньої фільтрації, валідації сумісності та підготовки даних для ML-моделі.

- repository: рівень доступу до даних, реалізований за допомогою Spring Data JPA, що забезпечує абстракцію над SQL-запитами до PostgreSQL.
- model / entity: опис сутностей бази даних (CPU, GPU, RAM тощо) як Java-об'єктів.
- dto (Data Transfer Objects): спеціалізовані об'єкти для обміну даними, що дозволяють передавати лише необхідну інформацію, приховуючи внутрішню структуру БД.

Інтеграція з базою даних та рівень персистенції. Для роботи з PostgreSQL використано технологію Hibernate, яка виступає в ролі ORM (Object-Relational Mapping). Це дозволяє оперувати комплектуючими як об'єктами класів, що значно пришвидшує розробку та мінімізує кількість помилок при роботі з типами даних. Кожна сутність (наприклад, Processor) містить анотації @Entity, @Table та описує поля, що відповідають схемі [36].

Реалізація механізму інтелектуального підбору. Ключовий процес у Spring Boot сервісі починається з отримання запиту від користувача. Алгоритм роботи контролера підбору можна розділити на наступні кроки:

1. Валідація запиту: Перевірка наявності обов'язкових параметрів (бюджет, цільове призначення).

2. Формування вектора ознак: Java-сервіс вилучає з бази даних потенційно підходящі компоненти та формує JSON-об'єкт для передачі у ML-сервіс.

3. Міжсервісна взаємодія (Internal API Call): Для звернення до Python-сервісу використано клієнт RestTemplate (або WebClient). Запит виконується асинхронно для запобігання блокуванню основного потоку.

4. Пост-обробка та верифікація: Отримавши прогноз від Random Forest (список ID рекомендованих вузлів), Spring Boot сервіс проводить фінальну перевірку на жорсткі обмеження сумісності (наприклад, габарити відеокарти та корпусу), які могли бути не враховані моделлю.

Забезпечення масштабованості та безпеки. Завдяки використанню Spring Boot, система підтримує зовнішню конфігурацію через application.yml, що дозволяє легко змінювати URL-адресу ML-сервісу або параметри підключення

до БД при розгортанні в Docker-контейнерах. Також реалізовано обробку виняткових ситуацій за допомогою @ControllerAdvice, що гарантує повернення зрозумілих JSON-помилок клієнту у разі технічних збоїв [37].

Реалізована серверна частина на Java Spring Boot забезпечує надійний фундамент для роботи всієї системи. Завдяки використанню сучасних патернів проектування та засобів автоматизації Spring Data, вдалося створити гнучкий API, здатний ефективно координувати роботу розподілених мікросервісів та забезпечувати високу швидкість обробки інтелектуальних запитів.

3.3 Програмна реалізація інтелектуального модуля на Python та інтеграція ML-моделі

Інтелектуальний модуль системи реалізований як автономний мікросервіс на мові Python. Такий підхід дозволяє ізолювати ресурсомісткі обчислення машинного навчання та використовувати багатий інструментарій екосистеми Data Science. Основна роль цього модуля полягає в обробці напівсинтетичних наборів даних, навчанні ансамблевої моделі та наданні прогнозів через REST API [38].

Програмний код модуля структурований за принципом функціонального розділення:

- Модуль завантаження та передобробки даних (data_processor.py): Використовує бібліотеку Pandas для зчитування даних із PostgreSQL або CSV-даптерів. Тут реалізовано логіку очищення даних від пропущених значень та векторизацію категоріальних ознак.
- Модуль моделювання (model_engine.py): Містить опис архітектури моделі Random Forest, процедури її навчання та збереження (серіалізації) у форматі .joblib або .pkl.
- Інтерфейсний модуль (app.py): Реалізований на базі легковажного фреймворку Flask (або FastAPI), що забезпечує приймання JSON-запитів від Java-сервісу та повернення результатів прогнозування.

Для побудови моделі використано клас RandomForestRegressor (або RandomForestClassifier) із бібліотеки Scikit-learn [39]. Враховуючи специфіку задачі – роботу з напівсинтетичними даними – було проведено підбір гіперпараметрів моделі для досягнення оптимального балансу між точністю та узагальнюючою здатністю:

- n_estimators = 100: Кількість дерев в ансамблі, що забезпечує стабільність прогнозу.
- max_depth: Обмеження глибини дерев для запобігання перенавчанню на специфічних синтетичних прикладах.
- random_state = 42: Для забезпечення відтворюваності результатів експериментів.

Математично процес прийняття рішення ансамблем для задачі регресії (оцінки рейтингу конфігурації) виглядає як усереднення прогнозів усіх N дерев.

Особливістю реалізації є блок адаптації даних [40]. Програма виконує нормалізацію вхідних характеристик компонентів за процентильним методом, що дозволяє звести різні одиниці виміру (МГц, ГБ, долари) до єдиної шкали.

Процес навчання моделі включає:

1. Розбиття вибірки: Використання функції train_test_split для поділу напівсинтетичного датасету на навчальну (80%) та тестову (20%) частини.

2. Навчання (Fitting): Процес побудови дерев рішень на основі згенерованих збалансованих конфігурацій.

3. Валідація: Оцінка метрик якості (MAE, RMSE, R^2), що детально розглядається у підрозділі 3.5.

Оскільки навчена модель Random Forest зберігається у бінарному вигляді, час виконання одного прогнозу (inference) становить менше 50 мс. Це дозволяє системі працювати в режимі реального часу, миттєво реагуючи на зміну фільтрів користувачем у веб-інтерфейсі.

Таблиця 3.3 – Перелік основних бібліотек Python та їх призначення у модулі

Бібліотека	**Версія**	**Функціональна роль**
Scikit-learn	1.2+	Реалізація Random Forest та метрик якості
Pandas	2.0+	Структурування напівсинтетичних даних
NumPy	1.24+	Високопродуктивні матричні обчислення
Flask/FastAPI	2.3+	Реалізація RESTful API мікросервісу
Joblib	1.2+	Серіалізація та десеріалізація моделей

Програмна реалізація інтелектуального модуля на Python підтвердила ефективність обраного стека технологій. Використання ансамблевих методів дозволило створити стійку модель, яка успішно інтегрується в загальну мікросервісну архітектуру та забезпечує високу точність підбору комплектуючих навіть в умовах використання синтетично згенерованих навчальних вибірок.

3.4 Розробка клієнтської частини системи на базі бібліотеки React JS

Клієнтська частина системи виконує роль інтерфейсу взаємодії між кінцевим користувачем та мікросервісною інфраструктурою. Основним завданням фронт-енд модуля є абстрагування складних технічних процесів (векторизація запитів, робота ML-моделі) та подання їх у вигляді інтуїтивно зрозумілого покрокового алгоритму підбору [41].

Використання бібліотеки React JS дозволило реалізувати декларативний підхід до побудови інтерфейсу, що є критичним для систем із великою кількістю динамічних елементів.

Основні переваги реалізації:

- Компонентно-орієнтована архітектура: Весь інтерфейс декомпоновано на незалежні функціональні блоки (форми вводу, картки комплектуючих, модальні вікна порівняння), що полегшує тестування та подальше розширення системи.
- Управління станом (State Management): Для збереження вибору користувача та результатів прогнозування використано вбудовані хуки useState та useContext, що забезпечує синхронізацію даних між різними частинами інтерфейсу без необхідності перевантаження сторінки.
- Ефективність оновлення (Virtual DOM): Система миттєво реагує на зміну вхідних параметрів (наприклад, зміна бюджету за допомогою слайдера), оновлюючи лише необхідні частини інтерфейсу.

Логічна структура інтерфейсу користувача [42]. Клієнтський додаток побудований за принципом односторінкового додатку (SPA – Single Page Application) та включає наступні основні модулі:

1. Форма реєстрації: Вхідна точка в систему, яка відповідає за реєстрацію користувача. Наведена на рисунку 3.1.

