Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)
Кафедра Інформатики
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

РОЗРОБЛЕННЯ ВЕБЗАСТОСУНКУ З АЛГОРИТМІЧНИМ ПІДБОРОМ ВЕРСТАТІВ З ЧИСЛОВИМ ПРОГРАМНИМ КЕРУВАННЯМ НА ОСНОВІ ВИМОГ КОРИСТУВАЧА
(тема)

Виконав:
здобувач 4 року навчання,
групи ІТІНФ-22-3
Костенко Н.Р.
(прізвище, ініціали)

Спеціальність 122 Комп'ютерні науки
(код і повна назва спеціальності)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

Керівник доц. Руденко Д. О.
(посада, прізвище, ініціали)

Допускається до захисту

Завідувач кафедри інформатики _____ Кобилін О. А.
(підпис) (прізвище, ініціали)
2026 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту

Кафедра Інформатики

Рівень вищої освіти перший (бакалаврський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 2026 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Костенку Нікіті Романовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розроблення вебзастосунку з алгоритмічним підбором верстатів з числовим програмним керуванням на основі вимог користувача

затверджена наказом університету від 26 травня 2026 року № 435Ст
2. Термін подання здобувачем роботи до екзаменаційної комісії 22 червня 2026 р.
3. Вихідні дані до роботи Науково-методична та науково-технічна література з веброзробки та архітектури інформаційних систем, матеріали науково-практичних конференцій, дані інтернет-мережі, технічні специфікації та каталоги промислового обладнання з ЧПК, офіційна документація мови Python, вебфреймворку Flask та бібліотеки об'єктно-реляційного відображення SQLAlchemy.

4. Перелік питань, що потрібно опрацювати в роботі
1. Аналіз сучасного стану автоматизації підбору верстатів з ЧПК, огляд існуючих онлайн-платформ та формування функціональних і нефункциональних вимог до програмного забезпечення.
2. Математичне та алгоритмічне забезпечення багатокритеріального вибору обладнання, проєктування структури реляційної бази даних та об'єктно-орієнтоване моделювання системи з використанням мови UML.
3. Програмна реалізація користувацького інтерфейсу, бізнес-логіки конфігуратора та модуля каталогу верстатів, забезпечення інформаційної безпеки системи та комплексне функціональне тестування вебзастосунку.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) Актуальність проблеми та постановка задачі, порівняльний аналіз платформ-аналогів, схема трирівневої архітектури взаємодії інструментальних засобів, UML-діаграми (прецедентів, активності та послідовності), специфікація полів бази даних, екранні форми користувацького інтерфейсу форми конфігуратора та карток результатів підбору обладнання, таблиці функціонального тестування системи.

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Строк / терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	13.04.2026	
2	Аналіз завдання, підбір літератури	14.04.26-18.04.26	
3	Аналіз літератури з проблеми, що вирішується	19.04.26-21.04.26	
4	Аналіз існуючих методів розпізнавання	22.04.26-23.04.26	
5	Формування кроків вирішення проблеми	24.04.26-07.05.26	
6	Програмна реалізація	28.04.26-25.05.26	
7	Аналіз отриманих результатів	26.05.26-04.06.26	
8	Оформлення пояснювальної записки	05.06.26-09.06.26	
9	Перевірка на нормоконтроль	10.06.26-19.06.26	
10	Перевірка на плагіат	11.06.26-30.06.26	
11	Рецензування	20.06.26-30.06.26	
12	Підготовка презентації та доповіді	20.06.26-30.06.26	
13	Занесення роботи в електронний архів	30.06.26-09.07.26	
14	Попередній захист кваліфікаційної роботи	30.06.26-09.07.26	

Дата видачі завдання 13 квітня 2026 р.

Здобувач _____
(підпис)

Керівник роботи _____ доц. Руденко Д. О.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ/ABSTRACT

Пояснювальна записка до кваліфікаційної роботи: 74 с., 6 табл., 23 рис., 31 джерело.

ВЕБЗАСТОСУНОК, ВЕРСТАТ З ЧПК, АЛГОРИТМ ФІЛЬТРАЦІЇ, БАГАТОКРИТЕРІАЛЬНИЙ АНАЛІЗ, PYTHON, FLASK, SQLITE.

Об’єкт роботи: процеси алгоритмічного підбору та фільтрації промислового обладнання з ЧПК на основі вимог користувача.

Мета роботи: розроблення інтерактивного вебзастосунку для автоматизованого підбору оптимальних моделей верстатів із ЧПК з урахуванням просторових, економічних та технологічних обмежень.

Методи дослідження: багатокритеріальна оптимізація, об'єктно-орієнтоване проєктування, методи веброзробки (Flask, SQLAlchemy).

Практична цінність: автоматизація інженерних розрахунків при виборі верстатів для мінімізації фінансових ризиків.

Висновки: розроблено вебзастосунок, що розраховує технологічну складність обробки, виконує просторову фільтрацію та генерує вибірку оптимального обладнання.

Область застосування**:** підприємства обробної промисловості, відділи продажу обладнання, інжинірингові компанії, B2B-платформи.

WEB APPLICATION, CNC MACHINE, FILTRATION ALGORITHM, MULTI-CRITERIA ANALYSIS, PYTHON, FLASK, SQLITE.

Object of work: processes of algorithmic selection and filtration of industrial CNC equipment based on user requirements.

Goal of work: development of an interactive web application for automated selection of optimal CNC machines, considering spatial, economic, and technological constraints.

Methods: multi-criteria optimization, object-oriented design, web development methods (Flask, SQLAlchemy).

Practical value: automation of engineering calculations when choosing machines to minimize financial risks.

Conclusions: a web application has been developed that calculates technological complexity, performs spatial filtration, and generates a selection of optimal equipment.

Applications: manufacturing enterprises, equipment sales departments, engineering companies, B2B platforms.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

БД – база даних

ПЗ – програмне забезпечення

САПР – система автоматизованого проєктування

СУБД – система керування базами даних

ЧПК – числове програмне керування (в англомовній літературі – CNC)

API – Application Programming Interface (прикладний програмний інтерфейс, використовується для обміну даними між клієнтом і сервером або різними сервісами)

B2B – Business-to-Business (бізнес-модель, у якій компанія продає обладнання або програмні послуги іншим компаніям

CAM – Computer-Aided Manufacturing (автоматизована система технологічної підготовки виробництва)

CSS – Cascading Style Sheets (каскадні таблиці стилів)

DOM – Document Object Model (об'єктна модель документа)

HTML – HyperText Markup Language (мова гіпертекстової розмітки)

HTTP – HyperText Transfer Protocol (протокол передавання гіпертексту)

JSON – JavaScript Object Notation (текстовий формат обміну даними)

MVC – Model-View-Controller (архітектурний шаблон проєктування «Модель-Вигляд-Контролер»)

ORM – Object-Relational Mapping (технологія об'єктно-реляційного відображення бази даних у класи мови програмування)

SaaS – Software as a Service (програмне забезпечення як послуга)

SQL – Structured Query Language (мова структурованих запитів)

UI / UX – User Interface / User Experience (користувацький інтерфейс / користувацький досвід)

ВСТУП

Сучасний розвиток промисловості характеризується переходом до концепції «Індустрія 4.0», що передбачає масштабну цифровізацію виробництва та широке використання обладнання з числовим програмним керуванням (ЧПК) [1, 2]. Ефективність функціонування профільних підприємств напряму залежить від технічно обґрунтованого вибору верстатів. Проте підбір такого обладнання є складною багатокритеріальною задачею, яка вимагає врахування габаритів робочих зон, фізико-механічних властивостей матеріалів та жорстких фінансових обмежень [3]. На практиці початківці та представники малого бізнесу часто здійснюють вибір суб'єктивно або спираючись виключно на ціну, що призводить до технологічних помилок, швидкого зносу вузлів, підвищеного браку або неефективного заморожування капіталу в надлишково потужних машинах. Існуючі комерційні платформи переважно орієнтовані на досвідчених технологів і вимагають введення вузькоспеціалізованих параметрів. Тому виникає гостра потреба у створенні доступних інтелектуальних вебсистем, здатних автоматично трансформувати бізнес-завдання користувача у точні апаратні конфігурації за допомогою математичних алгоритмів [4].

Актуальність теми роботи зумовлена необхідністю автоматизації процесів інженерного аналізу та прийняття рішень при виборі промислового обладнання з ЧПК, а також відсутністю на ринку легковагових кросплатформених інструментів, що інтегрують ієрархічні моделі складності обробки матеріалів для гнучкого підбору верстатів без перевантаження користувача надлишковими технічними параметрами.

Об'єктом роботи є процеси алгоритмічного підбору, багатокритеріального аналізу та фільтрації промислового обладнання з числовим програмним керуванням на основі вимог кінцевого користувача.

Предметом роботи є архітектура, математичні моделі ієрархічного

аналізу сумісності матеріалів, алгоритми збереження стану та програмні інструменти реалізації вебзастосунку «CNC MARKET».

Метою кваліфікаційної роботи є розробка, архітектурне проєктування та програмна реалізація сучасного інтерактивного вебзастосунку «CNC MARKET» з вбудованим алгоритмом багатокритеріальної ієрархічної фільтрації для автоматизованого підбору оптимальних моделей верстатів із ЧПК з урахуванням просторових, економічних та технологічних обмежень замовника.

Для досягнення поставленої мети в роботі було використано комплекс наукових методів, серед яких базовими є методи системного аналізу предметної області для виділення критеріїв класифікації верстатів, теорія багатокритеріальної оптимізації та логічного моделювання для розробки системи фільтраційних обмежень, методи об’єктно-орієнтованого аналізу та проєктування з використанням мови UML для побудови архітектури системи, а також сучасні інструменти вебпрограмування на базі мови Python, мікрофреймворку Flask, об’єктно-реляційного відображення SQLAlchemy та реляційних баз даних SQLite.

Практичне значення отриманих результатів полягає у створенні працездатного, масштабованого та готового до інтеграції програмного продукту, який автоматизує інженерні розрахунки при виборі верстатів. Впровадження системи «CNC MARKET» дозволяє суттєво мінімізувати фінансові та технологічні ризики підприємств, оптимізує процеси взаємодії між постачальниками обладнання та кінцевими клієнтами, знижує навантаження на менеджерів з продажу за рахунок автоматичного відсікання невідповідних конфігурацій, а також надає користувачам технічно обґрунтовані інженерні рішення у вигляді оптимальної вибірки обладнання в режимі реального часу.

1 АНАЛІЗ СУЧАСНОГО СТАНУ ПІДБОРУ ВЕРСТАТІВ З ЧПК ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Аналіз сучасного стану ринку та класифікація верстатів з ЧПК

Сучасний етап розвитку світової промисловості характеризується стрімким впровадженням технологій автоматизації та цифровізації виробничих процесів, що отримало назву «Індустрія 4.0» [5]. Основним компонентом цих змін стало використання обладнання з числовим програмним керуванням, яке сприяє мінімізації впливу людського фактора, забезпечує високу точність повторюваності виробів та істотно підвищує продуктивність праці.

Завдяки ЧПК-верстатам механічна обробка матеріалів трансформувалася з ремісничого ремесла у високотехнологічний процес, де ключову роль відіграє алгоритмічне налаштування траєкторій інструменту. Однак швидке розширення ринку такого обладнання породило новий виклик – складність у виборі оптимальної конфігурації верстата, яка відповідала б конкретним виробничим потребам.

За габаритами та рівнем жорсткості конструкції верстати з ЧПК доцільно поділити на промислові та настільні (хобійні або напівпрофесійні). Настільні портальні верстати зазвичай мають алюмінієву або полегшену сталеву станину та крокові двигуни [2]. Вони чудово підходять для обробки пластику, дерева, композитів та м’яких металів (алюмінію, латуні), проте не здатні забезпечити необхідну жорсткість для фрезерування сталі. Типовий зовнішній вигляд легкого настільного фрезерного верстата з ЧПК показано на рисунку 1.1.

Рисунок 1.1 – Зовнішній вигляд настільного портального фрезерного верстата з ЧПК

Невдало підібране обладнання може мати серйозні наслідки. Наприклад, недостатня жорсткість станини ускладнить обробку твердих матеріалів, зокрема сталі чи чавуну, спричиняючи вібрації, що пришвидшують зношування інструментів і знижують якість виробів. Навпаки, використання надмірно потужного устаткування для обробки м’яких матеріалів, таких як пароніт, алюміній чи композити, може призвести до перевитрати енергії та завищення вартості обладнання.

Ще одним важливим аспектом автоматизації є вибір привідної системи. Крокові двигуни можна успішно використовувати для завдань із невеликими вимогами до швидкості й точності, однак під час серійного виробництва, де важлива стабільна динаміка виконання завдань і відсутність пропусків кроків,

необхідно застосовувати сервоприводи. Аналогічно, некоректний вибір системи керування, такої як NC Studio, Mach3 або промислові стійки типу Siemens чи Fanuc, може значною мірою обмежити можливості масштабування виробництва або інтеграції верстата в загальну мережу підприємства.

На сьогодні вибір обладнання часто базується на суб'єктивних рекомендаціях продавців чи обмеженому досвіді персоналу, що не гарантує досягнення оптимальних техніко-економічних результатів. У зв'язку з цим актуальним завданням є створення інтелектуальних систем, здатних здійснювати багатофакторний аналіз вимог замовника та пропонувати найбільш ефективні апаратні рішення.

1.2 Огляд існуючих програмних рішень для підбору обладнання

Для визначення напрямків розробки власного алгоритму вибору верстатів з ЧПК було проведено аналіз існуючих програмних рішень та онлайн-платформ, доступних на ринку України. Наразі більшість виробників та дистриб'юторів обладнання користуються класичними інтернет-каталогами та маркетплейсами для демонстрації своєї продукції. Хоча такі платформи пропонують широкий асортимент, вони переважно орієнтовані на користувачів із глибокими інженерними знаннями, які вже чітко розуміють свої вимоги до технічних характеристик обладнання[6, 7].

В якості першого аналога було проаналізовано онлайн-платформу українського виробника обладнання з ЧПК – компанії «Dominant CNC». Сайт являє собою добре структурований каталог (рис. 1.2), у якому обладнання розподілено за основними категоріями: фрезерні, лазерні та плазмові верстати. Інтерфейс ресурсу надає можливість користувачу детально ознайомитися з технічними характеристиками обраної моделі, включаючи розміри робочого поля (наприклад, 1000×1000 мм або 2000×3000 мм), потужність шпинделя та тип напрямних.

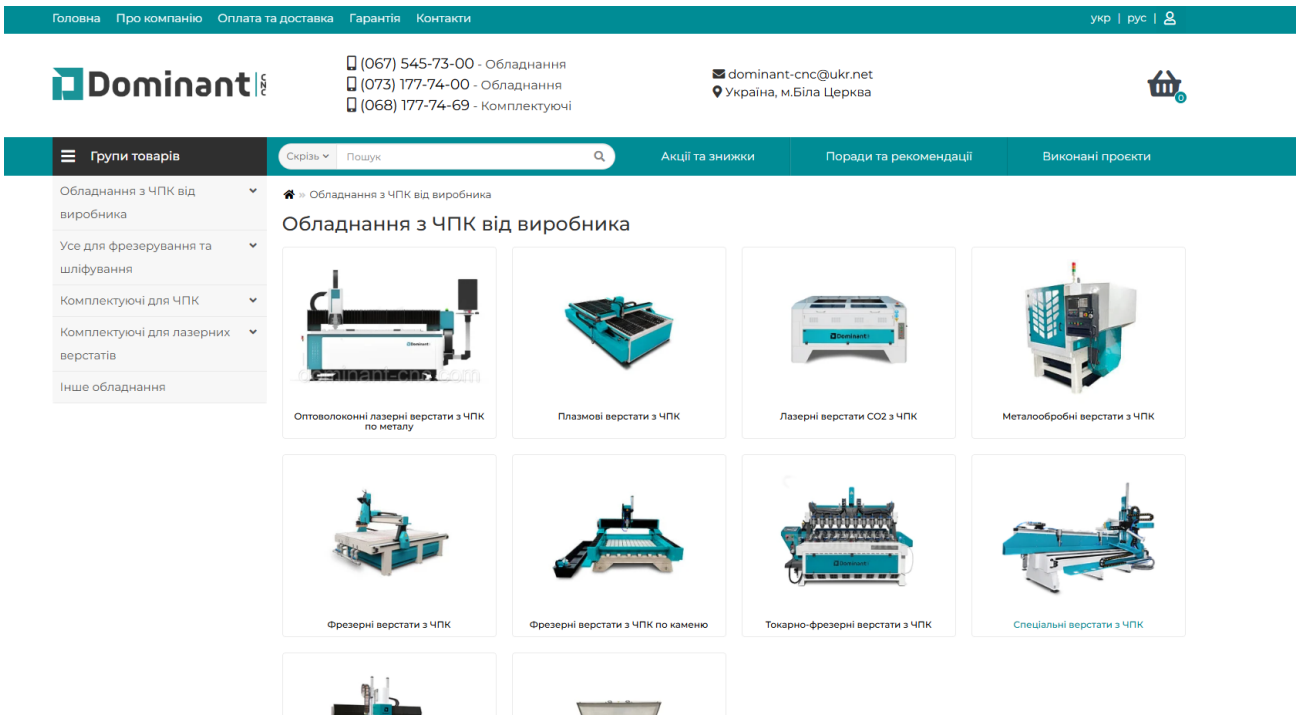

Рисунок 1.2 – Інтерфейс каталогу обладнання на платформі компанії Dominant CNC

Детальний аналіз функціональних можливостей платформи Dominant CNC виявило кілька суттєвих недоліків з точки зору зручності для клієнта. Зокрема, система пошуку обмежена статичними фільтрами та рубрикатором. Це створює проблеми для початківців і представників малого бізнесу, які орієнтуються лише на базові параметри, як-от тип матеріалу (наприклад, фанера 18 мм або листовий алюміній) та доступний бюджет. У поточному вигляді платформа неспроможна алгоритмічно запропонувати оптимальне рішення, змушуючи користувача самостійно переглядати й аналізувати описи кожної моделі або звертатися за порадами до менеджера. Такий підхід значно уповільнює процес ухвалення рішення.

Іншим показовим прикладом є вебсайт українського виробника «Javelin CNC», який спеціалізується на виготовленні фрезерних та плазмових верстатів (рис. 1.3).

Аналіз вебсайту Javelin CNC показує, що хоча виробник надає вичерпну технічну інформацію про комплектуючі (тип приводів, системи керування NC

Studio або DSP-контролери), платформа не має вбудованої системи автоматизованого підбору Відсутня можливість багатокритеріального фільтрування на основі технологічних завдань. Тобто система діє як цифрова вітрина, а не як інтелектуальний помічник.

