МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра Могили

Факультет комп'ютерних наук

Кафедра інтелектуальних інформаційних систем

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувач кафедри інтелектуальних інформаційних систем

_____ Євген СІДЕНКО

«____»_____ 2026 р.

КВАЛІФІКАЦІЙНА РОБОТА

НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ БАКАЛАВРА

ВЕБЗАСТОСУНОК ДЛЯ ПРОГНОЗУВАННЯ ПОПИТУ НА ТОВАРИ ЕЛЕКТРОННОЇ КОМЕРЦІЇ З ВИКОРИСТАННЯМ МАШИННОГО НАВЧАННЯ ТА ЗОВНІШНІХ API

Спеціальність 122 Комп'ютерні науки

Освітня програма «Комп'ютерні науки»

Здобувач _____Дмитро КОСТЮК

«___»_____ 2026 р.

Керівник канд. техн. наук, доцент _____Галина КОНДРАТЕНКО

«___»_____ 2026 р.

Миколаїв – 2026

Чорноморський національний університет імені Петра Могили

Факультет	Комп’ютерних наук
Кафедра	Інтелектуальних інформаційних систем
Рівень вищої освіти	Перший (бакалаврський)
Освітній ступень	Бакалавр
Спеціальність	122 Комп’ютерні науки
Освітня програма	Комп’ютерні науки

ЗАТВЕРДЖУЮ

Завідувач кафедри інтелектуальних інформаційних систем

_____ Євген СІДЕНКО

«____» _____ 2025 р.

ЗАВДАННЯ
на кваліфікаційну роботу здобувача

Костюка Дмитра Олеговича

1. Тема кваліфікаційної роботи «Вебзастосунок для прогнозування попиту на товари електронної комерції з використанням машинного навчання та зовнішніх API».

Керівник роботи: Кондратенко Галина Володимирівна, доцент кафедри інтелектуальних інформаційних систем канд. техн. наук, доцент.

Затверджена наказом ЧНУ ім. Петра Могили від «25» грудня 2025 р. № 353.

2. Строк представлення кваліфікаційної роботи « » червня 2026 р.

3. Очікуваним результатом є розробка вебзастосунку DemandForecast для прогнозування попиту в e-commerce сегменті. Програмний комплекс автоматизує процеси збереження й перевірки історичних даних про продажі, агрегує контекстну інформацію через зовнішні API, обчислює прогнозні значення за допомогою ML-

модуля та візуалізує аналітику в графічному інтерфейсі. Проектування системи спирається на результати аналізу наявних програмних аналогів, декомпозицію технічних вимог до ML-компонентів і критерії масштабованості обраного технологічного стеку.

4. Перелік питань, що підлягають розробці: системний аналіз предметної галузі, бізнес-логіки та технічних вимог до платформи прогнозування; огляд та порівняння методів машинного навчання для задач прогнозування в електронній комерції; проєктування клієнт-серверної архітектури системи з урахуванням інтеграції ML–моделей та зовнішніх сервісів; реалізація бази даних для збереження історичних даних про продажі та результатів прогнозування; розробка бекенду для обробки запитів користувачів, навчання моделей та взаємодії із зовнішніми API; створення фронтенду з інтерактивною візуалізацією прогнозів: тестування точності моделей машинного навчання та оптимізація продуктивності вебзастосунку.

5. Перелік графічних матеріалів: презентація.

Керівник роботи _____ Галина КОНДРАТЕНКО
(Особистий підпис) *(Власне ім'я ПРІЗВИЩЕ)*

Здобувач _____ Дмитро КОСТЮК
(Особистий підпис) *(Власне ім'я ПРІЗВИЩЕ)*

Дата видачі завдання «22» грудня 2025 р.

КАЛЕНДАРНИЙ ПЛАН
кваліфікаційної роботи

Тема: Вебзастосунок для прогнозування попиту на товари електронної комерції з використанням машинного навчання та зовнішніх API

№	Найменування роботи	Початок	Закінчення	Примітки
1	Отримання завдання на виконання КР	24.12.2025	25.12.2025	Виконано
2	Огляд літератури та аналіз аналогів	15.01.2026	16.01.2026	Виконано
3	Складання календарного плану КР	22.01.2026	23.01.2026	Виконано
4	Аналіз предметної області та визначення вимог	02.02.2026	15.02.2026	Виконано
5	Проєктування архітектури вебзастосунку	16.02.2026	01.03.2026	Виконано
6	Розробка бази даних	02.03.2026	15.03.2026	Виконано
7	Реалізація бекенду та інтеграція зовнішних API	16.03.2026	02.04.2026	Виконано
8	Розробка та навчання ML–моделей	03.04.2026	19.04.2026	Виконано
9	Реалізація фронтенду вебзастосунку	20.04.2026	04.05.2026	Виконано
10	Тестування та відлагодження функціоналу	05.05.2026	11.05.2026	Виконано
11	Оформлення КР та презентації	12.05.2026	24.05.2026	Виконано
12	Перший попередній захист КР на засіданні комісії кафедри	25.05.2026	25.05.2026	Виконано
13	Корегування роботи за результатами попереднього захист	26.05.2026	04.06.2026	Виконано
14	Другий попередній захист КР на засіданні комісії кафедри та завершення оформлення КР	05.06.2026	05.06.2026	Виконано
15	Захист кваліфікаційної роботи	23.06.2026	23.06.2026	Виконано

Керівник роботи _____ Галина КОНДРАТЕНКО
(Особистий підпис) *(Власне ім'я ПРІЗВИЩЕ)*

Здобувач _____ Дмитро КОСТЮК
(Особистий підпис) *(Власне ім'я ПРІЗВИЩЕ)*

Дата складання календарного плану
«29» січня 2026 р.

АНОТАЦІЯ

до кваліфікаційної роботи
здобувача групи 402 ЧНУ ім. Петра Могили

Костюка Дмитра Олеговича

на тему: **“ВЕБЗАСТОСУНОК ДЛЯ ПРОГНОЗУВАННЯ ПОПИТУ НА ТОВАРИ ЕЛЕКТРОННОЇ КОМЕРЦІЇ З ВИКОРИСТАННЯМ МАШИННОГО НАВЧАННЯ ТА ЗОВНІШНІХ API”**

Актуальність теми зумовлена стрімким розвитком електронної комерції та зростанням конкуренції між онлайн-платформами. За таких умов бізнесу дедалі частіше потрібні інструменти, які дають змогу точніше керувати попитом і швидше реагувати на зміни ринку. Традиційні підходи до прогнозування не завжди враховують сезонність, погодні умови та споживчі тренди, через що виникають надлишкові або, навпаки, недостатні товарні запаси, а це призводить до фінансових витрат. Використання методів машинного навчання разом із даними зовнішніх API відкриває можливість підвищити точність прогнозів і приймати більш обгрунтовані управлінські рішення в сфері e-commerce.

Об'єкт роботи – процес прогнозування попиту на товари в системах електронної комерції.

Предмет роботи – методи машинного навчання, моделі часових рядів і механізми інтеграції зовнішніх API, що застосовуються для побудови системи прогнозування попиту.

Мета роботи – розробка інтелектуального вебзастосунку, який на основі історичних даних продажів і зовнішніх факторів прогнозуватиме попит та формуватиме рекомендації щодо оптимізації товарних запасів і цінової політики.

Методи дослідження – аналіз часових рядів, методи машинного навчання (Prophet, LSTM), порівняльна оцінка моделей та інтеграція зовнішніх API.

У першому розділі здійснено аналіз предметної області, досліджено особливості прогнозування попиту в електронній комерції, виконано порівняння

програмних аналогів і сформульовано постановку задачі дослідження.

У другому розділі обґрунтовано вибір методів прогнозування часових рядів (Prophet, LSTM) та підхід до комбінування прогнозів із оптимізованими вагами, визначено систему метрик для оцінювання якості, а також обґрунтовано технологічний стек розробленої системи.

У третьому розділі описано структуру та архітектуру системи, продемонстровано роботу розробленого вебзастосунку та проведено порівняльний аналіз точності моделей Prophet, LSTM і схеми комбінування прогнозів на синтетичних даних за відповідними метриками.

Четвертий розділ містить керівництво користувача з покроковим описом роботи із системою, а також результати функціонального й нефункціонального тестування розробленого вебзастосунку.

Кваліфікаційна робота викладена на 105 сторінках машинописного тексту, складається зі вступу, 4 розділів, загальних висновків, 4 додатків, 39 джерел у переліку джерел посилання. Праця містить 15 таблиць та 44 рисунки.

Ключові слова: прогноз попиту, часові ряди, машинне навчання, Prophet, LSTM, електронна комерція, Wikimedia Pageviews, оптимізація запасів, дашборд, REST API.

ABSTRACT

to the qualification work by the student of the group 402 of Petro Mohyla Black Sea National University

Kostiuk Dmytro

“WEB APPLICATION FOR FORECASTING DEMAND FOR E-COMMERCE GOODS USING MACHINE LEARNING AND EXTERNAL APIS”

The relevance of the topic is driven by the rapid growth of the e– commerce market and increasing competition among online platforms, which encourages businesses to seek intelligent demand management tools. Traditional forecasting methods fail to account for dynamic external factors such as seasonal fluctuations, weather conditions, and consumer trends, leading to overstocking or stockouts and significant financial losses. The application of machine learning methods combined with external API data opens new opportunities for improving forecast accuracy and supporting informed decision– making in the e–commerce sector.

The object of research is the process of demand forecasting for goods in e–commerce systems.

The subject of research is machine learning methods, time series models and mechanisms for integrating external APIs to build a demand forecasting system.

The purpose of the work is to develop an intelligent web application that forecasts product demand based on historical sales data and external factors and generates recommendations for optimizing inventory and pricing policies.

Research methods include time series modeling, machine learning methods (Prophet, LSTM), comparative model analysis, and external API integration.

The first section analyzes the subject area, examines the specifics of demand forecasting in e-commerce, provides a comparative analysis of software analogues, and formulates the research problem statement.

The second section justifies the selection of time series forecasting methods (Prophet, LSTM) and the forecast combination scheme with optimized weights, defines

the quality evaluation metrics system, and substantiates the technological stack of the developed system.

The third section describes the structure and architecture of the system, demonstrates the operation of the developed web application, and performs a comparative accuracy analysis of the Prophet, LSTM, and forecast combination models on synthetic data using the corresponding metrics.

The fourth section contains a user guide with a step-by-step description of working with the system and the results of functional and non-functional testing of the developed web application.

The qualification work is presented on 105 pages, consists of an introduction, 4 sections, general conclusions, 4 appendices, and 39 references. The work contains 15 tables and 44 figures.

Key words: demand forecasting, time series, machine learning, Prophet, LSTM, e–commerce, Wikimedia Pageviews, inventory optimization, dashboard, REST API.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

МСБ – малий та середній бізнес

ADF – Augmented Dickey-Fuller test

API – Application Programming Interface

ARIMA – AutoRegressive Integrated Moving Average

ASGI – Asynchronous Server Gateway Interface

BPTT – Backpropagation Through Time

CAGR – Compound Annual Growth Rate

HTTP – HyperText Transfer Protocol

JSON – JavaScript Object Notation

JWT – JSON Web Token

LSTM – Long Short–Term Memory

MAE – Mean Absolute Error

MAPE – Mean Absolute Percentage Error

ML – Machine Learning

REST – Representational State Transfer

RNN – Recurrent Neural Network

RMSE – Root Mean Square Error

SHAP – SHapley Additive exPlanations

SPA – Single Page Application

SQL – Structured Query Language

UI – User Interface

UX – User Experience

ВСТУП

Ринок електронної комерції генерує дедалі більші масиви транзакційних даних, і бізнес гостро потребує інструментів, здатних перетворювати ці дані на конкретні управлінські рішення. Цифрова трансформація торгівлі зробила ринок швидшим: ціни оновлюються в реальному часі, споживачі порівнюють пропозиції за секунди, а попит реагує на зовнішні події значно гостріше, ніж у традиційному ритейлі. Прогнозування попиту – одна з центральних задач у цьому контексті: помилка в оцінці майбутніх продажів безпосередньо конвертується у надлишкові складські залишки або дефіцит товару, а обидва сценарії несуть прямі фінансові втрати. Підприємство, яке покладається на інтуїцію або застарілі методи планування, системно програє конкурентам що приймають рішення на основі даних.

Великі гравці Amazon, Shopify, Alibaba вирішують цю задачу власними ML-платформами. Корпоративний сегмент обслуговують SAP, SAS та IBM. Теоретичний фундамент методів прогнозування часових рядів закладено, зокрема, у працях Роба Дж. Гіндмана. Рекурентні нейронні мережі типу LSTM здатні вловлювати довгострокові нелінійні залежності у часових рядах, адитивні моделі типу Prophet дозволяють явно враховувати сезонність і ефект акційних подій, а інтеграція зовнішніх даних через API відкриває можливість будувати прогнози з урахуванням реального ринкового середовища. Жодне з наявних комерційних рішень не адаптоване під реалії малого та середнього бізнесу: ліцензійна вартість, технологічна замкненість і відсутність автоматизованої інтеграції зовнішніх контекстних даних роблять їх практично недосяжними для МСБ.

Необхідність розробки зумовлена розривом між потребою і доступністю інструментів. Існуючі рішення спираються на класичні статистичні моделі, погано адаптуються до змін ринку, не автоматизують збір контекстних даних через API і мають високу вартість впровадження. Саме МСБ найгостріше потребує інструментів оптимізації запасів – через обмеженість ресурсів і меншу здатність поглинати збитки від надлишкових залишків або дефіциту товару.

Сфера застосування результатів:

- інтернет–магазини та маркетплейси;
- аналітичні підрозділи компаній з електронної комерції;
- логістичні центри для планування складських запасів і навантаження.

Декларація про використання ШІ. Під час підготовки наукової роботи (академічного тексту) було використано інструмент генеративного штучного інтелекту Claude Sonnet 4.6. Його було застосовано для таких допоміжних завдань: генерування та оптимізація фрагментів програмного коду; розробка шаблонів для візуалізації даних; вичитування, редагування та реформатування чорнових варіантів тексту для адаптації та коригування емоційного тону відповідно до академічного стилю; переклад технічної документації; оцінювання якості матеріалу та отримання рекомендацій щодо покращення структури розділів. Весь основний текст, аналіз архітектурних рішень, інтерпретація результатів моделювання та висновки є результатом моєї власної інтелектуальної праці. Я перевірив всю інформацію, отриману від ШІ, на точність та достовірність і несу повну особисту відповідальність за зміст цієї роботи.

1 АНАЛІЗ ПРЕДМЕТНОЇ СФЕРИ ТА ПОСТАНОВКА ЗАДАЧІ ПРОГНОЗУВАННЯ ПОПИТУ В ЕЛЕКТРОННІЙ КОМЕРЦІЇ

Перший розділ присвячено аналізу предметної сфери та формулюванню задачі дослідження. Підрозділ 1.1 розкриває стан ринку e-commerce, специфіку прогнозування попиту та фактори, що визначають точність моделей. Підрозділ 1.2 містить порівняльний аналіз трьох програмних аналогів – Salesforce Einstein Analytics, Amazon Forecast та Google Cloud Retail AI. У підрозділі 1.3 на основі виявлених прогалин сформульовано постановку задачі з визначенням об'єкту, предмету, мети та переліку завдань.

1.1 Опис предметної сфери

Ринок електронної комерції розвивається під тиском двох паралельних процесів: глобальної цифровізації та зміни споживчої поведінки. За даними Statista, онлайн-продажі щорічно зростають на 10–12% у загальному обсязі світового ритейлу [1], що формує конкурентне середовище, де швидкість прийняття рішень безпосередньо визначає виживання бізнесу. У цих умовах точність прогнозування попиту – науково обґрунтованого передбачення майбутніх обсягів продажів – набуває вирішального значення для забезпечення операційної ефективності.

E-commerce-прогнозування відрізняється від традиційного ритейлу за трьома характеристиками:

– висока динамічність: ціни, акції та асортимент можуть змінюватися кілька разів на добу, і кожна така зміна миттєво відбивається на обсягах замовлень;

– цифровий слід: поведінкові сигнали користувачів – перегляди товару, додавання в обране, пошукові запити – несуть прогностичну інформацію не менш значущу, ніж сама історія продажів;

– реакція на зовнішні збурення: погодні аномалії, сплески медійної активності або логістичні затримки впливають на онлайн-попит швидше і помітніше, ніж на офлайн-точки.

Узагальнену динаміку ринку та прогнозні показники до 2028 року представлено на рисунку 1.1.

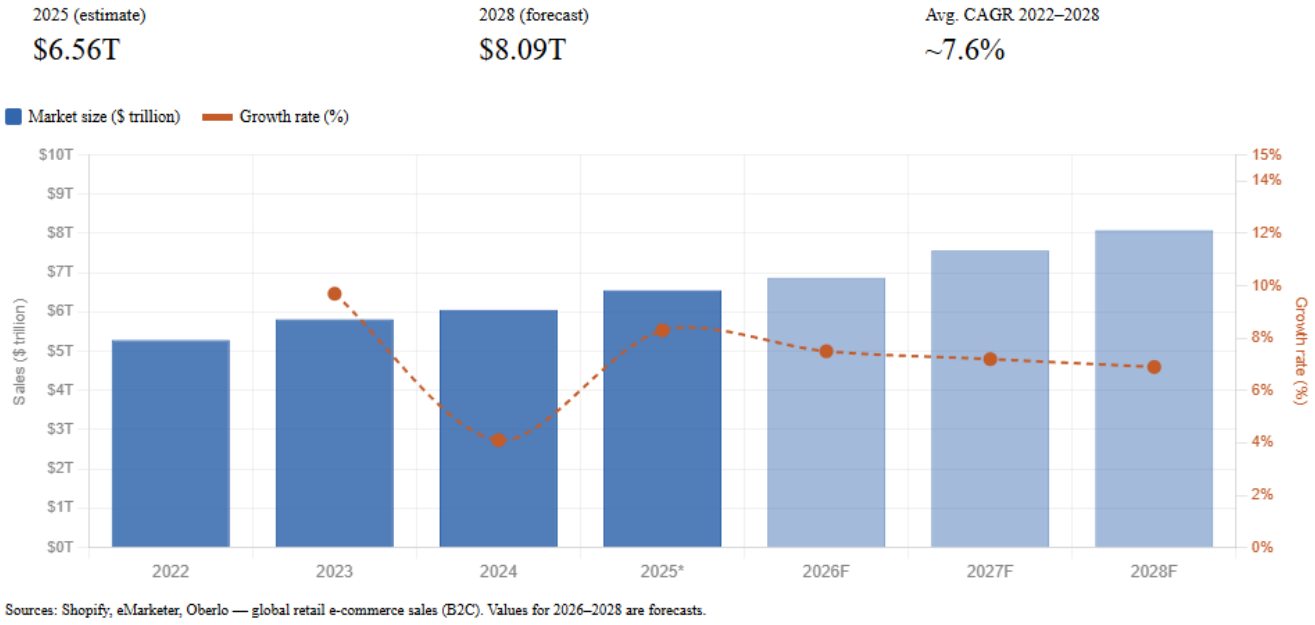


Рисунок 1.1 – Динаміка та прогноз обсягу світового ринку електронної комерції, 2022–2028 рр. (млрд дол. США)

Дані рисунку 1.1 фіксують стійке зростання: з 5,3 трлн дол. у 2022 році до очікуваних 8,09 трлн дол. у 2028-му при CAGR ≈ 7,6%. Масштаб галузі унеможливлює ручне управління аналітичними процесами – потрібна автоматизація.

Попит в e-commerce розподіляється за трьома типами: стабільний (товари першої необхідності з низькою варіативністю), сезонний (прив'язаний до свят і пір року) та переривчастий (intermittent) – найскладніший для моделювання через стохастичний характер виникнення.

Класичні статистичні методи – експоненціальне згладжування, сімейство ARIMA погано справляються зі складними ринковими сценаріями: вони апроксимують лінійні закономірності і не здатні фіксувати різкі структурні аномалії. Дослідження на IEEE Xplore підтверджують, що архітектури глибокого навчання, зокрема LSTM-мережі, знижують RMSE прогнозу завдяки здатності

утримувати довгострокові залежності та виявляти складні патерни у великих часових рядах [2].

Окремий напрям підвищення точності – інтеграція зовнішніх контекстних даних через API. Врахування сигналів поза внутрішньою звітністю компанії дозволяє моделі реагувати на екзогенні фактори задовго до їх відображення у даних продажів [3].

Open-Meteo API постачає погодні показники критичні для категорій сезонного одягу, техніки та спортивного інвентарю. Wikimedia Pageviews API фіксує щоденну статистику переглядів статей Вікіпедії як проксі-індикатор споживчого інтересу, що зазвичай випереджає фактичну покупку на один-два тижні [11].

Практичним наслідком неточного прогнозування є bullwhip effect – ефект батога, коли незначні коливання кінцевого попиту багаторазово підсилюються на рівні складських замовлень. Це призводить або до дефіциту (out-of-stock) з відповідною втратою лояльності покупців, або до надлишкових залишків (overstock), що заморожують обігові кошти підприємства.

1.2 Огляд та аналіз наявних аналогів і публікацій

Аналіз наявних програмних рішень передує розробці власної системи з практичної необхідності: без розуміння того, що вже існує на ринку, неможливо обґрунтувати архітектурні рішення і сформулювати реальні конкурентні переваги. Для порівняння для порівняння взято три платформи, що домінують у сегменті інтелектуального прогнозування попиту для електронної комерції.

Salesforce Einstein Analytics – хмарна аналітична платформа, вбудована в екосистему Salesforce CRM [4]. Для побудови прогнозів попиту та аналізу клієнтської поведінки платформа задіює алгоритми AutoML і нейронні мережі; результати відображаються через інтерактивні дашборди з інструментом Einstein Discovery (рис. 1.2).

Рисунок 1.2 – Логотип компанії Salesforce Einstein Analytics

З технічного боку платформа побудована на Java і Python, використовує власний фреймворк Salesforce Lightning на фронтенді та гібридні хмарні NoSQL-сховища для часових рядів. Архітектура – багатокористувацька SaaS з інтегрованим шаром AutoML.

Серед сильних сторін – глибока нативна інтеграція з CRM-даними, автоматичний підбір моделей і здатність обробляти великі датасети в хмарному середовищі. Проте для задач цієї роботи критичнішими є обмеження. Вартість ліцензії стартує від $25 на користувача і суттєво зростає при масштабуванні. Підключення зовнішніх контекстних джерел – погодних API або трендових даних – вимагає платного проміжного сервісу MuleSoft. Платформа жорстко прив'язана до екосистеми Salesforce (Vendor Lock-in), а для МСБ залишається фактично недосяжною за вартістю впровадження..

Amazon Forecast – повністю автоматизований хмарний сервіс прогнозування часових рядів від Amazon Web Services [5]. Підтримує широкий набір алгоритмів – DeepAR+, CNN-QR, Prophet, ARIMA, ETS – з автоматичним підбором найкращої моделі під конкретний датасет (рис. 1.3).

Рисунок 1.3 – Логотип компанії Amazon Forecast

Стек: Python SDK Boto3 для взаємодії через REST API, Amazon S3 як сховище датасетів, Amazon QuickSight як опційний інструмент візуалізації. Механізм Explainability забезпечує часткову інтерпретацію прогнозів, проте реалізований у закритому вигляді і суттєво поступається гнучкості SHAP [6].

Сервіс справляється з мільйонами часових рядів одночасно і безшовно інтегрується з рештою інфраструктури AWS. Разом з тим повна залежність від AWS-екосистеми (Vendor Lock-in) і непрозора модель ціноутворення роблять його обтяжливим для МСБ. Нативного інтерфейсу для кінцевого користувача немає – робота ведеться виключно через SDK або консоль. Інтеграція зовнішніх API на кшталт Open-Meteo або Wikimedia Pageviews потребує написання окремого коду через AWS Lambda і не входить до стандартної функціональності.

Google Cloud Retail AI – хмарна платформа від Google для прогнозування попиту та персоналізації асортименту, орієнтована на великих рітейлерів [7]. Ядро платформи будується на власних нейромережевих архітектурах Google з глибокою інтеграцією в екосистему BigQuery та Looker (рис. 1.4).

Рисунок 1.4 – Логотип компанії Google Cloud Retail AI

Технологічний стек: TensorFlow Extended (TFX) для навчання моделей, Vertex AI як MLOps-платформа, Cloud Dataflow і BigQuery ML для обробки даних. Управління – через консоль GCP або клієнтські бібліотеки Google Cloud.

Платформа демонструє високу точність на великих розрізнених наборах даних і забезпечує автоматичне масштабування в межах GCP. Водночас поріг входу для МСБ – один з найвищих серед розглянутих рішень. Розгортання можливе

виключно в межах Google Cloud (Cloud-only), зовнішні джерела даних поза Google-екосистемою не підтримуються через відкритий API, а впровадження потребує залучення сертифікованих дата-інженерів із досвідом налаштування MLOps-пайплайнів.

Таблиця 1.1 – Порівняння функціональних можливостей систем–аналогів

Функціональна можливість	**Salesforce Einstein**	**Amazon Forecast**	**Google Retail AI**	**Власний проєкт**
Прогнозування часових рядів	+	+	+	+
Використання глибокого навчання (DL)	+	+	+	+
Автоматичний підбір моделей (AutoML)	+	+	+	–
Пряма інтеграція з зовнішніми API (Погода / Тренди)	–	–	–	+
Візуалізація результатів «з коробки»	+	– (QuickSight)	+ (Looker)	+
Пояснюваність прогнозів (Explainability)	Частково	Частково	–	+ (SHAP)

Таблиця 1.2 – Технологічні стеки та архітектурні особливості

Критерій	**Salesforce Einstein**	**Amazon Forecast**	**Google Retail AI**	**Власний проєкт**
Основна мова	Java, Python	Python (Boto3 SDK)	Python (TFX)	Python (FastAPI)
База даних	NoSQL (Custom Cloud)	Amazon S3	BigQuery	Supabase (PostgreSQL)
Фронтенд	Lightning Framework	QuickSight (опційно)	Looker	React (TypeScript)
Моделі ML	Пропрієтарні (AutoML)	DeepAR+, CNN–QR, ARIMA	Vertex AI Models	Prophet, LSTM
Зовнішні дані	MuleSoft (платно)	AWS Lambda (custom)	Cloud Dataflow	Open– Meteo, Wikimedia REST API
Тип системи	Закрита (SaaS)	Хмарна (PaaS)	Хмарна (PaaS)	Відкрита (Open–source)
Ліцензія	Комерційна	Комерційна	Комерційна	Відкрита

Три розглянуті платформи – Salesforce Einstein Analytics, Amazon Forecast і Google Cloud Retail AI – попри суттєві відмінності в архітектурі та позиціонуванні, мають спільні структурні обмеження, що роблять їх непридатними для МСБ.

Усі три є комерційними закритими продуктами з високим порогом вартості. Salesforce стартує від $25 на користувача із значними витратами на впровадження. AWS використовує непрозору модель ціноутворення, яка зростає разом з обсягом даних. Google Cloud Retail AI практично недосяжна для малого бізнесу без виділеної команди дата-інженерів.

Жодна з платформ не підтримує нативної інтеграції зовнішніх контекстних джерел – погодних даних або пошукових трендів – без платних проміжних сервісів: MuleSoft у Salesforce, AWS Lambda в Amazon Forecast. Пояснюваність прогнозів або відсутня повністю (Google Cloud Retail AI), або реалізована у закритому вигляді без можливості розширення (Amazon Forecast Explainability) – на відміну від відкритого SHAP. Результати порівняльного аналізу систематизовано у таблицях 1.1 та 1.2.

Саме ці прогалини обґрунтовують розробку власного рішення на відкритому стеку: Python/FastAPI, React, Supabase [8], Prophet [9] і LSTM – з прямою інтеграцією Open-Meteo [10] та Wikimedia Pageviews [11], вбудованою підтримкою SHAP-пояснень та орієнтацією на доступність для МСБ без ліцензійних витрат.

Теоретична база роботи сформована за результатами систематизованого пошуку публікацій у наукометричних базах IEEE Xplore, Scopus, MDPI та Google Scholar за запитами «demand forecasting e-commerce», «LSTM time series», «Prophet retail forecasting», «external API demand forecasting». Відбиралися роботи за 2019–2024 роки, що стосуються прогнозування часових рядів у роздрібній торгівлі та електронній комерції.

Огляд Massaoudi et al. [12], що охоплює 119 публікацій зі Scopus за 2015–2024 роки, фіксує характерну динаміку: 73% з них з'явились у 2021–2024 роках. Це свідчить не просто про актуальність теми, а про те, що галузь переживає активну

фазу формування нових підходів – саме в той період, коли глибоке навчання стало практично застосовним для задач прогнозування попиту. Результати аналізу публікацій систематизовано у таблиці 1.3.