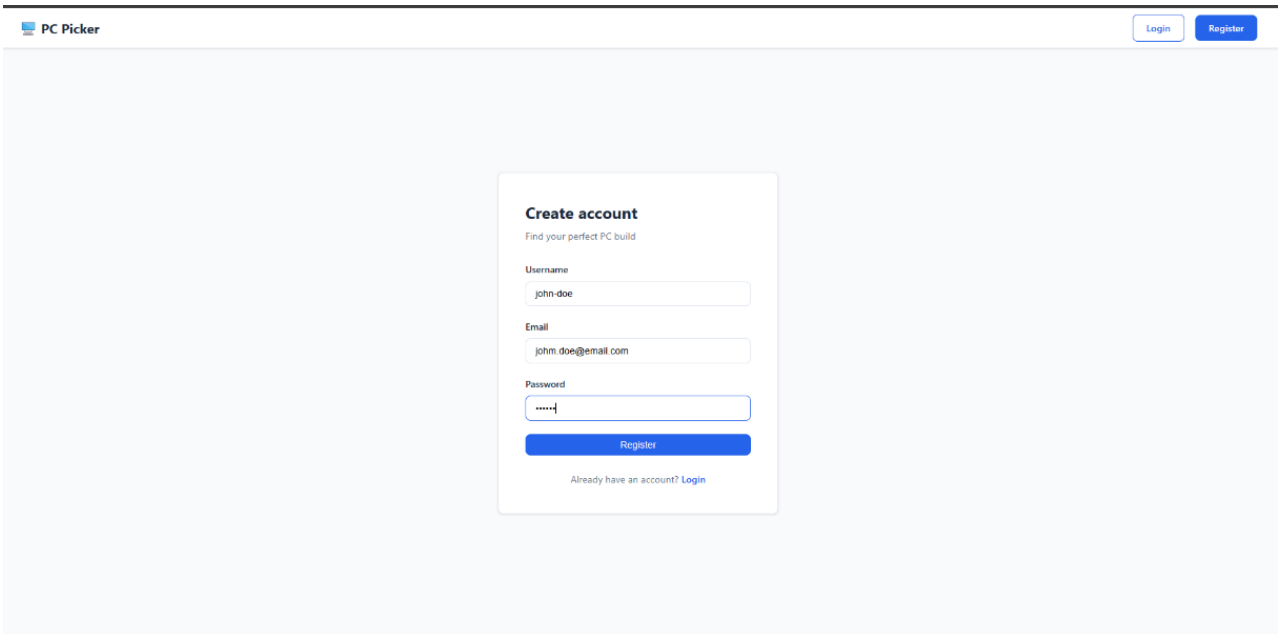

Рисунок 3.1 – Форма реєстрації користувача

2. Форма авторизації: Вхідна точка в систему, яка відповідає за авторизацію користувача. Наведена на рисунку 3.2.

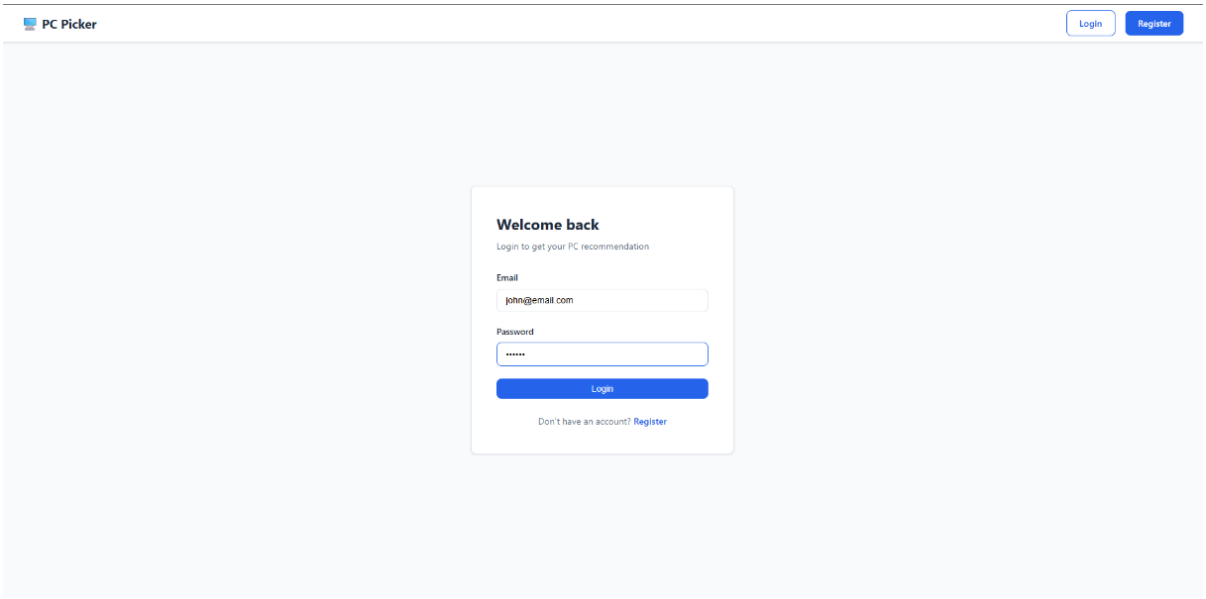

Рисунок 3.2 – Форма авторизації користувача

3. Конфігуратор: Головний вузол взаємодії, де користувач вказує тип ПК (Gaming, Office, Professional), цільові програми та граничний бюджет [43]. Дані з цієї форми агрегуються в єдиний JSON-об'єкт для відправки на сервер. Наведена на рисунку 3.3.

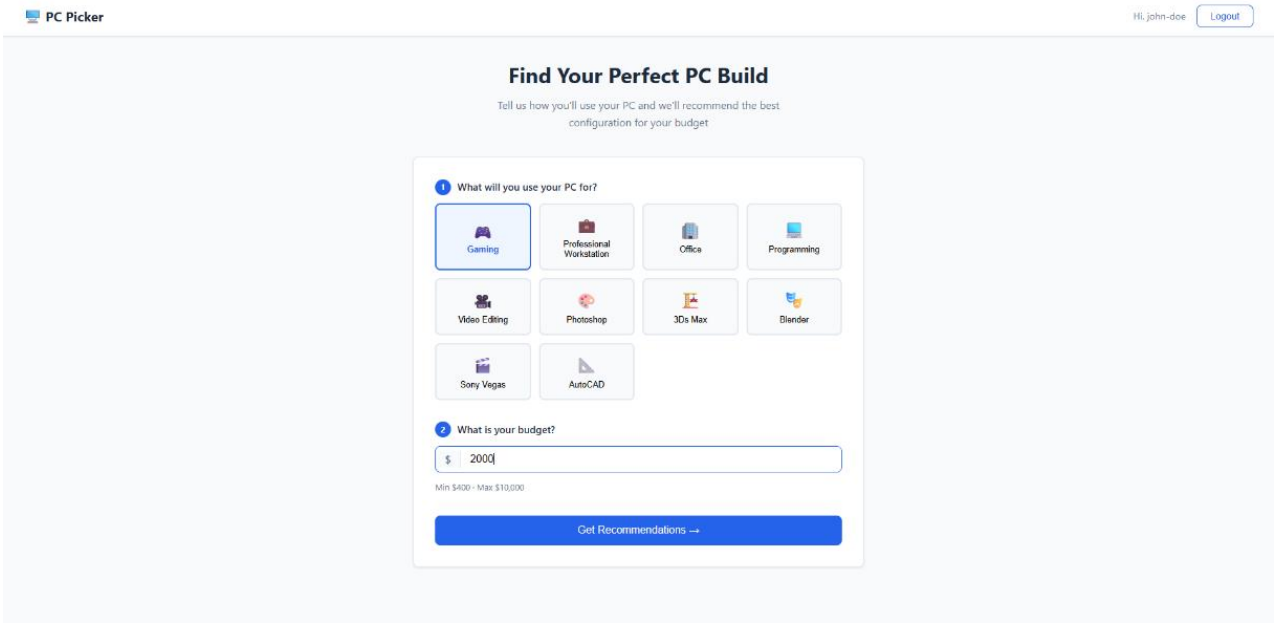

Рисунок 3.3 – Конфігуратор

4. Візуалізатор результатів: Після обробки запиту ML-моделлю, цей модуль відображає згенеровану конфігурацію [44]. Кожен компонент представлений окремою карткою з технічними характеристиками та рейтингом оптимальності, розрахованим системою. Наведена на рисунку 3.4.