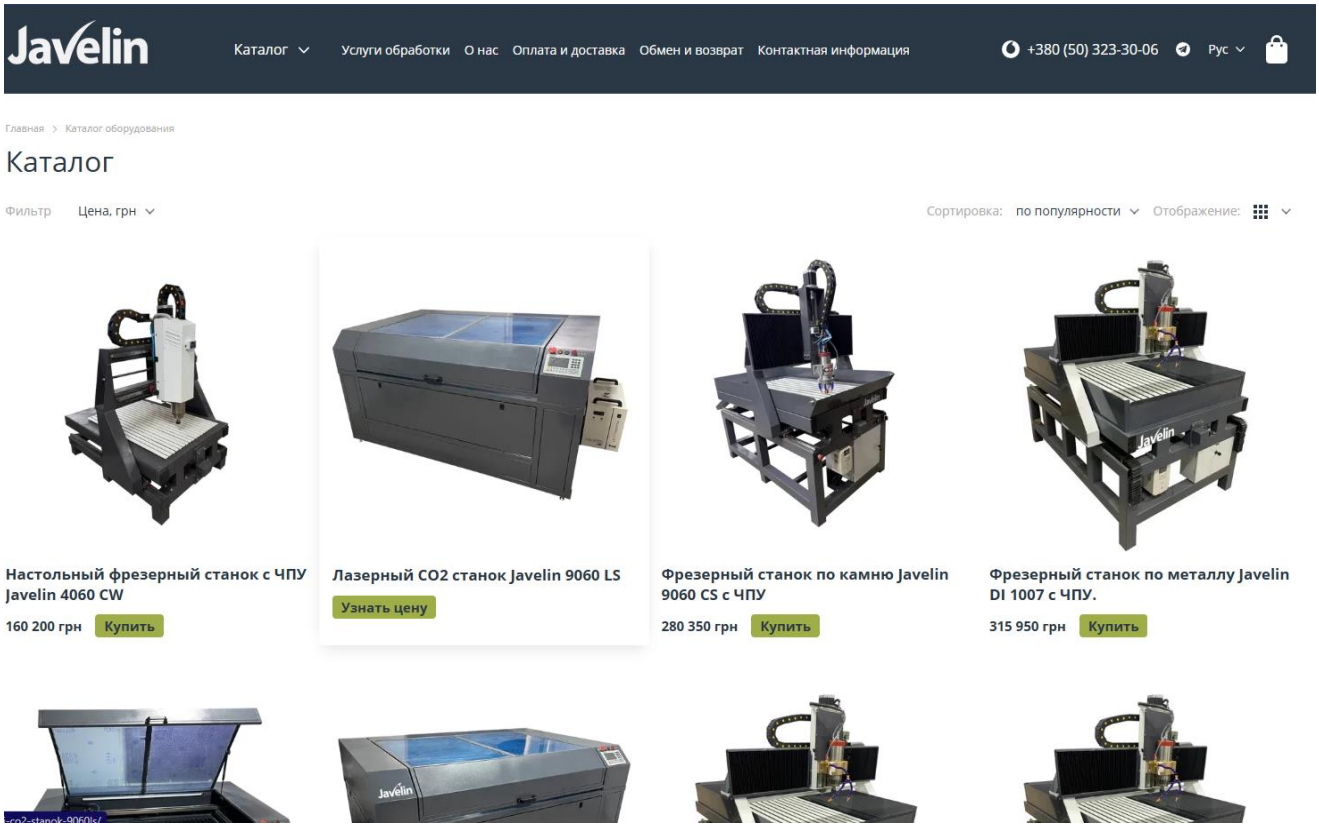

Рисунок 1.3 – Представлення серій обладнання з ЧПК на платформі Javelin CNC

Загальним недоліком проаналізованих рішень є відсутність клієнтоорієнтованого алгоритмічного підбору. Існуючі вебсайти вимагають від користувача введення готових апаратних параметрів (потужність у кВт, точність у мм), замість того щоб запитувати виробничі цілі (тип обробки, матеріал, габарити заготовки) та самостійно конвертувати їх у технічні вимоги до верстата.

З метою структурування виявлених недоліків і переваг була створена порівняльна таблиця 1.1 існуючих платформ і вебзастосунку, запропонованого в рамках кваліфікаційної роботи.

Таблиця 1.1 – Порівняльний аналіз функціональних можливостей систем підбору верстатів

Критерій порівняння	**Платформа Dominant CNC**	**Платформа Javelin CNC**
Наявність структурованого каталогу	є	є
Фільтрація за жорсткими технічними параметрами	є	є
Врахування матеріалу обробки під час пошуку	Немає	Немає
Автоматичний розрахунок необхідної потужності	Немає	Немає
Алгоритмічні рекомендації на основі бюджету	Немає	Немає
Орієнтованість на користувачів без інженерного досвіду	Немає	Немає

Враховуючи результати аналізу, можна зробити висновок, що на вітчизняному ринку існує незадоволена потреба у розробленні спеціалізованого вебзастосунку. Цей застосунок повинен базуватися на алгоритмічній моделі, яка буде виконувати роль технічного експерта, переводячи бізнес-вимоги замовника у конкретну специфікацію верстата з ЧПК.

1.3 Обґрунтування та формування вимог до програмного забезпечення

Проектування сучасних інформаційних систем та вебзастосунків потребує попереднього детального аналізу, класифікації та формалізації вимог до програмного продукту. Процес інженерії вимог для розроблюваної системи

алгоритмічного підбору обладнання з числовим програмним керуванням «CNC MARKET» спрямований на визначення меж проєкту, мінімізацію ризиків логічних помилок на етапі кодування та забезпечення відповідності підсумкового продукту потребам кінцевих користувачів. Відповідно до загальноприйнятих інженерних стандартів, архітектурні вимоги до системи розділено на три базові категорії, які охоплюють вимоги до ролей користувачів, функціональні сервіси, що визначають перелік безпосередніх завдань ПЗ, а також нефункціональні параметри, які окреслюють якісні характеристики, обмеження середовища та експлуатаційні характеристики застосунку.

Для забезпечення високого рівня зручності використання та оптимізації інтерфейсу вебзастосунку було проведено аналіз потенційної цільової аудиторії системи. Враховуючи специфіку предметної області, виділено єдину базову роль користувача системи – Гість або Клієнт-інженер. У межах розроблюваного мінімально життєздатного продукту відсутня необхідність упровадження обов’язкової авторизації або створення особистих кабінетів, оскільки ключова бізнес-логіка полягає у миттєвому наданні релевантних інженерних рішений без створення додаткових бар’єрів для користувача. Роль Клієнта-інженера передбачає повний доступ до всього обчислювального та інформаційного функціоналу системи. При цьому користувач цього типу може володіти різним рівнем технічної підготовки. Наприклад, категорія новачків, до якої відносяться підприємці або початківці у сфері ЧПК, зазвичай не володіє глибокими знаннями про апаратну частину верстатів, проте чітко знає параметри свого бізнес-завдання, такі як тип матеріалу, габарити деталі та фінансові обмеження. Для цієї категорії критично важливим є інтуїтивно зрозумілий покроковий інтерфейс конфігуратора. Натомість категорія професіоналів, куди входять технологи та інженери виробництва, вільно орієнтується у технічних характеристиках обладнання та використовує систему для швидкого моніторингу ринку, порівняння наявних моделей верстатів або швидкої перевірки сумісності габаритів заготовок із наявною

інструментальною базою через модуль повного каталогу.

Функціональні вимоги визначають безпосередню поведінку системи та описують сервіси, які програмний продукт має надавати користувачеві для успішного вирішення завдання багатокритеріального підбору верстатів. По-перше, система повинна надавати інтерактивну графічну форму для введення параметрів запиту користувача, яка включає вибір типу обробки, конструкційних матеріалів, введення лінійних габаритів деталі та обмеження максимальної вартості. При цьому компонент введення матеріалів повинен підтримувати множинний вибір за допомогою чекбоксів, згрупованих за рівнями технологічної складності обробки. Блок вказівки фінансових обмежень реалізується у вигляді динамічного повзунка, граничні мінімальні та максимальні значення якого автоматично підтягуються з бази даних на основі обраного типу обладнання. Серверна частина застосунку має здійснювати обов’язкову типізацію та очищення даних, блокувати від’ємні числові значення та обробляти винятки некоректного введення без аварійного завершення сесії.

По-друге, бекенд-модуль системи повинен реалізувати математичну модель багатокритеріального аналізу, послідовно пропускаючи масив даних через технологічний, геометричний, економічний та ієрархічно-матеріальний фільтри. Застосунок має підтримувати логіку адаптивного керування інтерфейсом, тому у разі вибору лазерного типу обробки параметр висоти заготовки по осі Z повинен програмно ігноруватися алгоритмом фільтрації та приховуватися у клієнтській частині. Сама логіка обробки матеріалів базується на моделі підгруп складності, забезпечуючи автоматичне включення верстатів вищого технологічного класу до результатів пошуку простіших матеріалів.

По-третє, програмний комплекс має забезпечувати відображення результатів та керування станом, виводячи результуючу вибірку, обмежену трьома найбільш оптимальними моделями верстатів, відсортованими за критерієм мінімізації ціни. Кожна картка верстата у результатах пошуку

повинна містити графічне фото обладнання, повну назву моделі, тип, лінійні характеристики робочої зони та актуальну вартість. Додатково впроваджується механізм керування станом в межах поточної сесії, завдяки чому при поверненні зі сторінки результатів на головну сторінку всі раніше введені параметри та прапорці мають відновлюватися автоматично. Нарешті, вебзастосунок повинен містити відокремлений функціональний модуль повного каталогу обладнання, доступний за окремим маршрутом, який відображає весь перелік наявних у базі даних верстатів без обмеження кількості результатів, без застосування жорстких геометричних чи матеріальних фільтрів, та підтримує інтерактивну навігацію для швидкого перемикання між класами обробки з базовим ранжуванням моделей за зростанням ціни.

Нефункціональні вимоги описують якісні характеристики системи, які безпосередньо не пов’язані з конкретними функціями, проте визначають надійність, продуктивність, безпеку та зручність експлуатації продукту. У контексті продуктивності та ефективності встановлено, що час відгуку сервера при обробці POST-запиту конфігуратора та виконанні алгоритму фільтрації в базі даних не повинен перевищувати 200 мілісекунд за умов нормального мережевого з’єднання, а рендеринг сторінок має відбуватися без створення помітних затримок в інтерфейсі. Що стосується надійності та доступності, програмний продукт повинен демонструвати стабільну роботу в режимі двадцять чотири на сім із середнім часом безвідмовної роботи не менше 99.5%, а також коректно обробляти помилки відсутності зв’язку з базою даних SQLite, виводячи безпечне інформаційне повідомлення замість критичного збою всього вебсервера.

Вимоги до сумісності та адаптивності визначають, що клієнтська частина вебзастосунку повинна мати адаптивний дизайн, забезпечуючи коректне відображення інтерфейсу, форм, таблиць та карток верстатів на пристроях із різною роздільною здатністю екрана від мобільних телефонів до стаціонарних моніторів, а також підтримувати кросбраузерність у всіх

сучасних версіях популярних браузерів. Зручність використання забезпечується виконанням інтерфейсу системи у сучасному мінімалістичному промисловому стилі із використанням темної кольорової палітри та контрастних індустріальних оранжевих акцентів для привернення уваги до ключових елементів керування, а чіткість текстів досягається використанням шрифтів родини IBM Plex з дотриманням оптимальних відступів за сіткою Bootstrap 5. Нарешті, вимоги до безпеки та супроводжуваності гарантують, що дані користувацьких сесій захищені від підробки на стороні клієнта за допомогою криптографічного підписання сесій зашифрованим ключем, а архітектура коду відповідає принципам модульності та чистоти [8], де використання ORM SQLAlchemy ізолює логіку запитів від конкретної СУБД і забезпечує легку можливість швидкої міграції системи на потужніші реляційні бази даних без переписування коду бізнес-логіки.

1.4 Практична значущість та техніко-економічне обґрунтування розробки

Аналіз реальних процесів експлуатації, а також безпосереднього складання та налагодження ЧПК-обладнання на вітчизняних підприємствах виявляє критичний розрив між маркетинговими пропозиціями постачальників та реальними інженерними потребами замовників. Більшість помилок при купівлі верстатів виникає через нерозуміння кінцевими клієнтами фізики процесу обробки та механіки верстата. Наприклад, при фрезеруванні листових матеріалів, що схильні до вібрацій або відриву від робочого столу, критично важливим є не лише потужність шпинделя, але й правильний підбір обладнання, здатного працювати зі специфічним ріжучим інструментом – зокрема, з фрезами, що мають низхідний напрямок відведення стружки (down-cut), які фізично притискають деталь до столу та забезпечують стабільність матеріалу. Якщо ж клієнт, орієнтуючись лише на ціну, купує легкий

портальний верстат для твердих в’язких матеріалів, це неминуче призводить до резонансу станини, швидкого зносу механічних вузлів та масового браку виробів.

Саме тому впровадження розробленої автоматизованої системи «CNC MARKET» має високу практичну значущість. Вона переносить фокус із ручного, суб’єктивного вибору на суворий математичний розрахунок. Алгоритм діє як віртуальний інженер-налагоджувальник, який ще до моменту покупки відсікає технологічно несумісні конфігурації. З економічної точки зору, це дозволяє підприємствам уникати двох найпоширеніших фінансових ризиків: недооснащення (коли купується занадто слабкий верстат, що не справляється з цільовим матеріалом і потребує швидкої заміни) та переоснащення (коли підприємство заморожує зайвий капітал, купуючи надлишково потужне та дороге обладнання для простих повсякденних завдань).

Крім того, розроблений програмний продукт суттєво оптимізує роботу менеджерів з продажу металообробного та деревообробного обладнання. Використання формалізованої математичної моделі підбору зменшує час на технічну консультацію одного клієнта, оскільки система автоматично формує первинний список релевантних моделей на основі введених габаритів та матеріалів. Клієнт отримує не просто прайс-лист, а технічно обґрунтовану інженерну пропозицію, яка гарантує, що обрана машина фізично здатна виконати його виробничу задачу з дотриманням необхідних допусків. Таким чином, розроблений вебзастосунок є не лише зручним IT-інструментом, але й важливим елементом оптимізації сучасних виробничих і бізнес-процесів у галузі верстатобудування, що підтверджує доцільність його створення та подальшого впровадження.

1.5 Постановка задачі

Широкий вибір верстатів із числовим програмним керуванням на сучасному ринку створює суттєві складнощі для підприємств та приватних майстрів, які не завжди володіють глибокими інженерними знаннями у сфері станкобудування. Наявні онлайн-каталоги та маркетплейси часто не в змозі забезпечити якісний підбір обладнання відповідно до ключових технологічних вимог, таких як тип оброблюваного матеріалу, розміри заготовки, необхідна точність та фінансові рамки. Відсутність автоматизованих експертних систем у цій сфері нерідко призводить до неоптимальних інвестиційних рішень і зниження загальної ефективності виробництва. У зв’язку з цим виникає гостра потреба у створенні спеціалізованого інструменту, який би на основі заданих критеріїв рекомендував оптимальні моделі верстатів. Це є актуальною та практично значущою задачею, вирішення якої дозволить значно спростити та покращити процес вибору обладнання.

Об’єктом роботи обрано процес алгоритмічного підбору верстатів із ЧПК на основі індивідуальних вимог користувача.

Метою роботи є створення вебзастосунку, який автоматизує процес вибору ЧПК-верстатів відповідно до специфічних потреб користувачів, сприяючи таким чином оптимізації процесу закупівлі виробничого обладнання.

Для досягнення поставленої мети необхідно виконати такі завдання:

– провести аналіз існуючих програмних рішень та платформ для підбору металообробного обладнання:

– ідентифікувати основні критерії та технологічні параметри, що визначають вибір верстатів із ЧПК;

– розробити алгоритмічну модель для прийняття рішень щодо найбільш відповідного обладнання;

– спроєктувати та реалізувати ієрархічну математичну модель оброблюваних матеріалів за групами їхньої щільності та складності;

– обґрунтувати вибір програмних засобів для створення та реалізації вебзастосунку;

– спроєктувати архітектуру та користувацький інтерфейс вебзастосунку, щоб забезпечити зручне введення початкових даних;

– розробити модуль повноцінного каталогу обладнання з можливістю сортування моделей без обмежень алгоритму фільтрації;

– створити сучасний користувацький інтерфейс у промисловому стилі з візуалізацією обладнання;

– реалізувати програмний продукт, що інтегрує базу даних доступних верстатів та алгоритм їх підбору;

– провести тестування створеного вебзастосунку на реальних даних і здійснити аналіз отриманих результатів.

2 МАТЕМАТИЧНЕ ТА АЛГОРИТМІЧНЕ ЗАБЕЗПЕЧЕННЯ ВЕБЗАСТОСУНКУ

2.1 Обґрунтування критеріїв підбору та вибір інструментальних засобів

Для розроблення ефективного вебзастосунку, здатного автоматизувати процес прийняття рішень у сфері вибору складного технологічного обладнання, необхідно насамперед формалізувати систему критеріїв оцінювання. На відміну від стандартних систем фільтрації, які пропонують виробники, розроблювана система має базуватися на технологічних потребах кінцевого користувача, що дозволяє усунути розрив між бізнес-завданням та апаратною реалізацією.

Фундаментальним критерієм алгоритмічного підбору є фізико-механічні властивості матеріалу, який підлягає обробці. У процесі налаштування та експлуатації верстатів з ЧПК встановлено, що різні групи матеріалів диктують специфічні вимоги до жорсткості станини, типу приводу та характеристик шпиндельного вузла. Залежно від природи оброблюваного матеріалу ці вимоги можуть кардинально відрізнятися, що робить даний критерій одним із визначальних на етапі формування рекомендацій.

Так, у випадку м’яких та в’язких матеріалів – таких як алюміній, латунь, пароніт або різноманітні композити – критичним параметром є забезпечення високої частоти обертання шпинделя у поєднанні з можливістю прецизійного регулювання вильоту інструменту. Саме ці характеристики дозволяють мінімізувати налипання стружки на різальну кромку та забезпечити необхідну якість обробленої поверхні, що є принципово важливим для деталей з підвищеними вимогами до чистоти.

Принципово інша ситуація складається при роботі з твердими матеріалами, зокрема зі сталлю та чавуном. Тут пріоритет зміщується від швидкісних характеристик у бік статичної та динамічної жорсткості всієї конструкції верстата. Ефективна обробка таких матеріалів потребує

застосування масивних чавунних або важких сталевих станин, які здатні ефективно поглинати та гасити вібрації, що виникають у процесі різання. Не менш важливою є наявність приводів із високим крутним моментом на низьких частотах обертання, що забезпечує стабільне знімання матеріалу без ризику зупинки шпинделя або втрати точності позиціонування.

Алгоритмічна модель вебзастосунку повинна оперувати чітко структурованою системою параметрів, яку доцільно розподілити на три взаємопов'язані блоки. Кожен із них відповідає за окремий аспект процесу вибору обладнання та в сукупності забезпечує повноцінне покриття базових потреб користувача – від технічних вимог до фінансових обмежень. Така структуризація не лише спрощує логіку обробки вхідних даних, а й дозволяє системі формувати обґрунтовані рекомендації, що відповідають реальним умовам конкретного виробничого середовища.

Перший блок – технологічно-матеріальний. Цей блок є первинним фільтром і охоплює характеристики, що безпосередньо визначають тип необхідної обробки (фрезерна, лазерна або плазмова) та перелік підтримуваних матеріалів. Оскільки різні моделі верстатів орієнтовані на різну щільність матеріалів, алгоритм повинен вміти здійснювати аналіз запиту користувача та зіставляти його з базою можливостей кожної машини. До цього ж блоку належить показник потужності головного робочого органу (шпинделя або лазерного випромінювача), який визначає здатність верстата прорізати матеріал заданої товщини.

Другий блок – просторово-геометричний. Цей блок стосується фізичних обмежень верстата і визначається габаритами робочої зони по осях X, Y та Z. У межах алгоритмічної моделі ці параметри виконують функцію жорсткого відсікання: верстат фізично не здатний обробити заготовку, розміри якої перевищують його робоче поле. Важливою особливістю математичної моделі є динамічна обробка осі Z: алгоритм враховує її для фрезерного та плазмового обладнання, де глибина обробки є критичною, і виключає її з розрахунків для лазерних верстатів, специфіка яких полягає у двовимірному площинному

розкрої матеріалів.