Таблиця 1.3 – Аналіз наукових публікацій за темою дослідження

Автор, рік	**Тема дослідження**	**Метод / підхід**	**Ключовий результат**	**Відношення до теми роботи**
Bandara K. et al., 2019 [2]	Sales Demand Forecast in E–commerce Using LSTM	Глобально навчена LSTM–мережа на ієрархії продуктів	Конкурентні результати на даних Walmart.com; перевищує ARIMA та ETS	Обґрунтування LSTM для прогнозування попиту в e–commerce
Khan M. A. et al., 2020 [3]	Effective Demand Forecasting Using Business Intelligence and ML	Методи ML + BI: регресія, нейронні мережі	Поєднання ML та BI підвищує точність порівняно з ізольованими	Обґрунтування гібридного підходу та метрик MAE/RMSE
Li Z., Zhang N., 2022 [38]	Short–Term Demand Forecast of E–Commerce Platform Based on ConvLSTM	ConvLSTM, LSTM, LGBM; метрика MAPE	ConvLSTM знижує MAPE на 0.42 проти LSTM та на 0.68 проти LGBM	Порівняльний аналіз архітектур; обґрунтування вибору MAPE як метрики
Falatouri T. et al., 2022 [13]	Predictive Analytics for Demand Forecasting: SARIMA vs LSTM in Retail SCM	SARIMA та LSTM; дані роздрібної торгівлі	LSTM перевершує SARIMA на нерегулярних та сезонних рядах	Підтверджує перевагу LSTM над класичними статистичними моделями
Massaoudi M. et al., 2024 [12]	ML and DL Models for Demand Forecasting in Supply Chain: A Critical Review	Огляд 119 публікацій зі Scopus (2015–2024)	73% публікацій з'явились у 2021–2024; глибоке навчання домінує	Підтверджує актуальність теми та обґрунтовує вибір DL–методів
Qureshi N. et al., 2024 [14]	Demand Forecasting with Weather and External Data Integration	ML–моделі з погодними API як зовнішніми регресорами	Врахування погодних факторів знижує MAPE на 10–24% для сезонних категорій	Обґрунтування інтеграції Open–Meteo API у систему
Lim B., Zohren S., 2021 [20]	Time–Series Forecasting with Deep Learning: A Survey	Огляд архітектур: LSTM, Transformer, TCN, N–BEATS	LSTM залишається конкурентоздатною архітектурою для коротко– та середньострокових рядів	Обґрунтування вибору LSTM серед архітектур глибокого навчання

Аналіз наукових публікацій дозволяє окреслити стан досліджень і визначити

місце даної роботи в загальному науковому контексті. Роботи Falatouri et al. [13] та Khan et al. фіксують стабільну перевагу LSTM-архітектур над класичними статистичними моделями – ARIMA та SARIMA – на нерегулярних і сезонних рядах роздрібної торгівлі.

Гібридні підходи, що поєднують адитивні моделі типу Prophet з нейромережевими методами через зважене комбінування, систематично знижують похибку прогнозу відносно кожного компонента окремо.

Qureshi et al. [14] показали, що інтеграція погодних даних як екзогенних регресорів знижує MAPE на 10–24% для сезонних товарних категорій – результат, який прямо обґрунтовує підключення Open-Meteo API у даній роботі. Використання індикаторів пошукового інтересу як випереджаючих сигналів попиту також отримало теоретичне підтвердження в сучасній літературі.

Водночас у проаналізованих роботах простежується характерна прогалина: переважна більшість досліджень орієнтована або на великі підприємства з масивними датасетами, або зосереджена на ізольованих теоретичних аспектах моделювання.

Комплексний вебзастосунок з відкритим кодом, що поєднує Prophet, LSTM і зовнішні API в єдиному пайплайні та орієнтований на МСБ без значних інфраструктурних витрат, у літературі практично не представлений. Результати аналізу підтверджують обґрунтованість обраного підходу і окреслюють практичну нішу, яку покликана заповнити розроблювана система.

1.3 Постановка задачі

Аналіз підрозділу 1.2 зафіксував три спільні обмеження наявних платформ: фінансова недоступність для МСБ, відсутність автоматизованої інтеграції зовнішніх даних і закриті або відсутні механізми пояснюваності прогнозів. Усунення цих обмежень і є предметом даної роботи.

Актуальність теми обумовлена тим, що конкуренція між онлайн-

платформами змушує бізнес шукати інструменти, який дедалі гостріше потребує інструментів, здатних передбачати попит, а не лише фіксувати його постфактум. Класичні методи прогнозування не враховують динамічних зовнішніх сигналів – сезонних коливань, погодних умов, споживчих трендів – і через це генерують або надлишкові залишки на складі, або дефіцит товару. Обидва сценарії конвертуються у прямі фінансові втрати.

Поєднання методів машинного навчання з контекстними даними зовнішніх API дозволяє вийти за межі внутрішньої звітності компанії і будувати прогнози з урахуванням реального ринкового середовища.

Об'єкт роботи – процес прогнозування попиту на товари в системах електронної комерції.

Предмет роботи – методи машинного навчання, моделі часових рядів та механізми інтеграції зовнішніх API для побудови інтелектуальної системи прогнозування попиту.

Мета роботи – розробка інтелектуального вебзастосунку, який на основі історичних даних продажів та зовнішніх факторів здійснює прогноз попиту на товари електронної комерції й формує практичні рекомендації для оптимізації товарних запасів і цінової політики підприємства.

Методи дослідження – у роботі застосовано такі методи: моделювання часових рядів через адитивну модель Prophet і рекурентну нейронну мережу LSTM; порівняльний аналіз якості моделей за метриками RMSE і MAPE; пояснюваність ML-результатів через SHAP і feature importance; збагачення навчальних вибірок контекстними даними через зовнішні API.

Для досягнення поставленої мети необхідно вирішити такі завдання:

– провести аналіз предметної області та дослідити існуючі підходи до прогнозування попиту в електронній комерції, виявити недоліки наявних рішень і сформулювати вимоги до системи;

– виконати огляд та порівняльний аналіз методів машинного навчання для

задач прогнозування часових рядів;

– спроєктувати архітектуру вебзастосунку з урахуванням інтеграції ML–компонентів та зовнішніх API, розробити структуру бази даних та сформувати специфікацію вимог до програмного забезпечення;

– реалізувати модуль збору та обробки даних з інтеграцією зовнішніх API, розробити та навчити моделі прогнозуванням з оцінюванням точності та пояснюваністю;

– реалізувати серверну частину, клієнтський інтерфейс та провести тестування з порівняльним аналізом моделей і рекомендаціями щодо впровадження.

Практична цінність роботи полягає в тому, що застосунок автоматизує планування закупівель і управління складськими запасами без ліцензійних витрат.

Відкритий технологічний стек і безкоштовне розгортання роблять систему реально доступною для МСБ – саме того сегменту, де інтелектуальні інструменти прогнозування залишаються практично недосяжними через обмеженість ресурсів.

Висновки до розділу 1

У першому розділі було проведено грунтовний аналіз предметної області та сформульовано задачу дослідження. Аналіз ринку e-commerce підтвердив стабільне зростання галузі: з 5,29 трлн дол. у 2022 році до прогнозованих 8,09 трлн дол. у 2028-му при CAGR ≈ 7,6%. У такому середовищі точність прогнозування попиту перестає бути конкурентною перевагою – вона стає операційною необхідністю. Виявлено три характеристики що відрізняють e-commerce від традиційного ритейлу: висока динамічність ринку, вплив цифрової поведінки користувачів і підвищена чутливість до зовнішніх збурень. Саме ці властивості роблять класичні статистичні методи недостатніми і обґрунтовують застосування ML у поєднанні із зовнішніми API.

Порівняльний аналіз трьох комерційних аналогів – Salesforce Einstein

Analytics, Amazon Forecast і Google Cloud Retail AI – виявив спільні структурні обмеження: висока вартість впровадження, технологічна замкненість і відсутність нативної інтеграції зовнішніх контекстних джерел без платних проміжних сервісів. Жодна з платформ не забезпечує відкритої пояснюваності прогнозів. Ці обмеження підтверджують практичну прогалину і обґрунтовують розробку власного рішення на відкритому стеку.

На основі виявлених недоліків сформульовано постановку задачі, визначено об'єкт і предмет дослідження, мету і перелік завдань. Отримані результати сформували вимоги до архітектури системи і стали відправною точкою для вибору методів і технологічного стеку у другому розділі.

2 МЕТОДИ МАШИННОГО НАВЧАННЯ ТА ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ ДЛЯ ПРОГНОЗУВАННЯ ПОПИТУ

Прогнозування попиту в e-commerce – задача, де класичні підходи системно програють. Нестаціонарність вхідних даних, множинна сезонність, різкі аномалії і суттєвий вплив зовнішніх чинників роблять її складнішою за більшість задач регресії на часових рядах. Жоден з розглянутих аналогів не вирішує її повноцінно для малого та середнього бізнесу (МСБ) – кожен закриває лише частину проблеми.

Другий розділ розкриває математичний апарат і технологічний інструментарій системи. Підрозділ 2.1 охоплює адитивну модель декомпозиції, алгоритм Prophet, архітектуру LSTM, метод комбінування прогнозів Bates–Granger, механізми збагачення даних через зовнішні API та систему метрик оцінювання якості. Підрозділ 2.2 описує технологічний стек – від серверної частини і ML-бібліотек до бази даних і клієнтського інтерфейсу.

2.1 Методи прогнозування попиту на основі машинного навчання

Перший розділ зафіксував ключові характеристики задачі: нестаціонарність рядів, множинна сезонність, зовнішні збурення. На основі цього у цьому підрозділі обґрунтовано гібридний підхід, що поєднує статистичне моделювання з глибоким навчанням, описано математичний апарат обраних методів і визначено систему метрик для порівняльного оцінювання якості прогнозів. Практичну реалізацію методів наведено у додатках А та Б. Пояснюваність результатів моделей забезпечується методом SHAP [15]. Емпіричне підтвердження відповідності обраних методів характеристикам конкретного набору даних – у підрозділі 3.1.5.

2.1.1 Математичне моделювання часових рядів

Обсяги продажів розглядаються як часовий ряд – упорядкована послідовність спостережень $Y = \{y_1, y_2, ..., y_n\}$, де кожен елемент y_t є результатом взаємодії детермінованих і стохастичних процесів. Декомпозиція ряду на структурні

компоненти дозволяє моделювати кожен фактор впливу окремо і тим самим підвищити прогностичну точність системи.

Задача формалізується через класичну адитивну модель:

$$Y(t) = T(t) + S(t) + C(t) + \varepsilon(t), \quad (2.1)$$

де *T(t)* – трендова компонента, що відображає довгостроковий напрям розвитку попиту;

S(t) – сезонна складова з регулярними циклічними коливаннями (тижневими, місячними, річними);

C(t) – циклічна компонента, зумовлена макроекономічними факторами;

ε(t) – випадкове відхилення («білий шум»), що систематичному моделюванню не піддається.

Адитивна форма обрана свідомо. Мультиплікативна альтернатива $Y(t) = T(t) \times S(t) \times C(t) \times \varepsilon(t)$ виправдана лише тоді, коли амплітуда сезонних коливань пропорційно зростає разом із рівнем тренду. Для e-commerce МСБ це не характерно: сезонні піки – різдвяний розпродаж, «чорна п'ятниця» – мають стабільну абсолютну амплітуду незалежно від поточного рівня продажів. STL-декомпозиція ряду Sports_drink (підрозділ 3.1.5) підтверджує це числово.

Більшість методів прогнозування передбачає стаціонарність ряду – незмінність його статистичних властивостей у часі [16]. Реальні ряди попиту через наявність тренду цій умові не відповідають [17]. Для формальної перевірки застосовується розширений тест Діккі–Фуллера (ADF); результати для досліджуваних рядів наведено у підрозділі 3.1.5.

2.1.2 Адитивна модель Prophet для аналізу бізнес– циклів

Для моделювання тренду, сезонності та ефекту акційних подій у системі застосовується байєсівська адитивна модель Prophet, розроблена інженерами Meta і описана у роботі Taylor і Letham [9]. Перед класичними моделями ARIMA [17]

Prophet має три практично значущі переваги: автоматичне виявлення точок перелому тренду (changepoints) без ручного налаштування, стійкість до пропущених значень і аномалій, характерних для даних онлайн-продажів, і вбудована підтримка зовнішніх регресорів для інтеграції API-даних.

Математична основа Prophet – адитивне рівняння:

$$y(t) = g(t) + s(t) + h(t) + \varepsilon_t, \tag{2.2}$$

де *g(t)* – функція тренду з автоматичним виявленням точок зміни швидкості зростання;

s(t) – сезонна складова, що апроксимується рядами Фур'є;

h(t) – компонента впливу святкових та акційних подій;

ε_t – залишковий шум.

Тренд g(t) задається як кусково-лінійна функція: швидкість зростання змінюється у точках перелому, які алгоритм виявляє автоматично через регуляризацію. Сезонна складова s(t) будується на усіченому ряді Фур'є – це дозволяє моделювати складні нерегулярні патерни без їхнього явного задання.

Компонента h(t) має пряме практичне значення для e-commerce. Через передачу переліку дат і вікон впливу вона явно враховує ефект «чорної п'ятниці», новорічних розпродажів та інших промо-подій – саме тих періодів, де точність прогнозу критична для бізнесу і де ARIMA без модифікацій системно помиляється.

2.1.3 Рекурентні нейронні мережі з довгостроковою пам’яттю (LSTM)

Нелінійні залежності у структурі попиту – те, з чим адитивні статистичні моделі принципово не справляються [13]. Для їх виявлення у системі реалізовано архітектуру LSTM (Long Short-Term Memory) – спеціалізований різновид рекурентних нейронних мереж. Класичні RNN мають відомий структурний недолік: при навчанні методом BPTT (Backpropagation Through Time) градієнт експоненційно затухає на довгих послідовностях [18], і модель втрачає здатність

враховувати залежності на значних часових відстанях. LSTM усуває цю проблему через внутрішній стан комірки C_t, який підтримується механізмом спеціалізованих воріт (gates).

Ворота забування f_t визначають, яку частину попереднього стану комірки слід обнулити:

$$f_t = \sigma(Wf \times [h_{t-1}, x_t] + bf), \tag{2.3}$$

де σ – сигмоїдна функція активації;

Wf – матриця ваг воріт забування;

h_{t-1} – прихований стан попереднього кроку;

x_t – вхідний вектор поточного кроку;

bf – вектор зміщення.

Вхідні ворота і кандидат нового стану контролюють, яка нова інформація записується до комірки. Оновлення стану відбувається за формулою:

$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t, \tag{2.4}$$

де $\odot$ – поелементне множення (операція Адамара);

$\tilde{C}_t$ – кандидат нового стану комірки.

Вихідні ворота o_t формують прихований стан h_t, який передається на наступний крок і слугує основою для генерації прогнозу..

У роботі реалізовано двошарову архітектуру: перший LSTM-шар містить 64 нейрони і повертає повні послідовності, другий – 32 нейрони. Після рекурентних шарів розташовано Dropout з коефіцієнтом 0,2 для запобігання перенавчанню і Dense-шар для формування скалярного прогнозованого значення. Навчання виконується оптимізатором Adam [19] з ранньою зупинкою при відсутності покращення на валідаційній вибірці протягом 10 епох. Вхідний вектор x_t – вікно з 30 попередніх спостережень разом із зовнішніми ознаками з API (реалізацію наведено у додатку Б.2).

2.1.4 Метод комбінування прогнозів

Ключова відмінність розробленої системи від аналогів – зважене комбінування прогнозів Prophet і LSTM в єдиному пайплайні. Принцип диверсифікації моделей обґрунтовує цей підхід: різна математична природа компонентів породжує різні типи систематичних похибок, і їх об'єднання знижує загальну дисперсію прогнозу порівняно з кожною моделлю окремо.

Prophet у цій схемі виконує роль базового екстрактора – виділяє лінійний тренд і основну сезонність, забезпечуючи інтерпретованість результату. LSTM фокусується на нелінійних залежностях і реакції на зовнішні сигнали: погодні умови та пошукові тренди. Фінальний прогноз формується як зважена комбінація:

$$\hat{y}_{final(t)} = \alpha \times \hat{y}_{Prophet(t)} + (1 - \alpha) \times \hat{y}_{LSTM(t)}. \quad (2.5)$$

Схема реалізує метод Bates–Granger (1969) [21], де коефіцієнт α ∈ [0, 1] визначається мінімізацією MAE на валідаційній вибірці: α* = argminα MAE(yval, α·Prophet + (1−α)·LSTM). На відміну від бекінгу (Bootstrap Aggregating) і бустингу, що працюють з однотипними моделями одного класу, Bates–Granger поєднує різнорідні прогнози через теорему про диверсифікацію помилок: некорельовані або слабко корельовані похибки складових моделей дають лінійну комбінацію з меншою загальною дисперсією. Реалізацію наведено у додатку А.2.

Просте усереднення (α = 0,5) – не оптимальний варіант. Prophet точніше моделює низькочастотну річну сезонність через компоненти Фур'є, LSTM краще відтворює короткостроковий авторегресійний патерн через механізм клітин пам'яті. Дисперсії залишків двох моделей порівнювані (відношення std ≈ 1.13 для обох тестових товарів), але не ідентичні – оптимальне α зміщується від 0,5 і автоматично визначається для кожного товару окремо.

Це підтверджує, що моделі несуть різну взаємодоповнюючу інформацію про структуру ряду, а не дублюють одна одну.

Порівняльну характеристику обох методів наведено у таблиці 2.1.

Таблиця 2.1 – Порівняльна характеристика методів Prophet та LSTM

Характеристика	**Prophet**	**LSTM**
Тип моделі	Статистична байєсівська адитивна)	Нейромережева (рекурентна, глибоке навчання)
Математична основа	Ряди Фур'є + точки перелому (changepoints)	Градієнтний спуск + BPTT (зворотне поширення крізь час)
Інтерпретованість	Висока – тренд, сезонність	Низька – потребує SHAP
Мінімальний обсяг	~100 спостережень	~500–1000+ спостережень
Виявлення нелінійних залежностей	Обмежене (лінійна апроксимація тренду)	Висока здатність (багатошарова нелінійна обробка)
Стійкість до пропущених значень	Висока – вбудована обробка пропусків	Потребує попереднього заповнення (imputation)
Врахування зовнішніх регресорів	Вбудована підтримка через параметр extra_regressors	Через вхідний вектор ознак x_t
Швидкість навчання	Висока (секунди на типових наборах)	Середня (хвилини; залежить від архітектури)
Роль у комбінуванні	Базова компонента – виділення тренду та сезонності; вища вага при стабільному сезонному попиті	Коригуюча компонента – нелінійні залежності та зовнішні фактори; вища вага при волатильному попиті
Тип похибки	Систематична на нерегулярних сплесках	Систематична на довгострокових трендах

Таблиця 2.1 фіксує цю комплементарність кількісно: Prophet забезпечує інтерпретованість, стійкість до малих вибірок і вбудовану підтримку акційних подій; LSTM вирізняється здатністю виявляти нелінійні патерни і ефективно використовувати зовнішні ознаки у вхідному векторі. Некорельованість похибок у поєднанні з різною математичною природою – достатня умова для застосування Bates–Granger. У сукупності це дозволяє поєднати сильні сторони обох підходів і

отримати більш стабільний та точний прогноз порівняно з використанням кожної моделі окремо.

Загальну архітектуру пайплайну прогнозування попиту наведено на рисунку 2.1.

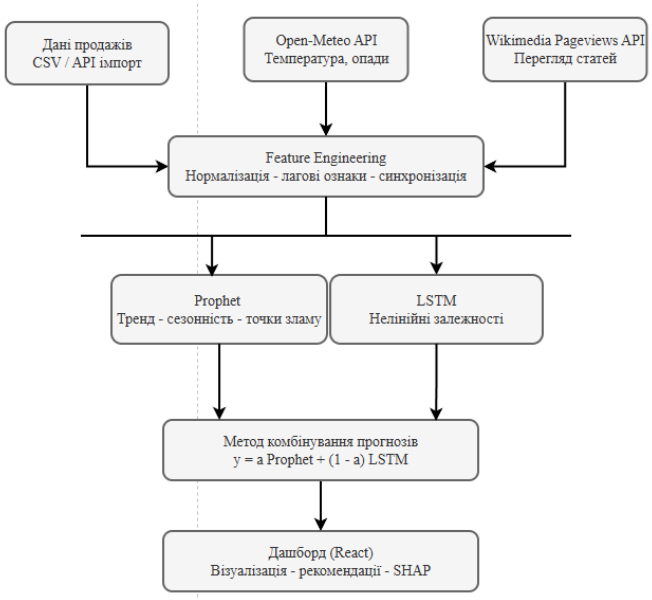


Рисунок 2.1 – Архітектура пайплайну прогнозування попиту вебзастосунку

Загальну архітектуру пайплайну наведено на рисунку 2.1. Пайплайн проходить п'ять етапів: збір даних з трьох джерел (продажі, Open-Meteo API, Wikimedia Pageviews API), формування вектору ознак через Feature Engineering, паралельне навчання Prophet і LSTM, комбінування прогнозів методом Bates–Granger і відображення результатів на дашборді з SHAP-поясненнями. Модульна структура забезпечує незалежність компонентів – кожен блок замінюється або розширюється без зміни загальної логіки пайплайну.

2.1.5 Методи збагачення даних через інтеграцію зовнішніх API

Автоматизована інтеграція зовнішніх контекстних даних через API – практична відмінність системи від усіх розглянутих аналогів. Feature Engineering на основі зовнішніх сигналів дозволяє виходити за межі внутрішньої звітності

компанії і знижує похибку прогнозу для товарів із вираженою залежністю від зовнішніх умов.

Зовнішні дані система отримує з двох джерел:

– Open–Meteo API: безкоштовний відкритий метеорологічний сервіс з архівом з 1940 року і прогнозами на 16 днів наперед. З нього витягуються денна температура (°C), рівень опадів (мм) і УФ-індекс. Статистично значуща кореляція цих показників із попитом на сезонні категорії підтверджена у [14, 22]: температура визначає продажі кліматичної техніки та одягу, опади – товарів для дому і спортінвентарю;

– Wikimedia REST API: відкритий сервіс без автентифікації і API-ключів, що надає щоденну статистику переглядів статей Вікіпедії за заданим ключовим словом. Дані нормуються до шкали 0–100 і використовуються як випереджаючий індикатор: зростання переглядів передує зростанню продажів на 1–2 тижні.

Усі зовнішні ознаки проходять три етапи попередньої обробки: синхронізація часових шкал з гранулярністю ряду продажів через ресемплінг; мін-макс нормалізація до діапазону [0, 1]:

$$x_{norm} = \frac{(x - x_{min})}{(x_{max} - x_{min})}. \quad (2.6)$$

формування лагових копій із зміщенням lag ∈ {1, 7, 14} днів для врахування відкладеного ефекту зовнішніх факторів на споживчу поведінку. Сформований вектор ознак є єдиним вхідним представленням для обох моделей – це забезпечує коректне порівняння їх якості за ідентичних умов навчання (реалізацію наведено у додатку А.3).

2.1.6 Метрики оцінки точності прогнозів та адекватності моделей

Для порівняння прогнозів обрано чотири метрики, що охоплюють абсолютні і відносні показники похибки та узагальнену оцінку пояснювальної здатності моделі. Усі метрики обчислюються на тестовій вибірці – останніх 20% часового

ряду. Hold-out validation виключає витік інформації з майбутніх даних і гарантує оцінку на спостереженнях, яких модель не бачила під час навчання [23].

MAE (Mean Absolute Error) – рівномірно враховує всі відхилення незалежно від їхньої величини:

$$MAE = (\frac{1}{n}) \times \sum_{t=1}^{n} |y_t - \hat{y}_t|, \quad (2.7)$$

RMSE (Root Mean Square Error) – через квадратичне зважування сильніше штрафує великі відхилення – властивість критична для запобігання логістичним збоям через різкі хибні прогнози [24]:

$$RMSE = \sqrt{((\frac{1}{n}) \times \sum_{t=1}^{n} (y_t - \hat{y}_t)^2)}, \quad (2.8)$$

MAPE (Mean Absolute Percentage Error) – основна метрика порівняння моделей у роботі. Відсоткове вираження похибки робить її найбільш інтерпретованою для бізнес-аналітики [25]: відхилення від реального попиту оцінюється без прив'язки до абсолютного масштабу ряду:

$$MAPE = (\frac{100\%}{n}) \times \sum_{t=1}^{n} \frac{|y_t - \hat{y}_t|}{y_t}, \quad (2.9)$$

R^2 (Coefficient of Determination) – характеризує частку дисперсії цільової змінної, пояснену моделлю відносно базового прогнозу у вигляді простого середнього:

$$R^2 = 1 - \frac{\sum_t (y_t - \hat{y}_t)^2}{\sum_t (y_t - \bar{y})^2}, \quad (2.10)$$

де $\bar{y}$ – середнє значення фактичного ряду. Значення $R^2 \in [0, 1]$, де 1 відповідає ідеальному прогнозу; від'ємне значення свідчить про те, що модель гірша за просте середнє.

Зведену характеристику метрик і інтерпретаційних орієнтирів для e-commerce наведено у таблиці 2.2.

Таблиця 2.2 – Метрики оцінки точності прогнозів

Метрика	**Формула**	**Інтерпретація**	**Орієнтир для e– commerce**
MAE	$(\frac{1}{n}) \times \sum_{t=1}^{n} \lvert y_t - \hat{y}_t \rvert$	Середня абсолютна похибка	< 10% від середнього обсягу продажів
RMSE	$\sqrt{((\frac{1}{n}) \times \sum_{t=1}^{n} (y_t - \hat{y}_t)^2)}$	Квадратичне зважування	< 15% – прийнятно; > 25% – модель потребує корекції
MAPE	$(\frac{100\%}{n}) \times \sum_{t=1}^{n} \frac{\lvert y_t - \hat{y}_t \rvert}{y_t}$	Відносна похибка у відсотках	< 10% – відмінно; 10–20% – прийнятно; > 20% – незадовільно

Точнісні метрики фіксують фізичне відхилення прогнозованих значень від фактичних продажів. Повна оцінка моделі цим не обмежується – потрібна ще діагностика залишків на статистичну адекватність. Три відповідні критерії зведено у таблиці 2.3.

Таблиця 2.3 – Критерії адекватності моделі

Критерій/Метрика	**Визначення / формула**	**Зміст критерію**	**Умова адекватності**
Льюнг–Бокс	Перевірка гіпотези H_0	Оцінює залишки на відповідність «білого шуму»	p > 0.05: залишки випадкові, модель адекватна
Середнє залишків	$\bar{e} = \frac{1}{n} \sum_{t=1}^{n} e_t$	Перевірка наявності систематичного зсуву	$\bar{e} \approx 0$ – відсутнє систематичне зміщення прогнозу
Стандартне відхилення залишків	$\sigma_e = \sqrt{\frac{1}{n} \sum_{t=1}^{n} (e_t - \bar{e})^2}$	Оцінює рівномірність та стабільність розподілу похибок	Мале відносно середнього y_t – похибки рівномірно розподілені
R^2	$1 - \frac{\sum_t (y_t - \hat{y}_t)^2}{\sum_t (y_t - \bar{y})^2}$	Оцінює частку дисперсії часового ряду моделлю	> 0.80 – модель адекватна

Метрики таблиці 2.2 характеризують числову точність прогнозу, критерії таблиці 2.3 підтверджують або спростовують статистичну коректність моделі – повноцінна оцінка спирається на обидві групи.

2.2 Технологічний стек розроблюваної системи

Вибір технологічного стеку безпосередньо визначає продуктивність, масштабованість і зручність супроводу системи. Застосунок поєднує три принципово різні задачі – ML-обчислення, обробку зовнішніх даних і інтерактивний інтерфейс – тому технології для кожного рівня підбирались під конкретні функціональні вимоги, а не за загальними міркуваннями популярності.

Критерії відбору: сумісність компонентів між собою, якість документації, активна спільнота розробників, відкритий код і можливість розширення функціональності без переписування архітектури.

2.2.1 Загальна архітектура системи

Система реалізована за трирівневою клієнт-серверною архітектурою з чітким розподілом відповідальності між компонентами. Взаємодія між рівнями – через стандартизований REST API [26] з обміном даними у форматі JSON.

Клієнтський рівень (React SPA) надсилає запити до серверного (FastAPI), який керує ML-ядром і базою даних. Загальну структуру рівнів наведено у таблиці 2.4. Трирівнева схема дає конкретну практичну перевагу: зміни в одному компоненті не вимагають переробки решти. Клієнтська, серверна і аналітична частини розвиваються незалежно – це спрощує як поточний супровід, так і подальше масштабування системи.