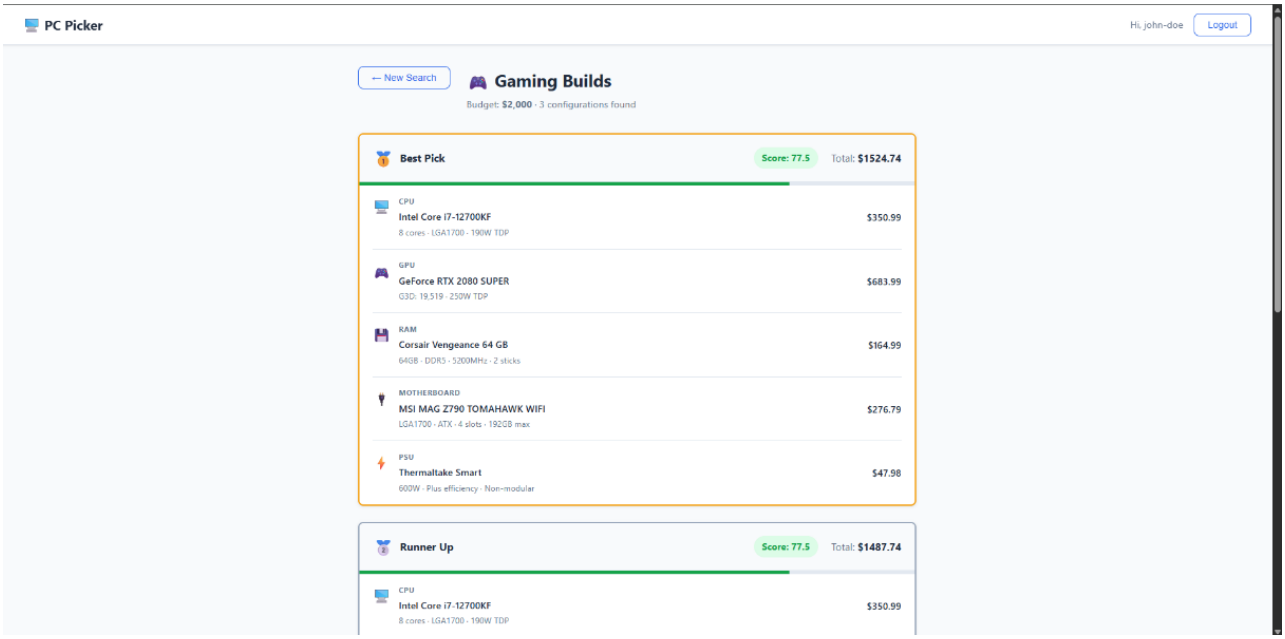

Рисунок 3.4 – Візуалізатор результатів

Міжсервісна взаємодія на рівні клієнта. Обмін даними між React-додатком та Java Spring Boot API реалізовано за допомогою бібліотеки Axios. Взаємодія побудована на асинхронних запитах, що дозволяє інтерфейсу залишатися чуйним під час очікування відповіді від ML-моделі.

Адаптивність та користувацький досвід (UX). Оскільки система орієнтована на користувачів без спеціальних технічних знань, у дизайні інтерфейсу застосовано принципи Material Design. Використання бібліотек стилів дозволило створити адаптивну верстку, яка коректно відображається на різних пристроях. Якщо користувач намагається вручну змінити компонент на несумісний або незбалансований, фронт-енд отримує валідаційні попередження від бізнес-логіки Java та миттєво інформує про це користувача.

Таблиця 3.4 – Основні компоненти та технології фронт-енд модуля

Компонент / Бібліотека	**Призначення**	**Функціональна роль**
React Hooks	Управління станом	Збереження параметрів підбору
Axios	HTTP-клієнт	Зв'язок із Spring Boot API
React Router	Маршрутизація	Навігація між сторінками каталогу та конфігуратора
SASS / CSS Modules	Стилізація	Забезпечення візуальної цілісності інтерфейсу
Lucide React	Графічні елементи	Візуалізація категорій комплектуючих

Розробка клієнтської частини на базі React JS забезпечила високу динамічність та інтерактивність системи. Завдяки компонентному підходу вдалося створити інтерфейс, який ефективно приховує складність внутрішніх обчислень, надаючи користувачеві зручний інструмент для інтелектуального проектування ПК. Реалізована схема асинхронної взаємодії гарантує стабільну роботу системи в розподіленому мікросервісному середовищі.

3.5 Опис експериментального дослідження: підготовка даних, навчання та валідація моделі

Метою експериментального дослідження є верифікація працездатності запропонованої методики формування напівсинтетичних даних та оцінка точності моделі Random Forest у задачах прогнозування оптимальних конфігурацій. Експеримент проводився у три етапи: підготовка розширеного датасету, гіперпараметричне налаштування моделі та статистична оцінка результатів.

Оскільки початкова база даних містила лише технічні характеристики окремих вузлів, було реалізовано алгоритм синтезу конфігурацій.

1. Генерація: На основі 500+ унікальних записів про CPU та GPU було згенеровано понад 10,000 технічно сумісних комбінацій.

2. Нормалізація: Всі числові параметри (продуктивність, ціна, енергоспоживання) були зведені до безрозмірних величин у діапазоні [0, 1] за процентильним методом. Це дозволило уникнути домінування ознак із великими абсолютними значеннями (наприклад, частоти в МГц над кількістю ядер).

3. Маркування (Labeling): Кожній збірці було присвоєно цільовий рейтинг згідно з математичною моделлю, описаною у підрозділі 2.2. Саме цей рейтинг виступає як цільова змінна (target) для навчанн.

Для навчання обрано ансамбль із 100 дерев рішень. Процес навчання проводився на обчислювальній станції з використанням бібліотеки *Scikit-learn*. Датасет був розділений у пропорції 80/20.

Навчальна вибірка: 8,000 записів (для формування структури дерев).

Тестова вибірка: 2,000 записів (для незалежної перевірки точності).

Під час експерименту було проведено крос-валідацію (K-fold cross-validation, k=5) для підтвердження того, що модель не перенавчилася на специфічних послідовностях синтетичних даних. Результати тестування моделі на різних типах конфігурацій наведені в таблиці 3.5.

Таблиця 3.5 – Показники точності моделі Random Forest на тестовій вибірці

Категорія конфігурацій	**MAE**	**RMSE**	**R2 Score**
Office (Низький бюджет)	0,024	0,031	0,96
Gaming (Середній/Високий)	0,035	0,048	0,94
Professional (Робочі станції)	0,042	0,055	0,91
Загальний показник	0,033	0,044	0,93

Високий показник R^2 = 0.93 свідчить про те, що модель успішно вловила нелінійні залежності між ціною та продуктивністю компонентів. Дещо нижча

точність у категорії "Professional" пояснюється більшою складністю факторів (необхідність врахування специфічних вимог ПЗ до потоків CPU та об'єму VRAM).

Проведене експериментальне дослідження підтвердило гіпотезу про можливість використання напівсинтетичних даних для навчання інтелектуальних систем підбору. Отримані метрики точності (помилка менше 4%) є цілком прийнятними для MVP-версії системи та дозволяють надавати користувачеві релевантні рекомендації, що базуються на об'єктивних математичних розрахунках, а не лише на експертних правилах.

3.6 Аналіз результатів роботи системи на контрольних сценаріях

Було обрано три типових сценарії (Gaming, Professional, Office), результати підбору для яких порівнювалися з рекомендаціями експертів.

Підібрана «Gaming» конфігурація зображена на рисунку 3.5. Конфігурація задовільняє більшість вимог, які типово асоціюються з даним типом ПК.