Третій блок – економічно-ранжувальний. Цей блок замикає систему параметрів і дозволяє оцінити доцільність вибору з фінансової точки зору. До нього входить гранична вартість придбання обладнання (бюджет користувача). Алгоритм не лише відсікає моделі, що перевищують фінансові ліміти замовника, але й використовує економічний критерій для фінального ранжування результатів пошуку, пропонуючи користувачу оптимальні варіанти за критерієм мінімізації вартості серед технічно придатних моделей.

Запропонована структура критеріїв є масштабованою і в майбутньому дозволяє легко розширювати базу даних додатковими конструктивними або програмними параметрами (наприклад, типами приводів чи сумісністю із CAM-системами) без кардинальної зміни логіки ядра застосунку.

Для реалізації зазначених критеріїв у формі функціонального програмного продукту необхідно обрати сучасний і гнучкий стек вебтехнологій. Архітектура додатку буде розділена на клієнтську (Frontend) та серверну (Backend) частини, кожна з яких має свої завдання.

Клієнтська частина повинна забезпечити зрозумілий і зручний інтерфейс у вигляді покрокового конфігуратору, де користувач вводить свої вимоги до системи. Це дозволить створити інтерактивний досвід для користувачів і скоротити час на введення даних.

Серверна частина, зі свого боку, відповідатиме за обробку отриманих запитів, взаємодію з реляційною базою даних, у якій зберігаються параметри десятків моделей обладнання, та проведення розрахунків на основі введених вимог. Такий розподіл обов’язків між частинами забезпечить ефективність, масштабованість і зручність у подальшому доопрацюванні системи.

Для розроблення алгоритмічного ядра та серверної частини (Backend) вебзастосунку обрано високорівневу мову програмування Python. Цей вибір є безальтернативним стандартом для подібних задач, що обґрунтовано високою ефективністю мови при роботі з математичними моделями, наявністю

розгалуженої екосистеми бібліотек для обробки даних та оптимальною швидкістю розгортання вебсервісів.

На основі обраної мови програмування, як програмний фреймворк було обрано Flask. Для обґрунтування цього архітектурного рішення було проведено порівняльний аналіз популярних вебфреймворків (табл. 2.1).

Таблиця 2.1 – Порівняння фреймворків для серверної розробки

Критерій порівняння	**Flask**	**Django**	**FastAPI**
Складність архітектури	Мікрофреймворк (легкий)	Моноліт (важкий)	Асинхронний (середній)
Швидкість розгортання прототипів	Висока	Середня	Висока
Гнучкість налаштувань	Повна	Обмежена структурою	Висока
Підтримка SQLAlchemy	Рідна/повна	Власна ORM	Повна

Обраний мікрофреймворк Flask ідеально підходить для створення вузькоспеціалізованих застосунків, оскільки він не перевантажений зайвими модулями, що забезпечує мінімальні затримки при обробці HTTP-запитів і дозволяє зосередитися на логіці алгоритму. Для зберігання інформації про верстати обрано реляційну СКБД SQLite, яка завдяки своїй компактності та відсутності потреби в окремому сервері є ідеальною для реалізації дипломного проєкту. Взаємодія з базою даних здійснюється за допомогою технології Object-Relational Mapping (ORM) у вигляді бібліотеки SQLAlchemy.

Процес розроблення та налагодження коду здійснюється в середовищі PyCharm. Це дозволяє використовувати професійні інструменти профілювання коду, інтегровані засоби тестування та зручне керування віртуальними середовищами, що критично важливо для стабільної роботи

алгоритмів підбору.

Таким чином, інтеграція сучасних засобів веброзробки з експертними даними про технологічні режими обробки матеріалів створює надійне підґрунтя для реалізації інтелектуального помічника у виборі ЧПК-обладнання.

2.2 Проєктування бази даних та структури зберігання інформації

Ефективність функціонування алгоритму автоматизованого підбору обладнання безпосередньо залежить від раціональної організації зберігання технічних характеристик. Для реалізації вебзастосунку було обрано реляційну модель даних, яка дозволяє чітко типізувати параметри верстатів та забезпечує високу швидкість фільтрації як за числовими, так і за текстовими критеріями. Реляційний підхід також створює надійне підґрунтя для масштабування системи у разі розширення номенклатури обладнання або введення додаткових параметрів підбору в майбутньому.

Як систему керування базами даних обрано SQLite, що зумовлено її компактністю, автономністю та відсутністю потреби в окремому серверному процесі [10,11]. Вся база даних зберігається в єдиному файлі machines.db, що суттєво спрощує розгортання системи та процедуру резервного копіювання. При цьому SQLite забезпечує повну підтримку стандартів SQL та демонструє достатню продуктивність в умовах локального вебзастосунку з обмеженим числом одночасних звернень.

Взаємодія програмного коду з базою даних реалізована через об’єктно-реляційне відображення за допомогою бібліотеки SQLAlchemy, що дозволяє описувати структуру даних у вигляді класів мови Python[11, 12]. Такий підхід абстрагує бізнес-логіку від деталей SQL-запитів, підвищує читабельність коду та полегшує його подальшу підтримку й модифікацію.

Центральною сутністю архітектури даних є таблиця machines, структура

якої розроблена таким чином, щоб охопити всі ключові параметри, що беруть участь у логіці підбору обладнання. Детальна специфікація полів таблиці наведена в таблиці 2.2.

Таблиця 2.2 – Специфікація полів таблиці

Назва поля	**Тип даних**	**Опис та призначення**
id	Integer	Унікальний ідентифікатор запису (первинний ключ)
name	String(200)	Комерційна назва верстата або серії
type	String(50)	Категорія обладнання («Фрезерний», «Лазерний», «Плазмовий»)
material_supported	Text	Перелік матеріалів, що підтримуються для обробки
work_area_x	Float	Максимальний розмір обробки за віссю *X* (мм)
work_area_y	Float	Максимальний розмір обробки за віссю *Y* (мм)
work_area_z	Float	Хід інструмента за віссю *Z* (мм)
spindle_power_kw	Float	Потужність робочого вузла (шпинделя або випромінювача), кВт
price_uah	Integer	Вартість верстата в національній валюті (грн)
image_filename	String(255)	Ім’я файлу зображення верстата. Допускається значення NULL

Поля work_area_x, work_area_y та work_area_z мають тип Float, що дозволяє зберігати габарити робочої зони з високою точністю та без втрати дробової частини значення. Для верстатів, специфіка яких не передбачає повноцінної обробки за віссю *Z* – зокрема, для установок лазерного розкрою –

у відповідному полі зберігається значення 0, що забезпечує уніфіковану структуру запису для всіх типів обладнання без введення додаткових полів або nullable-значень.

Поле material_supported реалізоване як тип Text і призначене для зберігання переліку матеріалів, які підтримує конкретна модель обладнання. У фінальній архітектурі системи логіку обробки цього параметра було суттєво модернізовано: замість базового пошуку підрядка впроваджено ієрархічну модель груп складності обробки (MATERIAL_GROUPS_BY_TYPE) [13]. Усі конструкційні матеріали класифіковано за рівнями їхньої щільності та в’язкості. Для кожного верстата в базі визначається максимальний підтримуваний рівень технологічної групи: якщо обладнання здатне обробляти складніший матеріал (наприклад, кольорові метали чи сталь), воно автоматично вважається придатним для всіх простіших груп (наприклад, деревини чи полімерів). Крім того, на рівні інтерфейсу користувач може обрати кілька матеріалів одночасно за допомогою системи чекбоксів, які обробляються алгоритмом за логікою диз’юнкції (оператор OR), що забезпечує високу гнучкість формування вибірки.

Процес первинного розгортання та ініціалізації бази даних повністю автоматизовано за допомогою функції init_db, яка інтегрована в головний модуль застосунку app.py. При кожному запуску програми зазначена функція перевіряє наявність службового каталогу instance, а також створює директорію static/images/machines/ для зберігання графічних асетів. За допомогою методу db.create_all() формується необхідна структура таблиць у файлі machines.db. Для забезпечення еволюції схеми даних без ризику втрати наявної інформації у функцію інтегровано метод _ensure_schema(). Цей алгоритм динамічно аналізує поточний стан реляційної схеми і, у разі виявлення нових полів (зокрема, атрибуту image_filename, який відповідає за збереження імен файлів фотографій обладнання), автоматично виконує легку міграцію, додаючи відповідні колонки до існуючої структури.

Первинне наповнення бази даних (data seeding) реалізовано через

відокремлений скрипт seed_db.py. Він автоматично наповнює каталог базовим набором із 8 промислових моделей верстатів різних технологічних класів, серед яких представлено фрезерне (серії «Кречет 6090», «Ястреб 6090» та «Ястреб 1212-250»), лазерне (моделі «ЛСК» та «ЛС») та плазмове обладнання (моделі серії «СМ-1212», «СМ-2020», «СМ-R 1212») [14]. Це гарантує негайну готовність системи «CNC MARKET» до демонстрації та тестування алгоритмів підбору одразу після встановлення проєктних залежностей.

2.3 Математична модель багатокритеріального вибору обладнання

Основою розробленого вебзастосунку є алгоритм багатокритеріальної фільтрації та ранжування, який дозволяє автоматизувати процес підбору обладнання [3,15, 16]. З математичної точки зору, завдання зводиться до пошуку оптимальної підмножини верстатів із загальної множини доступних варіантів [17], що задовольняють заданій системі обмежень (вимогам користувача), з подальшим застосуванням цільової функції мінімізації вартості.

Нехай $S = \{M_1, M_2, \dots , M_n\}$ – загальна множина верстатів, наявних у базі даних каталогу. Кожен i-й верстат описується вектором техніко-економічних параметрів:

$$M_i = (t_i , mat_i , x_{mi} , y_{mi} , z_{mi} , p_i), \tag{2.1}$$

де t_i – тип обладнання (фрезерний, лазерний, плазмовий);

mat_i – множина підтримуваних матеріалів (зберігається як текстовий рядок);

x_{mi}, y_{mi}, z_{mi} – габарити робочої зони у міліметрах;

p_i – вартість верстата у гривнях.

Запит користувача формалізується у вигляді аналогічного вектора обмежень:

$$U = (T_u\ , M_u\ , X_u\ , Y_u\ , Z_u\ , B_u), \tag{2.2}$$

де T_u – обраний користувачем тип обробки;

M_u – цільовий матеріал;

X_u, Y_u, Z_u – мінімально необхідні габарити заготовки;

B_u – фінансовий бюджет.

Процес прийняття рішення реалізується через систему жорстких фільтрів (кон'юнкцію логічних умов), які послідовно застосовуються до кожного елемента множини S. Алгоритм перевіряє виконання наступних умов для кожного верстата M_i :

1. Умова технологічної відповідності (тип обладнання):

$$C_1(M_i) = \begin{cases} 1\ , \text{ якщо } t_i = T_u \\ 0\ , \text{ якщо } t_i \neq T_u. \end{cases} \tag{2.3}$$

2. Умова матеріальної сумісності (ієрархічна модель груп складності): замість прямого лексемічного порівняння, алгоритм використовує ієрархічну модель щільності матеріалів [18, 19, 20]. Нехай $G = \{G_0, G_1, \dots, G_k\}$ – впорядкована множина груп складності матеріалів для певного типу обладнання (від найм'якіших G_0 до найтвердіших G_k).

Для фрезерного обладнання ієрархія має вигляд: м′які матеріали → деревина → полімери → алюміній → латунь. Для плазмового: лист → металопрокат/сталь → профілі → труби.

Введемо функцію *tier(m)*, яка повертає числовий індекс групи складності для матеріалу m. Тоді максимальний рівень складності, який підтримує верстат M_i , визначається як: $maxTier(M_i) = max_{m \in mat_i} tier(m)$.

Логіка фільтрації базується на технологічному принципі: якщо верстат

здатний обробляти матеріал вищої групи складності, він гарантовано підходить для обробки матеріалів нижчих груп. Відповідно, умова набуває вигляду:

$$C_2(M_i) = \begin{cases} 1, \text{ якщо } M_u = \emptyset \text{ або } maxTier(M_i) \geq max_{m \in mat_i} tier(m) \\ 0, \text{ інакше.} \end{cases} \quad (2.4)$$

3. Умова просторової сумісності (геометричний фільтр): для успішного проходження цього фільтра габарити робочої зони верстата мають бути більшими або дорівнювати габаритам деталі. Враховуючи специфіку лазерного обладнання, для якого вісь Z не є критичною, умова формалізується так:

$$C_3(M_i) = \begin{cases} 1, \text{ якщо } (X_{mi} \geq X_u) \wedge (Y_{mi} \geq Y_u) \wedge (t_i = \text{"Лаз."} \vee Z_{mi} \geq Z_u) \\ 0, \text{ інакше.} \end{cases} \quad (2.5)$$

4. Умова економічної доступності (бюджетний фільтр):

$$\mathrm{C}_4(M_i) = \begin{cases} 1, \text{ якщо } p_i \leq B_u \\ 0, \text{ якщо } p_i > B_u. \end{cases} \quad (2.6)$$

Результуюча множина $M_{filtred}$, яка містить лише ті верстати, що повністю задовольняють вимоги користувача, визначається як перетин усіх описаних умов:

$$M_{filtred} = \{M_i \in M \mid C_1(M_i) \times C_2(M_i) \times \mathrm{C}_3(M_i) \times \mathrm{C}_4(M_i) = 1\}. \quad (2.7)$$

На фінальному етапі до множини $M_{filtred}$ застосовується критерій оптимізації. Оскільки метою користувача є мінімізація капітальних витрат за умови дотримання всіх технологічних вимог, цільова функція набуває такого

вигляду:

$$F(M_i) = p_i \rightarrow min. \quad (2.8)$$

Програмна реалізація алгоритму сортує елементи множини $M_{filtred}$ за зростанням показника вартості та виводить користувачу підмножину, обмежену трьома найдешевшими варіантами, що забезпечує оптимальний вибір без перевантаження надмірною інформацією.

2.4 Алгоритмічна та архітектурна реалізація процесу фільтрації

Програмна реалізація математичної моделі підбору обладнання у межах розроблюваної системи «CNC MARKET» виконана на базі патерну проєктування MVC (Model-View-Controller), де мікрофреймворк Flask виконує роль контролера, скеровуючи потоки даних між користувацьким інтерфейсом та реляційною базою даних[21,22]. Архітектура процесу обробки запитів побудована за принципом послідовної валідації, типізації та ієрархічної фільтрації, що забезпечує передбачувану та стійку поведінку системи навіть за умови некоректного введення даних клієнтом.

Взаємодія між клієнтською та серверною частинами відбувається через фіксовані кінцеві точки, кожна з яких відповідає за ізольований етап бізнес-логіки. Повну специфікацію цих точок доступу та логіку їхньої маршрутизації наведено у табл. 2.3. Повний цикл обробки HTTP-запитів та логіку маршрутизації потоків даних відображено на розробленій UML-діаграмі активності (рис. 2.1).

Життєвий цикл запиту на підбір обладнання починається на стороні клієнта, де користувач заповнює параметри конфігуратора у відповідній формі інтерфейсу. Сформовані дані передаються на сервер методом HTTP POST через точку доступу /results. Оскільки всі значення з HTML-форми надходять

у вигляді рядкових типів, першим обов'язковим етапом обробки є сувора типізація та перетворення вхідних параметрів, що запобігає критичним збоям у подальшій роботі алгоритму та бази даних.

Таблиця 2.3 – Специфікація маршрутів вебзастосунку «CNC MARKET»

URL-шлях	**HTTP-метод**	**Призначення та архітектурна логіка**
/	GET	Рендеринг головної HTML-сторінки конфігуратора з автоматичним відновленням стану форми із сесії
/results	POST	Прийом вхідного вектора параметрів, валідація, виклик функції filter_machines та рендеринг результатів
/results	GET	Захисний перенаправлений маршрут (Redirect) на головну сторінку для запобігання помилкам повторного відправлення форм
/catalog	GET	Рендеринг модуля повного каталогу наявних моделей верстатів із підтримкою фільтрації за параметром ?type=

Для безпечного перетворення рядкових значень у числові типи з рухомою комою Float та цілі числа Integer розроблено спеціалізовані функції-парсери _parse_form_float та _parse_form_int. Ці функції автоматично очищують вхідні дані від зайвих пробілів методом strip, виконують заміну ком на крапки для коректного розпізнавання десяткових дробів у різних локалях, а також перехоплюють винятки ValueError, повертаючи безпечне значення None у разі введення некоректних символів. Такий підхід унеможливлює потрапляння недійсних значень до логіки фільтрації.

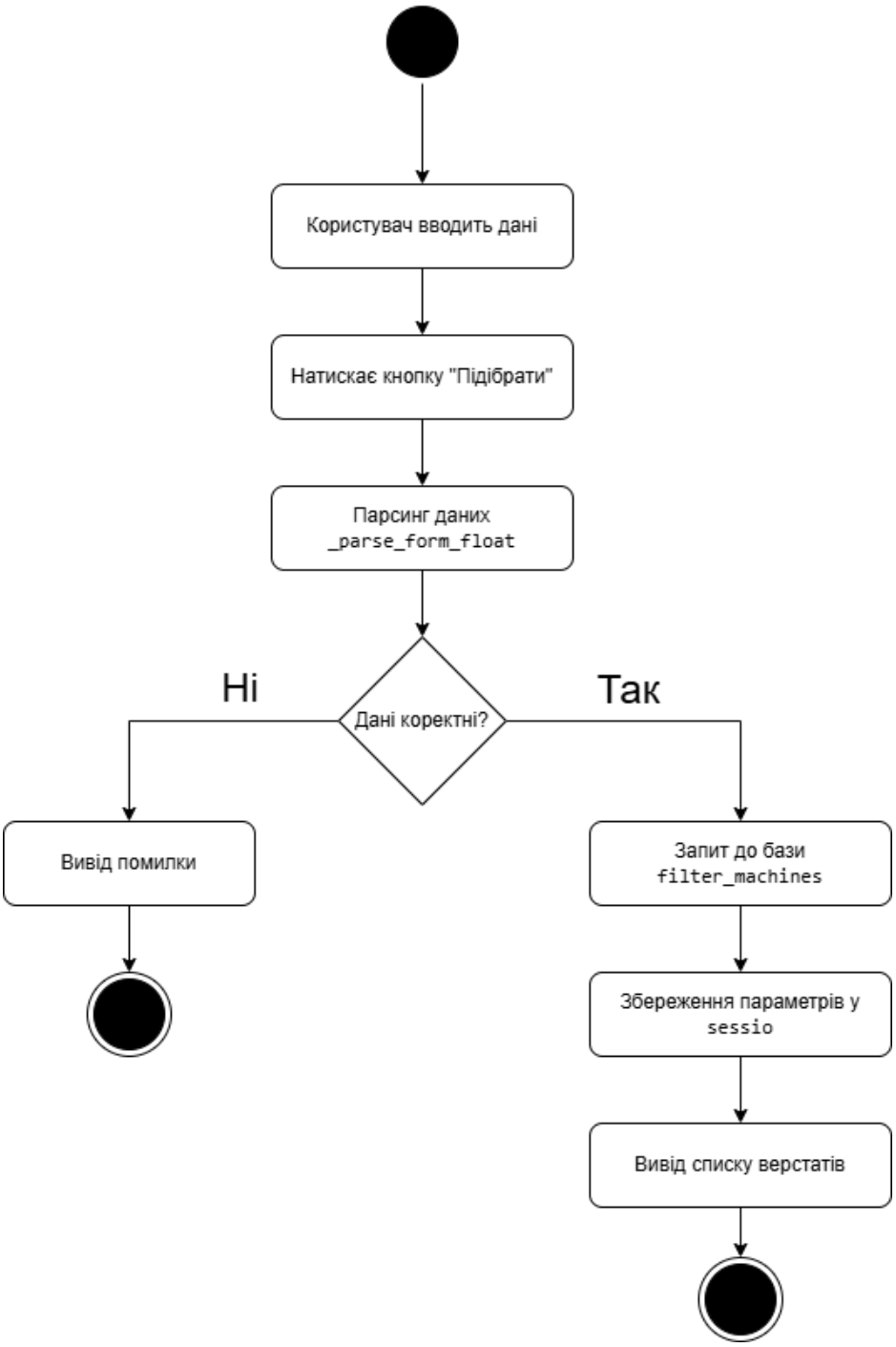

Рисунок 2.1 – UML-діаграма активності процесу обробки запиту користувача

Після успішного парсингу алгоритм виконує багаторівневу валідацію бізнес-логіки запиту, яка реалізована як послідовний комплекс взаємопов’язаних перевірок. Спочатку система здійснює верифікацію типу обробки, звіряючи введений користувачем параметр із дозволеною множиною константних значень обладнання, що включає фрезерний, лазерний та плазмовий класи. На наступному етапі контроль логічної цілісності даних

блокує виконання розрахунків, якщо обов'язкові лінійні габарити по осях *X* і *Y* або фінансовий бюджет відсутні в запиті чи набувають недопустимого від'ємного значення. Далі відбувається комплексна перевірка матеріалів, під час якої отриманий масив обраних користувачем позицій зіставляється з глобальним словником ієрархічних груп складності MATERIAL_GROUPS_BY_TYPE для динамічного обчислення граничного технологічного рівня верстата. Фінальним етапом валідації є умовний аналіз просторової осі *Z*, специфіка якого залежить від обраного класу інструменту: для фрезерних і плазмових установок система суворо вимагає введення додатного значення висоти ходу порталу, тоді як для лазерного обладнання цей параметр автоматично та програмно ігнорується як нерелевантний для площинного розкрою.