Таблиця 2.4 – Архітектурні рівні та технології системи

Рівень	**Компонент**	**Функція**	**Технологія**
Клієнтський	SPA– інтерфейс	Відображення дашборду, графіків та рекомендацій	React + TypeScript + Recharts
Серверний	REST API	Обробка HTTP– запитів, маршрутизація, авторизація	Python + FastAPI
ML– ядро	Модуль прогнозування	Навчання моделей, генерація прогнозів, SHAP– аналіз	Prophet + TensorFlow + SHAP

Кінець таблиці 2.4

Рівень	Компонент	Функція	Технологія
Дані	Сховище	Збереження продажів, прогнозів, параметрів моделей	Supabase (PostgreSQL)
Інтеграція	Зовнішні API	Отримання погодних даних та індексів пошукових трендів	Open– Meteo + Wikimedia REST API

2.2.2 Серверна частина: Python та FastAPI

Python обрано основною мовою серверної частини і ML-модуля з конкретної причини: жодна інша сучасна мова не має порівнянної екосистеми бібліотек для науки про дані та машинного навчання. Бібліотеки Prophet, TensorFlow, Pandas, Scikit-learn і SHAP мають нативну Python-підтримку і активно розвиваються – переписування пайплайну на іншу мову означало б або втрату цих інструментів, або суттєві накладні витрати на інтеграцію.

FastAPI – асинхронний вебфреймворк для побудови REST API на базі стандарту ASGI. Порівняно з Flask [27] і Django REST Framework, FastAPI [28] обробляє запити асинхронно, що дає вищу продуктивність під навантаженням, і автоматично генерує інтерактивну документацію OpenAPI/Swagger [29] на основі анотацій типів Python. Строга типізація через Pydantic v2 додатково знижує кількість помилок на межі клієнт–сервер: невалідний запит відхиляється на рівні валідації до виконання бізнес-логіки. Це безпосередньо скорочує час на тестування і клієнтську інтеграцію API.

2.2.3 ML – бібліотеки та база даних

ML-модуль будується на п'яти спеціалізованих відкритих бібліотеках, кожна з яких закриває конкретну задачу пайплайну:

– Prophet 1.1.5 (Meta) реалізує адитивне прогнозування, математичний апарат якого описано у підрозділі 2.1. Бібліотека надає Python API для навчання моделі і візуалізації компонент (додаток Б.1);

– TensorFlow/Keras 2.15 (Google) [30] забезпечує побудову і навчання LSTM-архітектури. Keras як високорівневий API TensorFlow дозволяє декларативно описувати архітектуру мережі – код стає суттєво коротшим порівняно з чистим TensorFlow;

– Pandas 2.1 [31] обробляє табличні дані у форматі DataFrame: завантаження CSV з історією продажів, ресемплінг зовнішніх ознак, формування лагових копій і підготовка навчальних вибірок;

– Scikit-learn 1.3 [32] обчислює метрики якості прогнозів (MAE, RMSE, R^2) і виконує мін-макс нормалізацію ознак через клас MinMaxScaler;

– SHAP 0.43 оцінює внесок кожної ознаки у прогноз через значення Шеплі – забезпечує пояснюваність ML-моделей на рівні окремих предикторів [6].

Для зберігання даних обрано Supabase – платформу з відкритим кодом на базі PostgreSQL [33] з вбудованим REST API, системою авторизації і веб-консоллю. Перевага над Firebase [34] і MongoDB Atlas визначається трьома факторами: PostgreSQL ефективно обробляє запити до часових рядів з фільтрацією за датою; автоматично згенерований REST API для всіх таблиць знімає частину навантаження із серверного рівня; безкоштовний tier достатній для розробки і демонстрації прототипу.

2.2.4 Клієнтська частина: React та TypeScript

Клієнтська частина реалізована як SPA на основі React 18 [35] з реактивним оновленням інтерфейсу через Virtual DOM і компонентною архітектурою на хуках. TypeScript [36] замість JavaScript – принципове технічне рішення: статична типізація виявляє помилки на етапі компіляції, а не у runtime, що критично для складних форм введення параметрів прогнозування і обробки відповідей API.Recharts [37] обрано для візуалізації через нативну інтеграцію з React-компонентами і підтримку інтерактивних SVG-графіків: лінійні графіки фактичних і прогнозованих продажів, теплові карти сезонності, стовпчасті діаграми метрик.

Адаптивний дизайн будується на Tailwind CSS – utility-first фреймворку, де стилі формуються безпосередньо в JSX-розмітці без окремих CSS-файлів.

2.2.5 Зовнішні API та обґрунтування стеку

Інтеграція зовнішніх джерел реалізована на серверному рівні через асинхронні HTTP-запити бібліотекою httpx – це виключає блокування основного потоку FastAPI при очікуванні відповіді від зовнішніх сервісів.

Open-Meteo API не потребує реєстрації і API-ключа. Відповідь надходить у JSON і автоматично конвертується у Pandas DataFrame для подальшої обробки.

Wikimedia REST API також не вимагає автентифікації. Запит формується з параметрами назви статті і часового діапазону; відповідь конвертується у Pandas DataFrame з нормалізацією до шкали 0–100.

Обраний стек відповідає вимогам підрозділу 1.3: весь інструментарій відкритий, усі сервіси мають безкоштовний рівень, інфраструктурні витрати мінімальні. Зведену характеристику технологій за рівнями системи наведено у таблиці 2.5.

Таблиця 2.5 – Технологічний стек розроблюваної системи

Рівень системи	**Технологія**	**Версія**	**Обґрунтування вибору**
Backend	Python + FastAPI	3.11 / 0.104	Асинхронна обробка запитів, автогенерація OpenAPI– документації, нативна підтримка ML– бібліотек
ML– модуль	Prophet + TensorFlow/Keras	1.1.5 / 2.15	Prophet – адитивне прогнозування з підтримкою свят; Keras – гнучка побудова LSTM– архітектур
Обробка даних	Pandas + scikit–learn	2.1 / 1.3	Стандарт галузі для маніпуляцій з табличними даними та обчислення метрик якості
Пояснюваність	SHAP	0.43	Єдиний уніфікований підхід до інтерпретації прогнозів ML– моделей (feature importance)
База даних	Supabase (PostgreSQL)	-	Відкритий код, вбудований REST API, безкоштовний хмарний хостинг, реляційна модель для часових рядів

Кінець таблиці 2.5

Рівень системи	**Технологія**	**Версія**	**Обґрунтування вибору**
Frontend	React + TypeScript	18 / 5.0	Компонентна SPA– архітектура, строга типізація, велика екосистема бібліотек
Візуалізація	Recharts	3.8.1	Нативна інтеграція з React, SVG– графіки, підтримка інтерактивних часових рядів
Стилі	Tailwind CSS	3.4	Utility– first підхід, адаптивний дизайн без написання власного CSS
Зовнішні дані	Open– Meteo + Wikimedia REST API	-	Безкоштовні відкриті сервіси без необхідності API– ключів, стабільний публічний доступ

Цей стек формує технологічну основу для реалізації, описаної у третьому розділі. Поєднання FastAPI, Prophet, LSTM і Supabase на єдиній відкритій платформі дозволяє розгорнути повноцінну систему прогнозування без ліцензійних витрат – що і є ключовою практичною перевагою над комерційними аналогами, розглянутими у підрозділі 1.2.

Висновки до розділу 2

Другий розділ обґрунтував методологічну і технологічну основу системи прогнозування попиту.

Теоретичним фундаментом визначено адитивну модель декомпозиції часового ряду $Y(t) = T(t) + S(t) + C(t) + \varepsilon(t)$. Для практичної реалізації обрано Prophet – для виявлення тренду, сезонності та ефекту акційних подій — і двошарову архітектуру LSTM для моделювання нелінійних залежностей. Фінальний прогноз формується методом комбінування Bates–Granger (1969): $\hat{y} = \alpha \cdot \text{Prophet} + (1-\alpha) \cdot \text{LSTM}$, де коефіцієнт α визначається мінімізацією MAE на валідаційній вибірці окремо для кожного товару. Такий підхід знижує загальну дисперсію прогнозу порівняно з кожною моделлю окремо. Зовнішні сигнали Open-Meteo API і Wikimedia Pageviewsзбагачують вхідний вектор ознак як випереджаючі індикатори попиту. Оцінювання якості моделей спирається на дві групи критеріїв:

метрики точності прогнозів (MAE, RMSE, MAPE) і критерії адекватності залишків (тест Льюнга–Бокса, середнє залишків, стандартне відхилення). Інтерпретованість результатів забезпечується методом SHAP.

Технологічний стек: Python/FastAPI на серверному рівні, Prophet і TensorFlow/Keras у ML-ядрі, Supabase/PostgreSQL як реляційне сховище, React з TypeScript і Recharts на клієнтській стороні. Весь стек відкритий і безкоштовний — це безпосередньо усуває головне обмеження розглянутих аналогів і робить систему реально доступною для МСБ. Результати розділу сформували повну методологічну і технологічну базу для практичної реалізації, описаної у третьому розділі.

3 РЕАЛІЗАЦІЯ СИСТЕМИ ПРОГНОЗУВАННЯ ПОПИТУ ТА АНАЛІЗ РЕЗУЛЬТАТІВ

Третій розділ охоплює практичну реалізацію системи і аналіз отриманих результатів. Підрозділ 3.1 описує структуру системи: схему бази даних, перелік API-маршрутів і формат вхідних даних. Підрозділ 3.2 містить наскрізну демонстрацію роботи на прикладі товарної позиції «Ізотонічний напій» – від завантаження і автоматичної діагностики часового ряду до прогнозу, аналізу важливості ознак і логістичних рекомендацій. Підрозділ 3.3 наводить порівняльний аналіз точності моделей за метриками точності та якості прогнозу з рекомендаціями щодо практичного впровадження.

3.1 Опис вхідних даних та структури системи

Система реалізована як вебзастосунок із трьома логічно відокремленими рівнями: клієнтський інтерфейс, серверний застосунок і рівень зберігання даних та машинного навчання. Підрозділ описує структуру вхідних даних, схему бази даних, перелік API-ендпоінтів і конвеєр обробки даних.

3.1.1 Трирівнева архітектура системи

Систему побудовано за трирівневою клієнт-серверною моделлю (рис. 3.1): кожен рівень виконує чітко визначені функції і взаємодіє з сусідніми через стандартизовані інтерфейси.

Клієнтський рівень – SPA на React 18.3 з TypeScript і збірником Vite 8. Стилізація – Tailwind CSS 4. Глобальний стан застосунку керується через Zustand 5, який генерує менше шаблонного коду порівняно з Redux. HTTP-взаємодія з сервером реалізована через Axios, візуалізація прогнозних графіків і метрик – через Recharts 3.8. Усі запити до серверного API надсилаються з Bearer-токеном авторизації (реалізацію компонентів наведено у додатках Г.1–Г.2).

Серверний рівень побудовано на FastAPI 0.104 для Python 3.11. Фреймворк обрано через нативну підтримку ASGI, автоматичну генерацію OpenAPI-документації і строгу типізацію через Pydantic v2. Сервер запускається під Uvicorn 0.24 і обслуговує чотири групи маршрутів: автентифікація (/api/auth), товари (/api/products), продажі (/api/sales) і прогнози (/api/forecasts). Автентифікація реалізована через JWT-токени з BCrypt-хешуванням паролів (додаток В.1). Ресурсомісткі ML-операції виконуються через механізм BackgroundTasks: клієнт отримує відповідь 202 Accepted негайно, конвеєр ML –асинхронно (додатки В.3–В.4).

Третій рівень охоплює базу даних і ML-підсистему. Сховище – Supabase на базі PostgreSQL з вбудованим REST API і Row Level Security. ML-підсистема реалізована в модулі app/ml і складається з трьох компонентів: Prophet 1.1.5 [9], LSTM на TensorFlow-CPU 2.15 і модуль комбінування прогнозів, що визначає оптимальний α мінімізацією MAE на валідаційній вибірці. Зовнішні ознаки надходять з Open-Meteo і Wikimedia REST API (реалізацію конвеєра наведено у додатку А.1). Взаємодія між рівнями – виключно через HTTP/HTTPS. Клієнт надсилає REST-запити до сервера, сервер звертається до Supabase через supabase-py ≥2.7, зовнішні API-виклики виконуються асинхронно через httpx 0.27 з таймаутом 30 секунд. CORS-політика обмежує запити адресою клієнтського застосунку. Загальну схему архітектури наведено на рис. 3.1.

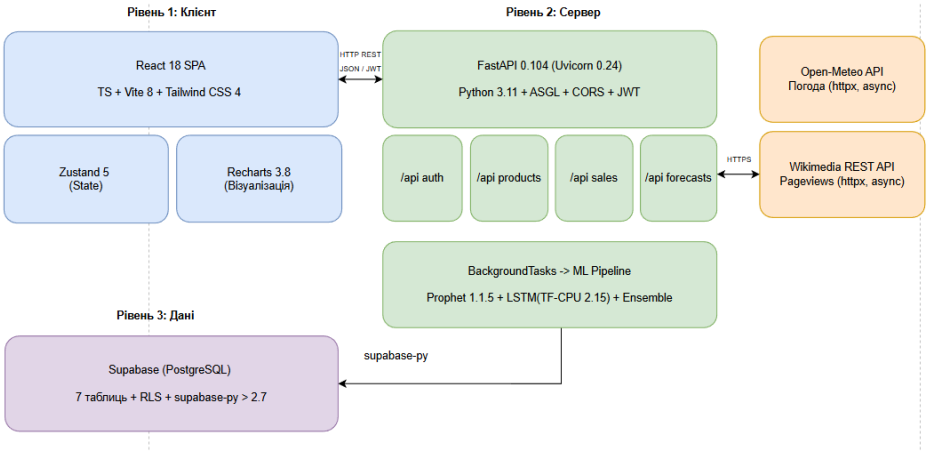


Рисунок 3.1 – Трирівнева архітектура системи прогнозування попиту

3.1.2 Структура бази даних

Реляційна схема бази даних PostgreSQL (Supabase) складається з семи таблиць. Мультикористувацька ізоляція забезпечується прив'язкою кожного запису товару до конкретного користувача через зовнішній ключ user_id. На всіх дочірніх таблицях налаштовано ON DELETE CASCADE – цілісність даних при видаленні батьківських записів гарантована на рівні схеми.

Центральні таблиці – users і products. Users зберігає облікові дані з BCrypt-хешем пароля. Products містить метадані товару: географічні координати (latitude, longitude) для запитів до Open-Meteo API і ключове слово search_keyword для Wikimedia Pageviews API. Координати за замовчуванням відповідають м. Київ (50.4501° пн.ш., 30.5234° сх.д.).

Sales зберігає хронологічний ряд обсягів продажів із щоденною гранулярністю. Унікальний обмежувач на пару (product_id, date) і механізм UPSERT виключають дублювання при повторному завантаженні CSV-файлів.

Forecasts – головний запис прогнозу. Містить параметри запуску (model_type, horizon_days), статус виконання (running / completed / failed), ваговий коефіцієнт α і зведені метрики якості (MAE, RMSE, MAPE, R^2) (реалізацію обчислення наведено у додатку А.4). Поля prophet_metrics і lstm_metrics типу JSONB зберігають метрики базових моделей для ендпоінту порівняння; prophet_components – компоненти декомпозиції Prophet для подальшої візуалізації.

Forecast_values містить покрокові прогнозні значення з межами довірчого інтервалу 95% (lower_bound, upper_bound). Shap_values зберігає SHAP-важливість ознак, обчислену через surrogate-модель GradientBoostingRegressor [39]. External_features слугує кешем зовнішніх даних – погоди і пошукового інтересу; оновлюється при кожному запуску прогнозу операцією UPSERT по парі (product_id, date), що виключає зайві запити до зовнішніх API при повторному прогнозуванні.

Повну ER-діаграму зв'язків між таблицями наведено на рис. 3.2.

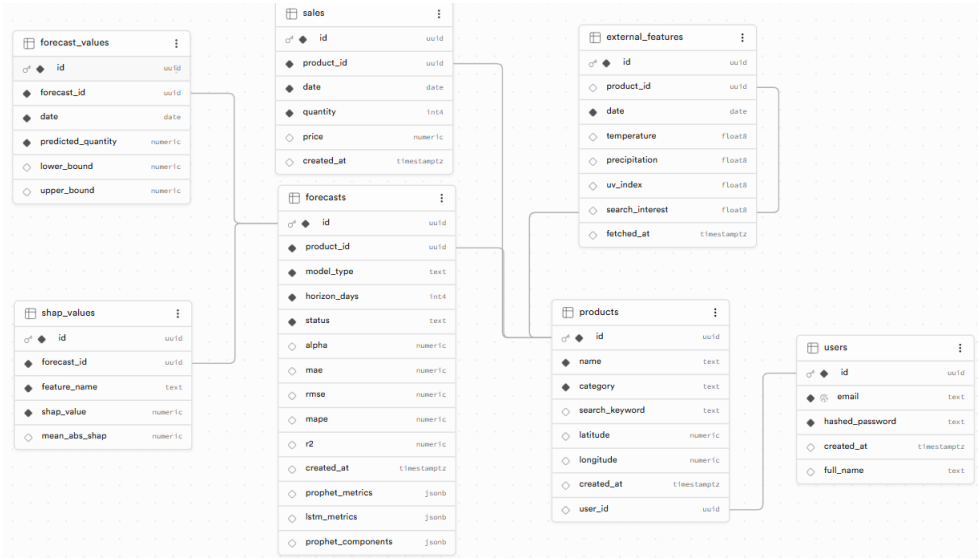


Рисунок 3.2 – ER– діаграма бази даних системи прогнозування попиту

3.1.3 REST API системи

Серверний застосунок надає 20 REST API ендпоінтів, розподілених на чотири функціональні групи. Усі маршрути, крім реєстрації та входу, захищені JWT-автентифікацією через заголовок Authorization: Bearer <token>. Формат обміну – JSON. Специфікація автоматично генерується FastAPI у форматах OpenAPI 3.0 і Swagger UI за адресою /docs.

Група автентифікації (3 ендпоінти) закриває реєстрацію нових користувачів, вхід з отриманням JWT-токена і отримання профілю поточного авторизованого користувача. Успішна реєстрація повертає HTTP 201; спроба зареєструвати вже існуючу адресу – HTTP 400 з описом помилки.

Група управління товарами (4 ендпоінти) реалізує повний CRUD для сутності Product. При створенні товару поле user_id витягується автоматично з JWT-токена – ізоляція даних між користувачами забезпечується на рівні сервера без додаткових перевірок. Видалення товару каскадно знищує всі пов'язані записи продажів, прогнозів і зовнішніх ознак.

Група управління продажами (4 ендпоінти) охоплює завантаження CSV, отримання переліку записів, видалення даних і отримання статистики датасету. Ендпоінт статистики повертає описові характеристики набору: кількість рядків,

діапазон дат, середнє, стандартне відхилення, мінімум і максимум, ознаку виявленої сезонності і прапорець достатності даних для запуску прогнозу.

Група управління прогнозами (9 ендпоінтів) – найбільш розгалужена. POST /api/forecasts/run/{product_id} приймає параметри horizon_days, model_type (prophet / lstm / ensemble), include_weather і include_trends, негайно повертає forecast_id зі статусом running (HTTP 202) і запускає ML-конвеєр у фоновому завданні через BackgroundTasks. Після завершення доступні: покрокові значення прогнозу з довірчим інтервалом 95%, SHAP-важливість ознак, автоматичні бізнес-рекомендації, компоненти декомпозиції Prophet, порівняння метрик між моделями і CSV-експорт результатів. Повний перелік ендпоінтів наведено у таблиці 3.1.

Таблиця 3.1 – Перелік REST API ендпоінтів системи

№	Метод	Шлях	Опис	Авт.
1	POST	/api/auth/register	Реєстрація нового користувача	-
2	POST	/api/auth/login	Вхід в систему, отримання JWT–токена	-
3	GET	/api/auth/me	Профіль поточного авторизованого користувача	+
4	GET	/api/products	Список товарів поточного користувача	+
5	POST	/api/products	Створення нового товару	+
6	GET	/api/products/{id}	Отримання деталей конкретного товару	+
7	DELETE	/api/products/{id}	Видалення товару (каскадне видалення даних)	+
8	POST	/api/sales/upload/{product_id}	Завантаження CSV з даними продажів (UPSERT)	+
9	GET	/api/sales/{product_id}	Отримання записів продажів товару	+
10	DELETE	/api/sales/{product_id}	Видалення всіх записів продажів товару	+
11	GET	/api/sales/{product_id}/stats	Статистика та аналіз датасету	+
12	POST	/api/forecasts/run/{product_id}	Запуск прогнозування (async, HTTP 202)	+
13	GET	/api/forecasts/{product_id}	Список прогнозів для товару	+
14	GET	/api/forecasts/{forecast_id}/status	Статус виконання та зведені метрики	+

Кінець таблиці 3.1

№	Метод	Шлях	Опис	Авт.
15	GET	/api/forecasts/{forecast_id}/values	Покрокові значення прогнозу + ДІ 95%	+
16	GET	/api/forecasts/{forecast_id}/shap	SHAP– важливість ознак (топ– 12)	+
17	GET	/api/forecasts/{forecast_id} /recommendations	Бізнес– рекомендації з управління запасами	+
18	GET	/api/forecasts/{forecast_id}/export	Експорт результатів прогнозу у форматі CSV	+
19	GET	/api/forecasts/{forecast_id}/components	Компоненти декомпозиції Prophet	+
20	GET	/api/forecasts/{forecast_id}/comparison	Порівняння метрик трьох моделей	+

3.1.4 Формат вхідних даних

Вхідні дані системи – CSV-файл з історичними щоденними обсягами продажів, що завантажується через веб-інтерфейс. Для верифікації підходу і тестування архітектури моделей сформовано два синтетичні набори даних – «Sports_drink» і «Cold_medicine», – кожен з яких охоплює 1095 записів за 2021–2023 роки (три повні календарні роки щоденних спостережень).

Обсяг перевищує мінімальні архітектурні вимоги обох алгоритмів: Prophet потребує щонайменше 14 рядків для ідентифікації тижневих компонент, LSTM – мінімум 90 рядків через специфіку формування послідовностей із вікном LOOKBACK = 30 кроків. Трирічний горизонт дозволяє виявити тижневу і річну сезонність, сформувати репрезентативний валідаційний набір для оптимізації гіперпараметрів і отримати статистично надійні SHAP-оцінки.

Файл повинен відповідати таким вимогам: кодування UTF-8, роздільник – кома, перший рядок – заголовок. Обов'язкові поля: date (ISO 8601, YYYY-MM-DD) і quantity (ціле невід'ємне число). Поле price – необов'язкове, допускає NaN. Приклад коректної структури:

```
date,quantity,price
2023-01-01,99,29.99
2023-01-02,124,29.99
```

2023-01-03,112,

Модуль validate_csv() перевіряє наявність обов'язкових колонок, відповідність дат формату ISO, числовий тип quantity і відсоток пропусків. Критичні дефекти – відсутність колонок або пошкоджена структура – перривають транзакцію і повертають HTTP 422 з деталізованим описом. Нефатальні попередження (малий обсяг, поодинокі пропуски) передаються клієнту у полі warnings.

Пропущені значення quantity відновлюються лінійною інтерполяцією через pandas.Series.interpolate [31]. Запис у сховище виконується як атомарна операція UPSERT за ключем (product_id, date) – дублювання і зашумлення ряду при повторних завантаженнях виключено.

Конвеєр формує п'ять типів вихідних даних:

– покрокові прогнозні значення з межами довірчого інтервалу 95%;

– метрики точності MAE, RMSE, MAPE і адекватності R^2 окремо для Prophet, LSTM і комбінування;

– SHAP-вектори важливості ознак;

– компоненти декомпозиції Prophet (тренд, тижнева і річна сезонність);

– автоматичні бізнес-рекомендації щодо управління запасами.

За потреби результати експортуються у CSV через /api/forecasts/{forecast_id}/export.

3.1.5 Попередній аналіз часового ряду

Вибір методу моделювання визначається характеристиками вхідного ряду – застосування моделі без попередньої діагностики даних систематично знижує якість прогнозу. Для товарної категорії Sports_drink проведено чотири види аналізу: тест ADF на стаціонарність, STL-декомпозицію, аналіз ACF і PACF.

Вихідний ряд щоденних продажів Sports_drink охоплює період січень 2021 – січень 2024 (рис. 3.3). Ковзне середнє з вікном 30 днів фіксує виражену річну

хвилю: пік у червні–серпні, спад у зимовий період – характерна сезонна структура для ізотонічних напоїв.

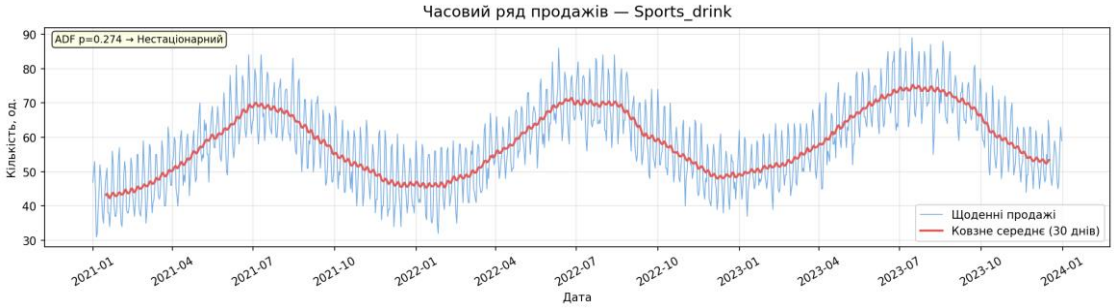


Рисунок 3.3 – Візуалізація вихідного часового ряду продажів Sports_drink

ADF-тест повернув p = 0.274 – вище критичного рівня 0.05. Ряд нестаціонарний [16]: математичне очікування і дисперсія змінюються в часі. Моделі ARIMA без попереднього диференціювання для таких рядів неефективні. Prophet нативно працює з нестаціонарними трендами через кусково-лінійні функції з гнучкими точками зламу – це і робить його методологічно виправданим вибором.

STL-декомпозиція (рис. 3.4) розклала ряд на три ортогональні складові. Тренд підтверджує нелінійне довгострокове зростання з циклічними макро-коливаннями. Сезонна складова фіксує стабільну амплітуду ±10 одиниць з вираженою тижневою частотою. Залишки розподілені навколо нуля, але містять локальні стохастичні сплески високої дисперсії – сигнал про вплив зовнішніх збурень, що додатково обґрунтовує підключення екзогенних регресорів (погодні дані, медіа-тренди).

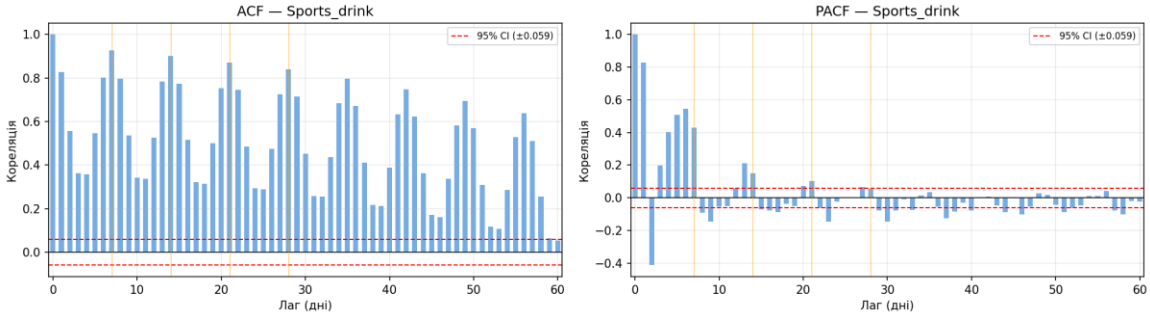


Рисунок 3.4 – STL-декомпозиція часового ряду Sports_drink

ACF (рис. 3.5) демонструє повільне згасання з періодичними піками на лагах, кратних 7 (7, 14, 21, 28 днів) – математичне свідчення глибокої довгострокової пам'яті в даних. Саме ця структура обґрунтовує LSTM: механізм гейтування дозволяє утримувати контекст віддалених лагів без ризику затухання градієнта. PACF фіксує значущі лаги на позиціях 1–5 – авторегресійна залежність ближнього порядку, яку LSTM апроксимує в межах загальної нелінійної функції.

Рисунок 3.5 – Функції автокореляції ACF та часткової автокореляції PACF

Чотири виявлені характеристики – нестаціонарність, мультисезонність, нелінійний тренд і тривалі лагові кореляції – виключають ефективність ізольованих моделей і обґрунтовують гібридну архітектуру. Prophet виділяє і екстраполює макро-компоненти (тренд і календарну сезонність), LSTM моделює високочастотні нелінійні залежності і лагові флуктуації. Фінальне об'єднання прогнозів виконується методом Bates–Granger (1969)через мінімізацію MAE на валідаційній вибірці.