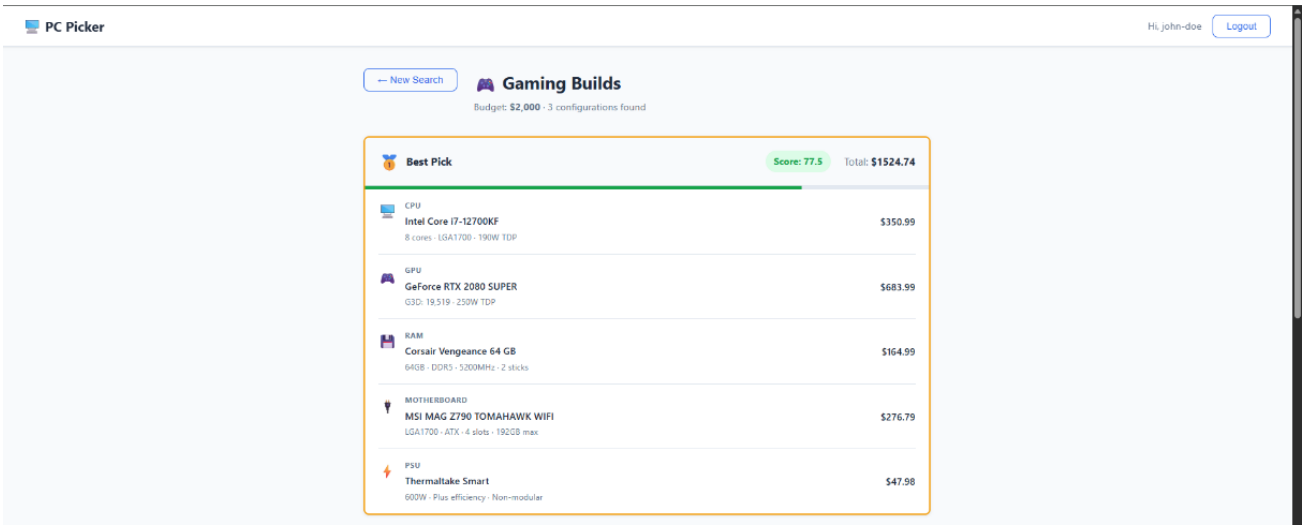

Рисунок 3.5 – Підбір «Gaming» конфігурації

Підібрана «Office» конфігурація зображена на рисунку 3.6. Конфігурація задовільняє всі вимоги, які типово асоціюються з офісними ПК.

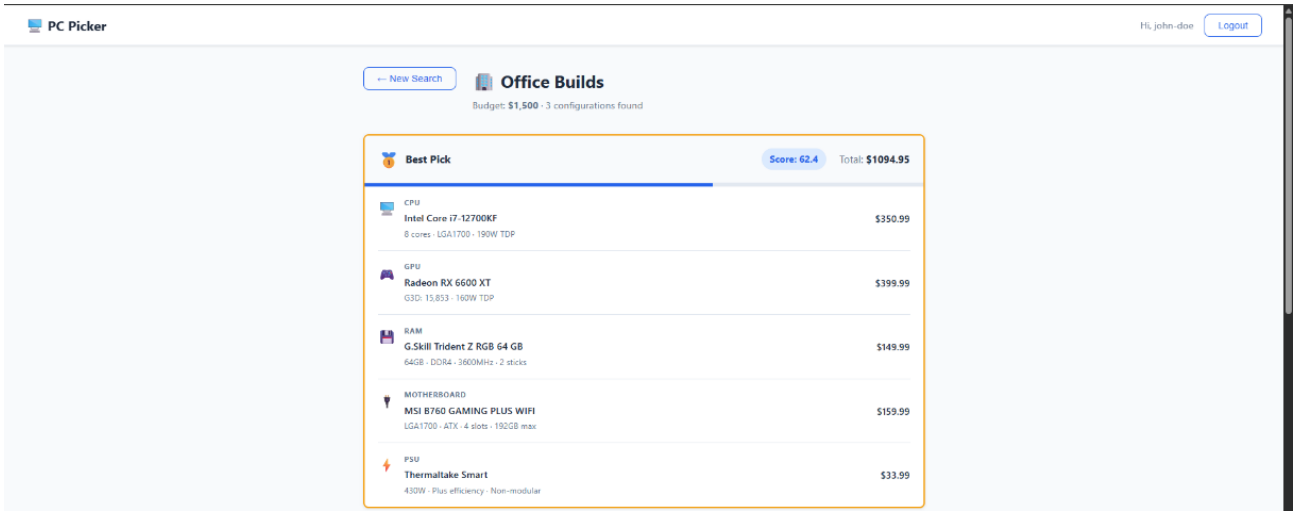

Рисунок 3.6 – Підбір «Office» конфігурації

Підібрана «3Ds Max» конфігурація зображена на рисунку 3.7.

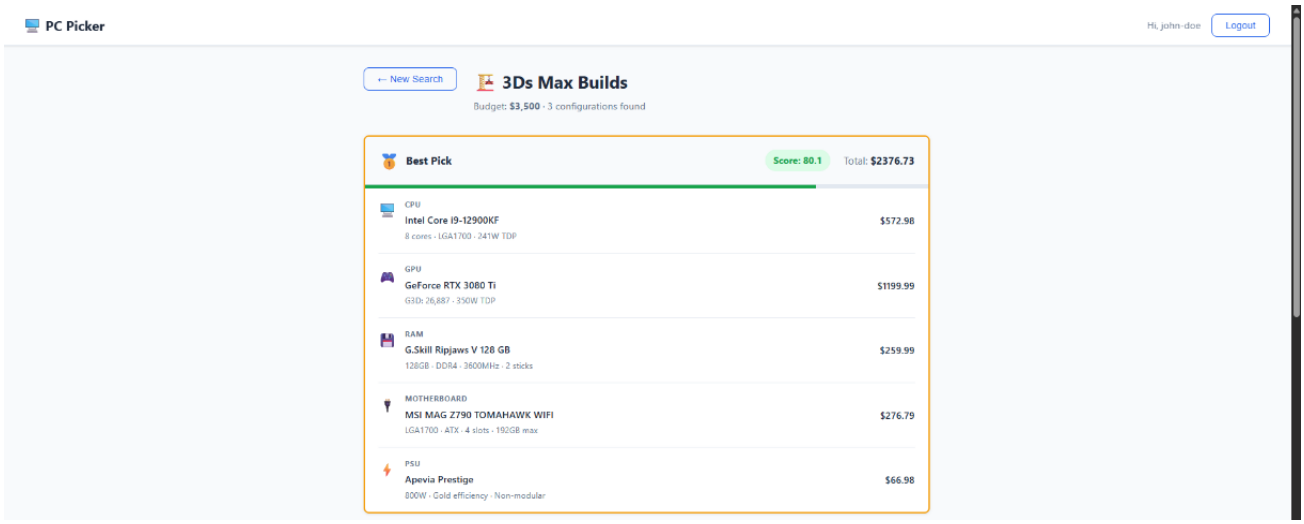

Рисунок 3.7 – Підбір «3Ds Max» конфігурації

Конфігурація задовільняє всі вимоги, для роботи в програмі 3Ds Max.

3.7 Висновок до третього розділу

У третьому розділі виконано практичну реалізацію та експериментальну перевірку інтелектуальної системи підбору комплектуючих ПК. За результатами проведеної роботи зроблено наступні висновки:

1. Обрано та обґрунтовано технологічний стек, що базується на поєднанні Java Spring Boot (бізнес-логіка), Python (машинне навчання) та React JS (інтерфейс). Такий розподіл дозволив ефективно використати переваги кожної платформи в межах мікросервісної архітектури.

2. Реалізовано серверний мікросервіс, який виконує роль оркестратора запитів та забезпечує надійну взаємодію з базою даних PostgreSQL через рівень абстракції JPA.

3. Розроблено інтелектуальний модуль на Python, у якому імплементовано алгоритм Random Forest. Модель оптимізована для роботи з гетерогенними даними та забезпечує швидкість прогнозування менше 50 мс.

4. Створено адаптивний клієнтський інтерфейс, який дозволяє користувачеві формувати складні запити на підбір ПК у зрозумілій формі, абстрагуючись від технічних складнощів архітектури.

5. Проведено експериментальне дослідження, яке продемонструвало високу ефективність методики генерації напівсинтетичних даних. Досягнуто коефіцієнт детермінації $R^2 = 0.93$, що підтверджує високу прогностичну здатність моделі та наукову обґрунтованість запропонованого підходу.

4 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

4.1 Характеристика життєдіяльності людини у системі «людина – машина – середовище існування»

Дослідження безпеки життєдіяльності та умов праці під час проектування, розробки та експлуатації сучасних інтелектуальних інформаційних систем вимагає застосування системного підходу. У межах цього підходу будь-який трудовий або комунікативний процес розглядається як функціонування трикомпонентної структури: «людина – машина – середовище існування» (Л–М–С). Проектований програмний комплекс, що складається з мікросервісів на Java Spring Boot, аналітичного модуля на Python із моделлю Random Forest та клієнтського інтерфейсу на React JS, є яскравим прикладом такої системи, де взаємодія між елементами має складний, нелінійний та інформаційно насичений характер [45]. У роботі компонент «Людина» має подвійну детермінацію.