У разі виявлення порушення хоча б однієї з перелічених умов процес фільтрації переривається, а користувачу повертається відповідне повідомлення про помилку (error), що дозволяє оперативно скоригувати введені дані без перезавантаження всієї форми. Якщо всі перевірки пройдено успішно, сервер викликає головну функцію фільтрації filter_machines, передаючи їй валідовані параметри. Сформована вибірка ТОП-3 моделей верстатів передається у HTML-шаблон для рендерингу результатів підбору.

Важливим архітектурним рішенням є впровадження механізму керування станом (State Management). Оскільки протокол HTTP за своєю природою є stateless і не зберігає стан між окремими запитами, для збереження введених параметрів між перезавантаженнями сторінок використовується серверний об'єкт session. Функція _save_search_form_to_session автоматично записує поточні параметри пошуку в зашифровану сесію користувача після кожного успішного запиту за такими ключами: search_machine_type, search_materials (масив обраних матеріалів), search_work_x, search_work_y, search_work_z та search_budget. Завдяки цьому при поверненні користувача на головну сторінку форма автоматично заповнюється збереженими даними через виклик session.get(), що суттєво покращує користувацький досвід і

позбавляє необхідності повторного введення параметрів.

2.5 Об’єктно-орієнтоване моделювання програмної системи

Ефективна програмна реалізація раніше формалізованої математичної моделі багатокритеріального вибору потребує попереднього об’єктно-орієнтованого аналізу та моделювання архітектури застосунку. Використання уніфікованої мови моделювання UML (Unified Modeling Language) дозволяє створити концептуальний міст між абстрактними математичними фільтрами та конкретними програмними компонентами в межах обраного архітектурного патерну MVC. На початковому етапі проєктування життєвого циклу системи розробляється модель прецедентів, загальну графічну структуру якої представлено на рисунку 2.2. Зазначена модель візуалізує функціональні можливості вебзастосунку з точки зору кінцевого користувача. Оскільки в системі «CNC MARKET» визначено єдину ключову роль актора, а саме Гостя або Клієнта-інженера, модель прецедентів фокусується на його взаємодії з інформаційними та обчислювальними модулями програми.

Головний прецедент використання описує процес інтерактивного підбору верстата з ЧПК, який ініціюється актором через заповнення параметричної форми на головній сторінці сайту. Цей процес безпосередньо пов’язаний із прецедентами введення лінійних габарити заготовки, фінансового бюджету та множинного вибору конструкційних матеріалів. Окремим важливим функціональним вектором є прецедент перегляду повного каталогу обладнання, який розширює можливості взаємодії з системою, дозволяючи користувачеві ігнорувати жорсткі фільтри конфігуратора та переходити до вільного моніторингу ринку верстатів. Цей прецедент додатково включає можливість інтерактивного перемикання між технологічними вкладками фрезерного, лазерного та плазмового обладнання з автоматичним ранжуванням усіх наявних моделей за критерієм вартості.

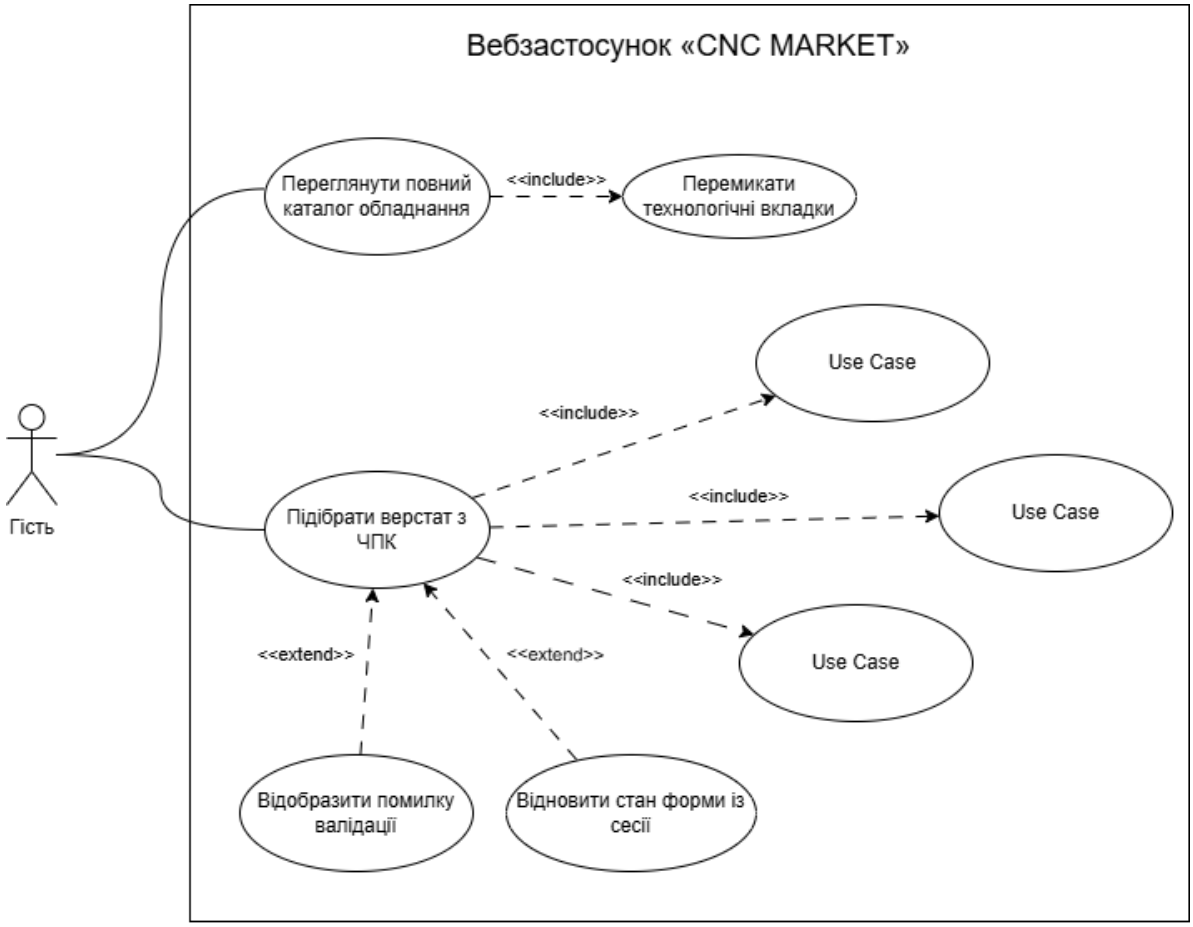

Рисунок 2.2 – UML-діаграма прецедентів використання системи «CNC MARKET»

Окрім позитивних сценаріїв, модель прецедентів передбачає обробку виняткових ситуацій, зокрема прецедент відображення повідомлень про помилки валідації, який активується сервером у разі введення недопустимих значень, та прецедент автоматичного відновлення стану форми із сесії, що оптимізує повторні звернення користувача до обчислювального модуля.

Для детального аналізу динаміки функціонування системи та опису часової послідовності обміну повідомленнями між компонентами застосунку розробляється модель послідовності, часову схему якої наведено на рисунку 2.3. Вона дозволяє покроково простежити життєвий цикл виконання ключового бізнес-сценарію – від моменту натискання користувачем кнопки ініціації пошуку до моменту фінального відображення результатів у браузері.

У цьому процесі беруть участь п’ять основних об’єктів системи: користувач у ролі актора, графічний інтерфейс клієнтської частини (Browser View), головний контролер Flask (app.py), внутрішній серверний об’єкт сесії (Session) та реляційна база даних (SQLite Database), взаємодія з якою ізольована через рівень ORM SQLAlchemy.

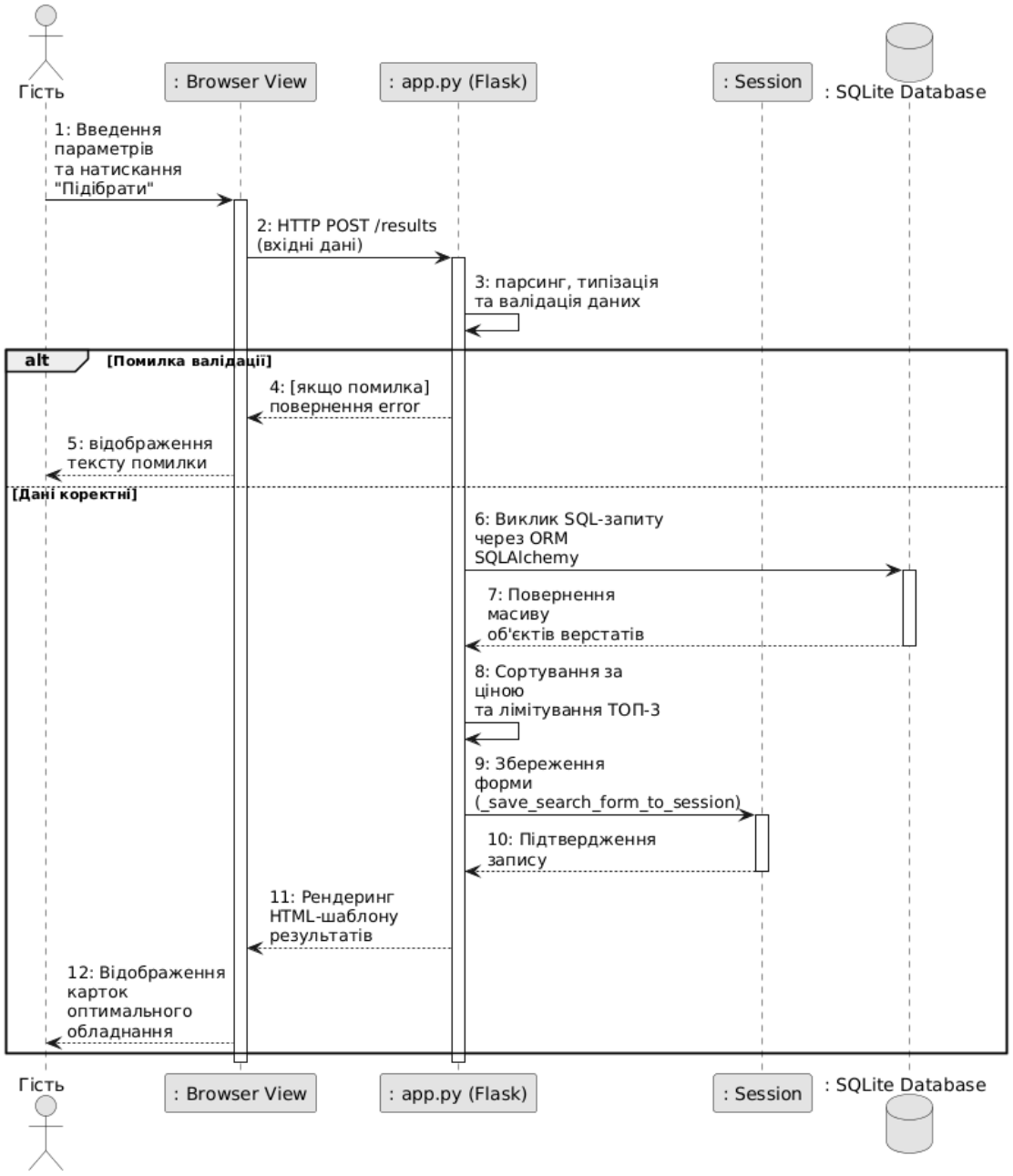

Рисунок 2.3 – UML-діаграма послідовності процесу багатокритеріального підбору обладнання

Процес починається з того, що актор заповнює поля форми та надсилає HTTP POST-запит через інтерфейсний об’єкт до контролера Flask на кінцеву точку /results. Отримавши вхідний вектор параметрів, об’єкт контролера тимчасово призупиняє лінійний хід програми для виконання внутрішніх методів парсингу та верифікації даних. Контролер послідовно звертається до внутрішнього логічного шару валідації, перевіряючи коректність числових типів та відповідність обраних матеріалів глобальному словнику груп складності. Якщо на цьому етапі виявляється логічна помилка, наприклад від’ємне значення бюджету, контролер миттєво генерує зворотне повідомлення з описом винятку та передає його об’єкту інтерфейсу для рендерингу текстової помилки користувачеві, завершуючи поточну ітерацію.

У разі успішного проходження всіх валідаційних перевірок, об’єкт контролера Flask ініціює наступну фазу взаємодії, надсилаючи структурований SQL-запит до об’єкта бази даних SQLite через ORM-шаблон. База даних виконує внутрішню фільтрацію записів таблиці за геометричними, економічними та ієрархічно-матеріальними критеріями, після чого повертає контролеру масив відфільтрованих об’єктів верстатів, відсортованих за зростанням ціни. Отримавши ці дані, контролер Flask виконує операцію обмеження вибірки, залишаючи лише перші три найбільш оптимальні моделі.

Потім, для забезпечення збереження стану інтерфейсу, контролер передає сформований масив параметрів пошуку об’єкту серверної сесії, який записує ці дані у зашифрований cookie-файл клієнта. На завершальному етапі об’єкт контролера передає фінальну ТОП-3 вибірку верстатів у HTML-шаблон результатів, викликає метод рендерингу та повертає сформовану вебсторінку клієнтському браузеру, який візуалізує готові картки обладнання для кінчевого користувача.

3 РОЗРОБЛЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ ВЕБЗАСТОСУНКУ

3.1 Обґрунтування вибору інструментальних засобів розроблення

Створення ефективного, масштабованого та надійного програмного продукту вимагає ретельного попереднього аналізу та обґрунтованого вибору стека технологій. У контексті розроблення вебзастосунку «CNC MARKET», архітектура якого передбачає виконання багатокритеріальних математичних обчислень та ієрархічної фільтрації даних у режимі реального часу, базовою мовою програмування серверної частини було обрано Python [23, 24]. Цей вибір зумовлений високим рівнем абстракції мови, її строгою динамічною типізацією та наявністю потужної стандартної бібліотеки, що дозволяє лаконічно реалізовувати складні алгоритмічні структури. На відміну від мов програмування з жорсткою компіляцією, Python забезпечує високу швидкість прототипування та написання чистого, легко підтримуваного коду, що є критично важливим на етапі розроблення мінімально життєздатного продукту (MVP).

Для побудови архітектури бекенду та реалізації патерну Model-View-Controller (MVC) було застосовано вебфреймворк Flask. Вибір на користь цього мікрофреймворку, на противагу монолітним рішенням на кшталт Django, обґрунтовується його легковаговістю, високою продуктивністю та модульністю. Flask не нав'язує розробнику надлишкового вбудованого інструментарію, залишаючи повний контроль над архітектурними рішеннями та маршрутизацією. Це дозволило побудувати оптимальну систему обробки кінцевих точок для конфігуратора та каталогу обладнання без втрати обчислювальних ресурсів сервера на підтримку непотрібних фонових процесів. Гнучкість Flask також забезпечила зручну інтеграцію механізмів криптографічного підписання користувацьких сесій для реалізації концепції керування станом.

Управління персистентними даними реалізовано на базі реляційної системи керування базами даних SQLite. Зважаючи на те, що застосунок «CNC MARKET» оперує структурованим, але відносно статичним каталогом моделей промислового обладнання, використання безсерверної БД є найбільш раціональним інженерним рішенням. SQLite зберігає всю базу даних у єдиному кросплатформеному файлі на стороні хоста, що повністю усуває необхідність розгортання, конфігурування та адміністрування окремого сервера БД (наприклад, MySQL чи PostgreSQL) на ранніх етапах розгортання проєкту. Для безпечної та об'єктно-орієнтованої взаємодії з базою даних застосовано бібліотеку SQLAlchemy, яка виконує роль технології об'єктно-реляційного відображення. Використання SQLAlchemy дозволило абстрагуватися від написання сирих SQL-запитів, трансформувавши таблиці бази даних у класи мови Python. Такий підхід не лише гарантує надійний захист застосунку від поширених вразливостей, зокрема SQL-ін'єкцій, шляхом автоматичного екранування вхідних параметрів, але й забезпечує високу портативність: у разі майбутнього масштабування проєкту та переходу на промислову клієнт-серверну СУБД, бізнес-логіка фільтрації не потребуватиме переписування коду.

Проєктування клієнтської частини вебзастосунку базується на використанні сучасних стандартів HTML5 та CSS3 у поєднанні з компонентним фреймворком Bootstrap 5 [25]. Вибір п'ятої версії цього фреймворку аргументується повною відмовою від застарілих залежностей, таких як бібліотека jQuery, що суттєво зменшило розмір завантажуваних скриптів та підвищило загальну швидкість рендерингу сторінок у браузері користувача. Завдяки потужній дванадцятиколонковій системі сіток та концепції Mobile-First, Bootstrap 5 дозволив реалізувати повністю адаптивний інтерфейс, який гарантує коректне відображення карток обладнання, складних навігаційних панелей та інтерактивних форм як на широкоформатних інженерних моніторах, так і на екранах мобільних пристроїв. Для

забезпечення інтерактивності інтерфейсу, зокрема роботи динамічного повзунка цінового діапазону та обробки логіки приховування нерелевантних полів вводу без перезавантаження сторінки, застосовано мову сценаріїв JavaScript (стандарт ES6). Сукупність обраних програмних рішень формує надійний, швидкий та масштабований технологічний стек, що повністю відповідає поставленим завданням та сучасним вимогам вебпроєктування.

Інтеграція всіх вищезазначених засобів формує цілісну трирівневу архітектуру системи. Клієнтський рівень (Frontend) відповідає за збір параметрів від користувача та відображення адаптивного інтерфейсу. Зв'язок із серверним рівнем здійснюється за допомогою протоколу HTTP. Серверний рівень (Backend), реалізований на Python з використанням Flask, виконує роль головного контролера: він приймає запити, проводить алгоритмічну валідацію та формує вибірки. Рівень даних представлений СУБД SQLite, до якої сервер звертається виключно через абстракцію SQLAlchemy ORM. Схематично архітектуру взаємодії обраного стека технологій вебзастосунку представлено на рисунку 3.1.