3.2 Демонстрація роботи розробленої системи прогнозування попиту

Для підтвердження працездатності системи проведено наскрізний експеримент на часовому ряді щоденних продажів «Ізотонічний напій» (категорія

Sport). Горизонт прогнозування – 30 днів, що відповідає практичному циклу оперативного планування поставок у роздрібній торгівлі. Вибір комбінованої моделі для демонстрації обґрунтовано результатами підрозділу 3.1.5: виявлена тижнева авторегресія (домінуючий лаг lag_7 за ACF) і підтверджена мультисезонність ряду вказують саме на гібридну архітектуру як методологічно найбільш відповідну структурі даних. Результати аналогічного експерименту для «Ліків від застуди» наведено у підрозділі 3.3 у форматі порівняльного аналізу точності моделей.

3.2.1 Автентифікація

Доступ до аналітичних модулів захищено JWT-автентифікацією (рис. 3.6). Stateless-архітектура авторизації виключає несанкціонований доступ до комерційних даних і забезпечує повне розмежування між обліковими записами. Після успішного входу система ініціалізує персональне середовище аналізу і надає доступ до всіх модулів відповідно до прав облікового запису.

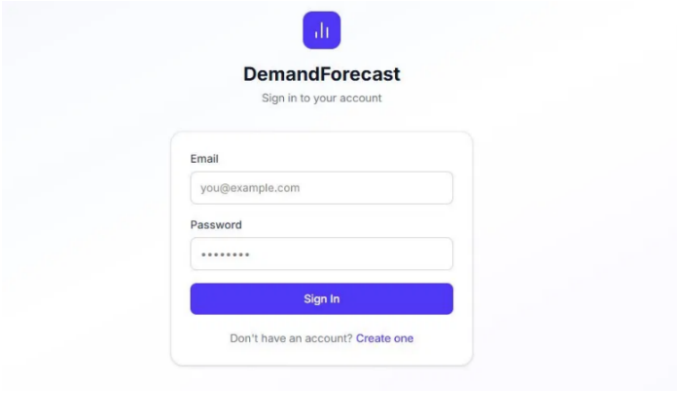


Рисунок 3.6 – Сторінка автентифікації системи DemandForecast

3.2.2 Завантаження та автоматична діагностика часового ряду

До модуля управління товарами завантажено часовий ряд «Ізотонічний напій» у форматі CSV (рис. 3.7). Система автоматично виконала парсинг дат YYYY-MM-DD, верифікацію структури файлу і первинну діагностику якості вхідних даних..

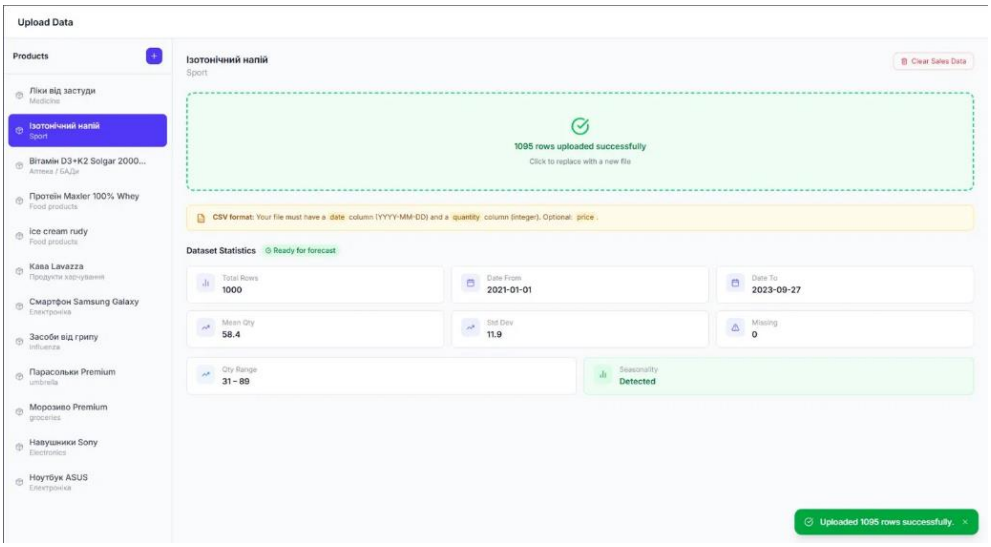


Рисунок 3.7 – Модуль управління товарами та інтерфейс завантаження даних

Автоматичний дескриптивний аналіз сформував статистичну характеристику датасету: 1 095 щоденних спостережень за 2021-01-01 – 2023-09-27; середній обсяг реалізації – 58,4 од./добу; стандартне відхилення – 11,9 од.; діапазон варіації – 31–89 од.; пропущених значень – 0. Індикатор «Seasonality: Detected» підтвердив наявність регулярних циклічних коливань – необхідну умову для декомпозиційного прогнозування. Статус «Ready for forecast» засвідчив придатність даних до моделювання без додаткової обробки.

3.2.3 Конфігурування моделей та запуск конвеєру машинного навчання

На етапі конфігурування визначено параметри експерименту (рис. 3.8): метод – «Комбінування», горизонт – 30 днів. З огляду на температурно-залежну природу попиту на спортивні напої активовано інтеграцію екзогенного сигналу через Open-Meteo API. Сигнал Wikimedia Trends не підключався: пошуковий попит не є релевантним драйвером реалізації ізотонічних напоїв.Після підтвердження конфігурації запущено ML-конвеєр: паралельне навчання Prophet і LSTM на тренувальній вибірці, підбір оптимального α мінімізацією MAE на валідаційній вибірці і формування фінального комбінованого прогнозу.

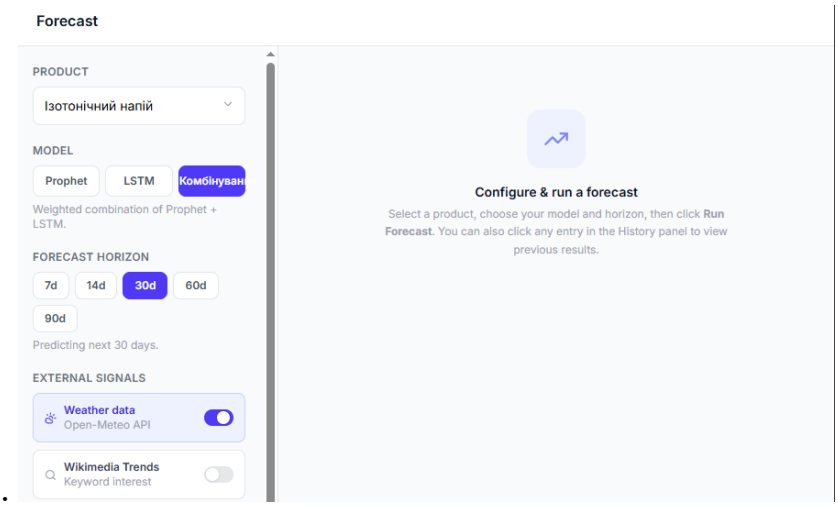


Рисунок 3.8 – Конфігурація прогнозу

3.2.4 Результати прогнозування

Після завершення конвеєру система сформувала аналітичну звітність у п'яти вкладках. Зведені метрики комбінованої моделі на тестовій вибірці: MAE = 1,80; RMSE = 2,18; MAPE = 2,61%; R^2 = 0,8481 при α = 0,4427.

Рис. 3.9 відображає 30-денний прогноз: штрихова лінія – фактичні спостереження тренувального періоду, суцільна – прогнозна траєкторія з 28 вересня, затінена область – 95% довірчий інтервал на основі стандартного відхилення залишків валідаційної вибірки. Прогнозна траєкторія відтворює тижневі коливання попиту зі збереженням загального рівня 55–80 одиниць. Вузький довірчий інтервал свідчить про низький рівень невизначеності прогнозу.

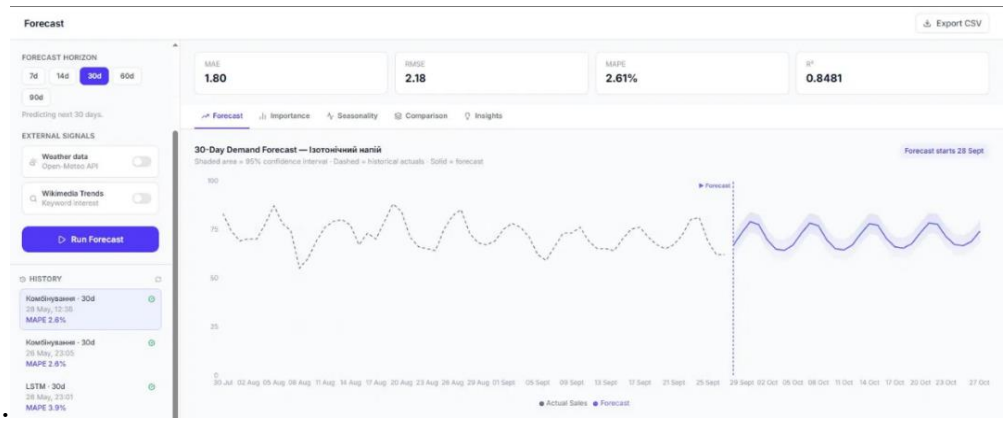


Рисунок 3.9 – Графік 30-денного прогнозу комбінованої моделі

Вкладка Importance (рис. 3.10) показала ієрархію SHAP-важливостей ознак LSTM. Домінує lag_7 зі значенням ~7,5 – майже вдвічі більше за наступну ознаку temperature (~3,2). Другий рівень важливості формують lag_14 (~3,0), day_of_week (~2,7) і rolling_7 (~2,4). Ієрархія підтверджує виражену авторегресійну природу попиту з тижневою циклічністю і помірним впливом температурного фактора – що узгоджується з фізичною природою продукту.

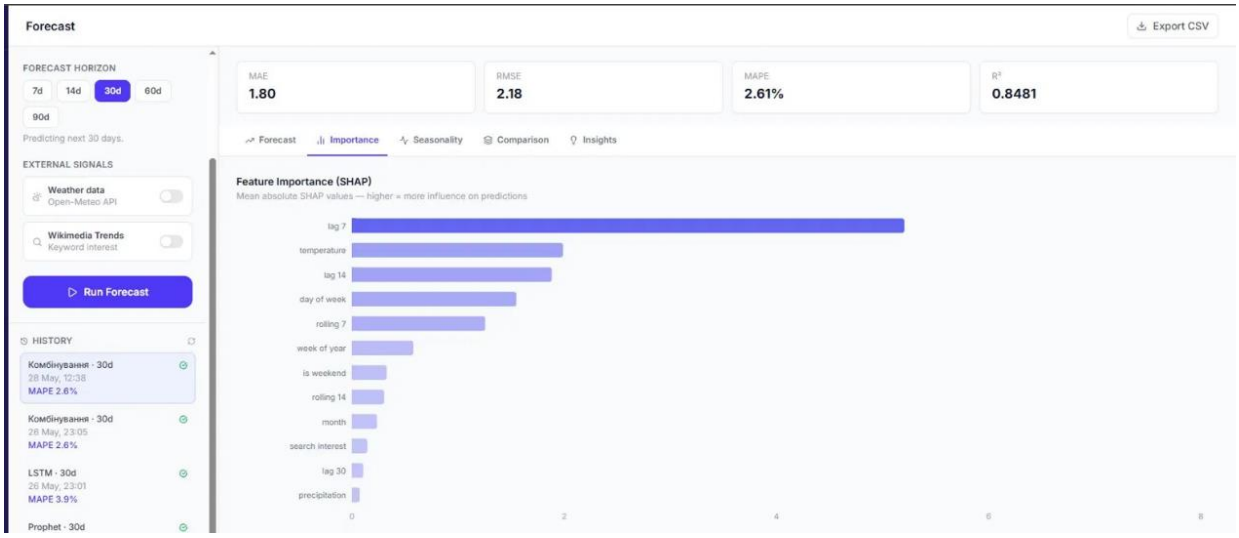


Рисунок 3.10 – Діаграма SHAP-важливості ознак для моделі LSTM

Вкладка Seasonality (рис. 3.11) відобразила три компоненти декомпозиції Prophet. Тренд тримається на стабільному рівні ~60 одиниць упродовж усього горизонту – систематичного зростання чи спадання не виявлено.

Тижнева сезонність має асиметричний профіль: субота і неділя дають позитивний ефект +7–8 одиниць, середа і вівторок – від'ємний -5–6 одиниць. Сезонний патерн у жовтні незначно спадає, що відображає осіннє зниження споживання спортивних напоїв.

Теплова карта (рис. 3.12) у розрізі «місяць × день тижня» підтверджує літньо-весняний характер попиту: найтемніші комірки зосереджені в червні–серпні, особливо у вихідні дні, – що кількісно корелює з піковими значеннями температурної ознаки у структурі SHAP.

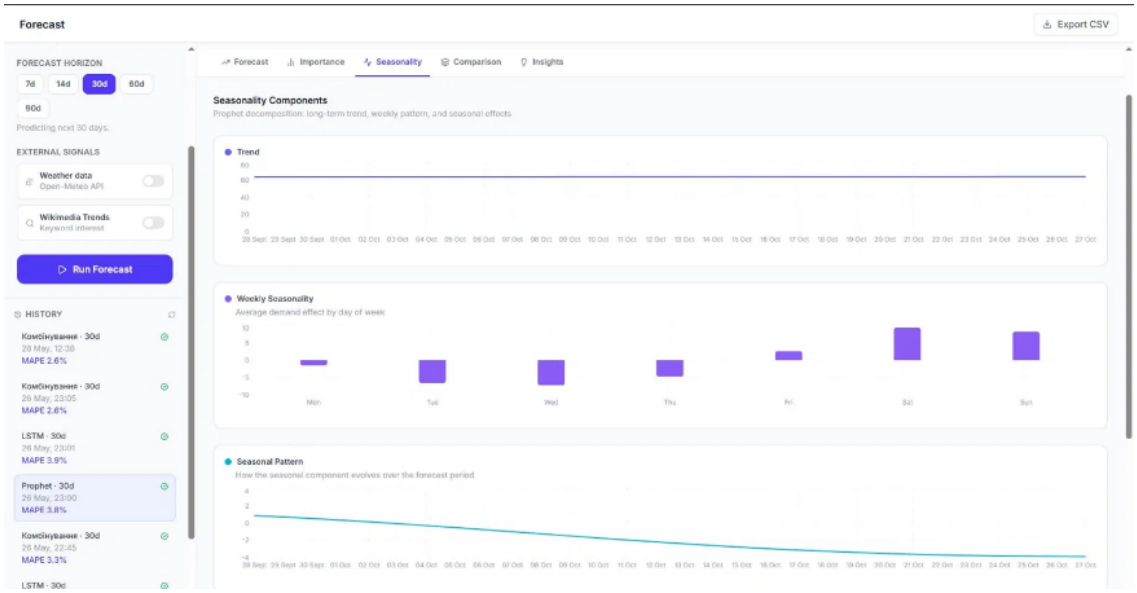


Рисунок 3.11 – Компоненти декомпозиції Prophet

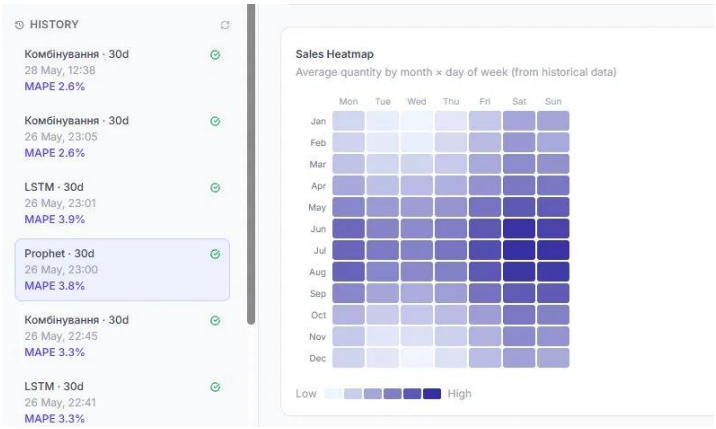


Рисунок 3.12 – Теплова карта середніх продажів у розрізі місяць × день тижня

Вкладка Comparison (рис. 3.13) містить зведені метрики трьох методів на тестовій вибірці. Комбінована модель перевершила обидві базові за всіма показниками: MAE = 1,80 проти 2,64 (Prophet) і 2,72 (LSTM); MAPE = 2,61% проти 3,83% і 3,90%. Скорочення похибки відносно найкращої базової моделі (Prophet) – 31,8% за MAE і 31,9% за MAPE.

Діагностика залишків виявила відмінність між базовими моделями: для Prophet $p = 0{,}160 > 0{,}05$ – залишки відповідають умові «білого шуму»; для LSTM $p = 0{,}032 < 0{,}05$ – у помилках присутня залишкова структура. Детальний статистичний аналіз цих результатів наведено у підрозділі 3.3.

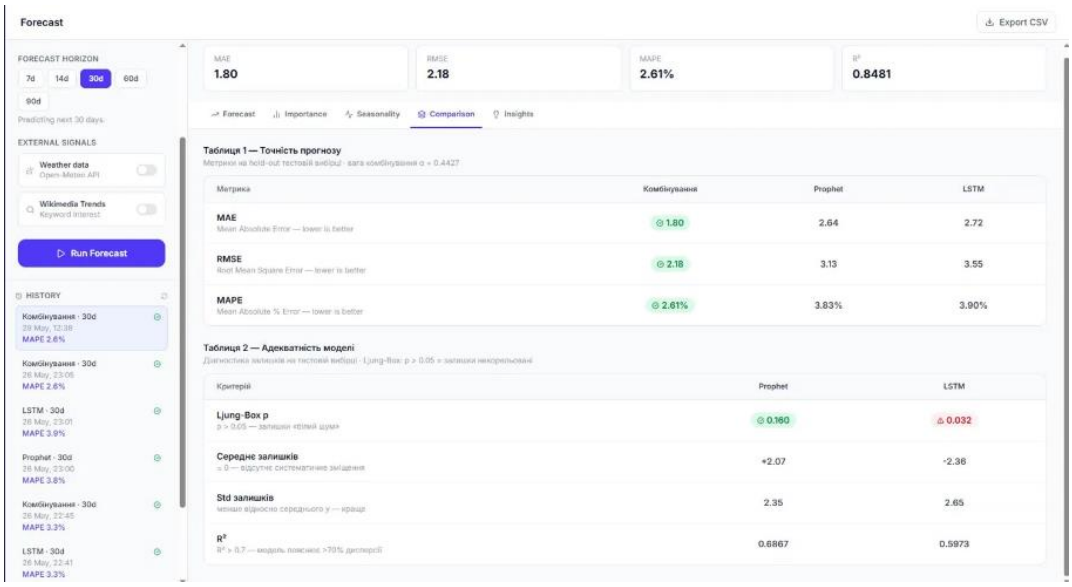


Рисунок 3.13 – Вкладка Comparison: порівняльні метрики точності та діагностика залишків моделей

3.2.5 Модуль автоматизованих бізнес-рекомендацій

Вкладка Insights (рис. 3.14) реалізує правило-базований модуль інтерпретації для підтримки логістичних рішень. Прогнозна траєкторія класифікована як «Growing demand» з приростом +32,2% на горизонті прогнозування. Рівень ризику – «Low Risk»: MAPE = 2,61% відповідає порогу достовірності для операційних логістичних рішень.На основі цієї класифікації система сформувала дві управлінські рекомендації: збільшити обсяг наступного замовлення на 15–20% відповідно до прогнозованого зростання попиту і розглянути розширення асортименту за рахунок суміжних позицій. Автоматизація інтерпретації скорочує час від отримання прогнозу до прийняття рішення без залучення аналітика.

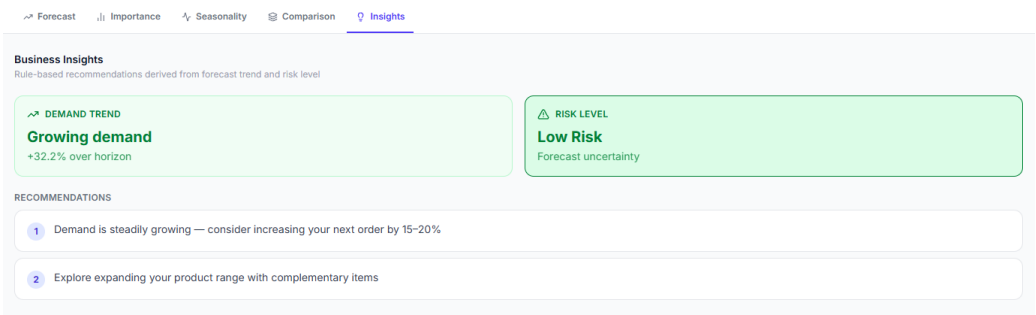


Рисунок 3.14 – Вкладка Insights: автоматизовані бізнес-рекомендації

3.2.6 Журнал прогнозів та відтворюваність результатів

Стан системи на момент проведення експерименту відображено на Dashboard (рис. 3.15): 12 зареєстрованих товарних позицій, 106 виконаних сесій прогнозування, 102 завершено успішно, середня точність по всіх товарах – 83,8% (100 − MAPE). Ці показники підтверджують, що система функціонує стабільно за межами одного тестового кейсу.

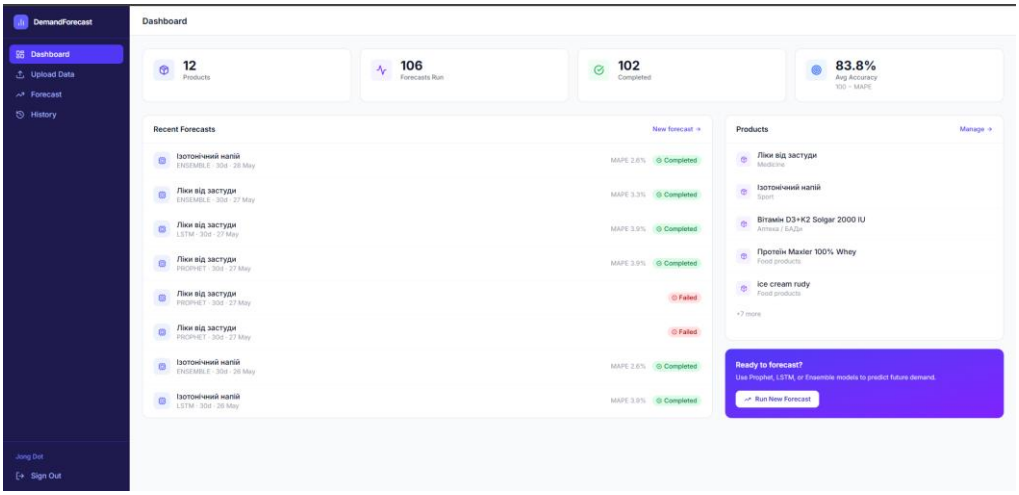


Рисунок 3.15 – Головна панель Dashboard: зведена статистика системи

Кожна сесія автоматично фіксується у журналі History (рис. 3.16) з повним набором метрик (MAE, MAPE, R^2), методом, горизонтом і датою запуску. Це забезпечує ретроспективне порівняння різних конфігурацій для однієї товарної позиції і аудитування результатів моделювання.

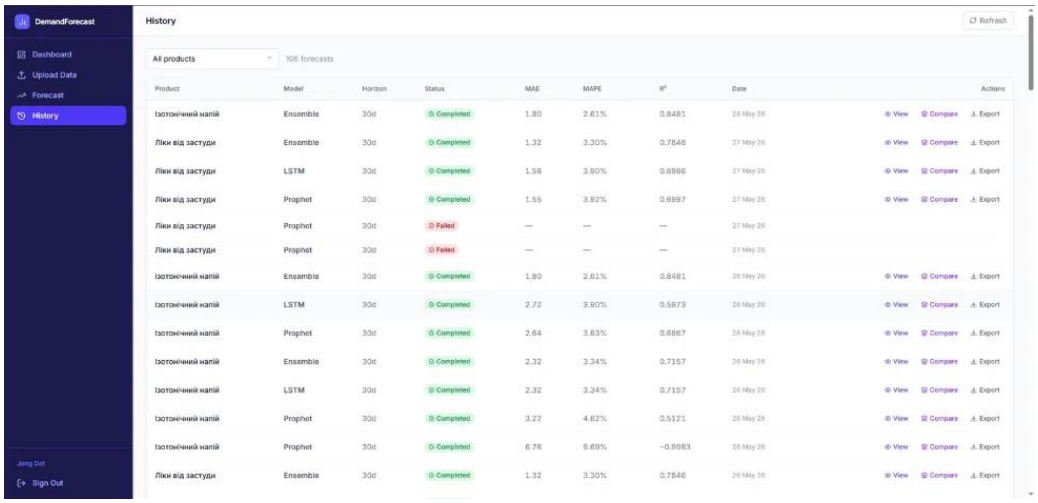


Рисунок 3.16 – Розділ History: журнал прогнозів з метриками та статусами

Для інтеграції з корпоративними ERP/SCM-системами реалізовано CSV-експорт прогнозних значень (рис. 3.17). Файл містить денні прогнози на 30-денний горизонт і завантажується безпосередньо до системи управління запасами або використовується для формування замовлень постачальникам.

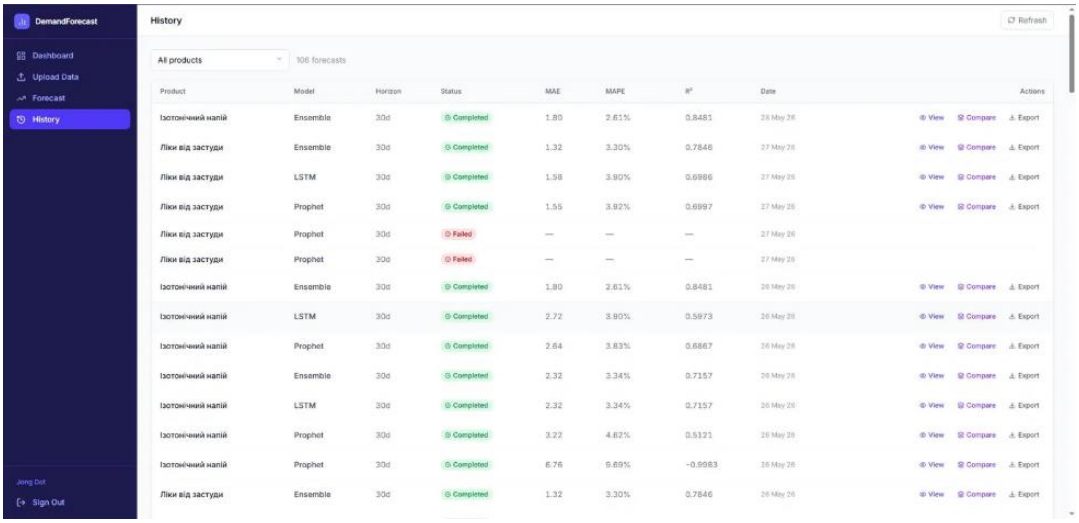


Рисунок 3.16 – Розділ History: журнал прогнозів з метриками та статусами виконання

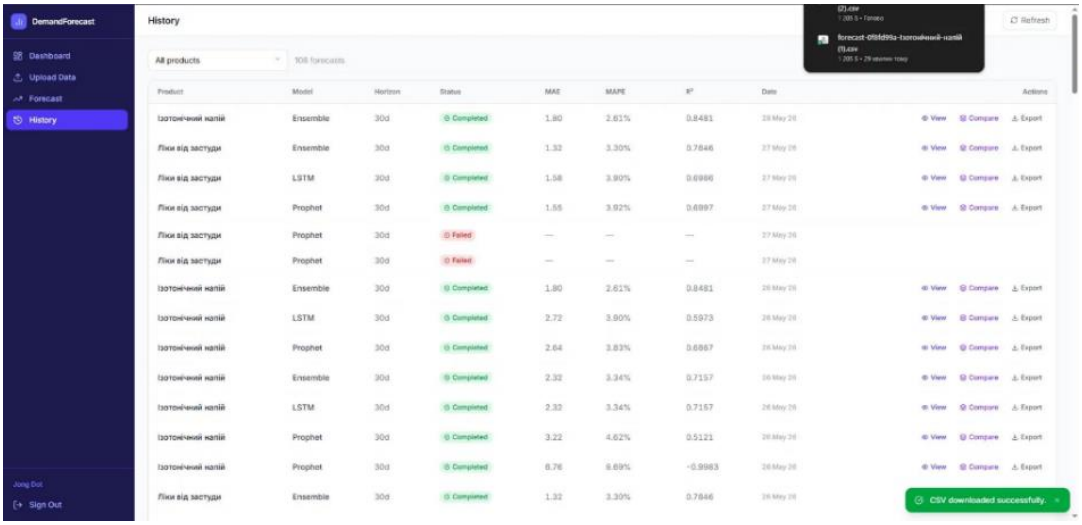


Рисунок 3.17 – Експорт результатів прогнозування у форматі CSV

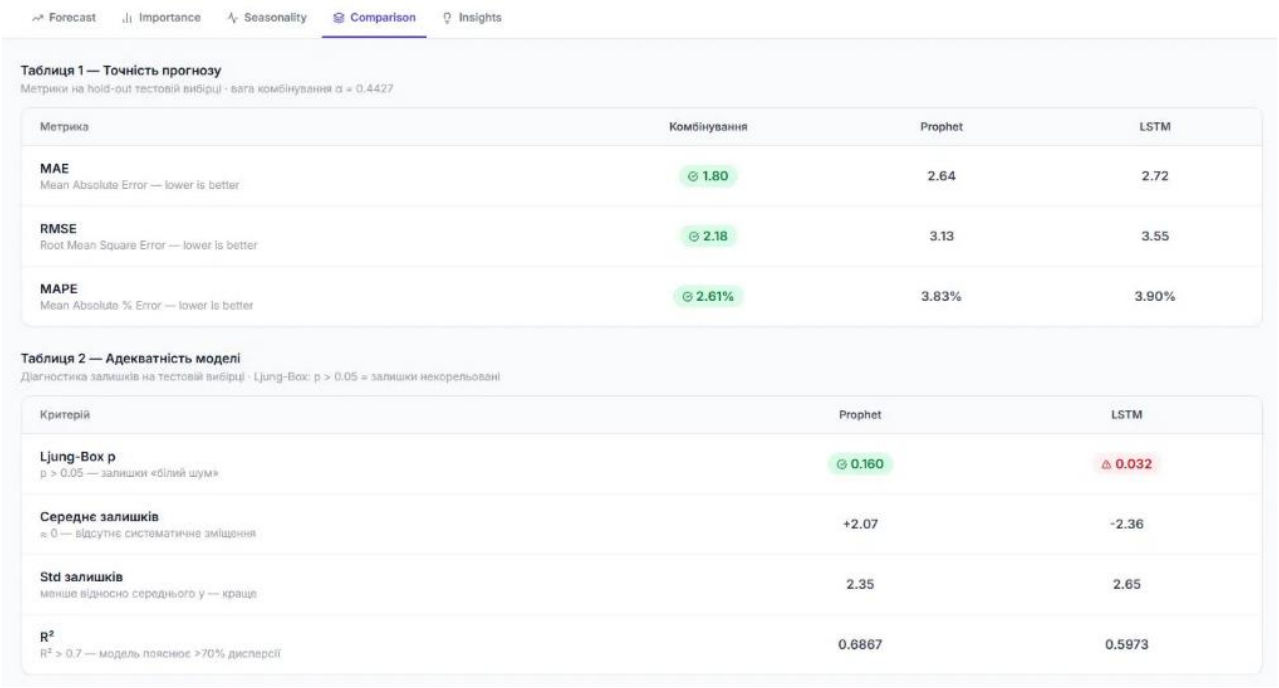
Наскрізний експеримент підтвердив коректність роботи всіх модулів системи: від імпорту даних і автоматичної діагностики часового ряду до генерації прогнозу, інтерпретації результатів і їх експорту. Порівняльний аналіз метрик точності наведено у підрозділі 3.3.