Інженер-програміст (оператор-розробник): здійснює проектування архітектури, написання програмного коду, навчання моделі машинного навчання та підтримку серверної інфраструктури. Його діяльність характеризується високою інтенсивністю розумової праці, тривалим статичним напруженням та значним навантаженням на зоровий аналізатор.

Кінцевий користувач (суб’єкт взаємодії): особа без спеціальної технічної підготовки, яка взаємодіє з інтерфейсом системи з метою отримання оптимальної конфігурації ПК. Його життєдіяльність у межах системи визначається рівнем когнітивного навантаження, психологічним комфортом та адаптивністю інтерфейсу до його обмежених знань про предметну область.

Компонент «Машина» у дослідженні інтегрує в собі як апаратний, так і програмний рівні. Апаратний рівень включає персональний комп'ютер розробника, серверне обладнання для розгортання мікросервісів та мережеву інфраструктуру. Програмний рівень охоплює розроблену архітектуру системи. Ефективність та безпека цього компонента визначаються надійністю алгоритмів, швидкістю відгуку інтерфейсу (responsiveness) та адекватністю прогнозів моделі

Random Forest. Збій у роботі будь-якого з мікросервісів або видача системою незбалансованої конфігурації (ефект «вузького місця») безпосередньо впливає на психофізіологічний стан людини, викликаючи роздратування, стрес та розумову втому.

Компонент «Середовище існування» визначає сукупність чинників, що оточують людину під час її взаємодії з машиною. Воно поділяється на фізичне та інформаційне середовище.

Фізичне середовище – мікроклімат робочої зони (температура, вологість, швидкість руху повітря), рівень освітленості, шум від роботи систем охолодження ПК та серверів, електромагнітні випромінювання від моніторів.

Інформаційне (когнітивне) середовище – щільність інформаційного потоку, структура розміщення елементів керування на екрані, логіка діалогових вікон.

Наукова новизна роботи, яка полягає в адаптації обмежених даних про комплектуючі у напівсинтетичні набори даних для ШІ-моделі, безпосередньо впливає на безпеку життєдіяльності кінцевого користувача. Завдяки тому, що модель Random Forest бере на себе функцію багатофакторного аналізу сумісності та балансу потужності, користувач звільняється від необхідності вивчати сотні специфікацій. Це мінімізує інформаційне навантаження та запобігає виникненню «парадоксу вибору», який є одним із головних чинників психологічного дискомфорту при роботі зі статичними конфігураторами.

Ергономічний аналіз системи Л–М–С у межах роботи показує, що мікросервісна архітектура (відноситься до компонента «Машина») має позитивний непрямий вплив на умови праці розробника. Завдяки ізоляції сервісів (Spring Boot та Python ML модуль функціонують незалежно), знижується рівень стресу розробника під час відладки та розгортання системи. Збій в одному модулі не призводить до краху всього програмного комплексу, що зменшує психоемоційне напруження, пов'язане з дефіцитом часу на пошук помилок (так званий «фактор терміновості»).

Таким чином, забезпечення високого рівня безпеки життєдіяльності у системі «людина – машина – середовище існування» для теми дослідження

досягається через раціональне проектування програмної архітектури та інтерфейсу користувача. Оптимізація взаємодії в тріаді Л–М–С дозволяє не лише підвищити технічну ефективність підбору комплектуючих ПК, а й гарантує збереження високої працездатності, мінімізацію професійних ризиків для розробника та забезпечення максимального ментального комфорту для кінцевого споживача.

4.2 Попередження та ліквідація наслідків техногенних надзвичайних ситуацій, викликаних пожежами та збоями в інфраструктурі серверних приміщень

Розгортання та експлуатація проектованого програмного комплексу, що функціонує на базі мікросервісної архітектури (Java Spring Boot та Python) з предиктивною моделлю машинного навчання та реляційною СУБД PostgreSQL, вимагає надійної серверної інфраструктури. З огляду на це, найбільш критичними техногенними надзвичайними ситуаціями (НС) для даної системи є пожежі та аварії на об'єктах зв'язку та телекомунікацій, що виникають внаслідок критичних збоїв електромереж або систем терморегуляції серверних приміщень (дата-центрів).

Основними чинниками виникнення пожежної небезпеки в інфраструктурі системи є:

- Електричне перевантаження: тривале обчислювальне навантаження на сервери під час навчання або інференсу моделі Random Forest призводить до підвищеного енергоспоживання та ризику короткого замикання.
- Відмова систем охолодження: збій у роботі прецизійних кондиціонерів викликає стрімке підвищення температури (тепловий розгін) процесорів (CPU) та графічних прискорювачів (GPU), що може спровокувати займання ізоляції провідників.

З метою запобігання (попередження) зазначених НС на об'єкті розгортання передбачено комплекс інженерно-технічних заходів відповідно до вимог НАПБ Б.03.005-2002 та ДБН В.2.5-56:2014 [46]. По-перше, серверні шафи оснащуються

системами раннього виявлення пожежі (аспіраційними димовими сповіщувачами). По-друге, ліквідація займань електронного обладнання під напругою унеможливлює використання води чи піни. Тому в приміщеннях інтегрується автоматична система газового пожежогасіння (АСГП) із застосуванням озонобезпечних газових вогнегасних речовин (наприклад, Хладон), які гасять полум'я шляхом зв'язування кисню та охолодження без пошкодження мікросхем і носіїв даних.

Оскільки мікросервіси Spring Boot та Python функціонують незалежно, техногенна стійкість системи на програмному рівні забезпечується через географічне резервування (кластеризацію).

У випадку виникнення НС ліквідація її наслідків здійснюється згідно з операційним планом евакуації та аварійного реагування. Алгоритм дій передбачає автоматичне знеструмлення ураженого сектора, запуск АСГП, активацію системи димовидалення та передачу сигналу тривоги на пульт ДСНС України. Одночасно з цим мережеві балансувальники навантаження (класична схема автоматизації відмовостійкості) миттєво перенаправляють трафік користувачів React-додатка на резервний серверний кластер. Такий підхід дозволяє локалізувати матеріальні збитки від техногенної аварії в межах одного приміщення, повністю зберігши працездатність інтелектуального сервісу підбору комплектуючих ПК та безпеку обслуговуючого персоналу.

4.3 Ергономічні вимоги до організації робочого місця інженера-програміста при розробці інтелектуальних систем

Процес проектування мікросервісної архітектури на Java Spring Boot, написання аналітичних скриптів на Python та верстки користувацького інтерфейсу на React JS супроводжується значним психофізіологічним навантаженням на організм розробника. Тривала робота за комп'ютером в умовах дефіциту часу та високої концентрації уваги вимагає суворого дотримання вимог ергономіки та виробничої санітарії, що регламентуються

НПАОП 0.00-7.15-18 «Вимоги щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями» [47].

Ергономічна оптимізація робочого простору інженера-програміста базується на просторовому плануванні та забезпеченні нормативних параметрів виробничого середовища:

- Організація робочої зони та меблів. Взаємне розташування елементів робочого місця має відповідати антропометричним характеристикам розробника. Висота робочої поверхні столу повинна встановлюватися в межах 680–800 мм. Робоче крісло має бути підйомно-поворотним, оснащеним механізмами регулювання висоти сидіння, кута нахилу спинки та висоти підлокітників для зниження статичного навантаження на хребет і м'язи плечового поясу. Відеодисплейний термінал (монітор) розміщують на відстані 600–700 мм від очей оператора, а верхня кромка екрана повинна знаходитися на рівні очей або на 10–20° нижче, що мінімізує втому м’язів шиї та зорового аналізатора [48].
- Параметри виробничого освітлення. Для запобігання зоровій втомі (астенопії) рівень освітленості на робочому столі в зоні розміщення документів та клавіатури повинен становити не менше 300–500 лк. Штучне освітлення реалізується за комбінованою системою, де загальне рівномірне освітлення доповнюється місцевим (настільними лампами). Слід уникати прямого та відбитого блиску на екрані монітора, для чого робоче місце розробника розміщують боком до віконних прорізів, а на вікна встановлюють регульовані жалюзі.
- Мікрокліматичні умови. Оскільки електронна апаратура виділяє додаткове тепло, приміщення розробки обладнується системами кондиціонування та припливно-витяжної вентиляції. Оптимальні параметри мікроклімату для категорії робіт Іа (легка розумова праця) у теплий період року становлять: температура повітря — 22–24°С, відносна вологість — 40–60%, швидкість руху повітря — не більше 0,1 м/с [49].