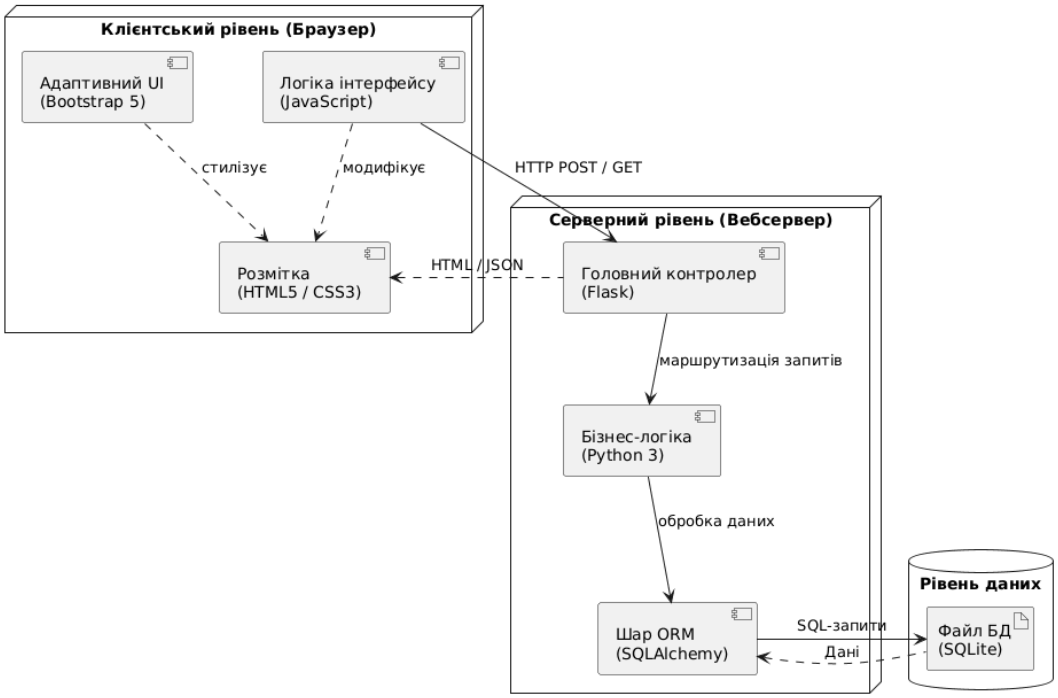

Рисунок 3.1 – Схема взаємодії інструментальних засобів вебзастосунку

3.2 Проєктування та реалізація користувацького інтерфейсу (UI/UX)

Ефективність будь-якого програмного продукту, орієнтованого на інженерне середовище, значною мірою залежить від якості проєктування користувацького інтерфейсу та загального користувацького досвіду [26]. Зважаючи на специфіку предметної області вебзастосунку «CNC MARKET» – промислове обладнання з числовим програмним керуванням та металообробка – візуальну концепцію проєкту було побудовано на базі сучасного індустріального стилю. Головним архітектурним рішенням у дизайні стало використання темної теми як базової колірної палітри інтерфейсу. Використання глибоких відтінків сірого та чорного кольорів, зокрема фонового кольору #121212 для сторінки та кольору #1e1e1e для контурів карток і блоків, дозволяє суттєво знизити навантаження на зоровій апарат інженера під час тривалого моніторингу специфікацій верстатів, а також створює стійкий асоціативний зв'язок із професійними інженерними САПР-системами.

Для забезпечення високого рівня контрастності, читабельності та ефективного керування увагою користувача було впроваджено трирівневу систему акцентних кольорів. Ключові елементи взаємодії, такі як кнопки відправки параметричних форм, активні стани чекбоксів та рухомі елементи повзунків бюджету, виділені яскравим індустріальним оранжевим кольором. Цей колір традиційно використовується у реальному верстатобудуванні для маркування активних та інструментальних вузлів обладнання, що додає цифровому інтерфейсу автентичності. Другорядні інформаційні блоки та текстові підписи технічних одиниць виміру (міліметри, кіловати, гривні) виконані у приглушених світло-сірих та білих тонах, що унеможливлює візуальне перевантаження сторінки.

Велику увагу приділено інженерній типографіці вебзастосунку. Для відображення всього текстового та числового контенту підключено сімейство шрифтів IBM Plex Sans з бібліотеки Google Fonts. Цей шрифт має чіткі,

акцентовані геометричні пропорції, що ідеально підходять для відображення складних технічних характеристик, маркувань моделей верстатів та числових лінійних габаритів. Головна сторінка застосунку зустрічає користувача лаконічною та жорстко зафіксованою навігаційною панеллю, яка містить у лівій частині текстовий фірмовий логотип проєкту, а у правій – інтерактивні елементи швидкого переходу між головним обчислювальним конфігуратором та модулем повного інформаційного каталогу обладнання. Візуальне оформлення початкового стану системи представлено на рисунку 3.2.

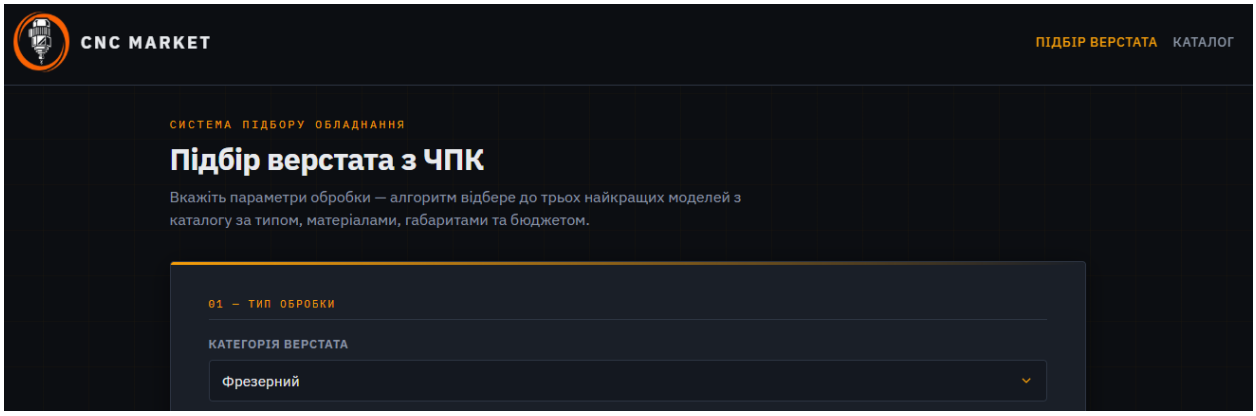

Рисунок 3.2 – Візуальне оформлення головної сторінки конфігуратора

Центральним елементом взаємодії Клієнта-інженера з обчислювальним ядром є параметрична форма введення вимог. Процес збору даних диференційовано за типами елементів керування для мінімізації помилок введення. Вибір базового класу обробки, що включає фрезерний, лазерний та плазмовий типи, реалізовано через випадаючий список. Введення лінійних розмірів майбутньої заготовки здійснюється через числові поля для осей X, Y та Z. На рівні HTML-розмітки ці поля мають атрибути обмеження мінімального значення, що дозволяє відсікати некоректні від'ємні параметри ще до моменту передачі запиту на серверний рівень.

Особливістю реалізованого інтерфейсу є блок вибору конструкційних матеріалів та вказівки фінансового бюджету, графічне представлення яких наведено на рисунку 3.3. Оскільки один і той самий верстат може бути

придатним для обробки кількох типів сировини, вибір матеріалів реалізовано у вигляді гнучкої сітки незалежних прапорців-чекбоксів, візуально згрупованих за технологічною складністю. Для введення фінансових обмежень розроблено інтерактивний динамічний повзунок. За допомогою клієнтських обробників подій мови JavaScript, переміщення цього повзунка користувачем миттєво оновлює текстове відображення обраної суми у гривнях на екрані, забезпечуючи миттєвий тактильний відгук інтерфейсу. Важливою UI-функцією є адаптивна поведінка форми: при виборі лазерного типу обробки поле введення висоти ходу по осі Z автоматично приховується за допомогою маніпуляцій з DOM-деревом сторінки, оскільки лазерний розкроювальник працює виключно у двовимірній площині.

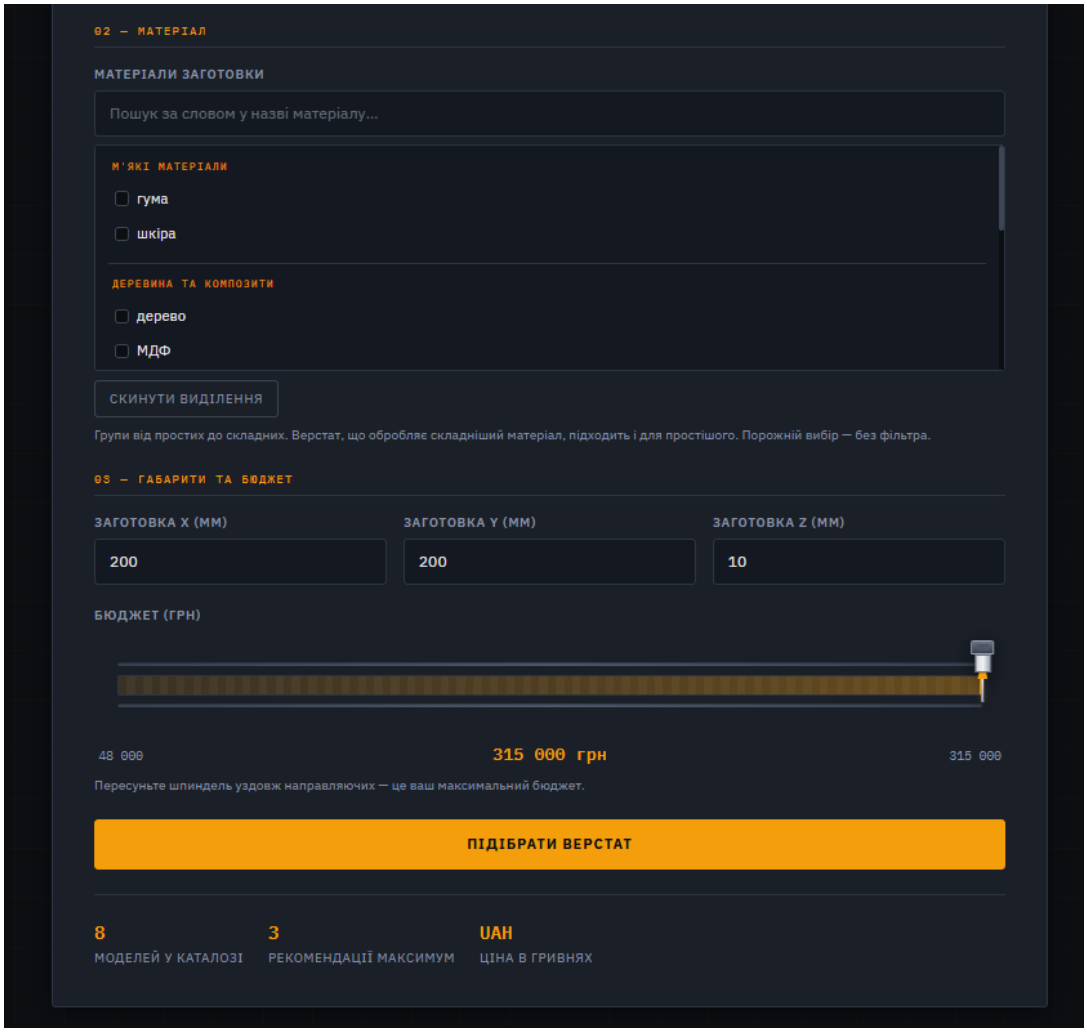

Рисунок 3.3 – Інтерфейс вибору матеріалів та цінового діапазону

Після натискання актором кнопки ініціації розрахунку та успішного виконання серверного алгоритму фільтрації, застосунок перенаправляє користувача на сторінку результатів підбору. Інтерфейс цієї сторінки спроєктовано за принципом максимальної інформативності та містить трирівневу горизонтальну або вертикальну сітку карток, яка обмежена трьома найбільш оптимальними моделями верстатів. Кожна картка верстата є ізольованим графічним компонентом, що містить блок візуалізації (фотографію промислового обладнання або графічну заглушку у разі відсутності зображення), чітку назву моделі, вказаний тип технологічного процесу, а також компактну таблицю технічних характеристик [27]. У нижній частині картки розташовано контрастний кольоровий бейдж із актуальною вартістю верстата у національній валюті. Приклад візуалізації згенерованих результатів підбору наведено на рисунку 3.4.

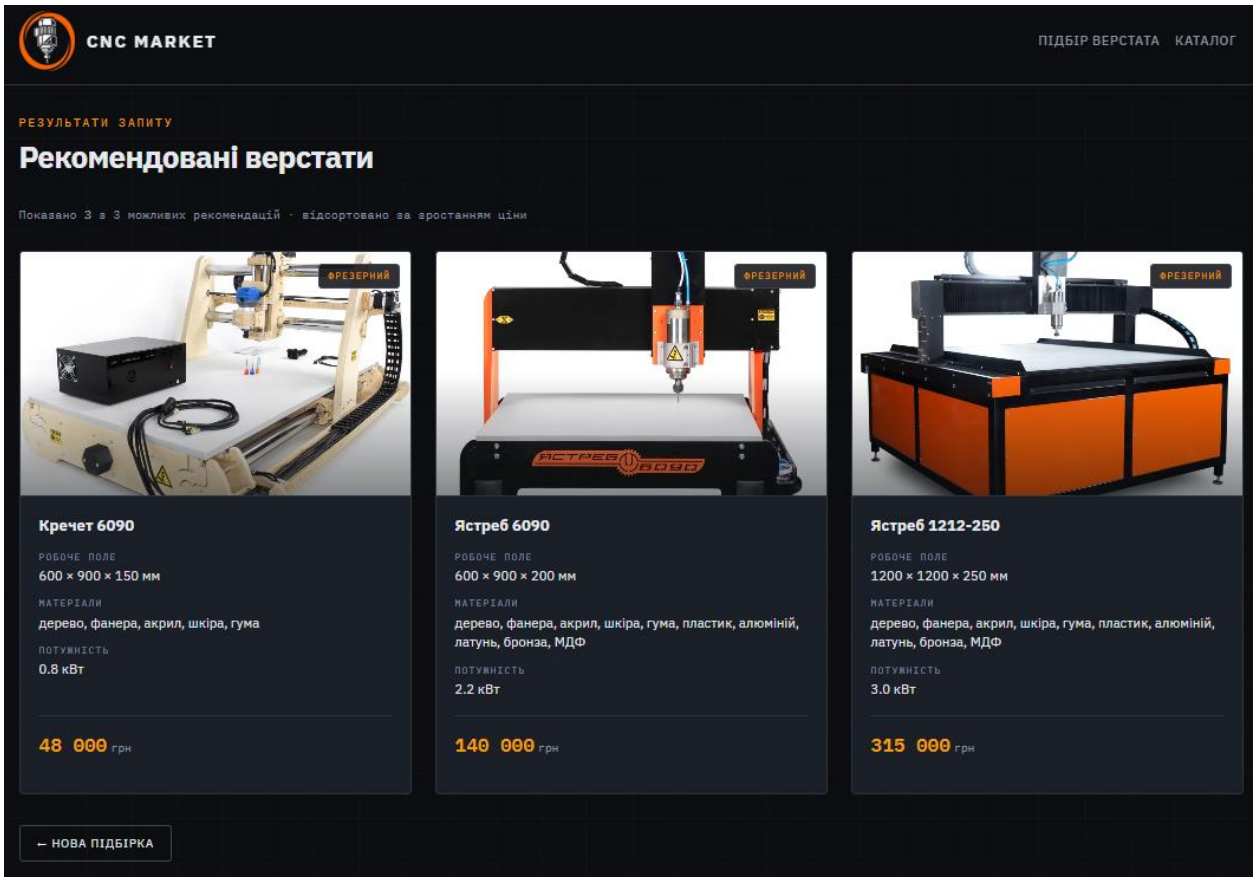

Рисунок 3.4 – Інтерфейс відображення результатів алгоритмічного підбору

Окремим функціональним екраном системи є сторінка повного каталогу обладнання, доступна через головне навігаційне меню. Візуальний стиль каталогу повторює загальну індустріальну концепцію, проте має іншу структуру розмітки сторінки. Замість обмеженої вибірки, тут реалізовано повноцінну інформаційну систему, де всі наявні в базі даних верстати відображаються без обмежень фільтрації. Для зручності навігації у верхній частині каталогу реалізовано інтерактивні вкладки Bootstrap 5, які дозволяють користувачеві в один клік відфільтрувати картки верстатів за класами: показати лише фрезерні, лише лазерні або лише плазмові машини. Ранжування моделей у каталозі автоматично налаштоване за критерієм зростання вартості, що дозволяє швидко проводити загальний моніторинг ринку обладнання. Візуальну структуру модуля повного каталогу представлено на рисунку 3.5.

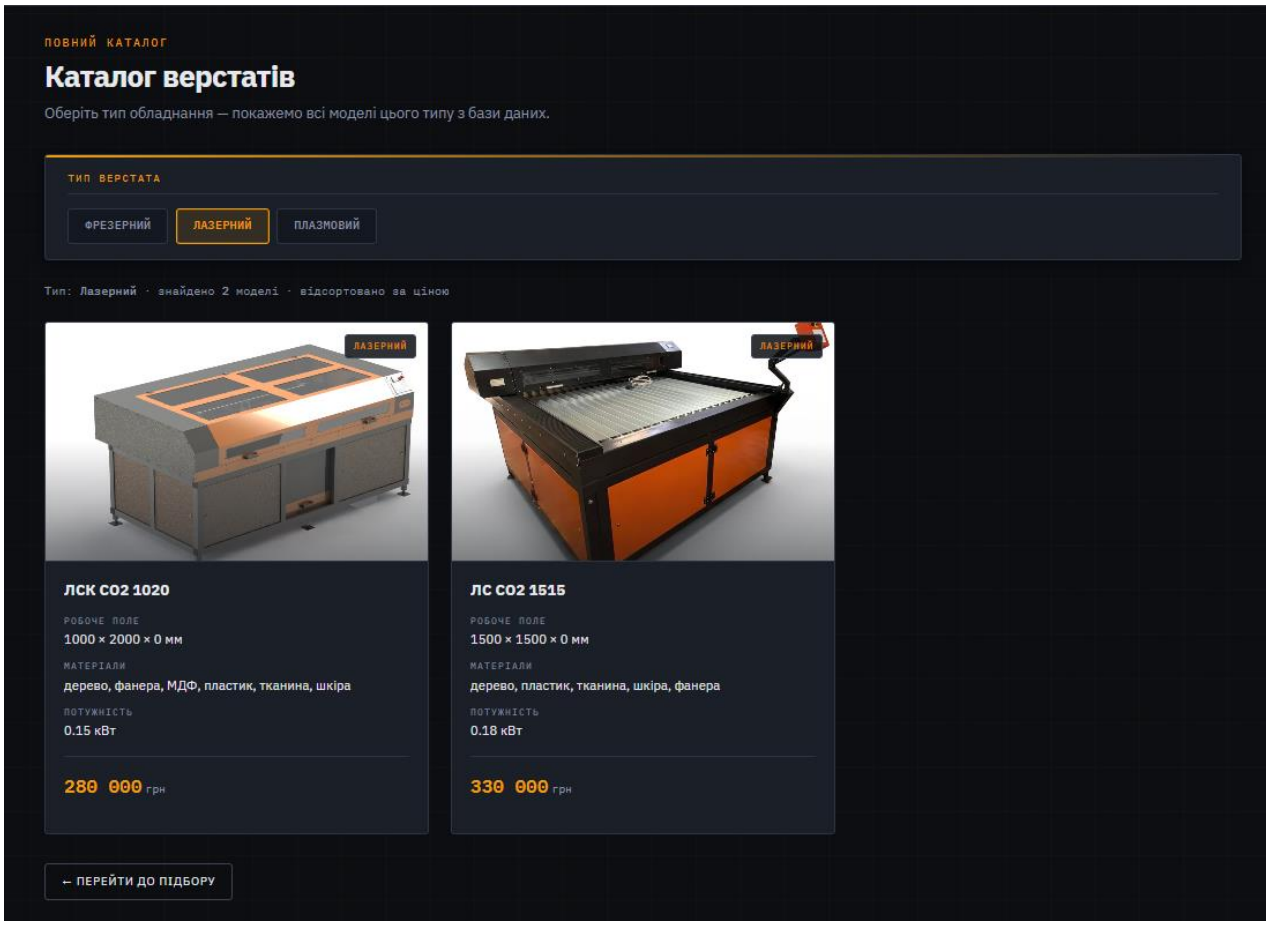

Рисунок 3.5 – Візуальна структура модуля повного каталогу обладнання

Завдяки використанню адаптивної дванадцятиколонкової сітки фреймворку Bootstrap 5, всі розроблені інтерфейсні рішення є повністю чуйними. Пристрій користувача динамічно змінює геометрію блоків: на широкоформатних моніторах форми та результати відображаються у кілька колонок із великими відступами, тоді як на екранах мобільних телефонів та планшетів елементи автоматично шикуються в одну вертикальну лінію, адаптуючи розміри кнопок та шрифтів для зручного натискання пальцем. Це гарантує стабільний та комфортний UX-досвід при експлуатації вебзастосунку за будь-яких умов виробництва.