3.3 Аналіз отриманих результатів прогнозування

Підрозділ містить кількісний і якісний аналіз результатів прогнозування за двокомпонентною методологією підрозділу 2.1.6: спочатку – діагностика статистичної адекватності залишків, потім – порівняльний аналіз точності на тестовій вибірці.Для репрезентативності дослідження обрано два товари з різною структурою попиту: «Ізотонічний напій» – виражена сезонність і стабільна авторегресія, «Ліки від застуди» – помірна сезонність із залежністю від зовнішніх факторів. Усі моделі тестувались на ідентичній hold-out вибірці (останні 20% ряду, ~219 спостережень) із горизонтом 30 днів від 28 вересня [23].

3.3.1 Аналіз результатів прогнозування «Ізотонічний напій»

Аналіз починається з перевірки статистичної адекватності моделей – відповідно до методологічної послідовності підрозділу 2.1.6. Результати діагностики залишків наведено у таблиці 3.2 (відтворює дані вкладки Comparison, рис. 3.18).


Forecast Importance Seasonality Comparison Insights

Таблиця 1 — Точність прогнозу

Метрики на hold-out тестовій вибірці · вага комбінування α = 0.4427

Метрика	Комбінування	Prophet	LSTM
MAE Mean Absolute Error — lower is better	1.80	2.64	2.72
RMSE Root Mean Square Error — lower is better	2.18	3.13	3.55
MAPE Mean Absolute % Error — lower is better	2.61%	3.83%	3.90%

Таблиця 2 — Адекватність моделі

Діагностика залишків на тестовій вибірці · Ljung-Box: p > 0.05 = залишки некорельовані

Критерій	Prophet	LSTM
Ljung-Box p p > 0.05 — залишки «білий шум»	0.160	0.032
Середнє залишків ≈ 0 — відсутнє систематичне зміщення	+2.07	-2.36
Std залишків менше відносно середнього y — краще	2.35	2.65
R^2 R^2 > 0.7 — модель пояснює >70% дисперсії	0.6867	0.5973

Рисунок 3.18 – Таблиці точності та адекватності для «Ізотонічний напій»

Таблиця 3.2 – Результати перевірки адекватності моделей («Ізотонічний напій»)

Критерій	Prophet	LSTM
Ljung-Box p	0.160	0.032
Середнє залишків	+2.07	-2.36
Std залишків	2.35	2.65
R^2	0.6867	0.5973

Перш ніж інтерпретувати результати адекватності, потрібно перевірити передумову методу Bates–Granger – співмірність дисперсій залишків базових моделей. Якщо дисперсії відрізняються більш ніж у 4–5 разів, комбінування втрачає теоретичне обґрунтування і доцільніше просто обрати кращу модель. Для «Ізотонічного напою» std залишків Prophet – 2.35, LSTM – 2.65, відношення 1.13. Для «Ліків від застуди» – 2.03 / 1.78 = 1.14. В обох випадках дисперсії порівнювані, комбінування методологічно виправдане.

Prophet пройшов перевірку адекватності повністю: p = 0.160 перевищує критичний рівень 0.05, залишки не містять автокореляційної структури і відповідають умові білого шуму. Модель вилучила з ряду тренд і сезонні компоненти без залишкової систематики.

LSTM отримала p = 0.032 < 0.05 – нульова гіпотеза про білий шум залишків відхиляється, в помилках присутня залишкова структура. Причина криється в природі ряду: як встановлено у підрозділі 3.1.5, для «Ізотонічного напою» домінує lag_7 з високим SHAP-значенням. При навчанні з вікном LOOKBACK = 30 LSTM розподіляє увагу по всьому вікну – за умов сильної тижневої авторегресії це призводить до неповного вилучення структури залишків. Додатковий сигнал дає систематичне зміщення: середнє залишків Prophet – +2.07, LSTM – −2.36 при ідеальному значенні близькому до нуля. Prophet систематично занижує прогноз, LSTM – завищує. Це протилежне зміщення і є одним із аргументів на користь їх зваженого комбінування. Після підтвердження адекватності Prophet і часткової

адекватності LSTM виконується порівняльний аналіз точності прогнозів на тестовій вибірці. Результати трьох моделей наведено у таблиці 3.3.

Таблиця 3.3 – Метрики точності прогнозів «Ізотонічний напій»

Метрика	Комбінування	Prophet	LSTM
MAE	1.8	2.64	2.72
RMSE	2.18	3.13	3.55
MAPE	2.61%	3.83%	3.90%

Усі три моделі потрапляють у категорію «відмінно» за шкалою e-commerce (MAPE < 10%), проте між ними є суттєві кількісні відмінності. R^2 комбінованої моделі – 0.8481 – перевищує цільовий орієнтир 0.80, визначений у таблиці 2.3.Prophet (рис. 3.19) показав MAPE = 3.83% і R^2 = 0.6867.

Графік відтворює тижневу ритмічність і загальний рівень попиту коректно, але прогнозна лінія є згладженою – модель не вловлює короткострокових сплесків через лінійну апроксимацію тренду і обмежену здатність до моделювання нелінійних авторегресійних залежностей.

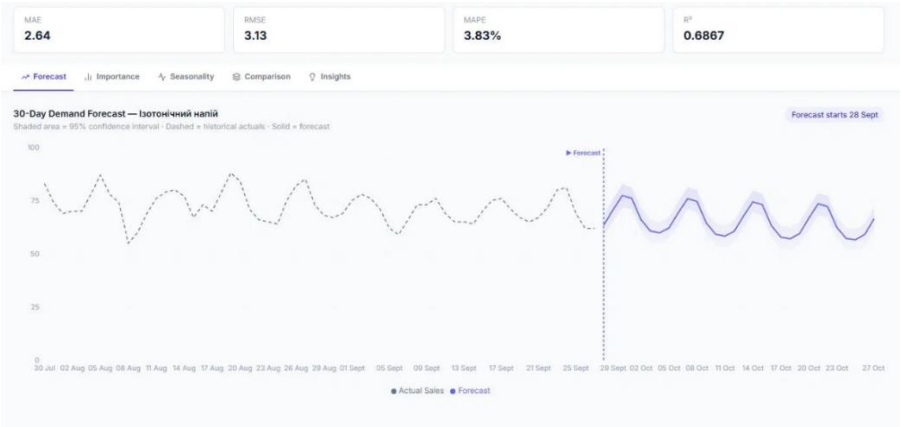


Рисунок 3.19 – Прогноз Prophet для «Ізотонічний напій»

LSTM (рис. 3.20) отримала MAPE = 3.90% і R^2 = 0.5973 – найнижчий коефіцієнт детермінації з трьох моделей. При схожому з Prophet значенні MAPE суттєво вищий RMSE = 3.55 проти 3.13 вказує на більші поодинокі відхилення. Це узгоджується з результатом тесту Льюнга–Бокса: залишки містять невилучену

структуру, що проявляється локальними помилками підвищеної амплітуди. Водночас LSTM краще реагує на різкі короткострокові зміни попиту – прогнозна лінія менш згладжена порівняно з Prophet.

Рисунок 3.20 – Прогноз LSTM для «Ізотонічний напій»

Комбінування Bates–Granger з $\alpha = 0.4427$ (рис. 3.21) дало найкращі результати за всіма чотирма метриками: MAPE = 2.61%, $R^2 = 0.8481$, MAE = 1.80, RMSE = 2.18. Відносно Prophet MAPE знизився на 0.32 в.п. (−8.4%), відносно LSTM – на 0.49 в.п. (−12.6%). Приріст R^2 з 0.6867 до 0.8481 означає, що модель пояснює 84.8% дисперсії ряду. Значення $\alpha \approx 0.44$ свідчить про майже рівний внесок обох компонентів: Prophet компенсує систематичне завищення LSTM, LSTM компенсує згладженість Prophet – саме та взаємодоповнюваність, яку передбачає теорема Bates–Granger.

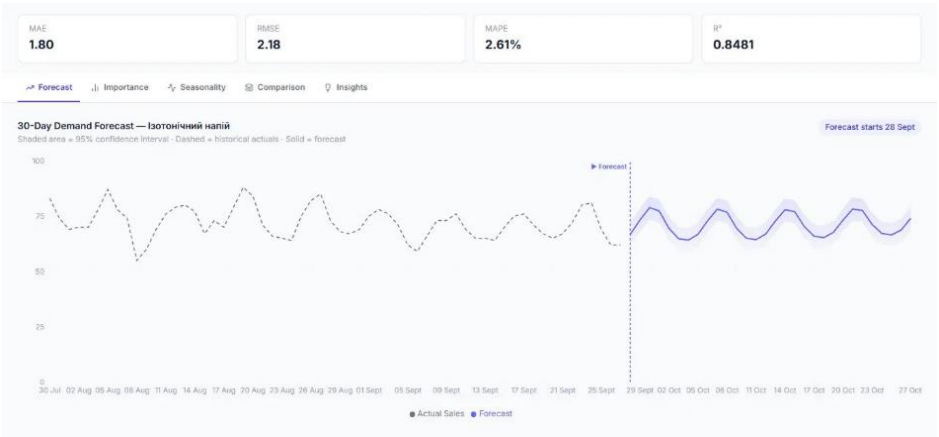


Рисунок 3.21 – Прогноз комбінування для «Ізотонічний напій»

SHAP-діаграма (рис. 3.22) розкриває ієрархію факторів, що визначають прогноз. Домінує lag_7 зі значенням ~7.5 – майже вдвічі більше за наступну ознаку. Це підтверджує висновок підрозділу 3.1.5: обсяг продажів конкретного дня найкраще пояснюється значенням рівно тиждень тому. Друге місце – температура повітря з Open-Meteo API, що підтверджує сезонну залежність попиту на ізотонічні напої від погодних умов і обґрунтовує практичну цінність інтеграції зовнішніх API [22]. Lag_14 і day_of_week займають третю і четверту позиції, підтверджуючи дворівневу тижневу авторегресійну структуру. Rolling_7 відображає короткостроковий тренд останнього тижня. Search_interest і precipitation мають низьку важливість – пошуковий інтерес слабо корелює з попитом на ізотонічні напої у досліджуваному горизонті.

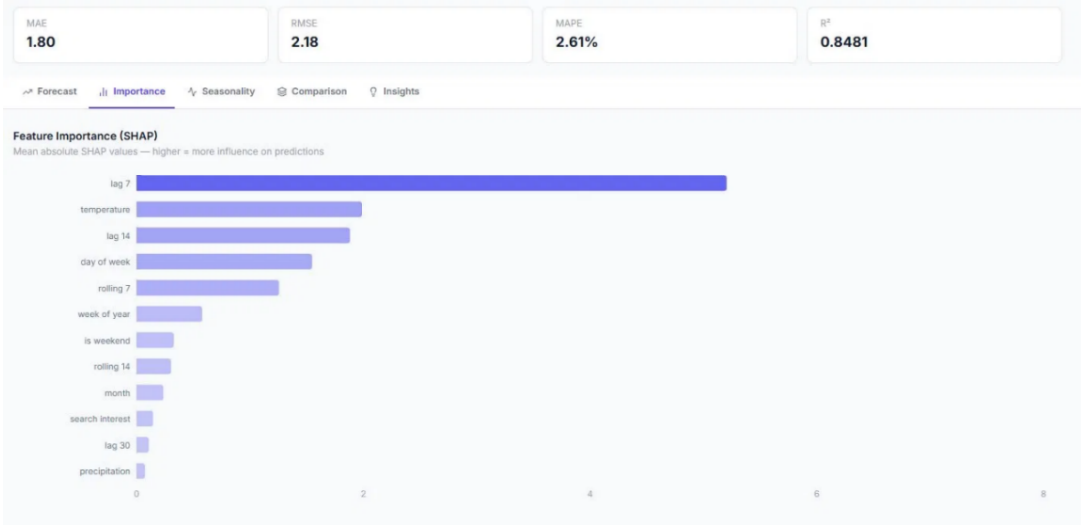


Рисунок 3.22 – SHAP-важливість ознак для «Ізотонічний напій» (Комбінування)

Декомпозиція Prophet (рис. 3.23) дає інтерпретовану картину структури попиту. Тренд тримається на горизонтальному рівні ~60 одиниць упродовж вересня–жовтня – точок зламу не виявлено. Тижнева сезонність асиметрична: субота і неділя дають +7–8 одиниць, вівторок і середа – від −4 до −5. Споживачі купують ізотонічні напої переважно у вихідні, що відповідає їх фізичній природі. Сезонний патерн поступово знижується з +1.0 на початку жовтня до −3.5 наприкінці – осіннє зменшення споживання після завершення літнього сезону.

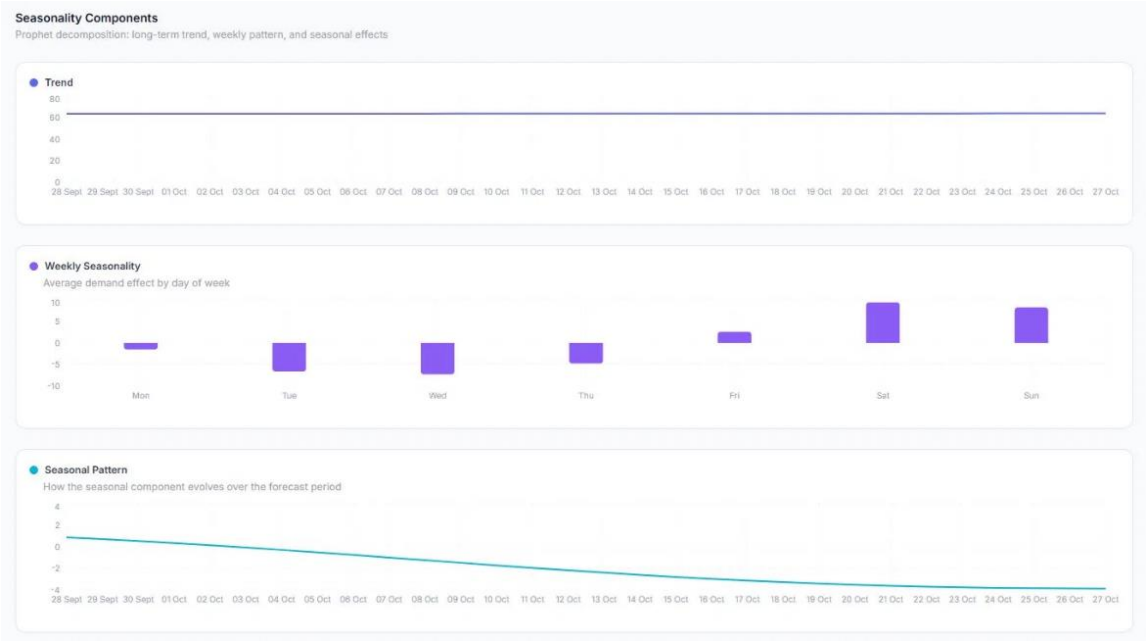


Рисунок 3.23 – Компоненти сезонності Prophet для «Ізотонічний напій»

Теплова карта (рис. 3.24) у розрізі місяць × день тижня підтверджує структуру декомпозиції. Найтемніші клітинки концентруються у червні–серпні, особливо у вихідні – літній пік споживання ізотонічних напоїв. Грудень і січень – найсвітліші, квітень–травень демонструють поступове зростання. Ця структура узгоджується з результатами STL-декомпозиції підрозділу 3.1.5 і підтверджує, що Prophet коректно ідентифікував і екстраполював сезонні компоненти.

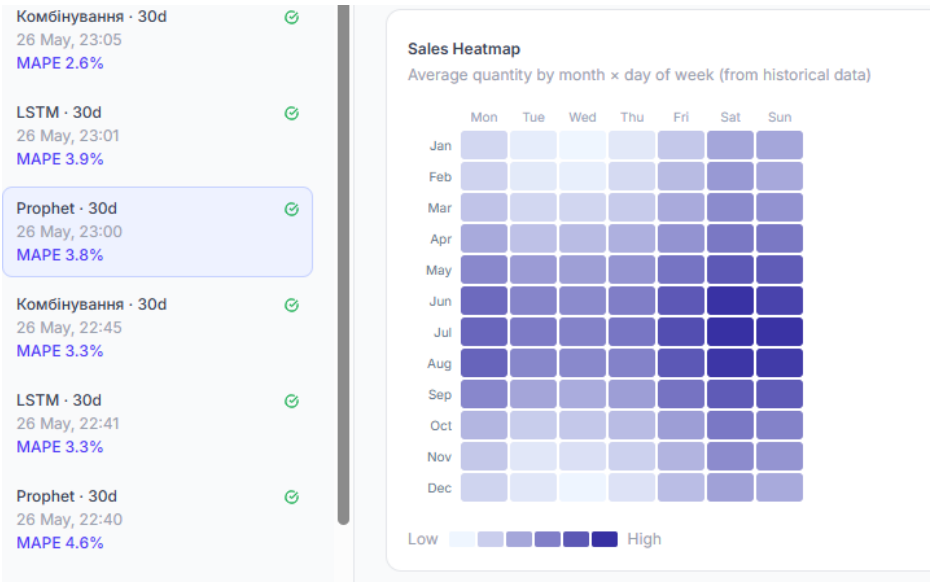


Рисунок 3.24 – Теплова карта продажів «Ізотонічний напій»: місяць × день тижня

Модуль Insights (рис. 3.25) класифікував попит як «Growing demand» з динамікою +32.2% і рівнем ризику «Low Risk». Зростаючий тренд у вересні–жовтні виглядає контрінтуїтивно на тлі осіннього спаду сезонної компоненти – пояснення у тому, що горизонт порівнюється з фактичними продажами кінця серпня–початку вересня, які вже нижчі за літній пік. «Low Risk» обумовлений MAPE = 2.61% – найкраща зона точності (< 5%). Система сформувала дві рекомендації: збільшити наступне замовлення на 15–20% і розглянути розширення асортименту супутніх товарів – обидві методологічно виправдані для товару зі зростаючим трендом і низькою невизначеністю прогнозу.

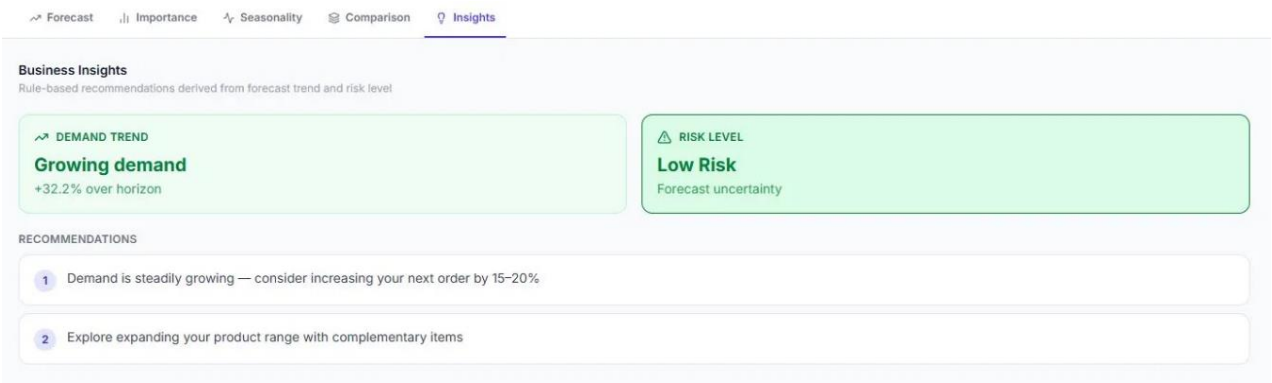


Рисунок 3.25 – Вкладка Insights для «Ізотонічний напій»

3.3.2 Аналіз результатів прогнозування «Ліки від застуди»

«Ліки від застуди» – порівняльний кейс із принципово іншою структурою попиту: помірна амплітуда коливань, залежність від сезону захворюваності і виражена реакція на пошуковий інтерес. Горизонт – 30 днів від 28 вересня, зовнішні сигнали – Wikimedia Trends і Open-Meteo API.

На відміну від «Ізотонічного напою», обидві базові моделі пройшли перевірку адекватності: Prophet – $p = 0.062$, LSTM – $p = 0.280$, обидва вище критичного рівня 0.05. Середнє залишків Prophet – −0.29, LSTM – +1.02: у Prophet систематичного зміщення немає, LSTM незначно завищує прогноз. Причина у природі ряду – ліки від застуди мають слабшу тижневу авторегресію порівняно з ізотонічним напоєм, що дозволяє LSTM коректніше вилучити структуру залишків.

Результати порівняльного аналізу точності наведено у таблиці 3.4.

Таблиця 3.4 – Метрики точності прогнозів «Ліки від застуди»

Метрика	Комбінування	Prophet	LSTM
MAE	1.32	1.55	1.58
RMSE	1.73	2.05	2.05
MAPE	3.30%	3.92%	3.90%

R^2 становить: Комбінування – 0.7846, Prophet – 0.6997, LSTM – 0.6986. Комбінована модель впритул наближається до орієнтиру > 0.80; базові моделі пояснюють ~70% дисперсії ряду – прийнятний результат для товару з помірною сезонністю. Prophet і LSTM дають практично ідентичні результати (MAPE 3.92% проти 3.90%, R^2 0.6997 проти 0.6986), проте їхні похибки частково некорельовані – саме це і забезпечує виграш комбінування. Значення $\alpha = 0.5416$ підтверджує майже рівний внесок обох компонентів з незначним пріоритетом Prophet – очікуваний результат для ряду з вираженою сезонністю. Графік прогнозу (рис. 3.26) відтворює стабільний рівень продажів 30–45 одиниць з тижневою ритмічністю; вузький довірчий інтервал 95% вказує на низьку невизначеність.

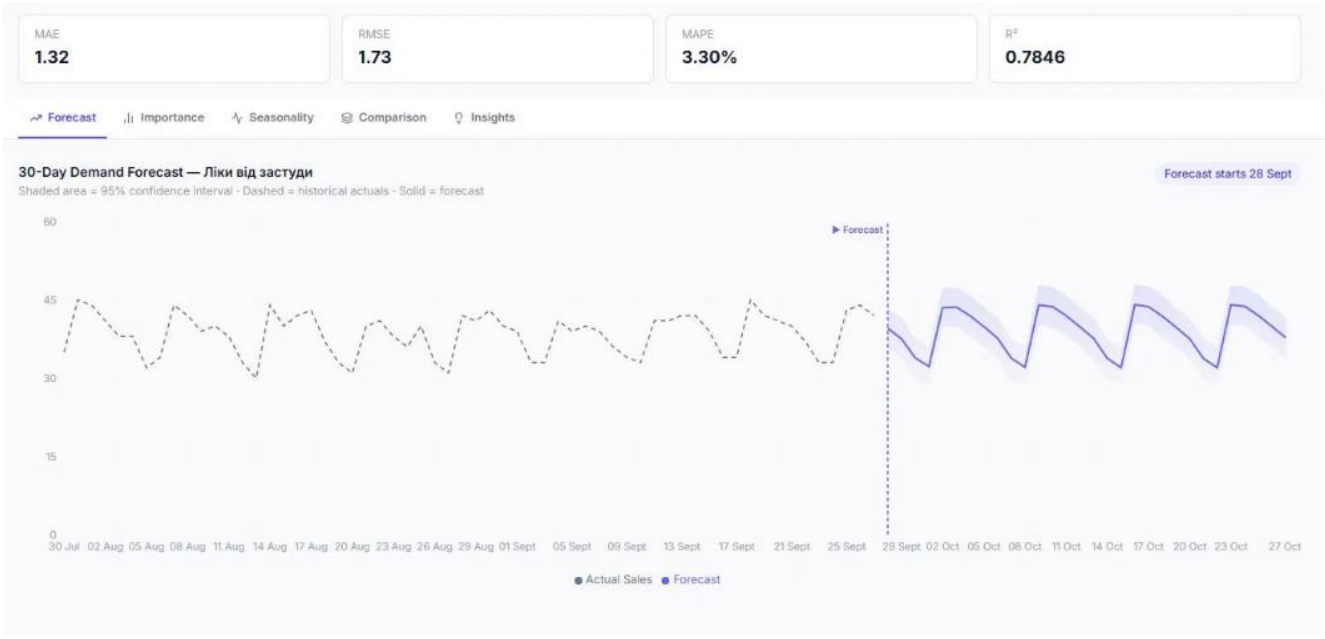


Рисунок 3.26 – Прогноз Комбінування для «Ліки від застуди»

Зіставлення двох товарів дає кілька методологічних висновків. Bates–Granger перевершив базові моделі в обох сценаріях: для «Ізотонічного напою» приріст R^2 відносно Prophet – +0.161, для «Ліків від застуди» – +0.085. Різний масштаб приросту пояснюється різницею в структурі рядів: сильніша авторегресія і протилежне зміщення залишків у першому кейсі дають більший виграш від комбінування.

Зведені результати наведено у таблиці 3.5.

Таблиця 3.5 – Зведені результати порівняльного аналізу моделей

Товар	Модель	MAE	RMSE	MAPE	R^2	Ljung-Box p
Ізотонічний напій	Prophet	2.64	3.13	3.83%	0.6867	0.160
Ізотонічний напій	LSTM	2.72	3.55	3.90%	0.5973	0.032
Ізотонічний напій	Комбінування	1.80	2.18	2.61%	0.8481	-
Ліки від застуди	Prophet	1.55	2.05	3.92%	0.6997	0.062
Ліки від застуди	LSTM	1.58	2.05	3.90%	0.6986	0.280
Ліки від застуди	Комбінування	1.32	1.73	3.3%	0.7846	-

Усі значення MAPE – в діапазоні 2.61–3.92%, категорія «відмінно» за шкалою e-commerce (< 10%). Bates–Granger перевершив Prophet і LSTM за всіма метриками в обох тестових сценаріях – практичне підтвердження теоретичного обґрунтування підрозділу 2.1.4: лінійна комбінація різнорідних прогнозів з некорельованими похибками дає меншу загальну дисперсію, ніж кожна модель окремо.