Важливим елементом охорони праці є раціональний режим праці та відпочинку. При виконанні високонапруженої творчої роботи розробника

(категорія В за характером навантаження) регламентовані перерви тривалістю 10–15 хвилин слід впроваджувати через кожні дві години роботи [50].

4.4 Висновок до четвертого розділу

У четвертому розділі кваліфікаційної роботи проведено детальний аналіз питань охорони праці, виробничої санітарії та безпеки в надзвичайних ситуаціях стосовно умов життєдіяльності розробника та кінцевого користувача проектованої системи:

1. На основі дослідження трикомпонентної системи «людина – машина – середовище існування» встановлено, що інтеграція предиктивних методів машинного навчання (Random Forest) суттєво знижує когнітивне навантаження на кінцевого користувача.

2. Визначено потенційні загрози та розроблено заходи щодо попередження та ліквідації наслідків техногенних надзвичайних ситуацій (зокрема пожеж) в інфраструктурі серверних приміщень. Обґрунтовано використання автоматичних систем газового пожежогасіння та програмного реплікування баз даних PostgreSQL для забезпечення високої відмовостійкості системи.

3. Сформульовано комплекс ергономічних та санітарно-гігієнічних вимог до організації робочого місця інженера-програміста відповідно до чинних нормативних актів України. Визначено оптимальні параметри мікроклімату, освітленості та просторового розміщення обладнання, що дозволяє мінімізувати вплив небезпечних і шкідливих виробничих факторів та зберегти високу працездатність у процесі розробки програмного продукту.

ВИСНОВКИ

В першому розділі кваліфікаційної роботи освітнього рівня «Магістр»:

- Подано детальний аналіз сучасного стану ринку комп’ютерного апаратного забезпечення у період 2024–2026 років із виокремленням домінуючого сегменту AI-орієнтованих систем.
- Розглянуто існуючі програмні рішення та онлайн-сервіси конфігурування ПК, виявлено їхні обмеження щодо інтелектуальної підтримки непідготовлених користувачів та «експертного бар'єру».
- Висвітлено теоретичні аспекти застосування методів машинного навчання у задачах класифікації та регресії для систем підтримки прийняття рішень.
- Проаналізовано критичні недоліки традиційних детермінованих алгоритмів підбору комплектуючих, що базуються на жорстких логічних фільтрах.
- Досліджено порівняльні характеристики моделей k-NN, штучних нейронних мереж та градієнтного бустингу в контексті роботи з гетерогенними табличними даними.
- Обґрунтовано вибір алгоритму Random Forest як найбільш стійкого до перенавчання в умовах використання напівсинтетичних вибірок та наявності категоріальних ознак.
- Сформовано дерево проблем та дерево цілей проекту на основі системного аналізу об'єкта автоматизації.

В другому розділі кваліфікаційної роботи:

- Описано концептуальну мікросервісну архітектуру системи, що базується на принципі розподілу функціональних обов'язків між Java Spring Boot, Python та React JS сервісами.
- Досліджено математичну модель оцінки сумісності та продуктивності, яка враховує вагові коефіцієнти для різних сценаріїв використання та мінімізує структурний дисбаланс («вузькі місця»).

- Подано порівняльний опис характеристик початкової та розширеної напівсинтетичної вибірок, а також спроектовано реляційну схему бази даних PostgreSQL, оптимовану для ML-аналітики.

В третьому розділі кваліфікаційної роботи:

- Розроблено програмний прототип серверної частини, що виконує роль API-оркестратора, та інтелектуальний ML-модуль, здатний виконувати прогноз оптимальних конфігурацій у режимі реального часу.
- Запропоновано адаптивний клієнтський інтерфейс, що абстрагує технічну складність процесу підбору для кінцевого користувача.
- Спроєктовано та імплементовано систему RESTful-взаємодії між мікросервісами, що забезпечує стабільну передачу даних у форматі JSON.
- Протестовано точність роботи навченої моделі Random Forest, у результаті чого підтверджено її високу ефективність із коефіцієнтом детермінації $R^2 = 0,93$.

ПЕРЕЛІК ДЖЕРЕЛ

1. Про схвалення Концепції розвитку цифрової економіки та суспільства України на 2018-2020 роки та затвердження плану заходів щодо її реалізації : Розпорядження Кабінету Міністрів України від 17 січ. 2018 р. № 67-р.

2. Закон України «Про Основні засади розвитку інформаційного суспільства в Україні на 2007-2015 роки» : від 09 січ. 2007 р. № 537-V.

3. ДСТУ ISO/IEC IEEE 12207:2020 (ISO/IEC/IEEE 12207:2017, IDT). Інженерія систем і програмного забезпечення. Процеси життєвого циклу програмного забезпечення. Київ : ДП «УкрНДНЦ», 2021. 118 с.

4. ДСТУ ISO/IEC 25010:2016 (ISO/IEC 25010:2011, IDT). Системна та програмна інженерія. Вимоги до якості систем і програмного забезпечення та її оцінювання (SQuaRE). Моделі якості систем та програмних продуктів. Київ : ДП «УкрНДНЦ», 2017. 46 с.

5. Субботін С. О. Подання й обробка знань у системах штучного інтелекту та підтримки прийняття рішень : навч. посіб. Запоріжжя : ЗНТУ, 2018. 342 с.

6. Ткаченко В. О. Автоматизовані системи прийняття рішень: архітектура та методи розробки. Комп'ютерні науки та інженерія. 2022. Т. 14, № 2. С. 45–52.

7. Hennessy J. L., Patterson D. A. Computer Architecture: A Quantitative Approach. 6th ed. Cambridge : Morgan Kaufmann, 2019. 936 p.

8. Tanenbaum A. S., Austin T. Structured Computer Organization. 6th ed. Boston : Pearson, 2016. 800 p.

9. Сміт Дж. Проблеми апаратної сумісності та балансування обчислювальних вузлів сучасних ЕОМ. Технічна кібернетика. 2023. № 4. С. 12–19.

10. Коваль Р. М. Системний аналіз ринку кастомних персональних комп'ютерів та засобів автоматизації їх збирання. Вісник технічного університету. 2024. Т. 28, № 1. С. 88–95.

11. Огляд та аналіз платформ конфігурування комп'ютерних систем споживчого рівня / О. П. Іванов та ін. Сучасні інформаційні технології. 2025. № 3. С. 104–112.

12. Бакалаврська кваліфікаційна робота на тему «Інформаційна система підбору комплектуючих ПК». Тернопіль : ТНТУ, 2024. 64 с.

13. Russell S., Norvig P. Artificial Intelligence: A Modern Approach. 4th ed. Hoboken : Pearson, 2020. 1168 p.

14. Глибоке навчання та інтелектуальні агенти в задачах багатофакторної фільтрації даних / В. С. Петров та ін. Штучний інтелект. 2024. № 2. С. 15–23.

15. Bishop C. M., Bishop H. A. Pattern Recognition and Machine Learning. 2nd ed. Cham : Springer, 2023. 756 p.

16. Sommerville I. Software Engineering. 10th ed. Boston : Pearson, 2016. 811 p.

17. ДСТУ ISO/IEC/IEEE 29148:2022 (ISO/IEC/IEEE 29148:2018, IDT). Інженерія систем і програмного забезпечення. Процеси життєвого циклу. Інженерія вимог. Київ : ДП «УкрНДНЦ», 2023. 92 с.

18. Волошин О. Ф., Мащенко С. О. Моделі та методи прийняття рішень : підручник. 2-ге вид., перероб. та допов. Київ : Видавничо-поліграфічний центр "Київський університет", 2015. 384 с.

19. Steuer R. E. Multiple Criteria Optimization: Theory, Computation, and Application. Malabar : Krieger Publishing, 2017. 546 p.

20. Антонов В. М. Математичне моделювання складних нелінійних систем взаємодії компонентів. Журнал обчислювальної математики. 2023. № 1. С. 67–74.

21. Breiman L. Random Forests. Machine Learning. 2001. Vol. 45, No. 1. P. 5–32.

22. Biau G., Scornet E. A random forest guided tour. TEST. 2016. Vol. 25, No. 2. P. 197–227.

23. Hastie T., Tibshirani R., Friedman J. The Elements of Statistical Learning: Data Mining, Inference, and Prediction. 2nd ed. New York : Springer, 2017. 745 p.