Для повноцінного досягнення унікального індустріального стилю (Industrial Design) лише стандартних компонентів фреймворку Bootstrap 5 виявилося недостатньо. Тому в архітектуру клієнтської частини було інтегровано кастомний файл каскадних таблиць стилів (CSS), який перевизначає стандартні змінні та додає глибоку темну палітру з помаранчевими акцентами. Особливу увагу було приділено стилізації динамічного повзунка вибору бюджету (Range Slider), який є ключовим елементом інтерактивної взаємодії.

Лістинг 3.1 програмного коду кастомних CSS-правил для інтерфейсу:

```
/* Базові налаштування темної індустріальної теми */
body {
    background-color: #121212;
    color: #e0e0e0;
    font-family: 'IBM Plex Sans', sans-serif;
    -webkit-font-smoothing: antialiased;
}

/* Стилізація карток обладнання */
.card {
    background-color: #1e1e1e;
```

```
    border: 1px solid #333;
    transition: transform 0.2s ease, box-shadow 0.2s ease;
}

/* Ефект наведення курсора (hover) */
.card:hover {
    transform: translateY(-5px);
    box-shadow: 0 8px 15px rgba(253, 126, 20, 0.15); /* Помаранчеве світіння */
}

/* Головна акцентна кнопка відправки форми */
.btn-industrial {
    background-color: #fd7e14;
    color: #121212;
    font-weight: 700;
    text-transform: uppercase;
    border: none;
}
```

3.3 Програмна реалізація логіки конфігуратора та каталогу

Центральним обчислювальним компонентом розробленого вебзастосунку є модуль алгоритмічної фільтрації, реалізований на стороні сервера за допомогою мови програмування Python. Архітектурно цей модуль відповідає за прийом даних від клієнтського інтерфейсу, їхню типізацію, безпечну обробку та формування фінальної вибірки обладнання. Ключовим інженерним рішенням у програмній реалізації стала імплементація математичної моделі ієрархії матеріалів [28, 29]. Замість створення

надлишкових зв'язків у реляційній базі даних, структуру технологічної складності обробки було винесено на рівень бізнес-логіки у вигляді типізованого глобального словника MATERIAL_GROUPS_BY_TYPE.

Ця структура даних зберігає впорядковані масиви об'єктів для кожного типу обладнання. Кожен об'єкт містить текстовий ярлик (ключ label), який використовується для зручного групування чекбоксів у клієнтському інтерфейсі, та безпосередньо множину матеріалів (ключ materials), відсортовану за правилами української локалі за допомогою допоміжної функції _sort_ua. Програмна реалізація цієї константної структури наведено у лістингу 3.2.

Лістинг 3.2 Програмна реалізація константної структури:

```
MATERIAL_GROUPS_BY_TYPE: dict[str, list[dict[str, str | list[str]]]] = {
    'Фрезерний': [
        {
            'label': 'М\'які матеріали',
            'materials': _sort_ua({'гума', 'шкіра'}),
        },
        {
            'label': 'Деревина та композити',
            'materials': _sort_ua({'дерево', 'фанера', 'МДФ'}),
        },
        {
            'label': 'Полімери',
            'materials': _sort_ua({'акрил', 'пластик'}),
        },
        {
            'label': 'Легкі метали',
            'materials': _sort_ua({'алюміній'}),
        },
        {
            'label': 'Кольорові сплави',
            'materials': _sort_ua({'латунь', 'бронза'}),
        },
    ],
    'Лазерний': [
        {
            'label': 'Текстиль та шкіра',
```

```
            'materials': _sort_ua({'тканина', 'шкіра'}),
        },
        {
            'label': 'Деревина та композити',
            'materials': _sort_ua({'дерево', 'фанера', 'МДФ'}),
        },
        {
            'label': 'Полімери',
            'materials': _sort_ua({'пластик'}),
        },
    ],
    'Плазмовий': [
        {
            'label': 'Листовий прокат',
            'materials': _sort_ua({'листовий метал'}),
        },
        {
            'label': 'Металопрокат та сталь',
            'materials': _sort_ua({'металопрокат', 'сталь'}),
        },
        {
            'label': 'Профілі',
            'materials': _sort_ua({'профілі'}),
        },
        {
            'label': 'Труби (4-осьова обробка)',
            'materials': _sort_ua({'труби'}),
        },
    ],
}
```

Наведений фрагмент коду дозволяє алгоритму динамічно обчислювати максимальний рівень складності для кожного верстата. Якщо клієнт обирає кілька матеріалів у формі конфігуратора, система визначає найвищий індекс складності серед обраних і порівнює його з можливостями кожної моделі з бази даних. Таким чином, верстат, здатний обробляти сталь, автоматично проходить перевірку на придатність до обробки деревини.

Основна логіка багатокритеріального аналізу інкапсульована у функції filter_machines. Ця функція приймає валідовані параметри та формує базовий

SQL-запит до об'єкта Machine через механізми SQLAlchemy ORM. Спочатку до запиту застосовується строгий фільтр за типом обладнання. Після цього алгоритм здійснює перевірку просторово-геометричних параметрів робочої зони. Важливою особливістю програмної реалізації є адаптивна обробка осі Z. Фрагмент коду, що відповідає за геометричну та економічну фільтрацію, наведено нижче:

Наведений фрагмент коду дозволяє алгоритму динамічно працювати з рівнями складності. Основна логіка багатокритеріального аналізу інкапсульована у функції filter_machines. Ця функція приймає суворо типізовані валідовані параметри (габарити req_x, req_y, req_z, бюджет budget та список матеріалів materials) і формує базовий SQL-запит до моделі Machine через механізми SQLAlchemy ORM. Спочатку до запиту застосовується строгий фільтр за типом обладнання, після чого накладаються просторово-геометричні та економічні обмеження. Фрагмент коду, що описує роботу ядра фільтрації, наведено у лістингу 3.3.

Лістинг 3.3 Код який описує роботу ядра фільтрації:

```
def filter_machines(
    machine_type: str,
    materials: list[str],
    req_x: float,
    req_y: float,
    budget: int,
    req_z: float | None,
) -> list[Machine]:
    """Фільтрація каталогу за типом, матеріалами (ієрархія складності), габаритами та бюджетом.
    Якщо materials непорожній – верстат підходить за OR-логікою і «стелею» складності.
    Повертає не більше трьох найдешевших збігів."""
    q = db.session.query(Machine).filter(Machine.type == machine_type)
    q = q.filter(Machine.work_area_x >= req_x, Machine.work_area_y >= req_y)
    q = q.filter(Machine.price_uah <= budget)
    if req_z is not None:
        q = q.filter(Machine.work_area_z >= req_z)
```

```
        rows = q.order_by(Machine.price_uah.asc()).all()

        if materials:
            rows = [
                m for m in rows
                if machine_matches_materials(machine_type, m.material_supported,
materials)
            ]

        return rows[:3]
```

Важливою особливістю програмної реалізації є гнучка адаптивна обробка осі Z. Оскільки для лазерного обладнання висота обробки не є релевантною, парсер клієнтських запитів передає у функцію значення None. Конструкція if req_z is not None дозволяє елегантно ігнорувати цей фільтр, забезпечуючи коректну роботу алгоритму з двовимірним площинним розкроєм.

Після формування SQL-запиту, база даних повертає попередню вибірку, одразу відсортовану за зростанням ціни (order_by(Machine.price_uah.asc())). Якщо користувач обрав матеріали обробки, алгоритм застосовує додатковий програмний фільтр на рівні мови Python. За допомогою генератора списків (list comprehension), кожен верстат із попередньої вибірки проходить перевірку через допоміжну функцію machine_matches_materials, яка реалізує логіку «стелі» складності (OR-логіку). Верстати, що не задовольняють вимогам щодо твердості матеріалів, відкидаються. Оскільки бізнес-вимога передбачає надання користувачеві лаконічної пропозиції, результуючий масив фінально обрізається до трьох елементів (ТОП-3) за допомогою стандартного синтаксису зрізу rows[:3], після чого передається у HTML-шаблон результатів.

Паралельно з логікою конфігуратора реалізовано модуль повного каталогу обладнання. Архітектурна відмінність цього маршруту полягає у повній відсутності жорстких геометричних та цінових фільтрів. Замість багатокритеріального аналізу, сервер отримує запит до бази даних лише за

поточним типом обладнання. Такий підхід забезпечує миттєве вивантаження всього наявного обладнання певного класу, що задовольняє потреби досвідчених технологів у вільному моніторингу ринку без проходження кроків конфігуратора.

3.4 Тестування програмного забезпечення вебзастосунку

Завершальним етапом життєвого циклу розроблення програмного забезпечення є комплексне тестування, мета якого полягає у перевірці відповідності реалізованого функціоналу початковим вимогам та забезпеченні стабільної роботи системи в умовах різноманітних користувацьких сценаріїв. Для вебзастосунку «CNC MARKET» було обрано стратегію ручного функціонального тестування за методом «чорного ящика. Цей підхід передбачає перевірку поведінки системи виключно через зовнішній інтерфейс, без прямого втручання у внутрішню структуру коду, що імітує реальні дії кінцевого користувача.

Основна увага під час тестування була приділена модулю багатокритеріального конфігуратора, оскільки він виконує ключову обчислювальну функцію і є найбільш вразливим до введення некоректних або логічно суперечливих даних. Для забезпечення максимального покриття тестовими сценаріями було застосовано техніки аналізу граничних значень та розбиття на класи еквівалентності. Це дозволило перевірити реакцію серверної частини на позитивні сценарії (коректні дані, що призводять до успішної вибірки), негативні сценарії (введення від'ємних чисел, порожніх значень або спеціальних символів), а також логіку обробки порожніх результатів пошуку (коли в базі даних відсутнє обладнання, що задовольняє надто жорстким вимогам клієнта). Аналіз даних, наведених у таблиці 3.2, підтверджує високий рівень відмовостійкості серверного контролера. Застосунок успішно перехоплює спроби введення некоректних даних та

замість аварійного роботи вебсервера
(помилки типу 500 Internal Server Error) повертає користувачеві безпечні інформаційні повідомлення. На рисунку 3.6 наведено приклад графічного відображення повідомлення про помилку валідації у разі введення від'ємного фінансового бюджету

Таблиця 3.1 – Результати функціонального тестування модуля конфігуратора

№	**Опис тестового сценарію (Test Case)**	**Вхідні параметри форми**	**Очікуваний результат системи**	**Фактичний результат**	**Статус**
1	Перевірка успішного пошуку фрезерного верстата базового рівня	Тип: Фрезерний; Матеріал: Дерево; X: 500; Y: 500; Z: 100; Бюджет: 300000	Відображення списку з 3-х або менше моделей, що підходять за габаритами та бюджетом	Сформовано вибірку коректних моделей	Успішно
2	Перевірка блокування від'ємного бюджету (Негативний тест)	Тип: Плазмовий; Матеріал: Сталь; X: 1000; Y: 1000; Z: 150; Бюджет: -50000	Переривання запиту. Відображення повідомлення про помилку: «Бюджет не може бути від'ємним»	Рендеринг повідомлення про помилку валідації	Успішно
3	Перевірка блокування від'ємних габаритів заготовки	Тип: Лазерний; Матеріал: Фанера; X: -200; Y: 400; Бюджет: 150000	Переривання запиту. Відображення помилки про недопустимість від'ємних лінійних розмірів	Рендеринг повідомлення про помилку валідації	Успішно

Продовження таблиці 3.1

№	Опис тестового сценарію (Test Case)	Вхідні параметри форми	Очікуваний результат системи	Фактичний результат	Статус
4	Перевірка обробки порожнього результату (відсутність моделей)	Тип: Фрезерний; Матеріал: Чавун; X: 3000; Y: 3000; Z: 500; Бюджет: 1000	Алгоритм відпрацьовує без збоїв. Відображається повідомлення: «На жаль, за вашими критеріями нічого не знайдено»	Інтерфейс коректно відображає відсутність результатів	Успішно
5	Тестування ігнорування осі Z для лазерних установок	Тип: Лазерний; Матеріал: Акрил; X: 400; Y: 400;	Успішна генерація вибірки лазерних верстатів, вісь Z	Сервер коректно відкинув	Успішно
6	Перевірка роботи ієрархії матеріалів (OR-логіка)	Тип: Фрезерний; Матеріали: Сталь + Пластик; X: 600; Y: 600; Z: 150; Бюджет: 500000	У результатах з'являються верстати флагманського рівня, здатні обробляти сталь	Фільтр матеріалів відпрацював коректно	Успішно

Застосунок успішно перехоплює спроби введення некоректних даних та замість аварійного завершення роботи вебсервера (помилки типу 500 Internal Server Error) повертає користувачеві безпечні інформаційні повідомлення. На рисунку 3.6 наведено приклад графічного відображення повідомлення про помилку валідації у разі введення від'ємного фінансового бюджету.

Другим важливим етапом стало тестування користувацького інтерфейсу (UI Testing) та перевірка механізмів керування станом сесій (Session State Management). Оскільки застосунок розроблено з використанням концепції адаптивного дизайну, тестування проводилося на різних роздільних здатностях екрана з використанням інструментів розробника браузера (Chrome DevTools). Результати цього етапу тестування наведено у таблиці 3.2.

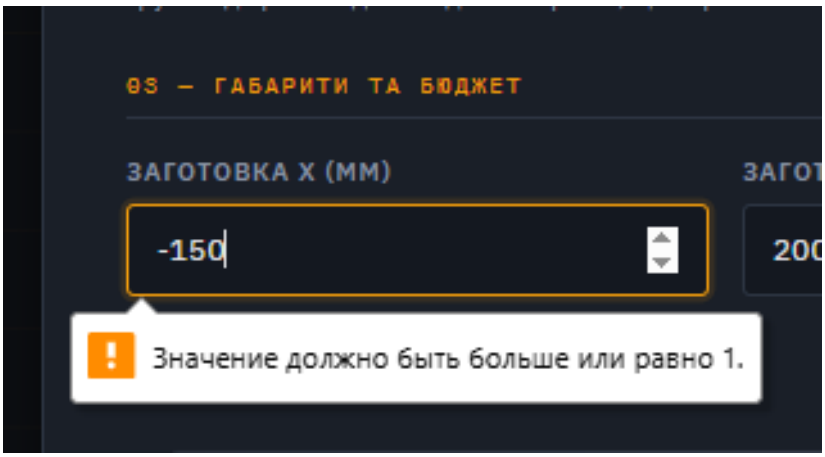

Рисунок 3.6 – Обробка виняткових ситуацій: повідомлення про помилку валідації

Проведене тестування підтверджує, що розроблений вебзастосунок «CNC MARKET» повністю відповідає заявленим функціональним та нефункціональним вимогам. Архітектура програми є стабільною, алгоритми фільтрації працюють математично точно відповідно до закладених ієрархічних моделей, а інтерфейс забезпечує комфортну та безперебійну роботу на пристроях будь-якого типу.

Таблиця 3.2 – Результати тестування UI/UX та механізмів сесії

№	**Опис тестового сценарію (Test Case)**	**Дія користувача в інтерфейсі**	**Очікуваний результат системи**	**Фактичний результат**	**Статус**
1	Перевірка адаптивності сітки конфігуратора на смартфонах	Зміна ширини вікна браузера до 375 пікселів (Viewport мобільного пристрою)	Форма трансформується у вертикальний список. Колонки X, Y, Z шикуються одна під одною	Інтерфейс коректно адаптувався без горизонтальної прокрутки	Успішно
2	Перевірка роботи динамічного повзунка бюджету (JavaScript)	Переміщення повзунка (Range Slider) мишкою або свайпом по екрану	Текстове значення суми в гривнях миттєво оновлюється під час руху повзунка	Подія oninput успішно оновлює DOM-елемент	Успішно

Продовження таблиці 3.2

№	Опис тестового сценарію (Test Case)	Дія користувача в інтерфейсі	Очікуваний результат системи	Фактичний результат	Статус
3	Динамічне приховування поля осі Z для лазера	Перемикання типу обробки в <select> на «Лазерний»	Блок із полем введення висоти Z миттєво зникає з екрана без перезавантаження сторінки	Подія onchange успішно відпрацювала	Успішно
4	Перевірка відновлення стану форми із серверної сесії	Виконання пошуку, перехід на сторінку результатів, а потім натискання кнопки «Назад» або логотипа	Усі раніше введені габарити, обрані чекбокси матеріалів та повзунок бюджету зберігають свої позиції	Об'єкт сесії session.get() успішно відновив дані у шаблоні	Успішно
5	Тестування модуля повного каталогу	Перехід за навігаційним посиланням /catalog	Миттєве вивантаження всього обладнання з бази даних, розділеного на 3 технологічні вкладки	Картки відображаються коректно, сортування за ціною працює	Успішно

3.5 Інструкція з розгортання та налаштування програмного середовища

Для забезпечення коректної роботи розробленого вебзастосунку «CNC MARKET» на локальному сервері або підготовки його до розгортання на промисловому хостингу (Production), необхідно виконати ряд конфігураційних кроків. Оскільки проєкт базується на екосистемі Python, процес встановлення є стандартизованим та кросплатформеним, що дозволяє запускати платформу на операційних системах Windows, macOS або Linux.

Мінімальні апаратні вимоги для локального розгортання сервера включають двоядерний процесор із тактовою частотою від 1.5 ГГц, не менше

2 ГБ оперативної пам'яті (RAM) та 500 МБ вільного простору на жорсткому диску для зберігання файлу бази даних SQLite та кешу графічних асетів. З програмного боку на цільовій машині має бути встановлений інтерпретатор Python версії не нижче 3.10, а також система керування пакунками pip.