Модуль Insights (рис. 3.27) класифікував попит як «Declining demand» з динамікою −20.6% і рівнем ризику «Low Risk». Спадний тренд у вересні–жовтні методологічно очікуваний: цей горизонт є перехідним перед осіннім піком захворюваності, попит ще не досяг сезонного максимуму. «Low Risk» при MAPE = 3.30% – коректна класифікація, точність прогнозу достатня для операційних рішень. Система рекомендує скоротити наступне замовлення на 10–15% і

розглянути акційну кампанію для вивільнення поточних запасів – практично виправдане рішення для товару зі спадаючим короткостроковим трендом перед майбутнім піком.

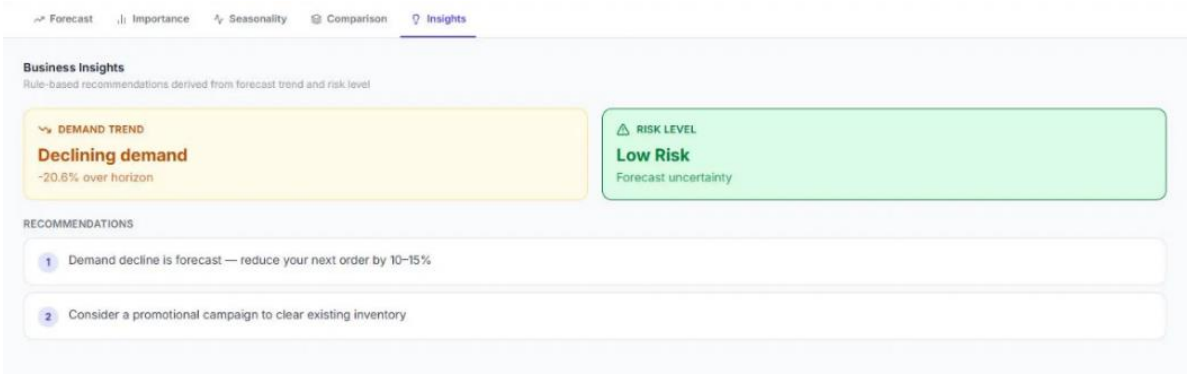


Рисунок 3.27 – Вкладка Insights для «Ліки від застуди»

Висновки до розділу 3

Третій розділ охопив практичну реалізацію системи – від проєктування структури до числових результатів на реальних даних.

Архітектура реалізована за трирівневою схемою, розроблено реляційну схему з семи таблиць і прописано двадцять API-маршрутів. Перед навчанням моделей часовий ряд пройшов статистичний аналіз і результати безпосередньо визначили вибір методів. Тест ADF повернув p=0.274: ряд нестаціонарний, що знімає зі столу моделі сімейства ARIMA без попереднього диференціювання. Графіки ACF зафіксували стійкі піки на лагах кратних семи математичне свідчення глибокої тижневої пам'яті, яку LSTM утримує через механізм клітин пам'яті, а лінійні методи ні. STL-декомпозиція підтвердила стабільну амплітуду сезонної складової (±10–15 одиниць), що обґрунтовує адитивну форму Prophet. Вибір методів спирається на конкретні характеристики вхідного ряду, а не на загальні міркування.

Перевірка повного сценарію роботи підтвердила функціональну завершеність застосунку: завантаження та валідація CSV, асинхронний запуск ML-пайплайну, побудова прогнозу з довірчим інтервалом 95%, SHAP-діаграма

важливості ознак, декомпозиція сезонних компонент Prophet та автоматичне формування рекомендацій щодо управління запасами.

Порівняльне тестування на двох товарах із різними патернами попиту підтвердило ефективність підходу. Метод комбінування Bates–Granger [21] перевершив кожну базову модель за всіма метриками в обох сценаріях практичне підтвердження теоретичного обґрунтування підрозділу 2.1.4. Залишки Prophet у обох випадках пройшли тест Льюнга–Бокса: некорельовані, без систематичного зміщення. SHAP-аналіз зафіксував температуру повітря з Open-Meteo API на другій позиції за впливом на прогноз сезонного товару пряме числове підтвердження практичної цінності інтеграції зовнішніх даних.

Система виконує поставлені вимоги щодо точності (MAPE 2.6–3.9%), інтерпретованості результатів та доступності для малого і середнього бізнесу.

4 КЕРІВНИЦТВО КОРИСТУВАЧА ТА ТЕСТУВАННЯ СИСТЕМИ

Четвертий розділ присвячено практичному аспекту розробленої системи – документації для кінцевого користувача та перевірці коректності роботи всіх компонентів. Підрозділ 4.1 містить покрокове керівництво, що охоплює реєстрацію, завантаження даних, налаштування і запуск прогнозу, інтерпретацію результатів і експорт. Підрозділ 4.2 – результати функціонального тестування за тест-кейсами та нефункціонального тестування на відповідність вимогам продуктивності, безпеки та надійності.

4.1 Керівництво користувача

Керівництво покроково описує взаємодію з платформою DemandForecast – від первинної реєстрації до експорту прогнозних значень. Система реалізована як вебзастосунок: встановлення стороннього програмного забезпечення не потрібне, доступ до всіх обчислювальних і графічних модулів забезпечується через браузер з підтримкою JavaScript (Google Chrome 111+, Mozilla Firefox 113+, Microsoft Edge 113+, Safari 16.4+). Орієнтація на браузерний доступ без інсталяції відповідає ключовій вимозі доступності для МСБ.

4.1.1 Реєстрація та вхід до системи

Для доступу до аналітичних модулів користувач відкриває адресу застосунку у браузері. Новий користувач натискає посилання «Create one» на сторінці автентифікації (рис. 4.1) і заповнює форму з електронною адресою та паролем. Зареєстрований користувач вводить облікові дані і натискає «Sign In». Після верифікації JWT-токена система перенаправляє на головну панель Dashboard, де одразу доступна персоналізована аналітична панель з навігацією до всіх модулів. Кожен обліковий запис ізольований на рівні сервера – дані одного користувача недоступні іншим без додаткових налаштувань.

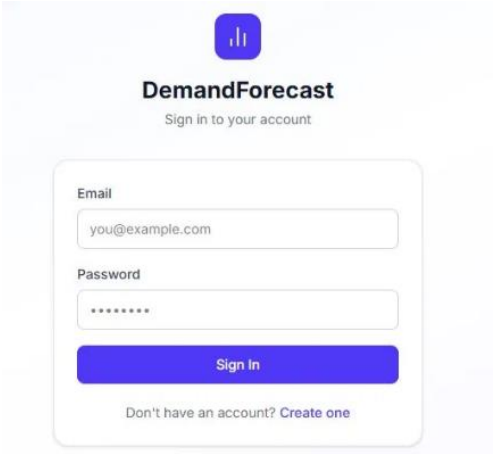


Рисунок 4.1 – Сторінка автентифікації системи DemandForecast

4.1.2 Головна панель Dashboard

Dashboard (рис. 4.2) відображає зведену статистику: кількість зареєстрованих товарів, виконаних і завершених сесій прогнозування та середню точність по всіх товарах у форматі (100 − MAPE). Ліва частина панелі містить список останніх прогнозів із зазначенням товару, методу, горизонту, дати та MAPE. Права – перелік зареєстрованих товарних позицій. Навігаційне меню ліворуч забезпечує перехід до модулів Upload Data, Forecast та History.

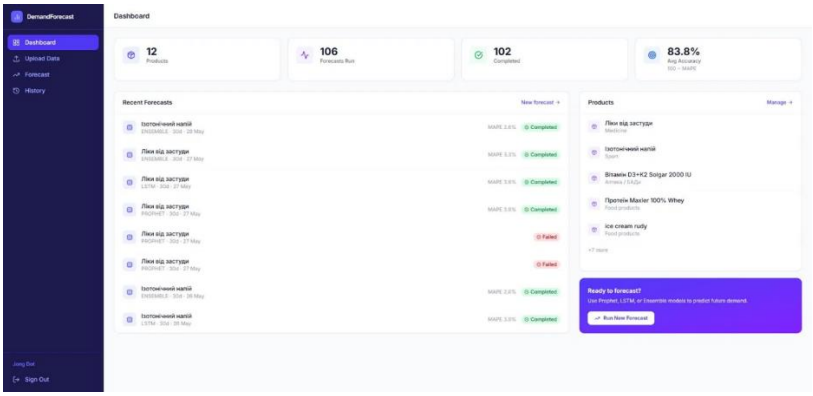


Рисунок 4.2 – Головна панель Dashboard: зведена статистика системи

4.1.3 Реєстрація товару та завантаження даних продажів

Перед запуском прогнозу необхідно зареєструвати товарну позицію і завантажити її історичні дані. У навігаційному меню обирається розділ «Upload

Data». Для реєстрації нового товару через форму у лівій панелі (рис. 4.3) заповнюються поля: «Product Name», «Category», «Wikimedia Trends keyword» – ключове слово для підключення сигналу пошукового інтересу, а також координати торгової точки («Latitude», «Longitude») для прив'язки до геолокованих погодних даних Open-Meteo API. Після заповнення форми натискається «+ Create».

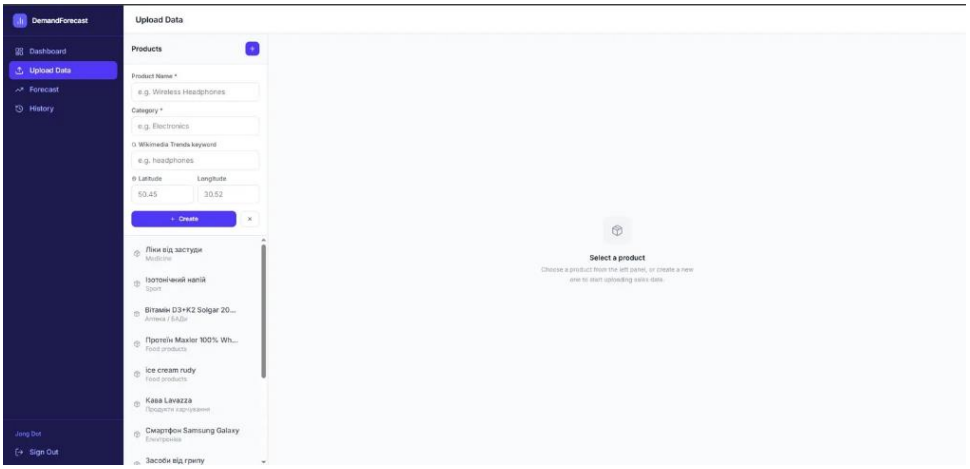


Рисунок 4.3 – Форма реєстрації нового товару в модулі Upload Data

Після вибору товару з переліку у правій частині екрану відкривається панель завантаження (рис. 4.4). CSV-файл переноситься у виділену область або обирається через файловий менеджер. Обов'язкові стовпці: date (формат YYYY-MM-DD) і quantity (ціла кількість продажів); стовпець price – необов'язковий. Мінімально рекомендований обсяг – 90 рядків. Після успішного імпорту система автоматично формує дескриптивну статистику датасету.

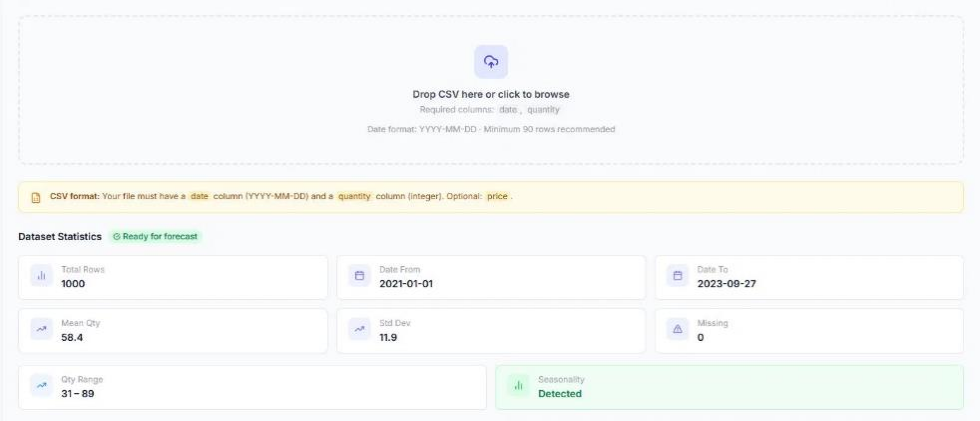


Рисунок 4.4 – Панель завантаження даних та автоматична статистика датасету

4.1.4 Налаштування та запуск прогнозу

У розділі «Forecast» ліва панель (рис. 4.5) містить усі параметри конфігурації. Зі спадного списку PRODUCT обирається товарна позиція.

У блоці MODEL вибирається один із трьох методів: Prophet, LSTM або Комбінування.

Горизонт прогнозування у блоці FORECAST HORIZON – 7, 14, 30, 60 або 90 днів.

У блоці EXTERNAL SIGNALS активуються зовнішні сигнали. «Weather data» підключає добові значення температури і опадів з Open-Meteo API для координат, заданих при реєстрації товару. «Wikimedia Trends» підключає індекс пошукового інтересу за вказаним ключовим словом.

Вибір сигналів визначається природою попиту: для температурно-залежних товарів (спортивні напої, морозиво) активується погодний сигнал; для товарів із сезонним пошуковим попитом (ліки, навчальне приладдя) – Wikimedia Trends.

Після налаштування параметрів натискається «Run Forecast» – система запускає ML-пайплайн у фоновому режимі і після завершення розрахунків відображає результати у п'яти аналітичних вкладках.

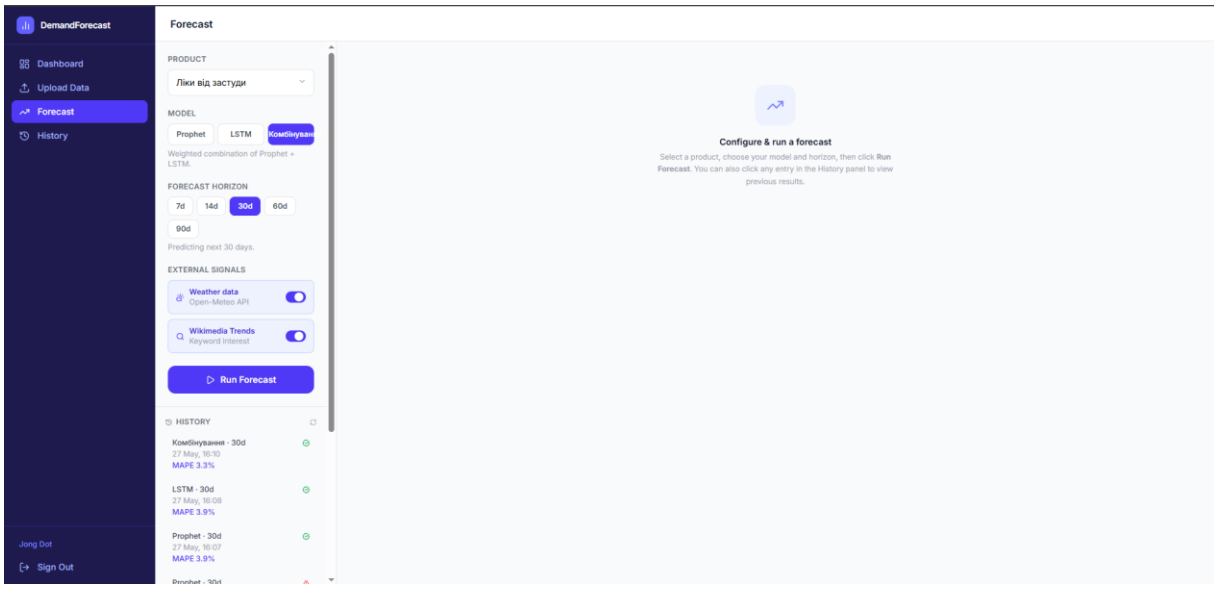


Рисунок 4.5 – Панель конфігурації прогнозу

4.1.5 Інтерпретація результатів прогнозування

У верхній частині екрану після завершення розрахунків відображаються зведені метрики: MAE, RMSE, MAPE та R^2. Нижче – п'ять аналітичних вкладок (рис. 4.6).

Forecast – графік прогнозу попиту на обраний горизонт з 95% довірчим інтервалом: штрихова лінія відповідає історичним даним, суцільна – прогнозній траєкторії, затінена область – межам невизначеності.

Importance – SHAP-діаграма важливості ознак, що показує які фактори найбільше вплинули на прогноз.

Seasonality – компоненти декомпозиції Prophet: довгостроковий тренд, тижнева сезонність, сезонний патерн на горизонті прогнозування та теплова карта середніх продажів у розрізі місяць × день тижня.

Comparison – таблиця метрик точності трьох методів на тестовій вибірці та діагностика залишків за критерієм Льюнга–Бокса. Зелений індикатор – відповідність критерію якості, помаранчевий – відхилення.

Insights – логістичні рекомендації на основі класифікації тренду прогнозної траєкторії (Growing / Declining / Stable) та рівня ризику (Low / Medium / High Risk), що визначається значенням MAPE. На підставі цих двох показників система формує конкретні рекомендації щодо коригування обсягу наступного замовлення.

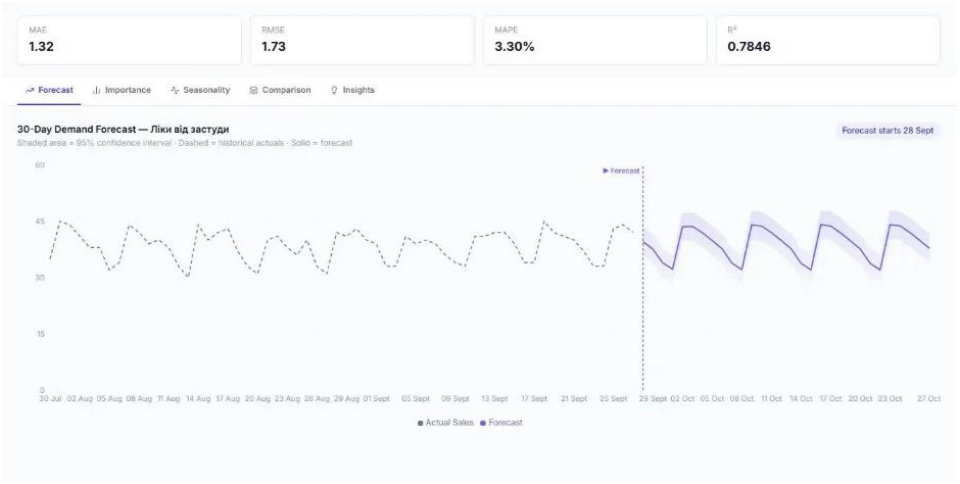


Рисунок 4.6 – Результати прогнозування

4.1.6 Перегляд журналу прогнозів

Розділ «History» (рис. 4.7) містить повний журнал виконаних сесій: товарна позиція, метод, горизонт, статус (Completed / Failed), метрики MAE, MAPE, R^2 та дата запуску. Фільтрація за конкретним товаром виконується через спадний список «All products» (рис. 4.8). Для кожного запису доступні три дії: «View» – перехід до повних результатів сесії, «Compare» – порівняння метрик з іншими сесіями, «Export» – вивантаження прогнозних значень у CSV.

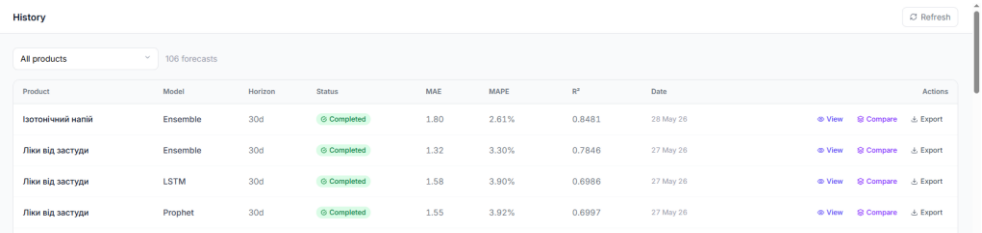


Рисунок 4.7 – Розділ History: повний журнал сесій прогнозування

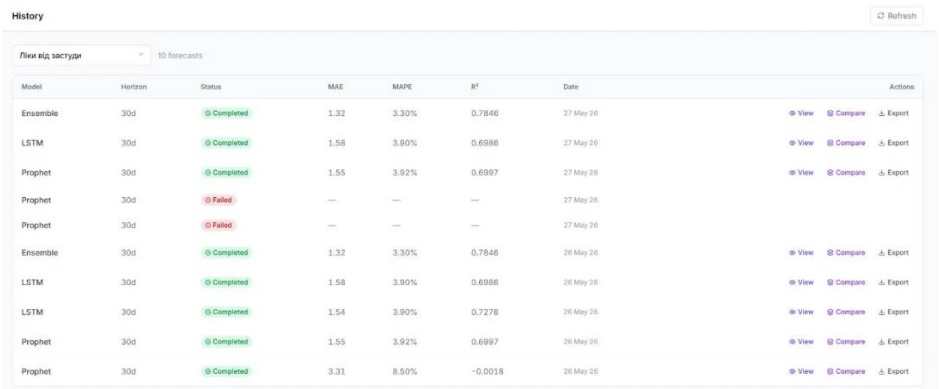


Рисунок 4.8 – Фільтрація журналу за товарною позицією

4.1.7 Експорт результатів прогнозування

Результати прогнозування можна передати до зовнішніх ERP-систем або платформ управління запасами через CSV-експорт. Він виконується двома способами: кнопка «Export CSV» у правому верхньому куті сторінки Forecast при перегляді активної сесії, або кнопка «Export» у рядку відповідного запису журналу History. Браузер зберігає файл з унікальним ідентифікатором сесії у назві (рис. 4.9). Файл містить денні прогнозні значення на весь визначений горизонт.

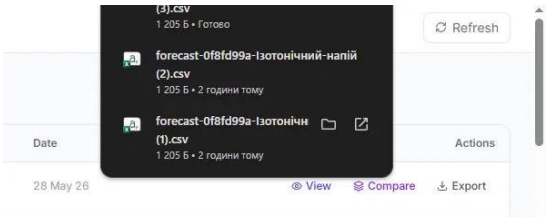


Рисунок 4.9 – Завантаження CSV-файлу з результатами прогнозування

4.2 Функціональне і нефункціональне тестування системи

Тестування DemandForecast проходило у два послідовних етапи. Функціональний етап перевіряв відповідність поведінки системи задокументованим вимогам у всіх ключових сценаріях. Нефункціональний – продуктивність, надійність і безпеку в умовах, наближених до реальної експлуатації. Обидва етапи виконувались методом «чорної скриньки»: виключно на основі вхідних даних і спостереження за реакцією системи без доступу до вихідного коду.

4.2.1 Функціональне тестування

Тестування охопило чотири функціональні модулі: автентифікацію, управління товарами та завантаження даних, прогнозування попиту, журналювання і експорт результатів. Для кожного з 19 тест-кейсів зафіксовано вхідні дані, очікуваний і фактичний результат та статус виконання. Результати наведено у таблиці 4.1.

Таблиця 4.1 – Результати функціонального тестування системи DemandForecast.

№	Назва тест-кейсу	Вхідні дані	Очікуваний результат	Фактичний результат	Статус
TC-01	Реєстрація з коректними даними	Унікальний email, пароль	Обліковий запис створено, перехід на сторінку входу	Обліковий запис створено, виконано перенаправлення	Passed
TC-02	Реєстрація з вже існуючим email	Email що вже зареєстрований	Повідомлення про помилку, залишення на сторінці реєстрації	Відображено повідомлення про те що email вже використовується	Passed

Продовження таблиці 4.1

№	Назва тест-кейсу	Вхідні дані	Очікуваний результат	Фактичний результат	Статус
TC-03	Вхід з коректними обліковими даними	Валідні email та пароль	Перехід на Dashboard, ініціалізація JWT	Перехід на Dashboard, сесія ініціалізована	Passed
TC-04	Вхід з невірним паролем	Валідний email, невірний пароль	Повідомлення про помилку автентифікації, залишення на сторінці входу	Відображено повідомлення про помилку	Passed
TC-05	Вхід з порожніми полями	Порожні поля email та пароль	Блокування відправки форми	Відправка форми заблокована браузером	Passed
TC-06	Доступ до сторінки без авторизації	Перехід без токена	Перенаправлення на сторінку входу	Перенаправлення виконано	Passed
TC-07	Реєстрація нового товару	Коректні назва, категорія, координати	Товар з'являється у переліку лівої панелі	Товар додано до переліку	Passed
TC-08	Завантаження коректного CSV-файлу	CSV з колонками date та quantity	Відображення статистики датасету, статус «Ready for forecast»	Статистика відображена	Passed
TC-09	Завантаження CSV без обов'язкової колонки	CSV без колонки quantity	Повідомлення про помилку структури файлу	Відображено повідомлення про некоректний формат	Passed
TC-10	Завантаження файлу некоректного формату	Файл формату .xlsx	Повідомлення про непідтримуваний формат	Відображено повідомлення про помилку	Passed
TC-11	Запуск прогнозу з коректними параметрами	Товар з завантаженим датасетом, модель Ensemble	Відображення результатів у п'яти вкладках, метрики точності у верхній частині	Результати відображено, всі вкладки доступні	Passed
TC-12	Запуск прогнозу без завантаженого датасету	Товар без CSV, модель Ensemble	Повідомлення про відсутність даних, блокування запуску	Запуск заблоковано, відображено повідомлення	Passed
TC-13	Активація зовнішнього сигналу Weather data	Товар з координатами, увімкнений перемикач Weather data	Інтеграція температурних даних Open-Meteo API у модель	Сигнал підключено, ознака temperature присутня у SHAP	Passed
TC-14	Активація зовнішнього сигналу Wikimedia Trends	Товар з вказаним keyword, увімкнений перемикач Wikimedia Trends	Інтеграція індексу пошукового інтересу у модель	Сигнал підключено, ознака search_interest присутня у SHAP	Passed

Кінець таблиці 4.1

№	Назва тест-кейсу	Вхідні дані	Очікуваний результат	Фактичний результат	Статус
TC-14	Активація зовнішнього сигналу Wikimedia Trends	Товар з вказаним keyword, увімкнений перемикач Wikimedia Trends	Інтеграція індексу пошукового інтересу у модель	Сигнал підключено, ознака search_interest присутня у SHAP	Passed
TC-15	Експорт результатів прогнозу у CSV	Завершена сесія прогнозування, натискання Export CSV	Завантаження файлу з денними прогнозними значеннями	Файл завантажено, містить коректні прогнозні значення	Passed
TC-16	Фільтрація журналу History за товаром	Вибір конкретного товару зі спадного списку	Відображення лише записів обраної позиції	Журнал відфільтровано коректно	Passed
TC-17	Перегляд результатів збереженої сесії	Натискання «View» у рядку журналу	Відображення повних результатів сесії з усіма вкладками	Результати відображено коректно	Passed
TC-18	Вихід із системи	Натискання «Sign Out»	Завершення сесії, видалення токена, перенаправлення на сторінку входу	Сесія завершена, перенаправлення виконано	Passed
TC-19	Завантаження CSV з недостатньою кількістю рядків	CSV з коректними колонками, 45 рядків	Відображення статистики з попередженням про недостатню кількість даних, статус не "Ready for forecast"	Дані завантажено, відображено попередження про недостатній обсяг, індикатор «Ready for forecast» відсутній	Passed

Усі 19 тест-кейсів отримали статус Passed, що підтверджує стабільність програмної логіки. Критичних відмов і неочікуваної поведінки не виявлено. На граничних випадках із некоректними вхідними даними система формувала інформативні повідомлення про помилки через стандартні коди відповідей HTTP (у модулі FastAPI), виключаючи аварійне завершення процесів (crash) або порушення цілісності даних у Supabase.

Покриття охоплює всі чотири ключові модулі – результати підтверджують відповідність системи функціональним вимогам.

4.2.2 Нефункціональне тестування

Нефункціональне тестування перевіряло три напрями: продуктивність, безпеку та надійність.

Продуктивність. Час відповіді API вимірювався інструментом Network у DevTools Google Chrome.

Для кожної комбінації моделі та горизонту зафіксовано три послідовні виміри тривалості запиту до ендпоінту /api/forecast. Результати наведено у таблиці 4.2.