24. Ignatenko V., Surkov A., Koltcov S. Random forests with parametric entropy-based information gains for classification and regression problems. PeerJ Computer Science. 2024. Vol. 10. P. e1775.

25. Kuhn M., Johnson K. Applied Predictive Modeling. New York : Springer, 2018. 600 p.

26. Методологія обробки розріджених даних та синтезу напівсинтетичних вибірок для навчання регресійних моделей / А. М. Шевченко та ін. Наукові записки НаУКМА. Комп'ютерні науки. 2025. Т. 8. С. 41–49.

27. Goodfellow I., Bengio Y., Courville A. Deep Learning. Cambridge : MIT Press, 2016. 800 p.

28. Allen Akselrud C. Random Forest Regression Models in Ecology: Accounting for Messy Biological Data and Producing Predictions with Uncertainty. SSRN Electronic Journal. 2024. DOI: 10.2139/ssrn.4865972.

29. Newman S. Building Microservices: Designing Fine-Grained Systems. 2nd ed. Sebastopol : O'Reilly Media, 2021. 614 p.

30. Fowler M. Microservices: a definition of this new architectural term. Martinfowler.com. 2014. URL: https://martinfowler.com/articles/microservices.html.

31. Richardson C. Microservices Patterns: With examples in Java. Shelter Island : Manning Publications, 2018. 520 p.

32. Walls C. Spring in Action. 6th ed. Shelter Island : Manning Publications, 2022. 525 p.

33. Yang W., Xing Y., Lyu Y. A Feature Dataset of Microservices-based Systems. arXiv preprint. 2024. arXiv:2404.01789.

34. Deshpande R. A. Application Of Spring Boot Microservice Architecture for Scaling Banking Applications. The American Journal of Engineering and Technology. 2025. Vol. 7, No. 9. P. 152–158.

35. Date C. J. An Introduction to Database Systems. 8th ed. Boston : Pearson, 2018. 1024 p.

36. Schönig H.-J. Mastering PostgreSQL 15: Advanced techniques to build and manage scalable, reliable, and fault-tolerant database applications. 5th ed. Birmingham : Packt Publishing, 2023. 580 p.

37. Системний аналіз та оптимізація реляційних баз даних великого об'єму в PostgreSQL / Д. О. Чернов та ін. Комп'ютерні системи та мережі. 2024. № 12. С. 34–42.

38. McKinney W. Python for Data Analysis: Data Wrangling with Pandas, NumPy, and Jupyter. 3rd ed. Sebastopol : O'Reilly Media, 2022. 578 p.

39. Pedregosa F. et al. Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research. 2011. Vol. 12. P. 2825–2830.

40. Grinberg M. Flask Web Development: Developing Web Applications with Python. 2nd ed. Sebastopol : O'Reilly Media, 2018. 314 p.

41. Banks A., Porcello E. Learning React: Modern Patterns for Developing Real-World Applications. 2nd ed. Sebastopol : O'Reilly Media, 2020. 370 p.

42. Чорний М. В. Розробка чуйних інтерфейсів Single Page Application з використанням компонентного підходу. Сучасні веб-технології. 2025. № 1. С. 77–83.

43. Chicco D., Warrens M. J., Jurman G. The coefficient of determination R-squared is more informative than SMAPE, MAE, MAPE, MSE and RMSE in regression analysis evaluation. PeerJ Computer Science. 2021. Vol. 7. P. e623.

44. Критерії оцінки адекватності прогнозних моделей штучного інтелекту в інженерних задачах / І. В. Дмитрієв та ін. Математичне моделювання. 2024. № 4. С. 19–27.

45. Безпека життєдіяльності : підручник / В. В. Березуцький та ін. ; за ред. В. В. Березуцького. Харків : Факт, 2017. 448 с.

46. ДБН В.2.5-56:2014. Системи протипожежного захисту (із Змінами № 1, № 2). Київ : Мінрегіон України, 2018. 134 с. / НАПБ Б.03.005-2002 Норми розрахунку необхідної кількості первинних засобів пожежогасіння.

47. НПАОП 0.00-7.15-18. Вимоги щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями : Затверджено Наказом Міністерства соціальної політики України від 14.02.2018 № 207.

48. ДБН В.2.5-28:2018. Природне і штучне освітлення. Київ : Мінрегіон України, 2018. 144 с.

49. ДБН В.2.2-40:2018. Інклюзивність будівель і споруд. Основні положення / ДСТУ EN ISO 9241-5:2022 Ергономічні вимоги до роботи з відеодисплейними терміналами. Частина 5. Вимоги до розташування робочого місця та постави. Київ : ДП «УкрНДНЦ», 2023.

50. Санітарні норми мікроклімату виробничих приміщень ДСН3.3.6.042-99 / ДСТУ EN ISO 7730:2011 Ергономіка теплового середовища. Київ : Держспоживстандарт України, 2012.

ДОДАТКИ

Додаток А

Публікації та сертифікати з наукових конференцій

Збірник з всеукраїнської студентської наукової конференції «ЕКСПЕРИМЕНТАЛЬНІ ТА ТЕОРЕТИЧНІ ДОСЛІДЖЕННЯ В КОНТЕКСТІ СУЧАСНОЇ НАУКИ», 20 березня 2026 р., м. Вінниця: https://go-vropejska.esclick.me/1r93g7y38n5e96ytGk

СЕКЦІЯ 12.
КОМП'ЮТЕРНА ТА ПРОГРАМНА ІНЖЕНЕРІЯ

СЕКЦІЯ 13.
ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ ТА СИСТЕМИ

СЕКЦІЯ 14.
ФІЛОЛОГІЯ ТА ЖУРНАЛІСТИКА

СЕКЦІЯ 13.
ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ ТА СИСТЕМИ

Тизунь Андрій Сергійович, здобувач другого магістерського рівня вищої освіти
Факультет комп'ютерно-інформаційних систем і програмної інженерії
Тернопільський національний технічний університет імені Івана Пулюя, Україна

ІНТЕЛЕКТУАЛЬНА СИСТЕМА АВТОМАТИЗОВАНОГО ПІДБОРУ КОМПЛЕКТУЮЧИХ ПК ДЛЯ НЕПРОФЕСІЙНИХ КОРИСТУВАЧІВ НА ОСНОВІ ТЕХНОЛОГІЙ МАШИННОГО НАВЧАННЯ

Цифровізація сучасного суспільства та зростання вимог до обчислювальних потужностей роблять питання придбання персонального комп'ютера (ПК) критичним як для професійної діяльності, так і для побутового використання. Проте на сьогоднішній день споживач стикається з гострою дилемою між придбанням готового рішення (брендової збірки) та самостійним підбором комплектуючих [1]. Аналіз ринку готових ПК демонструє низку суттєвих недоліків, зумовлених маркетинговими стратегіями виробників. По-перше, конфігурації часто базуються на принципі «незбалансованості», де потужний центральний процесор поєднується з критично слабкими або низькоякісними другорядними компонентами. Зокрема, для мінімізації собівартості часто використовуються блоки живлення (БЖ) з низьким ККД та відсутністю систем захисту, що ставить під загрозу стабільність роботи всієї системи та її довговічність. По-друге, використання пропрієтарних материнських плат та специфічних форм-факторів корпусів унеможливлює подальшу модернізацію (апгрейд), змушуючи користувача купувати повністю новий пристрій через кілька років експлуатації. До того ж, націнка за бренд та збірку може складати від 15% до 40% вартості апаратного забезпечення, що не є економічно виправданим.

З іншого боку, ринок окремих комплектуючих характеризується експоненціальним зростанням асортименту та складності. Велика кількість стандартів сокетів, чипсетів, типів оперативної пам'яті та вимог до тепловиділення (TDP) створює високий поріг входження для непрофесійного користувача [2]. Ризик придбання несумісних деталей або вибору компонентів, що створюють «ефект вузького місця» (bottleneck), залишається надзвичайно високим навіть за наявності базових знань.

Таким чином, виникає потреба у розробці інтелектуальної системи, що виступає експертним посередником, здатним на основі нетехнічних вимог користувача (бюджет, цільова сфера застосування) синтезувати технічно досконалу та надійну конфігурацію.