Процес розгортання починається з копіювання файлів проєкту у робочу директорію. Архітектура проєкту має чітку ієрархію, де головний скрипт app.py знаходиться в корені, HTML-шаблони розміщені у папці templates, а статичні ресурси (CSS-стилі, клієнтські скрипти, фотографії верстатів) – у директорії static. Загальну ієрархічну структуру директорій розроблюваного вебзастосунку наведено на рисунку 3.7.

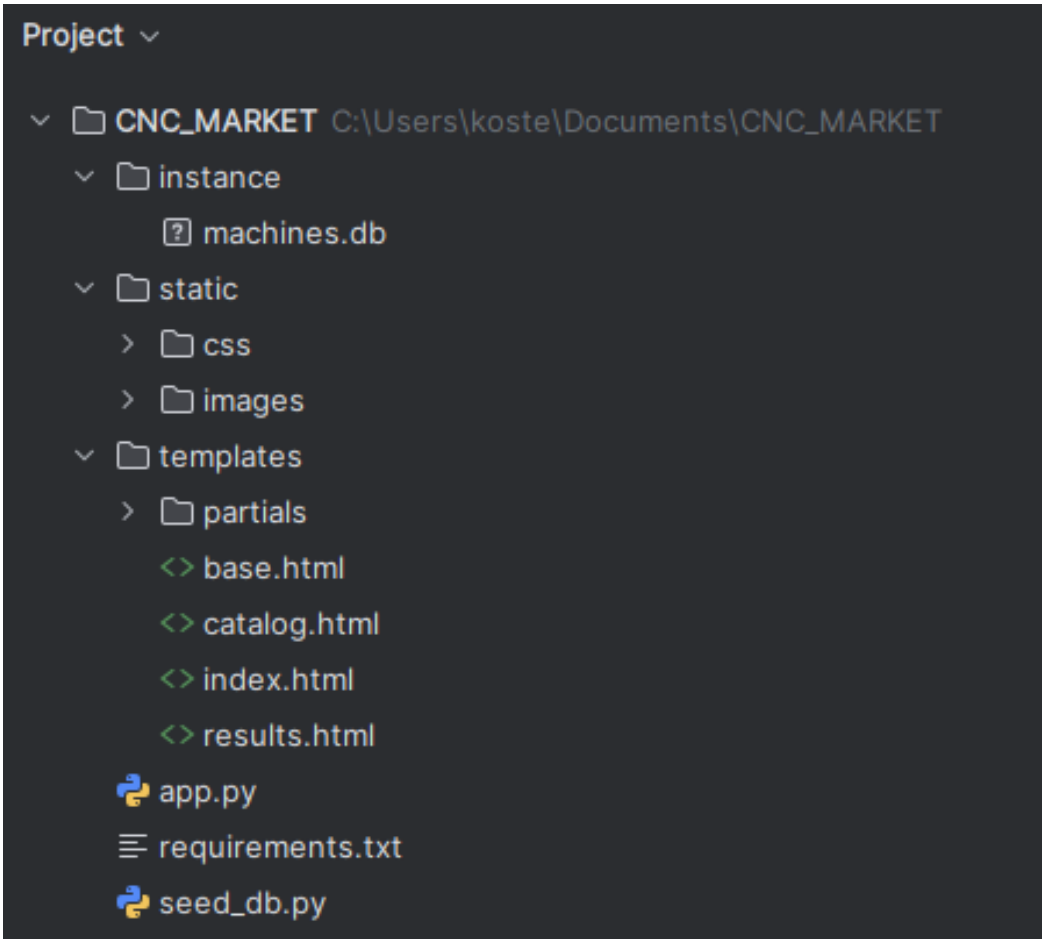

Рисунок 3.7 – Структура директорій вебзастосунку у середовищі розробки

Згідно з найкращими практиками інженерії програмного забезпечення (Software Engineering), запуск застосунку має відбуватися в ізольованому

віртуальному середовищі. Це унеможливлює конфлікти версій бібліотек із системними пакетами. Для ініціалізації віртуального середовища у кореневій теці проєкту через термінал необхідно виконати команду:

python -m venv venv

Після успішного створення директорії середовища, його необхідно активувати. Для операційних систем сімейства Windows використовується команда venv\Scripts\activate. Успішна активація супроводжується появою префікса (venv) у командному рядку термінала.

Наступним критичним кроком є встановлення всіх програмних залежностей, необхідних для роботи мікрофреймворку Flask та шару ORM SQLAlchemy. Усі необхідні пакети та їхні точні версії зафіксовані у конфігураційному файлі requirements.txt. Інсталяція виконується консольною командою pip install -r requirements.txt. Процес успішного встановлення пакетів залежностей через термінал середовища розробки відображено на рисунку 3.8.

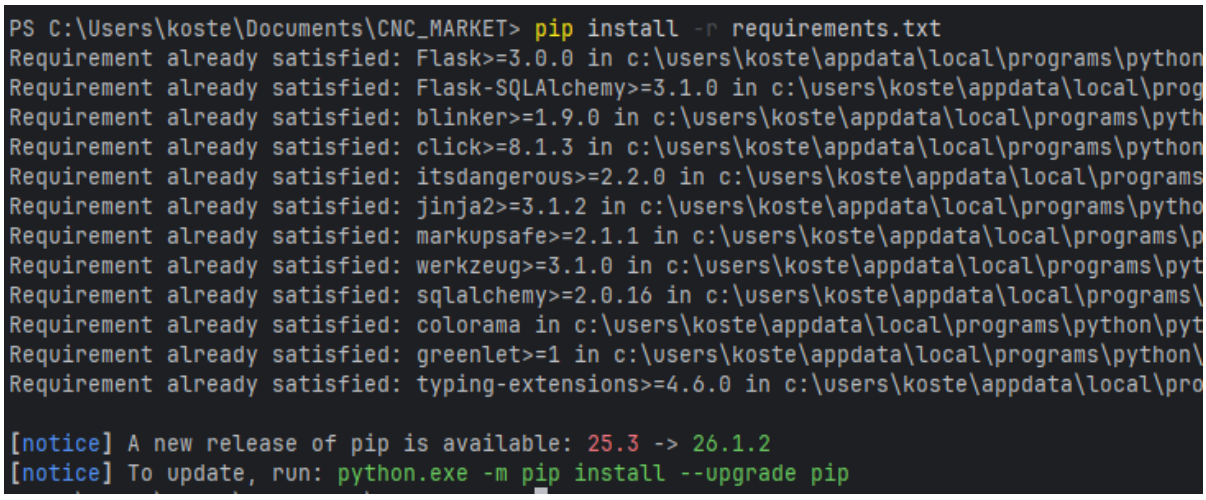

Рисунок 3.8 – Процес встановлення програмних залежностей через менеджер pip

Після розгортання середовища необхідно підготувати рівень даних. Оскільки розроблений застосунок використовує автономну СУБД SQLite, користувачеві не потрібно встановлювати додаткові сервіси баз даних.

Застосунок «CNC MARKET» підтримує автоматичну ініціалізацію схеми: при першому запуску головного файлу app.py контролер перевіряє наявність файлу machines.db у папці instance. Якщо файл відсутній, SQLAlchemy автоматично створює всі необхідні таблиці на основі класів моделей.

Для первинного наповнення каталогу тестовими даними, у структурі проєкту передбачено допоміжний скрипт. Виконання команди python seed_db.py генерує базовий набір промислового обладнання різних класів, що дозволяє негайно розпочати роботу з конфігуратором без необхідності ручного додавання верстатів у базу. Результат успішного додавання показано на рисунку 3.9.

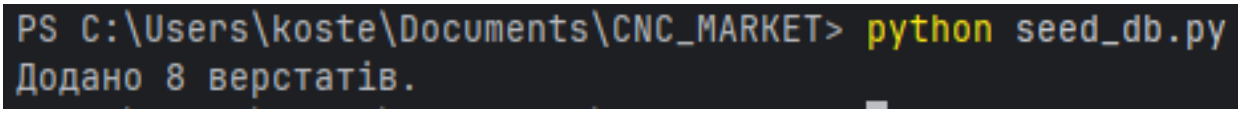

Рисунок 3.9 – Процесс додавання верстатів з бази даних

Фінальним етапом є запуск вбудованого WSGI-сервера для обслуговування HTTP-запитів клієнтів. Запуск ініціюється командою python app.py. Після виконання цієї команди сервер починає прослуховувати локальний порт (за замовчуванням http://127.0.0.1:5000/). Відповідна індикація успішного старту виводиться у консоль, після чого вебзастосунок стає повністю доступним для взаємодії. Індикацію успішного запуску локального вебсервера Flask наведено на рисунку 3.10.

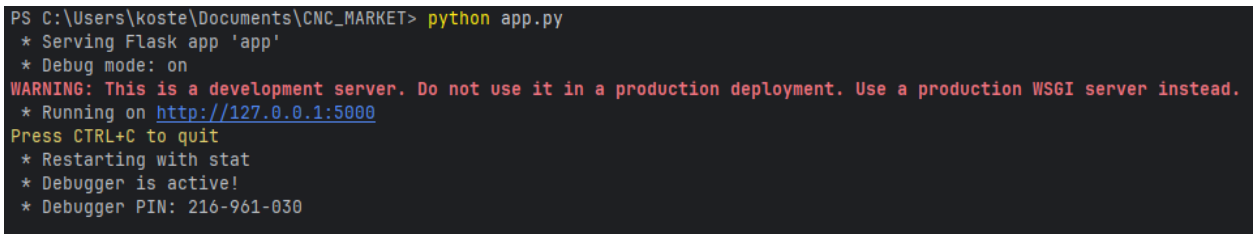

Рисунок 3.10 – Успішний запуск локального вебсервера Flask

3.6 Керівництво користувача та перспективи розвитку системи

Після успішного розгортання та налаштування програмного середовища вебзастосунок «CNC MARKET» повністю готовий до експлуатації кінцевими користувачами. Для забезпечення ефективної взаємодії Клієнта-інженера з системою було розроблено покрокове керівництво користувача, яке деталізує кожен етап роботи з багатокритеріальним конфігуратором. Інтерфейс системи спроєктовано за принципом інтуїтивної зрозумілості, що дозволяє мінімізувати час на адаптацію користувача. Робота з системою розпочинається на головній сторінці, де розміщено форму конфігуратора.

Крок 1. Вибір базового типу обладнання. Процес підбору розпочинається з визначення фундаментального класу верстата. У верхній частині форми користувачеві пропонується випадаючий список (<select>), який містить три доступні технології обробки: фрезерну, лазерну та плазмову. Від цього вибору залежить подальша логіка поведінки інтерфейсу. Наприклад, якщо користувач обирає лазерний тип обладнання, клієнтський скрипт миттєво приховає поле введення висоти заготовки, оскільки для площинного розкрою цей параметр є нерелевантним. Графічний інтерфейс першого кроку взаємодії наведено на рисунку 3.11.

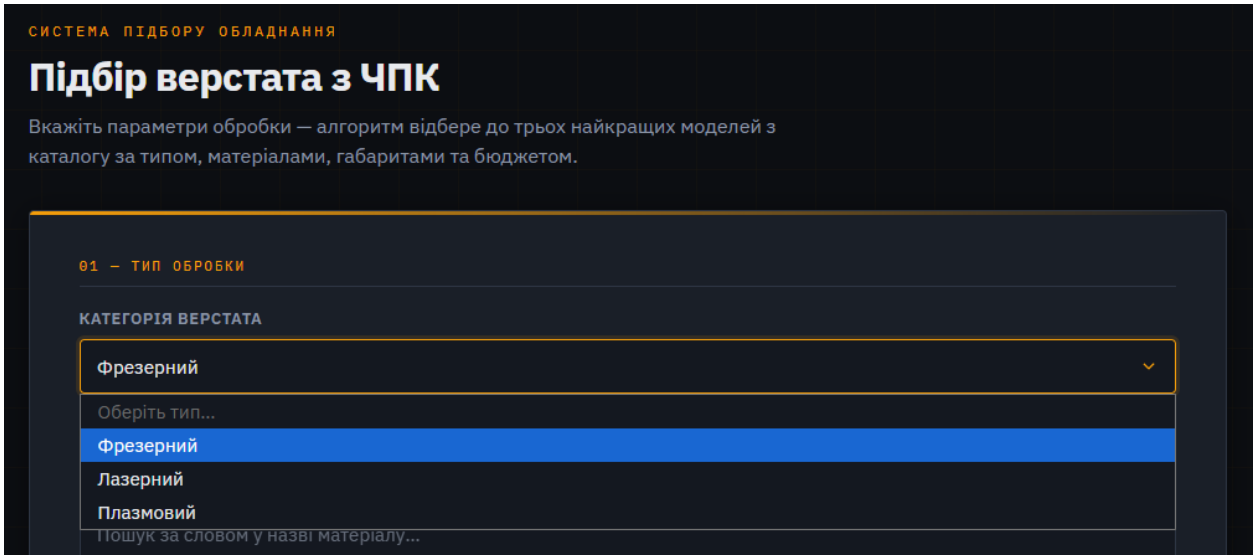

Рисунок 3.11 – Етап вибору базового типу технологічної обробки

Крок 2. Визначення конструкційних матеріалів**.** Наступним логічним етапом є вибір матеріалів, з якими планує працювати підприємство замовника. Інтерфейс цього блоку реалізовано у вигляді сітки незалежних прапорців (чекбоксів), візуально згрупованих за класами складності обробки. Користувач може обрати як один матеріал, так і кілька одночасно. Завдяки ієрархічній математичній моделі бекенду, рекомендується обирати всі цільові матеріали: серверний алгоритм самостійно визначить найскладніший з них і підбере верстат із відповідним запасом міцності та потужності шпинделя. Інтерфейс множинного вибору матеріалів представлено на рисунку 3.12.

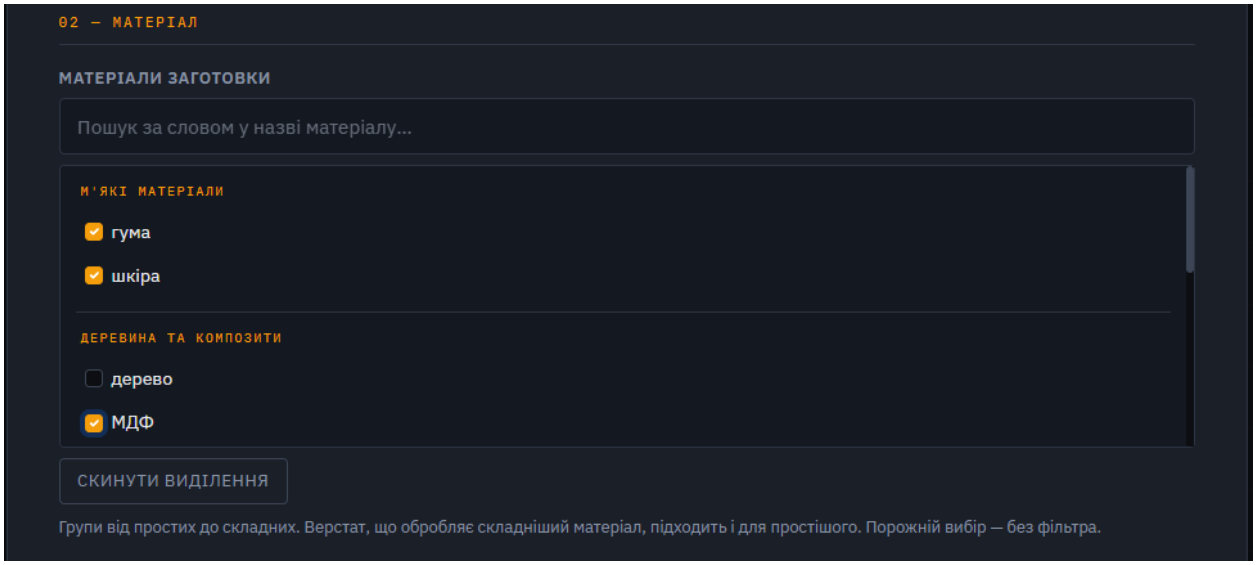

Рисунок 3.12 – Множинний вибір цільових матеріалів обробки

Крок 3. Введення лінійних габаритів заготовки. Після визначення матеріалів клієнт має вказати фізичні розміри майбутньої деталі у міліметрах. Для цього передбачено числовий блок осей координат (X, Y, Z). Користувач повинен ввести ширину, довжину та висоту заготовки за допомогою клавіатури. Важливою особливістю системи є вбудований захист від некоректного введення: HTML-атрибути та клієнтська валідація не дозволять ввести від'ємні значення, блокуючи відправку форми. Процес заповнення геометричних параметрів відображено на рисунку 3.13.

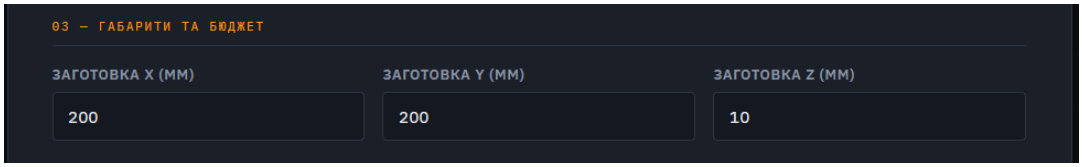

Рисунок 3.13 – Введення геометричних параметрів робочої зони

Крок 4. Налаштування фінансових обмежень та запуск пошуку. На фінальному етапі роботи з формою конфігуратора клієнт повинен встановити граничний фінансовий бюджет за допомогою інтерактивного динамічного повзунка (Range Slider). Під час переміщення повзунка його числове значення миттєво відображається на екрані, що дозволяє точно контролювати ціновий діапазон. Після остаточної перевірки всіх параметрів користувач ініціює обчислювальний процес натисканням акцентної помаранчевої кнопки «Підібрати обладнання». Інтерфейс блоку фінансових обмежень наведено на рисунку 3.14.

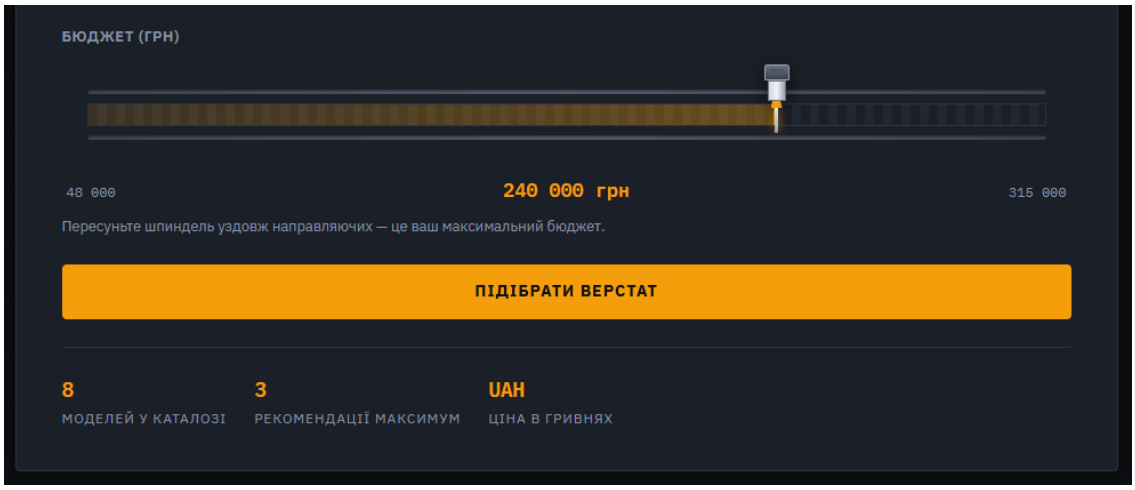

Рисунок 3.14 – Налаштування бюджету та ініціація алгоритмічного пошуку

Крок 5. Аналіз результатів підбору. У разі успішної обробки запиту користувач перенаправляється на сторінку результатів. Система генерує компактну вибірку з трьох найбільш оптимальних моделей верстатів, які задовольняють вимоги замовника. Користувач має змогу ознайомитися з фотографіями обладнання, технічними характеристиками та ціною. Завдяки

механізму збереження сесій, при поверненні на головну сторінку всі введені раніше параметри автоматично відновлюються, що дозволяє легко змінити критерії без повторного введення. Вигляд сформованих результатів підбору проілюстровано на рисунку 3.15.