Таблиця 4.2 – Результати вимірювання часу відповіді API прогнозування

Модель	Горизонт	Вимір 1, с	Вимір 2, с	Вимір 3, с	Середнє, с
Prophet	30 днів	6,41	5,85	6,12	6,13
LSTM	30 днів	125,3	127,8	125,6	126,2
Комбінування	7 днів	142,2	140,5	144,3	142,3
Комбінування	30 днів	160,7	163,2	164,5	162,8
Комбінування	60 днів	172,0	172,9	171,2	172,0

Різниця у часі виконання між моделями пояснюється їхньою математичною природою. Prophet відповідає найшвидше – в середньому 6,13 с – оскільки декомпозиційний підхід не потребує ітеративного навчання. LSTM значно повільніша (126,2 с): рекурентна мережа навчається заново при кожному запуску. Комбінована модель включає навчання обох компонентів і оптимізацію коефіцієнта α, що дає найбільший час – від 142,3 с для горизонту 7 днів до 172,0 с для 60 днів. Залежність часу від горизонту лінійна і пояснюється зростанням обсягу даних для генерації прогнозних значень. Для задач оперативного планування з горизонтом 30 днів час відповіді комбінованої моделі (~163 с) є прийнятним: прогноз запускається не частіше одного разу на добу.

Безпека. Перевірено три аспекти: захист API-маршрутів, захист облікових даних і ізоляція даних між користувачами.

Захист маршрутів перевірено прямим запитом до захищеного ендпоінту /api/products без заголовку авторизації через консоль браузера:

```
fetch('http://localhost:8000/api/products', {
  method: 'GET'
}).then(r => console.log(r.status))
```

Сервер повернув 403 Forbidden, що підтверджує блокування несанкціонованого доступу (рис. 4.10). Механізм HTTPBearer у FastAPI відхиляє запит із відсутнім заголовком Authorization з кодом 403; запит із невалідним або простроченим токеном повертає 401 Unauthorized. Усі захищені маршрути коректно відхиляють звернення без автентифікації.

Рисунок 4.10 – Результат запиту до захищеного API-маршруту без токена авторизації

Паролі захищено через bcrypt у модулі backend/app/core/security.py. При реєстрації пароль одразу перетворюється на незворотній хеш через hash_password() – у базу потрапляє виключно хеш. При автентифікації verify_password() порівнює введений пароль з хешем без зворотного розшифрування, що унеможливлює витік навіть при компрометації бази даних.

Ізоляцію даних перевірено реєстрацією двох незалежних облікових записів. Основний обліковий запис містить 12 товарних позицій (рис. 4.11); після входу під другим обліковим записом список відображає виключно його позиції без жодного

доступу до даних першого (рис. 4.12). Ізоляція реалізована серверною фільтрацією за ідентифікатором власника і не потребує додаткових перевірок на клієнті.

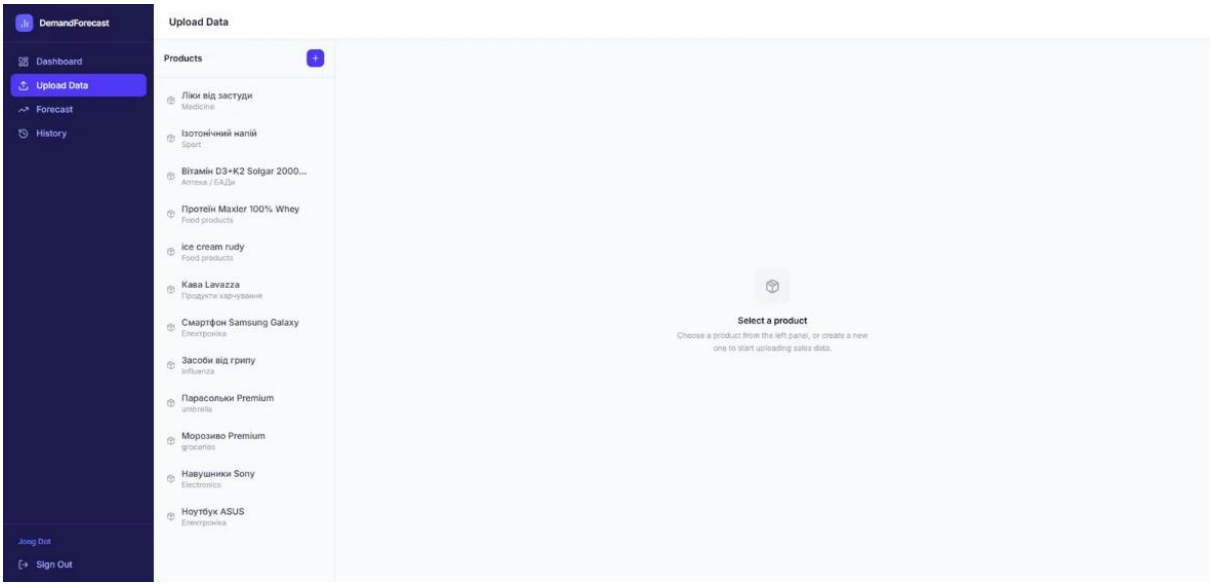


Рисунок 4.11 – Список товарів основного облікового запису (12 позицій)

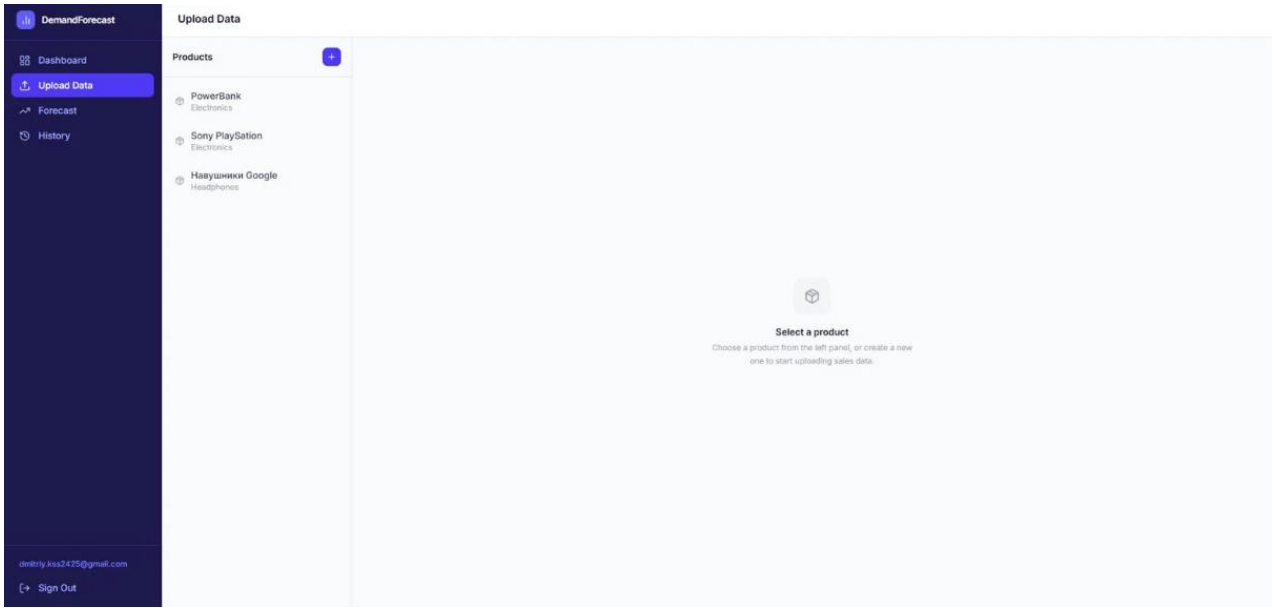


Рисунок 4.12 – Список товарів другого облікового запису

Надійність. При нештатних сценаріях із некоректними вхідними даними система формувала зрозумілі повідомлення про помилку без відображення внутрішніх технічних деталей і без переходу у неробочий стан. Завантаження пошкоджених файлів не порушувало роботу інших модулів.

Стабільність при тривалих обчисленнях забезпечено механізмом асинхронного polling: фронтенд з фіксованим інтервалом опитує /api/forecasts/{id}/status до отримання статусу завершення, що виключає таймаут з'єднання при навчанні LSTM (~126 с) і комбінованої моделі (~163 с). Якщо користувач закриває вкладку під час виконання прогнозу і повертається пізніше – результати збережено у журналі History і доступні без повторного запуску. Журнал забезпечує відтворюваність результатів і ретроспективне порівняння конфігурацій для однієї товарної позиції.

Час відповіді комбінованої моделі (~163 с) вкладається в межі оперативного планування. Захист маршрутів через HTTPBearer, bcrypt-хешування паролів і серверна ізоляція даних відповідають стандартам безпеки сучасних вебзастосунків. Механізм асинхронного polling і збереження результатів у журналі History підтверджують стабільність системи при тривалих обчисленнях і перериванні з'єднання.

Висновки до розділу 4

Четвертий розділ присвячено практичній стороні розробленої системи – документації для кінцевого користувача та перевірці коректності роботи всіх компонентів.

Керівництво користувача охоплює повний робочий цикл DemandForecast: реєстрацію та вхід, додавання товарів і завантаження CSV-файлів із продажами, вибір методу прогнозування, горизонту та зовнішніх сигналів, запуск ML-пайплайну і перегляд результатів через п'ять аналітичних вкладок. Окремо описано експорт прогнозних значень для інтеграції з обліковими системами підприємства. Застосунок не вимагає встановлення жодного додаткового ПЗ, достатньо сучасного браузера, що знижує поріг входу для МСБ.

Функціональне тестування методом «чорної скриньки» охопило 19 тест-кейсів у чотирьох модулях: автентифікація, управління товарами та даними,

прогнозування, журналювання та експорт. Усі кейси – Passed. Система коректно відпрацювала штатні сценарії і граничні випадки: некоректний формат CSV, порожній датасет, відсутній токен авторизації.

Нефункціональна перевірка охопила продуктивність, безпеку і надійність. Середній час відповіді Prophet – 6,13 с, комбінованої моделі на горизонті 30 днів – 162,8 с; обидва значення прийнятні для оперативного планування закупівель. Захист маршрутів через HTTPBearer повертає 403 Forbidden без токена; паролі хешуються bcrypt; ізоляція даних між користувачами реалізована серверною фільтрацією по ідентифікатору власника. Асинхронний механізм polling зберігає результати тривалих обчислень у журналі History незалежно від стану клієнтського з'єднання.

Сукупність отриманих результатів підтверджує придатність системи до розгортання в реальному середовищі роздрібної торгівлі та електронної комерції.

ВИСНОВКИ

Мету роботи досягнуто – розроблено вебзастосунок DemandForecast для прогнозування попиту на товари електронної комерції з інтеграцією моделей машинного навчання та зовнішніх API.

Порівняльний аналіз трьох провідних аналогів – Salesforce Einstein Analytics, Amazon Forecast і Google Cloud Retail AI – зафіксував спільні структурні обмеження: висока вартість ліцензування, технологічна замкненість і відсутність нативної підтримки зовнішніх контекстних джерел без платних проміжних сервісів. Ці обмеження і стали відправною точкою для розробки відкритого рішення, орієнтованого на малий та середній бізнеси.

Архітектура реалізована за трирівневою клієнт-серверною схемою: React SPA з TypeScript на клієнтському рівні, FastAPI на Python – на серверному, Supabase/PostgreSQL – як реляційне сховище. Специфікацію складають сім таблиць бази даних і двадцять REST API-маршрутів із JWT-автентифікацією.

ML-ядро поєднує три компоненти з різною математичною природою. Prophet моделює тренд і сезонність через адитивну декомпозицію з рядами Фур'є. Двошарова LSTM з механізмом гейтування виявляє нелінійні залежності та обробляє екзогенні регресори. Фінальний прогноз формується методом Bates–Granger: коефіцієнт α оптимізується мінімізацією MAE на валідаційній вибірці окремо для кожного товару. Вектор ознак збагачено даними Open-Meteo API та Wikimedia Pageviews API.

Тестування на двох товарних позиціях дало конкретні числа. Для «Ізотонічного напою» комбінована модель: MAE = 1,80; MAPE = 2,61%; R^2 = 0,8481 – перевищення точності відносно найкращої базової моделі 31,8%. Для «Ліків від застуди»: MAE = 1,32; MAPE = 3,30%; R^2 = 0,7846. Bates–Granger перевершив Prophet і LSTM за всіма метриками в обох сценаріях. SHAP-аналіз зафіксував температуру повітря з Open-Meteo на другій позиції за впливом на прогноз

сезонного товару – числове підтвердження практичної цінності інтеграції зовнішніх API.

Функціональне тестування за 19 тест-кейсами – усі Passed. Нефункціональна перевірка підтвердила відповідність вимогам продуктивності (час відповіді ~163 с для горизонту 30 днів), безпеки (HTTP 403 без токена, bcrypt, серверна ізоляція даних) та надійності (асинхронний polling зі збереженням результатів у журналі History).

Інтерпретованість прогнозів через SHAP і автоматичні рекомендації щодо управління запасами дозволяють аналітику без підготовки з машинного навчання приймати обґрунтовані логістичні рішення безпосередньо на основі виходів системи.

Подальший розвиток передбачає впровадження трансформерних архітектур, механізмів автоматичного перенавчання при виявленні концептуального дрейфу та багатотоварного прогнозування з урахуванням крос-кореляційних зв'язків між асортиментними позиціями.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Retail e– commerce sales worldwide from 2014 to 2027. Statista. 2024. URL: https://www.statista.com/statistics/379046/worldwide– retail– e– commerce– sales/ (date access: 15.01.2026).

2. Bandara K., Shi P., Bergmeir C., Hewamalage H., Tran Q., Seaman B. Sales Demand Forecast in E– commerce Using a Long Short– Term Memory Neural Network Methodology. Neural Information Processing: 26th International Conference, ICONIP 2019, Sydney, Australia. Springer, 2019. P. 462–474. URL: https://doi.org/10.1007/978–3– 030– 36718– 3_39 (date access: 15.01.2026).

3. Khan M. A., Saqib S., Alyas T., Rehman A., Saeed Y., Zeb A., Zareei M., Mohamed E. M. Effective Demand Forecasting Model Using Business Intelligence Empowered with Machine Learning. IEEE Access. 2020. Vol. 8. P. 116013–116023. URL: https://ieeexplore.ieee.org/document/9121220 (date access: 15.01.2026).

4. Salesforce, Inc. Einstein Analytics: AI– Powered Business Intelligence Platform. Official Documentation. 2024. URL: https://www.salesforce.com/products/einstein–analytics/overview/ (date access: 16.01.2026).

5. Amazon Web Services. Amazon Forecast Developer Guide: What is Amazon Forecast? AWS Documentation. 2024. URL: https://docs.aws.amazon.com/forecast/latest/dg/what–is–forecast.html (date access: 16.01.2026).

6. Lundberg S. M., Lee S.– I. A Unified Approach to Interpreting Model Predictions. Advances in Neural Information Processing Systems. 2017. Vol. 30. URL: https://arxiv.org/abs/1705.07874 (date access: 20.02.2026).

7. 6. Google LLC. Retail AI Solutions: Demand Forecasting and Personalization. Google Cloud Documentation. 2024. URL: https://cloud.google.com/solutions/retail (date access: 16.01.2026).

8. Supabase, Inc. Supabase Documentation: Open Source Firebase Alternative.

2024. URL: https://supabase.com/docs (date access: 17.02.2026).

9. Taylor S. J., Letham B. Forecasting at Scale. The American Statistician. 2018. Vol. 72, No. 1. P. 37–45. URL: https://doi.org/10.1080/00031305.2017.1380080 (date access: 03.02.2026).

10. Zippenfenig P. Open–Meteo: Open–Source Weather API. 2023. URL: https://open–meteo.com (date access: 10.03.2026).

11. Wikimedia Foundation. Wikimedia REST API: Pageviews Documentation. 2024. URL: https://wikitech.wikimedia.org/wiki/Data_Platform/AQS/Pageviews (date access: 10.03.2026).

12. Massaoudi M., Chihi I., Sidhom L., Refaat S. S., Abboudi A. Machine Learning and Deep Learning Models for Demand Forecasting in Supply Chain Management: A Critical Review. Applied System Innovation. 2024. Vol. 7, No. 5. Article 93. URL: https://doi.org/10.3390/asi7050093 (date access: 15.01.2026).

13. Falatouri T., Darbanian F., Brandtner P., Udokwu C. Predictive Analytics for Demand Forecasting – A Comparison of SARIMA and LSTM in Retail SCM. Procedia Computer Science. 2022. Vol. 200. P. 993–1003. URL: https://doi.org/10.1016/j.procs.2022.01.298 (date access: 15.01.2026).

14. Qureshi N. U. H., Javed S., Javed K., Naqvi S. M. R., Raza A., Saeed Z. Demand Forecasting in Supply Chain Management for Rossmann Stores Using Weather Enhanced Deep Learning Model. IEEE Access. 2024. Vol. 12. P. 145570–145581. URL: https://ieeexplore.ieee.org/document/10703040/ (date access: 02.05.2026).

15. Molnar C. Interpretable Machine Learning: A Guide for Making Black Box Models Explainable. 2nd ed. 2022. URL: https://christophm.github.io/interpretable–ml–book (date access: 20.02.2026).

16. Hyndman R. J., Athanasopoulos G. Forecasting: Principles and Practice. 3rd ed. OTexts, 2021. 442 p. URL: https://otexts.com/fpp3 (date access: 03.02.2026).

17. Box G. E. P., Jenkins G. M., Reinsel G. C., Ljung G. M. Time Series Analysis: Forecasting and Control. 5th ed. Hoboken: Wiley, 2015. 712 p. URL:

https://doi.org/10.1002/9781118619193 (date access: 03.02.2026).

18. Hochreiter S., Schmidhuber J. Long Short–Term Memory. Neural Computation. 1997. Vol. 9, No. 8. P. 1735–1780. URL: https://doi.org/10.1162/neco.1997.9.8.1735 (date access: 05.02.2026).

19. Kingma D. P., Ba J. Adam: A Method for Stochastic Optimization. 3rd International Conference on Learning Representations (ICLR 2015). 2015. URL: https://arxiv.org/abs/1412.6980 (date access: 05.02.2026).

20. Lim B., Zohren S. Time–Series Forecasting with Deep Learning: A Survey. Philosophical Transactions of the Royal Society A. 2021. Vol. 379, No. 2194. URL: https://doi.org/10.1098/rsta.2020.0209 (date access: 05.02.2026).

21. Bates J. M., Granger C. W. J. The Combination of Forecasts. Journal of the Operational Research Society. 1969. Vol. 20, No. 4. P. 451–468. URL: https://doi.org/10.1057/jors.1969.103 (date access: 15.01.2026).

22. Sharma A., Kulkarni A. Weather Impact on Retail Demand Forecasting Using Machine Learning. International Journal of Information Management Data Insights. 2022. Vol. 2, No. 2. P. 100–112. URL: https://doi.org/10.1016/j.jjimei.2022.100085 (date access: 11.03.2026).

23. Cerqueira V., Torgo L., Mozetič I. Evaluating Time Series Forecasting Models: An Empirical Study on Performance Estimation Methods. Machine Learning. 2020. Vol. 109. P. 1997–2028. URL: https://doi.org/10.1007/s10994– 020– 05910– 7 (date access: 10.02.2026).

24. Makridakis S., Spiliotis E., Assimakopoulos V. The M4 Competition: 100,000 Time Series and 61 Forecasting Methods. International Journal of Forecasting. 2020. Vol. 36, No. 1. P. 54–74. URL: https://doi.org/10.1016/j.ijforecast.2019.04.014 (date access: 10.02.2026).

25. Kolassa S. Why the "Best" Point Forecast Depends on the Error or Accuracy Measure. International Journal of Forecasting. 2020. Vol. 36, No. 1. P. 208–211. URL: https://doi.org/10.1016/j.ijforecast.2019.02.017 (date access: 10.02.2026).

26. Fielding R. T. Architectural Styles and the Design of Network– based Software Architectures: PhD dissertation. University of California, Irvine, 2000. URL: https://ics.uci.edu/~fielding/pubs/dissertation/top.htm (date access: 17.02.2026).

27. Grinberg M. Flask Web Development: Developing Web Applications with Python. 2nd ed. O'Reilly Media, 2018. 316 p. URL: https://www.oreilly.com/library/view/flask–web–development/9781491991725 (date access: 17.02.2026).

28. Ramírez S. FastAPI: Modern, Fast Web Framework for Building APIs with Python. GitHub. 2024. URL: https://github.com/tiangolo/fastapi (date access: 17.02.2026).

29. OpenAPI Initiative. OpenAPI Specification v3.1.0. Linux Foundation. 2021. URL: https://spec.openapis.org/oas/v3.1.0 (date access: 17.02.2026).

30. Abadi M., Barham P., Chen J. et al. TensorFlow: A System for Large–Scale Machine Learning. 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16). 2016. P. 265–283. URL: https://arxiv.org/abs/1603.04467 (date access: 05.04.2026).

31. McKinney W. Data Structures for Statistical Computing in Python. Proceedings of the 9th Python in Science Conference. 2010. P. 56–61. URL: https://doi.org/10.25080/Majora– 92bf1922–00a (date access: 02.04.2026).

32. Pedregosa F., Varoquaux G., Gramfort A. et al. Scikit–learn: Machine Learning in Python. Journal of Machine Learning Research. 2011. Vol. 12. P. 2825–2830. URL: https://jmlr.org/papers/v12/pedregosa11a.html (date access: 03.04.2026).

33. Momjian B. PostgreSQL: Introduction and Concepts. Addison– Wesley, 2001. 461 p. URL: https://www.postgresql.org/docs/ (date access: 18.02.2026).

34. Go ogle LLC. Firebase: App Development Platform. Official Documentation. 2024. URL: https://firebase.google.com/docs (date access: 19.02.2026).

35. Meta Platforms, Inc. React: A JavaScript Library for Building User Interfaces. Official Documentation. 2024. URL: https://react.dev (date access: 25.04.2026).

36. Microsoft Corporation. TypeScript: JavaScript with Syntax for Types. Official Documentation. 2024. URL: https://www.typescriptlang.org/docs (date access: 25.04.2026).

37. Recharts Group. Recharts: Redefined Chart Library Built with React and D3. GitHub. 2024. URL: https://recharts.org/en–US (date access: 24.04.2026).

38. Li Z., Zhang N. Short–Term Demand Forecast of E–Commerce Platform Based on ConvLSTM Network. Computational Intelligence and Neuroscience. 2022. Article 5227829. URL: https://doi.org/10.1155/2022/5227829 (date access: 15.01.2026).

39. Géron A. Hands–On Machine Learning with Scikit–Learn, Keras, and TensorFlow. 3rd ed. O'Reilly Media, 2022. 861 p. URL: https://www.oreilly.com/library/view/hands– on– machine– learning/9781098125967 (date access: 05.04.2026).

ДОДАТОК А
Код модуля машинного навчання

А.1 Конвеєр машинного навчання

```
import uuid
import asyncio
import logging
from functools import partial
import numpy as np
import pandas as pd
from datetime import date, timedelta
from supabase import Client

logger = logging.getLogger(__name__)

from app.services.open_meteo import fetch_weather
from app.services.google_trends import fetch_trends
from app.services.feature_eng import build_feature_matrix
from app.ml.prophet_model import train_predict_prophet
from app.ml.lstm_model import train_predict_lstm
from app.ml.ensemble import combine_ensemble
from app.ml.metrics import compute_metrics, compute_adequacy_metrics

MIN_ROWS      = 14
LSTM_MIN_ROWS = 60

async def run_forecast_pipeline(
    product_id: str, forecast_id: str, horizon_days: int,
    model_type: str, include_weather: bool, include_trends: bool, supabase: Client,
):
    try:
        await _run(product_id, forecast_id, horizon_days, model_type,
                   include_weather, include_trends, supabase)
    except Exception as exc:
        supabase.table("forecasts").update({"status": "failed"}).eq("id",
forecast_id).execute()
        raise exc

async def _run(product_id, forecast_id, horizon_days, model_type,
               include_weather, include_trends, supabase):

    sales_result = (
        supabase.table("sales").select("date,quantity")
        .eq("product_id", product_id).order("date").range(0, 49999).execute()
    )
    if not sales_result.data:
        raise ValueError("No sales data found for product")

    sales_df = pd.DataFrame(sales_result.data)
    sales_df["quantity"] = pd.to_numeric(sales_df["quantity"],
errors="coerce").fillna(0.0)
    sales_df = sales_df.sort_values("date").reset_index(drop=True)
    if len(sales_df) < MIN_ROWS:
        raise ValueError(f"Need at least {MIN_ROWS} days of sales data")

    product = (
```

```
        supabase.table("products").select("latitude,longitude,search_keyword")
        .eq("id", product_id).single().execute()
    ).data or {}
    lat     = float(product.get("latitude")  or 50.4501)
    lon     = float(product.get("longitude") or 30.5234)
    keyword = product.get("search_keyword")  or ""

    hist_start = sales_df["date"].iloc[0]
    today_str  = date.today().isoformat()
    future_end = (date.today() + timedelta(days=horizon_days)).isoformat()

    weather_df = pd.DataFrame(columns=["date", "temperature", "precipitation",
"uv_index"])
    if include_weather:
        try:
            weather_df = await fetch_weather(lat, lon, hist_start, future_end)
        except Exception:
            pass

    trends_df = pd.DataFrame(columns=["date", "search_interest"])
    if include_trends and keyword:
        try:
            trends_df = await asyncio.get_event_loop().run_in_executor(
                None, partial(fetch_trends, keyword, hist_start, today_str)
            )
        except Exception:
            pass

    hist_df    = build_feature_matrix(
        sales_df,
        weather_df[weather_df["date"] <= today_str] if not weather_df.empty else
weather_df,
        trends_df,
    )
    n          = len(hist_df)
    regressors = []
    if include_weather and not weather_df.empty:
        regressors += [c for c in ["temperature", "precipitation", "uv_index"] if c in
hist_df.columns]
    if include_trends and not trends_df.empty and "search_interest" in hist_df.columns:
        regressors.append("search_interest")

    future_df = pd.DataFrame({"date": [
        d.strftime("%Y-%m-%d")
        for d in pd.date_range(start=date.today() + timedelta(days=1),
periods=horizon_days, freq="D")
    ]})
    if regressors and not weather_df.empty:
        fw = weather_df[weather_df["date"] > today_str].copy()
        if not fw.empty:
            future_df =
future_df.merge(fw[["date","temperature","precipitation","uv_index"]],
                                        on="date", how="left")
        for col in ["temperature", "precipitation", "uv_index"]:
            fb = float(hist_df[col].mean()) if col in hist_df.columns else 0.0
            future_df[col] = future_df.get(col, pd.Series([fb]*len(future_df))).fillna(fb)
    if "search_interest" in regressors:
        future_df["search_interest"] = float(hist_df["search_interest"].iloc[-7:].mean())
```

```
    val_size = max(7, min(30, int(n * 0.2)))
    split    = n - val_size
    train_hist    = hist_df.iloc[:split].copy()
    val_actuals   = hist_df["quantity"].values[split:]
    val_future_df = hist_df.iloc[split:][["date"] + [c for c in regressors if c in
hist_df.columns]].copy()
    val_future_df["date"] = pd.to_datetime(val_future_df["date"]).dt.strftime("%Y-%m-%d")

    use_lstm    = model_type in ("lstm", "ensemble") and n >= LSTM_MIN_ROWS
    use_prophet = model_type in ("prophet", "ensemble") or (model_type == "lstm" and not
use_lstm)
    prophet_val_preds = lstm_val_preds = prophet_final = lstm_final = None

    if use_prophet:
        pv            = train_predict_prophet(train_hist, val_size, regressors,
val_future_df)
        prophet_val_preds = [p["predicted"] for p in pv["predictions"]]
        prophet_final = train_predict_prophet(hist_df, horizon_days, regressors,
future_df)

    if use_lstm:
        lv            = train_predict_lstm(train_hist, val_size, regressors,
val_future_df)
        lstm_val_preds = [p["predicted"] for p in lv["predictions"]]
        _m             = min(len(lstm_val_preds), len(val_actuals))
        lstm_final    = train_predict_lstm(
            hist_df, horizon_days, regressors, future_df,
            residual_std=float(np.std(val_actuals[-_m:] - np.array(lstm_val_preds[-
_m:]))),
        )

    if model_type == "ensemble" and prophet_final and lstm_final:
        final_preds, alpha = combine_ensemble(
            prophet_final["predictions"], lstm_final["predictions"],
            prophet_val_preds, lstm_val_preds, val_actuals,
        )
        ml = min(len(prophet_val_preds), len(lstm_val_preds), len(val_actuals))
        val_preds = (alpha * np.array(prophet_val_preds[-ml:])
                     + (1 - alpha) * np.array(lstm_val_preds[-ml:]))
        val_actuals_trimmed = val_actuals[-ml:]
    elif prophet_final:
        final_preds, alpha  = prophet_final["predictions"], 1.0
        ml = min(len(prophet_val_preds), len(val_actuals))
        val_preds, val_actuals_trimmed = np.array(prophet_val_preds[-ml:]), val_actuals[-
ml:]
    else:
        final_preds, alpha  = lstm_final["predictions"], 0.0
        ml = min(len(lstm_val_preds), len(val_actuals))
        val_preds, val_actuals_trimmed = np.array(lstm_val_preds[-ml:]), val_actuals[-ml:]

    metrics = compute_metrics(val_actuals_trimmed, val_preds)

    def _indiv_metrics(preds, actuals):
        ml = min(len(preds), len(actuals))
        return {**compute_metrics(actuals[-ml:], np.array(preds[-ml:])),
                **compute_adequacy_metrics(actuals[-ml:], np.array(preds[-ml:]))}
```

```
    prophet_metrics = _indiv_metrics(prophet_val_preds, val_actuals) if prophet_val_preds
else None
    lstm_metrics    = _indiv_metrics(lstm_val_preds,    val_actuals) if lstm_val_preds
else None

    supabase.table("forecast_values").insert([
        {
            "id":                str(uuid.uuid4()),
            "forecast_id":       forecast_id,
            "date":              p["date"],
            "predicted_quantity": round(float(p["predicted"]), 4),
            "lower_bound":       round(float(max(0.0, p.get("lower", 0.0))), 4),
            "upper_bound":       round(float(p.get("upper", p["predicted"] * 1.2)), 4),
        }
        for p in final_preds
    ]).execute()

    shap_data = await asyncio.get_event_loop().run_in_executor(
        None, partial(_compute_shap, hist_df, regressors)
    )
    if shap_data:
        supabase.table("shap_values").insert([
            {"id": str(uuid.uuid4()), "forecast_id": forecast_id,
             "feature_name": feat, "shap_value": round(float(v), 6),
             "mean_abs_shap": round(abs(float(v)), 6)}
            for feat, v in shap_data.items()
        ]).execute()

    update_data = {
        "status": "completed",
        "alpha":  round(float(alpha), 4),
        **{k: round(metrics[k], 4) for k in ("mae", "rmse", "mape", "r2")},
    }
    if prophet_metrics:
        update_data["prophet_metrics"] = {k: (round(v, 4) if v is not None else None)
                                          for k, v in prophet_metrics.items()}
    if lstm_metrics:
        update_data["lstm_metrics"]    = {k: (round(v, 4) if v is not None else None)
                                          for k, v in lstm_metrics.items()}
    if prophet_final:
        update_data["prophet_components"] = prophet_final.get("prophet_components")
    supabase.table("forecasts").update(update_data).eq("id", forecast_id).execute()

def _compute_shap(hist_df: pd.DataFrame, regressors: list) -> dict:
    try:
        import shap
        from sklearn.ensemble import GradientBoostingRegressor
        cols = [c for c in [
            "day_of_week","month","week_of_year","is_weekend",
            "lag_7","lag_14","lag_30","rolling_7","rolling_14",
        ] + regressors if c in hist_df.columns]
        X = hist_df[cols].fillna(0.0).values
        y = hist_df["quantity"].values.astype(float)
        if len(X) < 20:
            return {}
        gbm = GradientBoostingRegressor(n_estimators=100, max_depth=4, random_state=42)
        gbm.fit(X, y)
```

```
        shap_vals = shap.TreeExplainer(gbm).shap_values(X)
        return dict(zip(cols, np.abs(shap_vals).mean(axis=0).tolist()))
    except Exception:
        return {}
```