Головною концепцією системи є зміщення фокусу з жорстких технічних параметрів на реальні потреби людини через реалізацію підходу «від завдання до рішення». Це забезпечує інтуїтивну абстракцію вимог, за якої користувач взаємодіє із

Тизунь Андрій Сергійович, здобувач другого магістерського рівня вищої освіти
Факультет комп'ютерно-інформаційних систем і програмної інженерії
Тернопільський національний технічний університет імені Івана Пулюя, Україна

МЕТОДИКА ФОРМУВАННЯ СИНТЕТИЧНИХ НАВЧАЛЬНИХ ВИБІРОК ДЛЯ МОДЕЛЕЙ МАШИННОГО НАВЧАННЯ В ЗАДАЧАХ КОНФІГУРУВАННЯ ПЕРСОНАЛЬНОГО КОМП'ЮТЕРА

Розробка рекомендаційних систем у сфері апаратного забезпечення стикається з фундаментальною перешкодою — відсутністю структурованих великих даних (Big Data), що містять експертні оцінки оптимальності конфігурацій. На відміну від стандартних задач класифікації зображень чи тексту, де існують загальнодоступні датасети, оцінка збалансованості компонентів ПК вимагає глибокої технічної експертизи. Специфіка задачі полягає у необхідності врахування не лише фізичної сумісності, а й синергії характеристик для конкретних сценаріїв використання. У таких умовах стає актуальним перехід від збору реальних даних до методології генерації синтетичних навчальних вибірок, які б імітували експертну логіку.

Первинним джерелом інформації для формування системи виступають дискретні технічні характеристики компонентів та результати їх синтетичних випробувань (бенчмарків). Доступний масив даних включає: показники одно- та багатопоточної продуктивності центральних процесорів, обчислювальну потужність графічних адаптерів, швидкісні характеристики накопичувачів, показники тепловиділення (TDP) та ринкову вартість. Проте ці дані є лише «сировиною», оскільки вони не містять цільової мітки, яка б вказувала на якість конкретної комбінації цих параметрів у межах цілісної системи.

Для створення повноцінної навчальної вибірки запропоновано триетапний алгоритм адаптації. На першому етапі здійснюється генерація базових сумісних рішень (у дослідженні використано 1000 унікальних комбінацій), де перевірка сумісності базується на жорстких логічних фільтрах (відповідність сокетів, типів пам'яті, енергоспоживання).

Другий етап передбачає контекстне розширення даних шляхом присвоєння кожній базовій конфігурації специфічних ролей (наприклад: ігрова, офісна, професійна). Це дозволяє системі навчатися на одному й тому самому наборі компонентів, але з різними векторами цільових значень.

На третьому етапі реалізується функція багатокритеріального оцінювання (Scoring Function). Вона трансформує технічні характеристики у нормалізований рейтинг (від 0 до 100), враховуючи вагові коефіцієнти для кожної ролі та корелюючи продуктивність із ціною. В результаті адаптації формується датасет (в даному дослідженні, обсягом 4000 записів), де кожна конфігурація має чіткий математично обґрунтований рейтинг оптимальності.

Використання сформованої синтетичної вибірки дозволило провести ефективне навчання моделі на основі алгоритму Random Forest. На відміну від лінійних моделей, ансамблевий підхід продемонстрував здатність виявляти нелінійні залежності між

апаратними характеристиками, що критично важливо для виявлення «ефектів вузького місця» (bottleneck). Модель, навчена на адаптованих даних, здатна здійснювати предиктивну оцінку нових конфігурацій у режимі реального часу, виконуючи роль інтелектуального фільтра. Результати тестування прототипу підтвердили, що порогова фільтрація (score > 90) дозволяє автоматизовано відбирати рішення, які за рівнем збалансованості відповідають рекомендаціям профільних технічних експертів.

Висновок

У ході дослідження обґрунтовано методологію створення синтетичних навчальних вибірок як ефективний засіб подолання дефіциту експертних даних. Запропонований підхід до алгоритмічної генерації сумісних конфігурацій з наступним контекстним розширенням та багатокритеріальним оцінюванням дозволяє готувати якісні датасети для навчання моделей машинного навчання. Експериментальна перевірка на базі алгоритму Random Forest підтвердила високу точність предиктивного аналізу в задачах автоматизованого проектування ПК. Такий підхід є універсальним і може бути адаптований для конфігурування інших складних технічних систем, де прямий збір експертних даних є економічно недоцільним або технічно складним.

Список використаних джерел:

1. Goodfellow, I., Bengio, Y., & Courville, A. Deep Learning. MIT Press, 2016. URL: https://www.perlego.com/paid/book/5170018/deep-learning-pdf
2. Triantaphyllou, E. Multi-criteria Decision Making Methods: A Comparative Study. Kluwer Academic Publishers, 2000. URL: https://www.researchgate.net/publication/209805531_Multi-Criteria_Decision_Making_Methods_A_Comparative_Study
3. Breiman, L. Random Forests. Machine Learning, 45, 5–32, 2001. URL: https://link.springer.com/article/10.1023/A:1010933404324
4. Ricci, F., Rokach, L., & Shapira, B. Recommender Systems Handbook. Springer, 2022. URL: https://link.springer.com/book/10.1007/978-1-0716-2197-4
5. Zheng, A., & Casari, A. Feature Engineering for Machine Learning. O'Reilly Media, 2018. URL: https://www.oreilly.com/library/view/feature-engineering-for/9781491953235

системою через зрозумілі функціональні категорії, такі як «відеомонтаж» чи «геймінг», а інтелектуальне ядро самостійно інтерпретує ці запити у необхідну обчислювальну потужність підсистем [3]. Важливою перевагою такого підходу є об'єктивна багатокритеріальна оцінка, де прогнозний рейтинг моделі гарантує збалансованість конфігурації, відсутність «вузьких місць» та потенційних технічних конфліктів між компонентами.

Крім того, система гарантує високу надійність та тривалий життєвий цикл обчислювальної системи, оскільки алгоритм автоматично віддає пріоритет компонентам із надлишковою потужністю блоку живлення та наявністю вільних слотів для майбутньої модернізації. Економічна ефективність запропонованих рішень досягається шляхом аналізу ринку в режимі реального часу та вибору варіантів з найкращим показником продуктивності на одиницю вартості, що дозволяє уникнути надлишкових маркетингових націнок за бренд. Завдяки динамічній адаптації до актуальних ринкових пропозицій, користувач отримує сучасне та збалансоване рішення, що повністю відповідає його індивідуальним запитам без необхідності глибокого вивчення технічної документації.

Висновок

Розроблено архітектуру та програмний прототип інтелектуальної системи автоматизованого підбору комплектуючих ПК, орієнтованої на користувачів без спеціалізованого технічного досвіду. Запропоноване рішення на базі мікросервісного підходу [4] та моделі Random Forest [5], навченої на розширеному синтетичному датасеті, підтвердило свою ефективність у задачах багатокритеріальної оцінки апаратних рішень та розгорнута на хмарній платформі AWS [6]. Розроблена система успішно вирішує проблему «інформаційного бар'єра», транслюючи абстрактні вимоги користувача у технічно збалансовані специфікації.

Практична цінність прототипу полягає у можливості отримання конфігурацій, що перевершують готові ринкові аналоги за показниками надійності та цінової ефективності. Подальші дослідження будуть спрямовані на впровадження циклів зворотного зв'язку для перенавчання моделі на основі реального досвіду користувачів.

Список використаних джерел:

1. Gartner Report. Forecast: PCs, Tablets and Mobile Phones, Worldwide, 2023-2029, 3Q25 Update. URL: https://www.gartner.com/en/documents/6979966
2. Hennessy, J. L., & Patterson, D. A. Computer Architecture: A Quantitative Approach. Elsevier, 2017. URL: https://shop.elsevier.com/books/computer-architecture/hennessy/978-0-443-15406-5
3. Ricci, F., Rokach, L. Recommender Systems Handbook. Springer, 2022. URL: https://www.cs.ubbcluj.ro/~gabis/DocDiplome/SistemeDeRecomandare/Recommender_systems_handbook.pdf
4. Richardson, C. Microservices Patterns: With examples in Java. Manning Publications, 2018. URL: https://www.manning.com/books/microservices-patterns
5. Breiman, L. Random Forests. Machine Learning, 45, 5–32, 2001. URL: https://link.springer.com/article/10.1023/A:1010933404324
6. Varia, J., & Mathew, S. Architecting for the Cloud: AWS Best Practices. Amazon Web Services, 2014. URL: https://engineering.purdue.edu/ee695b/public-web/handouts/References/AWS_Cloud_Best_Practices.pdf