Крок 6. Робота з модулем повного каталогу**.** Альтернативним сценарієм є використання модуля вільного каталогу. Для доступу до нього користувачу необхідно натиснути посилання «Каталог» у верхній навігаційній панелі. Цей режим дозволяє вільно моніторити асортимент без проходження кроків конфігуратора. Перемикаючись між технологічними вкладками (Фрезерні, Лазерні, Плазмові), користувач отримує повний перелік обладнання, автоматично відсортований за критерієм зростання вартості. Інтерфейс модуля каталогу представлено на рисунку 3.16.

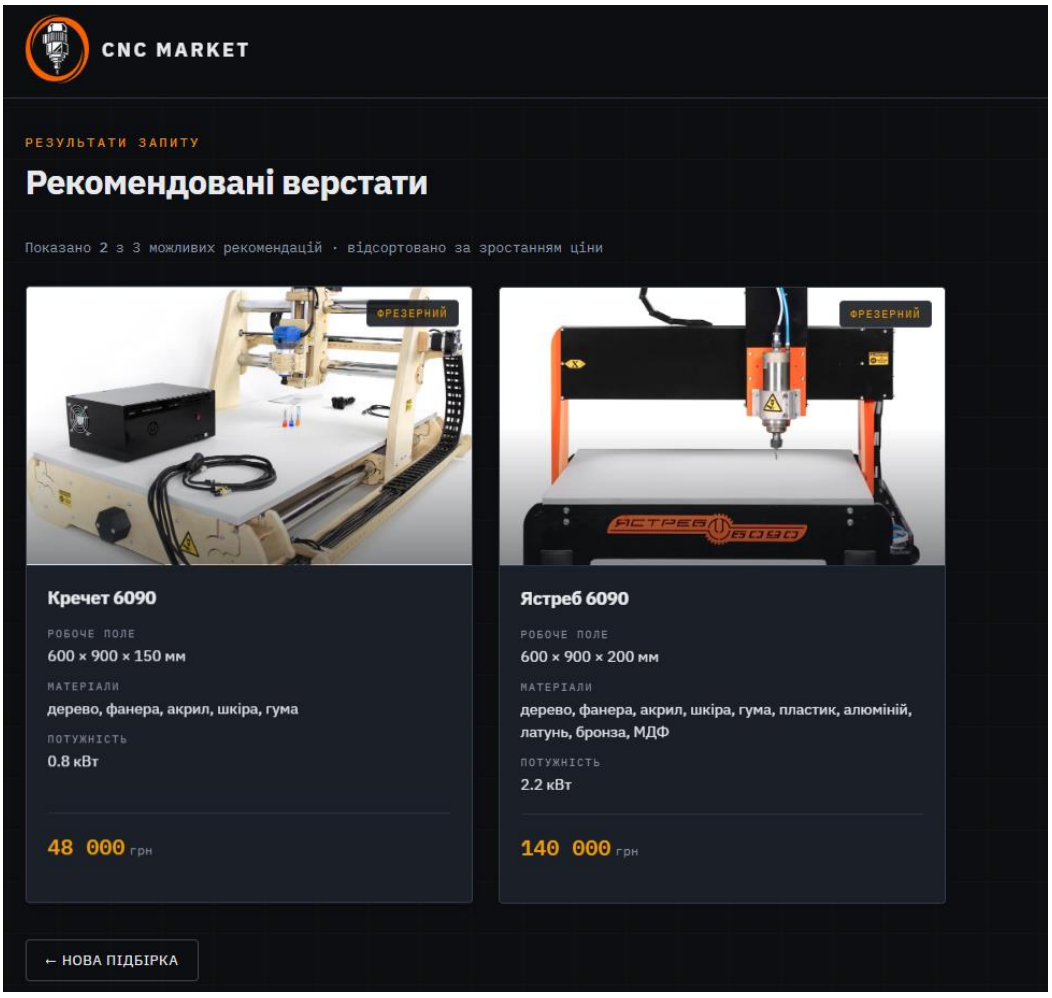

Рисунок 3.15 – Інтерфейс сторінки результатів багатокритеріального підбору

Розроблений вебзастосунок є мінімально життєздатним продуктом (MVP), який успішно вирішує базове завдання алгоритмічного підбору обладнання. Проте архітектура системи має значний потенціал для комерційного розвитку. Пріоритетним напрямком модернізації є розширення бази даних за рахунок додавання деталізованих характеристик обладнання (типи приводів, напрямних, сумісність із CAM-системами). Іншим перспективним вектором є інтеграція із зовнішніми API-сервісами постачальників для автоматичного оновлення цін, а також впровадження системи особистих кабінетів клієнтів із можливістю генерації комерційних пропозицій у форматі PDF.

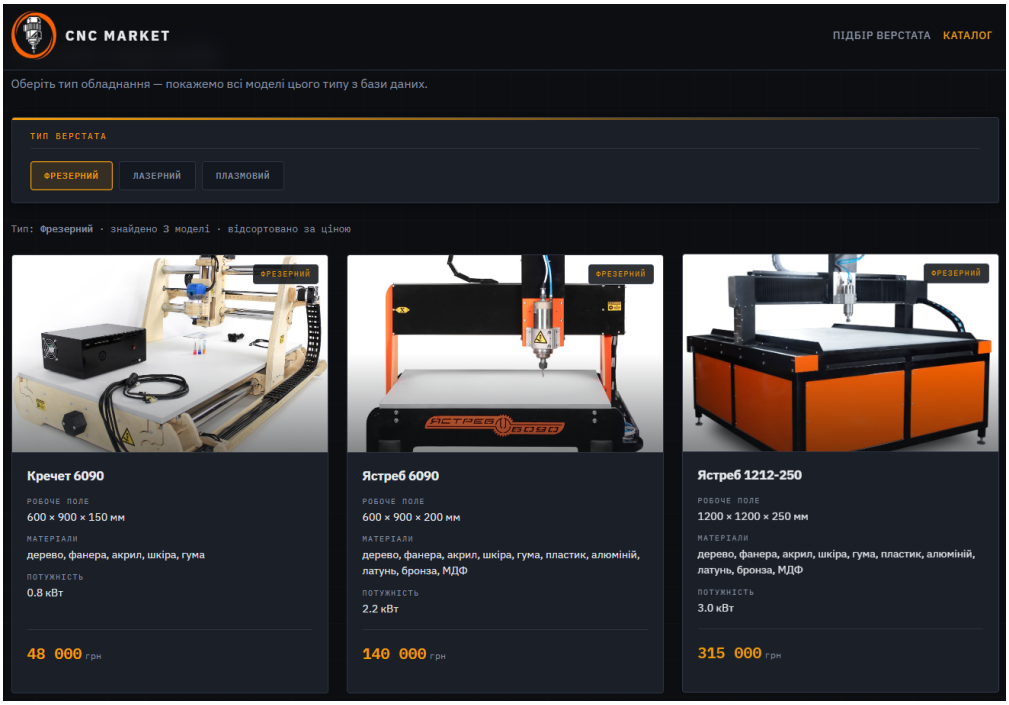

Рисунок 3.16 – Навігація по повному каталогу промислового обладнання

3.7 Забезпечення інформаційної безпеки та захисту даних вебзастосунку

Під час розроблення вебзастосунків, які обробляють запити користувачів та взаємодіють із реляційними базами даних, критично

важливим етапом є проєктування надійної архітектури безпеки (Security Architecture). Зважаючи на те, що система «CNC MARKET» може бути розгорнута у відкритому інтернет-середовищі та потенційно оброблятиме комерційні запити B2B-сектору, її програмний код був глибоко оптимізований для захисту від найбільш поширених вразливостей згідно з міжнародними стандартами безпеки OWASP Top 10 [30]. Такий підхід гарантує цілісність, конфіденційність та доступність даних.

Перш за все, архітектуру системи надійно захищено від атак типу впровадження SQL-коду (SQL Injection). Ця вразливість виникає, коли зловмисник передає через форму вводу шкідливі SQL-команди з метою несанкціонованого доступу до бази даних, викрадення інформації або її знищення. У розробленому застосунку цей ризик повністю нівельовано завдяки використанню технології об'єктно-реляційного відображення SQLAlchemy. Усі параметри, що надходять від клієнта (габарити заготовки, ціновий бюджет, тип верстата), проходять сувору типізацію та не конкатенуються безпосередньо у рядок SQL-запиту. Замість цього SQLAlchemy використовує механізм параметризованих запитів (Prepared Statements) на рівні драйвера SQLite. Інтерпретатор автоматично екранує всі спеціальні символи (наприклад, одинарні лапки або крапки з комою), перетворюючи їх на безпечні текстові літерали, що робить виконання шкідливого ін'єктованого коду технічно неможливим.

Другим вектором захисту є запобігання міжсайтовому скриптингу (Cross-Site Scripting, XSS). Цей тип атак передбачає впровадження шкідливих JavaScript-сценаріїв у вебсторінку, які потім виконуються у браузері кінцевого користувача і можуть призвести до викрадення сесійних даних. Оскільки результати багатокритеріального пошуку та картки верстатів генеруються динамічно на основі даних із БД, існує теоретичний ризик відображення небезпечного контенту. Для вирішення цієї проблеми успішно використано вбудований у мікрофреймворк Flask потужний шаблонізатор Jinja2. Він автоматично застосовує механізм Auto-Escaping (контекстне автоматичне

екранування) до всіх змінних, які виводяться у HTML-документ. Будь-які теги <script> або HTML-сутності, що можуть випадково чи навмисно міститися у назві верстата чи описі матеріалу, автоматично перетворюються на безпечний неактивний текст (наприклад, <script>), що унеможливлює несанкціоновану модифікацію DOM-дерева браузера.

Третім аспектом безпеки є механізм керування станом сесій (Session Management). Оскільки HTTP-протокол за своєю природою не зберігає стан (stateless), для запам'ятовування введених користувачем критеріїв пошуку використовується клієнтський cookie-файл. Для унеможливлення підробки цих даних (Session Tampering) на стороні клієнта, сесії у Flask криптографічно підписуються (Cryptographic Signing) за допомогою секретного ключа (app.secret_key). Цей ключ є абсолютно унікальним для кожного розгортання сервера і криптографічно безпечно генерується за допомогою системного модуля os.urandom. Будь-яка спроба зловмисника або користувача вручну змінити чи підробити вміст cookie-файлу неминуче призведе до порушення криптографічного підпису. У такому разі сервер автоматично відхилить скомпрометовану сесію і скине параметри пошуку до безпечних значень за замовчуванням, додатково захищаючи систему від атак із підміною ідентифікатора сесії.

ВИСНОВКИ

У рамках кваліфікаційної роботи було успішно вирішено актуальне науково-практичне завдання, яке полягає в комплексній автоматизації процесів інженерного аналізу та прийняття обґрунтованих рішень при виборі промислового обладнання в умовах переходу до концепції «Індустрія 4.0». Було спроєктовано, розроблено з нуля та успішно протестовано інтерактивний вебзастосунок «CNC MARKET», здатний здійснювати швидкий багатокритеріальний алгоритмічний підбір верстатів із числовим програмним керуванням на основі детальних технологічних вимог кінцевого користувача. У ході дослідження доведено, що існуючі комерційні платформи та маркетплейси виконують переважно функцію статичних електронних вітрин і не здатні алгоритмічно конвертувати конкретні бізнес-завдання замовника у точні апаратні конфігурації. Це беззаперечно підтверджує доцільність та своєчасність створення спеціалізованого експертного інструменту для B2B-сегменту та приватних майстерень.

У ході виконання роботи розроблено надійне математичне та алгоритмічне забезпечення процесу фільтрації промислового устаткування. Ключовим науковим здобутком стала розробка та впровадження ієрархічної моделі груп складності конструкційних матеріалів, що дозволило реалізувати інтелектуальну логіку «технологічної стелі». Завдяки цьому алгоритм автоматично пропонує обладнання з необхідним запасом міцності та потужності шпинделя, паралельно застосовуючи адаптивне ігнорування осі Z для лазерних установок. Виконано детальне об'єктно-орієнтоване проєктування архітектури системи за допомогою уніфікованої мови UML (розроблено діаграми прецедентів, активності та послідовності), де чітко формалізовано життєвий цикл обробки HTTP-запитів. Також спроєктовано оптимізовану та масштабовану реляційну модель бази даних, що забезпечує гнучке, структуроване зберігання технічних характеристик верстатів, їх

просторових габаритів та фінансової вартості без надмірної надлишковості даних.

Програмну реалізацію серверної та клієнтської частин вебзастосунку виконано з дотриманням патерну MVC та використанням сучасного стека технологій: мови Python, мікрофреймворку Flask, абстракції ORM SQLAlchemy, СУБД SQLite та компонентного фреймворку Bootstrap 5. Створено адаптивний користувацький інтерфейс у промисловому стилі з надійними механізмами збереження стану сесій та комплексним захистом від SQL-ін'єкцій і XSS-атак. Проведене тестування за методом «чорного ящика» підтвердило абсолютну відмовостійкість системи та математичну точність роботи алгоритму ранжування. Впровадження розробленого програмного продукту діє як віртуальний інженер-налагоджувальник, що дозволяє малим підприємствам уникати фінансових ризиків при закупівлі верстатів та оптимізує бізнес-процеси компаній-постачальників.

Результати роботи апробовано у вигляді 1 тези доповіді під час XXIV Міжнародної науково-практичної конференції «Modern technologies and innovations as new tools for the development of science» [31].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Brown A. (2025). Web architecture for industrial IoT and Industry 4.0, IEEE Internet of Things Journal, 12(4), pp. 450-462.

2. Smith J. (2023). CNC machining technology: Design, programming and execution: монографія. Springer, 240 с.

3. Johnson M., & Lee K. (2024). Multi-criteria decision making in industrial equipment selection, Journal of Manufacturing Systems, 65, pp. 120-135.

4. Котенко О.Є., Руденко Д.О., & Танянський О.С. (2024). Використання методів інтелектуального аналізу даних для медичної діагностики: навч. посібник. Харків: ХНУРЕ.

5. Suprun A., Tvoroshenko I., Gorokhovatskyi V., and Yakovleva O. (2025). Development and research of a method for the combined use of large language models for text generation, International Journal of Academic and Applied Research, 9(10), pp. 249-263.

6. Творошенко, І.С. (2021). Технології прийняття рішень в інформаційних системах: навч. посібник. Харків: ХНУРЕ.

7. Bohdan N., Tvoroshenko I., Gorokhovatskyi V., and Kobylin O. (2025). Development of a hybrid method to enhance context memory for a chatbot application based on large language models, International Journal of Academic Information Systems Research, 9(10), pp. 7-18.

8. Руденко Д.О., & Кириченко І.О. (2025). Основи програмування мовою С++: навч. посібник. Харків: ХНУРЕ.

9. SQLite documentation. URL: https://www.sqlite.org/docs.html (дата звернення 14.06.2026).

10. Date C.J. (2003). An introduction to database systems (8th ed.): навч. посібник. Pearson, 850 с.

11. Дацок Є., Дацок О., & Руденко Д.О. (2025). Analysis of effectiveness of using ORM DAPPER and SOLID principles in the design of a medical

information system, Bulletin of the National Technical University "KhPI". Series: Information and Modeling, 2, pp. 15-22.

12. SQLAlchemy 2.0 documentation. URL: https://docs.sqlalchemy.org/ (дата звернення 14.06.2026.

13. Gorokhovatskyi V., Tvoroshenko I., Kobylin O., & Vlasenko N. (2023). Search for visual objects by request in the form of a cluster representation for the structural image description, Advances in Electrical and Electronic Engineering, 21(1), pp. 19-27.

14. Каталог верстатів з ЧПК. (2026). CNC Machines. URL: https://cncmachines.com.ua/ (дата звернення 14.06.2026).

15. Гороховатський В.О., Творошенко І.С. (2022). Аналіз багатовимірних даних за описом у формі множини компонент: монографія. Харків: ХНУРЕ, 124 с.

16. Гороховатський В.О., Творошенко І.С., Чмутов Ю.В. (2022). Застосування систем ортогональних функцій для формування простору ознак у методах класифікації зображень, Сучасні інформаційні системи, 6(3), С. 5-12.

17. Шафроненко А.Ю., Бодянський Є.В., Руденко Д.О., & Шафроненко Є.О. (2026). Онлайн беггінг в задачах нечіткої кластеризації, Збірник наукових праць Харківського національного університету Повітряних Сил, 1, С. 87-95.

18. Bodyanskiy Y., Shafronenko A., Rudenko D., and Tanianskyi O. (2024). Online Stacking Credibilistic Fuzzy Clustering for Data Stream Mining, CEUR Workshop Proceedings, 3702, pp. 120-131.

19. Гороховатський В., Творошенко І. (2026). Продуктивні моделі аналізу даних у методах розпізнавання зображень: монографія. Харків: ХНУРЕ, 153 с.

20. Гороховатський В.О., Творошенко І.С. (2025). Оцінювання значущості ознак для підвищення продуктивності структурних методів класифікації зображень. Проблеми інформатики та моделювання (ПІМ-2025). Тези двадцять п'ятої міжнародної науково-технічної конференції (25 – 28

вересня 2025 року). Харків: НТУ «ХПІ», С. 38-43.

21. Flask documentation (3.0.x). URL: https://flask.palletsprojects.com/ (дата звернення 14.06.2026).

22. Gamma E., Helm R., Johnson R., & Vlissides J. (1994). Design patterns: Elements of reusable object-oriented software: монографія. Addison-Wesley, 395 с.

23. Python 3.12 documentation. URL: https://docs.python.org/3/ (дата звернення 14.06.2026.

24. Лутц М. (2021). Вивчаємо Python (5-те вид.): навч. посібник. Київ: Форс Україна, 1040.

25. Bootstrap 5.3 documentation. URL: https://getbootstrap.com/docs/5.3/ (дата звернення 14.06.2026).

26. Miller T. (2024). User interface design principles for complex engineering software, International Journal of Human-Computer Studies, 180, pp. 103-115.

27. Кобилін, О.А., & Творошенко, І.С. (2021). Методи цифрової обробки зображень: навч. посібник. Харків: ХНУРЕ.

28. Bodyanskiy Y., Shafronenko A., Rudenko D., Polubiekhin A., and Frolov D. (2024). Online Image Segmentation using Credibilistic Fuzzy Clustering, CEUR Workshop Proceedings, 3664, pp. 45-56.

29. Гороховатський В., Передрій О., Творошенко І., Марков Т. (2023). Матриця відстаней для множини компонентів структурного опису як інструмент для створення класифікатора зображень, Сучасні інформаційні системи, 7(1), С. 5-13.

30. OWASP top 10 web application security risks. URL: https://owasp.org/www-project-top-ten/ (дата звернення 14.06.2026).

31. Костенко Н. Р. (2026). Автоматизація багатокритеріального підбору обладнання з чпк на базі вебтехнологій. xxiv Міжнародна науково-практична конференція «Modern technologies and innovations as new tools for the development of science»: Тези доповідей. Мюнхен. С. 82–83.