А.2 Комбінування прогнозів

```
import numpy as np
from scipy.optimize import minimize_scalar
from typing import List, Tuple

def combine_ensemble(
    prophet_future: List[dict], lstm_future: List[dict],
    prophet_val_preds: List[float], lstm_val_preds: List[float],
    y_val: np.ndarray,
) -> Tuple[List[dict], float]:
    p_val = np.array(prophet_val_preds, dtype=float)
    l_val = np.array(lstm_val_preds,    dtype=float)
    y     = np.array(y_val,             dtype=float)
    ml    = min(len(p_val), len(l_val), len(y))
    p_val, l_val, y = p_val[-ml:], l_val[-ml:], y[-ml:]

    def mae_loss(a: float) -> float:
        return float(np.mean(np.abs(y - (a * p_val + (1 - a) * l_val))))

    alpha = float(minimize_scalar(mae_loss, bounds=(0.0, 1.0), method="bounded").x)

    blended = []
    for p, lp in zip(prophet_future[:len(lstm_future)],
lstm_future[:len(prophet_future)]):
        pred  = alpha * p["predicted"]             + (1 - alpha) * lp["predicted"]
        lower = alpha * p.get("lower", pred * 0.8) + (1 - alpha) * lp.get("lower", pred *
0.8)
        upper = alpha * p.get("upper", pred * 1.2) + (1 - alpha) * lp.get("upper", pred *
1.2)
        blended.append({"date": p["date"],
                        "predicted": max(0.0, pred),
                        "lower": max(0.0, lower),
                        "upper": max(0.0, upper)})
    return blended, alpha
```

А.3 Формування матриці ознак

```
import pandas as pd

def build_feature_matrix(
    sales_df: pd.DataFrame, weather_df: pd.DataFrame, trends_df: pd.DataFrame,
) -> pd.DataFrame:
    df = sales_df[["date", "quantity"]].copy()
    df["date"] = pd.to_datetime(df["date"])
    df = df.sort_values("date").reset_index(drop=True)

    df["day_of_week"]  = df["date"].dt.dayofweek
    df["month"]        = df["date"].dt.month
    df["week_of_year"] = df["date"].dt.isocalendar().week.astype(int)
    df["is_weekend"]   = (df["day_of_week"] >= 5).astype(int)
```

```
    df["lag_7"]        = df["quantity"].shift(7).fillna(0)
    df["lag_14"]       = df["quantity"].shift(14).fillna(0)
    df["lag_30"]       = df["quantity"].shift(30).fillna(0)
    df["rolling_7"]    = df["quantity"].rolling(7,  min_periods=1).mean()
    df["rolling_14"]   = df["quantity"].rolling(14, min_periods=1).mean()

    if not weather_df.empty:
        w = weather_df.copy()
        w["date"] = pd.to_datetime(w["date"])
        df = df.merge(w[["date","temperature","precipitation","uv_index"]], on="date",
how="left")
        df["temperature"]   = df["temperature"].ffill().fillna(0.0)
        df["precipitation"] = df["precipitation"].fillna(0.0)
        df["uv_index"]      = df["uv_index"].fillna(0.0)
    else:
        df["temperature"] = df["precipitation"] = df["uv_index"] = 0.0

    if not trends_df.empty:
        t = trends_df.copy()
        t["date"] = pd.to_datetime(t["date"])
        df = df.merge(t[["date","search_interest"]], on="date", how="left")
        df["search_interest"] = df["search_interest"].fillna(0.0)
    else:
        df["search_interest"] = 0.0
    return df
```

А.4 Обчислення метрик якості

```
import numpy as np
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

def compute_metrics(y_true, y_pred) -> dict:
    y_true = np.array(y_true, dtype=float)
    y_pred = np.array(y_pred, dtype=float)
    mask   = y_true != 0
    return {
        "mae":  float(mean_absolute_error(y_true, y_pred)),
        "rmse": float(np.sqrt(mean_squared_error(y_true, y_pred))),
        "mape": float(np.mean(np.abs((y_true[mask] - y_pred[mask]) / y_true[mask])) * 100)
                if mask.any() else 0.0,
        "r2":   float(r2_score(y_true, y_pred)),
    }

def compute_adequacy_metrics(y_true, y_pred) -> dict:
    from statsmodels.stats.diagnostic import acorr_ljungbox
    residuals = np.array(y_true, dtype=float) - np.array(y_pred, dtype=float)
    ljung_box_p = None
    try:
        lb_df = acorr_ljungbox(residuals, lags=[10], return_df=True)
        ljung_box_p = float(lb_df["lb_pvalue"].iloc[0])
    except Exception:
        pass
    return {
        "residual_mean": float(np.mean(residuals)),
        "residual_std":  float(np.std(residuals)),
        "ljung_box_p":   ljung_box_p,
    }
```

ДОДАТОК Б
Код моделей прогнозування

Б.1 Модуль прогнозування на основі Prophet

```
import warnings
import pandas as pd
import numpy as np
from prophet import Prophet
from typing import List, Optional

warnings.filterwarnings("ignore")


def train_predict_prophet(
    hist_df: pd.DataFrame,
    horizon: int,
    regressors: List[str],
    future_regressors: Optional[pd.DataFrame] = None,
) -> dict:
    valid_regs = [r for r in regressors if r in hist_df.columns]

    train = hist_df[["date","quantity"] + valid_regs].copy()
    train["date"] = pd.to_datetime(train["date"])
    train = train.rename(columns={"date": "ds", "quantity": "y"})

    model = Prophet(
        yearly_seasonality=True,
        weekly_seasonality=True,
        daily_seasonality=False,
        interval_width=0.95,
        changepoint_prior_scale=0.05,
    )
    for reg in valid_regs:
        model.add_regressor(reg)
    model.fit(train)

    future = model.make_future_dataframe(periods=horizon, freq="D")

    for reg in valid_regs:
        hist_reg = train.set_index("ds")[reg]
        future   = future.set_index("ds")
        future[reg] = hist_reg.reindex(future.index)

        if future_regressors is not None and reg in future_regressors.columns:
            fr = future_regressors.copy()
            fr["date"] = pd.to_datetime(fr["date"])
            fr_idx = fr.set_index("date")[reg]
            mask   = future[reg].isna()
            future.loc[mask, reg] = fr_idx.reindex(future.index[mask]).values

        fallback = (float(hist_reg.dropna().iloc[-7:].mean())
                    if len(hist_reg.dropna()) >= 7 else 0.0)
        future[reg] = future[reg].fillna(fallback)
        future = future.reset_index()

    forecast  = model.predict(future)
```

```
    hist_len  = len(train)
    future_fc = forecast.iloc[hist_len:][["ds","yhat","yhat_lower","yhat_upper"]]

    return {
        "predictions": [
            {
                "date":      row["ds"].date().isoformat(),
                "predicted": max(0.0, float(row["yhat"])),
                "lower":     max(0.0, float(row["yhat_lower"])),
                "upper":     max(0.0, float(row["yhat_upper"])),
            }
            for _, row in future_fc.iterrows()
        ],
        "prophet_components": {
            "dates":  [d.date().isoformat() for d in future_fc["ds"]],
            "trend":  forecast.iloc[hist_len:]["trend"].tolist()   if "trend"  in
forecast.columns else [],
            "weekly": forecast.iloc[hist_len:]["weekly"].tolist()  if "weekly" in
forecast.columns else [],
            "yearly": forecast.iloc[hist_len:]["yearly"].tolist()  if "yearly" in
forecast.columns else [],
        },
    }
```

Б.2 Модуль прогнозування на основі LSTM

```
import os, random, warnings
import numpy as np
import pandas as pd
from sklearn.preprocessing import MinMaxScaler

warnings.filterwarnings("ignore")
os.environ.setdefault("TF_CPP_MIN_LOG_LEVEL", "3")

LOOKBACK = 30
_SEED    = 42

def _set_seeds():
    random.seed(_SEED); np.random.seed(_SEED)
    import tensorflow as tf; tf.random.set_seed(_SEED)

def _build_sequences(data: np.ndarray, lookback: int):
    X, y = [], []
    for i in range(lookback, len(data)):
        seq = data[i - lookback:i].copy()
        if data.shape[1] > 1:
            seq[-1, 1:] = data[i, 1:]
        X.append(seq); y.append(data[i, 0])
    return np.array(X), np.array(y)

def train_predict_lstm(
    hist_df: pd.DataFrame,
    horizon: int,
    regressors: list | None = None,
    future_regressors_df: pd.DataFrame | None = None,
    residual_std: float | None = None,
) -> dict:
```

```
    _set_seeds()
    import tensorflow as tf; tf.get_logger().setLevel("ERROR")
    from tensorflow.keras.models import Sequential
    from tensorflow.keras.layers import LSTM, Dense, Dropout
    from tensorflow.keras.callbacks import EarlyStopping

    regressors   = regressors or []
    feature_cols = ["quantity"] + [r for r in regressors if r in hist_df.columns]
    n_features   = len(feature_cols)
    series       = hist_df[feature_cols].values.astype(float)
    n            = len(series)
    last_date    = pd.to_datetime(hist_df["date"].iloc[-1])
    future_dates = pd.date_range(last_date + pd.Timedelta(days=1), periods=horizon,
freq="D")

    if n < LOOKBACK + 10:
        mean_val = float(np.mean(series[:, 0])) if n > 0 else 0.0
        half     = 1.96 * residual_std if residual_std else mean_val * 0.2
        return {"predictions": [
            {"date": d.date().isoformat(), "predicted": mean_val,
             "lower": max(0.0, mean_val - half), "upper": mean_val + half}
            for d in future_dates
        ]}
    scaler       = MinMaxScaler()
    scaled       = scaler.fit_transform(series)
    X_all, y_all = _build_sequences(scaled, LOOKBACK)
    model = Sequential([
        LSTM(64, return_sequences=True, input_shape=(LOOKBACK, n_features)),
        Dropout(0.2),
        LSTM(32),
        Dropout(0.2),
        Dense(1),
    ])
    model.compile(optimizer="adam", loss="mse")
    model.fit(
        X_all, y_all,
        epochs=100, batch_size=16, validation_split=0.1,
        callbacks=[EarlyStopping(patience=20, restore_best_weights=True)],
        verbose=0,
    )
    last_seq, future_scaled_qty = scaled[-LOOKBACK:].copy(), []
    for _ in range(horizon):
        pred = float(model.predict(last_seq.reshape(1, LOOKBACK, n_features),
verbose=0)[0][0])
        future_scaled_qty.append(pred)
        new_row       = np.zeros(n_features)
        new_row[0]    = pred
        last_seq      = np.vstack([last_seq[1:], new_row])
    qty_dummy       = np.zeros((horizon, n_features))
    qty_dummy[:, 0] = future_scaled_qty
    future_vals     = np.clip(scaler.inverse_transform(qty_dummy)[:, 0], 0.0, None)
    half = 1.96 * residual_std if residual_std else None
    return {"predictions": [
        {"date": d.date().isoformat(), "predicted": float(v),
         "lower": float(max(0.0, v - half if half else v * 0.8)),
         "upper": float(v + half if half else v * 1.2)}
        for d, v in zip(future_dates, future_vals)
    ]}
```

ДОДАТОК В
Код серверної частини

В.1 Модуль безпеки

```
from datetime import datetime, timedelta
from jose import jwt
from passlib.context import CryptContext
from app.core.config import settings

pwd_context = CryptContext(schemes=["bcrypt"], deprecated="auto")

def hash_password(password: str) -> str:
    return pwd_context.hash(password)

def verify_password(plain: str, hashed: str) -> bool:
    return pwd_context.verify(plain, hashed)

def create_access_token(data: dict) -> str:
    to_encode = {**data, "exp": datetime.utcnow() +
timedelta(minutes=settings.ACCESS_TOKEN_EXPIRE_MINUTES)}
    return jwt.encode(to_encode, settings.JWT_SECRET, algorithm=settings.ALGORITHM)

def decode_token(token: str) -> dict:
    return jwt.decode(token, settings.JWT_SECRET, algorithms=[settings.ALGORITHM])
```

В.2 Перевірка JWT-токена та залежності

```
from fastapi import Depends, HTTPException, status
from fastapi.security import HTTPBearer, HTTPAuthorizationCredentials
from jose import JWTError
from app.core.security import decode_token
from app.db.supabase import get_supabase
bearer_scheme = HTTPBearer()

async def get_current_user(credentials: HTTPAuthorizationCredentials =
Depends(bearer_scheme)):
    exc = HTTPException(
        status_code=status.HTTP_401_UNAUTHORIZED,
        detail="Could not validate credentials",
        headers={"WWW-Authenticate": "Bearer"},
    )
    try:
        payload = decode_token(credentials.credentials)
        user_id = payload.get("sub")
        if user_id is None:
            raise exc
    except JWTError:
        raise exc
    result = get_supabase().table("users").select("*").eq("id",
user_id).single().execute()
    if not result.data:
        raise exc
    return result.data
```

```
def get_user_product(product_id: str, user_id: str, supabase) -> dict:
    result = supabase.table("products").select("*") \
        .eq("id", product_id).eq("user_id", user_id).execute()
    if not result.data:
        raise HTTPException(status_code=404, detail="Product not found")
    return result.data[0]
```

В.3 Модуль управління даними продажів

```
from fastapi import APIRouter, HTTPException, UploadFile, File, Depends
from app.db.supabase import get_supabase
from app.api.deps import get_current_user, get_user_product
import pandas as pd, io
router = APIRouter()

def validate_csv(df: pd.DataFrame) -> dict:
    errors, warnings = [], []
    missing = [c for c in ["date","quantity"] if c not in df.columns]
    if missing:
        errors.append(f"Відсутні обов'язкові колонки: {missing}")
        return {"errors": errors, "warnings": warnings, "valid": False}
    try:
        pd.to_datetime(df["date"])
    except Exception:
        errors.append("Колонка 'date' містить некоректний формат (очікується YYYY-MM-DD)")
    if not pd.to_numeric(df["quantity"], errors="coerce").notna().all():
        errors.append("Колонка 'quantity' містить нечислові значення")
    missing_pct = df["quantity"].isna().mean() * 100
    if missing_pct > 0:
        warnings.append(f"Знайдено {missing_pct:.1f}% пропущених значень - буде заповнено
інтерполяцією")
    if len(df) < 90:
        warnings.append(f"Мало даних ({len(df)} рядків) - рекомендується мінімум 90 днів")
    return {"errors": errors, "warnings": warnings, "valid": len(errors) == 0}

@router.post("/upload/{product_id}", status_code=201)
async def upload_sales(
    product_id: str, file: UploadFile = File(...),
    current_user: dict = Depends(get_current_user)
):
    supabase = get_supabase()
    get_user_product(product_id, current_user["id"], supabase)
    content = await file.read()
    try:
        df = pd.read_csv(io.StringIO(content.decode("utf-8")))
    except Exception:
        raise HTTPException(status_code=400, detail="Invalid CSV file")
    validation = validate_csv(df)
    if not validation["valid"]:
        raise HTTPException(status_code=422, detail={"message": "CSV validation failed",
            "errors": validation["errors"], "warnings": validation["warnings"]})
    df["date"] = pd.to_datetime(df["date"]).dt.strftime("%Y-%m-%d")
    df["quantity"]   = df["quantity"].astype(int)
    df["product_id"] = product_id
    if "price" not in df.columns:
        df["price"] = None
    records = df[["product_id","date","quantity","price"]].to_dict(orient="records")
```

```
    for i in range(0, len(records), 500):
        supabase.table("sales").upsert(records[i:i+500],
on_conflict="product_id,date").execute()
    return {"inserted": len(records), "warnings": validation["warnings"]}
```

В.4 Модуль управління прогнозами

```
from fastapi import APIRouter, HTTPException, BackgroundTasks, Depends
from fastapi.responses import StreamingResponse
from app.db.models import ForecastRequest
from app.db.supabase import get_supabase
from app.api.deps import get_current_user, get_user_product, get_user_forecast
import uuid, pandas as pd, io
router = APIRouter()

@router.post("/run/{product_id}", status_code=202)
async def run_forecast(
    product_id: str, request: ForecastRequest,
    background_tasks: BackgroundTasks,
    current_user: dict = Depends(get_current_user)
):
    supabase    = get_supabase()
    get_user_product(product_id, current_user["id"], supabase)
    forecast_id = str(uuid.uuid4())
    supabase.table("forecasts").insert({
        "id": forecast_id, "product_id": product_id,
        "model_type": request.model_type, "horizon_days": request.horizon_days,
        "status": "running",
    }).execute()
    from app.ml.pipeline import run_forecast_pipeline
    background_tasks.add_task(run_forecast_pipeline,
        product_id=product_id, forecast_id=forecast_id,
        horizon_days=request.horizon_days, model_type=request.model_type,
        include_weather=request.include_weather, include_trends=request.include_trends,
        supabase=supabase)
    return {"forecast_id": forecast_id, "status": "running"}
@router.get("/{forecast_id}/export")
def export_forecast(forecast_id: str, current_user: dict = Depends(get_current_user)):
    supabase = get_supabase()
    get_user_forecast(forecast_id, current_user["id"], supabase)
    result = (supabase.table("forecast_values").select("*")
              .eq("forecast_id", forecast_id).order("date").execute())
    if not result.data:
        raise HTTPException(status_code=404, detail="Forecast not found")
    df = (pd.DataFrame(result.data)
          [["date","predicted_quantity","lower_bound","upper_bound"]]
          .rename(columns={"predicted_quantity":"ensemble",
                           "lower_bound":"lower","upper_bound":"upper"}))
    df["actual"] = df["prophet"] = df["lstm"] = None
    stream = io.StringIO()
    df[["date","actual","prophet","lstm","ensemble","lower","upper"]].to_csv(stream,
index=False)
    stream.seek(0)
    return StreamingResponse(iter([stream.getvalue()]), media_type="text/csv",
        headers={"Content-Disposition": f"attachment;
filename=forecast_{forecast_id}.csv"})
```

ДОДАТОК Г
Клієнтська частина

Г.1 Хук асинхронного опитування стану прогнозу

```
import { useState, useEffect, useRef } from 'react';
import { forecastsApi } from '../api/forecasts';
import type { ForecastStatus } from '../types';

interface PollingState {
  status:   ForecastStatus | null;
  progress: number;
  metrics:  { mae?: number; rmse?: number; mape?: number; r2?: number } | null;
  error:    string | null;
}

const POLL_INTERVAL_MS = 3000;

function statusToProgress(s: string) {
  return s === 'pending' ? 10 : s === 'running' ? 55 : s === 'completed' ? 100 : 0;
}

export function useForecastPolling(forecastId: string | null) {
  const [state, setState] = useState<PollingState>(
    { status: null, progress: 0, metrics: null, error: null }
  );
  const intervalRef = useRef<ReturnType<typeof setInterval> | null>(null);

  const stop = () => { if (intervalRef.current) clearInterval(intervalRef.current); };

  useEffect(() => {
    if (!forecastId) return;
    setState({ status: 'pending', progress: 10, metrics: null, error: null });

    const poll = async () => {
      try {
        const data = await forecastsApi.getStatus(forecastId);
        setState({
          status:   data.status as ForecastStatus,
          progress: statusToProgress(data.status),
          metrics:  data.mae != null
            ? { mae: data.mae, rmse: data.rmse, mape: data.mape, r2: data.r2 } : null,
          error: null,
        });
        if (data.status === 'completed' || data.status === 'failed') {
          stop();
          if (data.status === 'failed')
            setState((p) => ({ ...p, error: 'Forecast failed. Please try again.' }));
        }
      } catch {
        stop();
        setState((p) => ({ ...p, error: 'Failed to get forecast status.' }));
      }
    };

    poll();
    intervalRef.current = setInterval(poll, POLL_INTERVAL_MS);
```

```
    return stop;
  }, [forecastId]);

  return state;
}
```

Г.2 Компонент сторінки прогнозування

```
import { useState, useEffect, useCallback } from 'react';
import { useProducts } from '../hooks/useProducts';
import { useForecastPolling } from '../hooks/useForecastPolling';
import { forecastsApi } from '../api/forecasts';
import { salesApi } from '../api/sales';
import type { ModelType, ResultsState, Tab } from '../types';

export function ForecastPage() {
  const { products } = useProducts();
  const [selectedProductId, setSelectedProductId] = useState('');
  const [modelType,   setModelType]   = useState<ModelType>('ensemble');
  const [horizonDays, setHorizonDays] = useState(30);
  const [inclWeather, setInclWeather] = useState(false);
  const [inclTrends,  setInclTrends]  = useState(false);
  const [currentForecastId, setCurrentForecastId] = useState<string | null>(null);
  const [results,   setResults]   = useState<ResultsState | null>(null);
  const [activeTab, setActiveTab] = useState<Tab>('forecast');

  const { status: pollStatus, progress } = useForecastPolling(currentForecastId);
  const isPolling = pollStatus === 'pending' || pollStatus === 'running';

  const loadResults = useCallback(async (id: string, pid: string, mt: ModelType) => {
    setResults(null);
    const [v, s, c, cmp, rec, sr] = await Promise.allSettled([
      forecastsApi.getValues(id, pid), forecastsApi.getShap(id),
      forecastsApi.getComponents(id),  forecastsApi.getComparison(id),
      forecastsApi.getRecommendations(id), salesApi.getSales(pid),
    ]);
    const values = v.status === 'fulfilled' ? v.value : null;
    if (values?.ensemble.length) {
      setResults({ forecastId: id, modelType: mt, values,
        shap:            s.status   === 'fulfilled' ? s.value   : null,
        components:      c.status   === 'fulfilled' ? c.value   : null,
        comparison:      cmp.status === 'fulfilled' ? cmp.value : null,
        recommendations: rec.status === 'fulfilled' ? rec.value : null,
        historicalSales: sr.status  === 'fulfilled'
          ? [...sr.value].sort((a, b) => a.date.localeCompare(b.date)) : [],
      });
      setActiveTab('forecast');
    }
  }, []);

  useEffect(() => {
    if (pollStatus === 'completed' && currentForecastId && selectedProductId)
      setTimeout(() => loadResults(currentForecastId, selectedProductId, modelType), 800);
  }, [pollStatus, currentForecastId, selectedProductId, modelType, loadResults]);

  const handleRun = async () => {
    setResults(null); setCurrentForecastId(null);
```

```
    const res = await forecastsApi.run(selectedProductId, {
      horizon_days: horizonDays, model_type: modelType,
      include_weather: inclWeather, include_trends: inclTrends,
    });
    setCurrentForecastId(res.forecast_id);
  };

  const handleExport = async () => {
    if (!results) return;
    const blob = await forecastsApi.exportCsv(results.forecastId);
    const a = Object.assign(document.createElement('a'), {
      href: URL.createObjectURL(blob),
      download: `forecast-${results.forecastId.slice(0, 8)}.csv`,
    });
    a.click();
  };

  return (
    <div className="flex h-screen">
      <aside className="w-72 bg-gray-50 border-r p-5 space-y-4">
        <select value={selectedProductId} onChange={(e) =>
setSelectedProductId(e.target.value)}>
          {products.map((p) => <option key={p.id} value={p.id}>{p.name}</option>)}
        </select>
        <div className="grid grid-cols-3 gap-1">
          {(['prophet','lstm','ensemble'] as ModelType[]).map((m) => (
            <button key={m} onClick={() => setModelType(m)}
              className={modelType === m ? 'bg-indigo-600 text-white rounded' : 'border
rounded'}>
              {m}
            </button>
          ))}
        </div>
        <div className="flex gap-1 flex-wrap">
          {[7,14,30,60,90].map((d) => (
            <button key={d} onClick={() => setHorizonDays(d)}
              className={horizonDays === d ? 'bg-indigo-600 text-white px-2 rounded' :
'border px-2 rounded'}>
              {d}d
            </button>
          ))}
        </div>
        <button onClick={handleRun} disabled={!selectedProductId || isPolling}
          className="w-full bg-indigo-600 text-white py-2 rounded-xl disabled:opacity-50">
          {isPolling ? 'Running…' : 'Run Forecast'}
        </button>
        {results && (
          <button onClick={handleExport} className="w-full border py-2 rounded-lg text-
sm">
            Export CSV
          </button>
        )}
      </aside>

      <main className="flex-1 overflow-y-auto p-6">
        {isPolling && (
          <div className="w-full bg-gray-100 rounded-full h-2 mb-4">
            <div className="h-2 bg-indigo-600 rounded-full transition-all"
```

```
              style={{ width: `${progress}%` }} />
          </div>
        )}
        {results && (
          <>
            <nav className="flex border-b mb-4">
              {(['forecast','shap','components','comparison','insights'] as Tab[]).map((t)
=> (
                <button key={t} onClick={() => setActiveTab(t)}
                  className={activeTab === t
                    ? 'border-b-2 border-indigo-600 text-indigo-600 px-3 py-2 text-sm'
                    : 'px-3 py-2 text-sm text-gray-500'}>
                  {t}
                </button>
              ))}
            </nav>
            <div>
              {activeTab === 'forecast'   && results.values          && <ForecastChart
data={results.values} />}
              {activeTab === 'shap'       && results.shap            && <ShapChart
data={results.shap} />}
              {activeTab === 'components' && results.components      && <ComponentsChart
data={results.components} />}
              {activeTab === 'comparison' && results.comparison      && <ComparisonTable
data={results.comparison} />}
              {activeTab === 'insights'   && results.recommendations && <InsightsPanel
data={results.recommendations} />}
            </div>
          </>
        )}
      </main>
    </div>
  );
}
```