ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Розробка програмного забезпечення для планування закупівель та обліку витратних матеріалів кав'ярні»

на здобуття освітнього ступеня бакалавра
зі спеціальності 121 Інженерія програмного забезпечення
освітньо-професійної програми «Інженерія програмного забезпечення»

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Єлизавета ОЛЕКСІЄВА
(підпис)

Виконала: здобувачка вищої освіти групи ПД-42

Єлизавета ОЛЕКСІЄВА

Керівник: Ігор ГАМАНЮК
ст. викладач кафедри ІПЗ

Рецензент: _____

Київ 2025

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Інженерії програмного забезпечення

Ступінь вищої освіти Бакалавр

Спеціальність 121 Інженерія програмного забезпечення

Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного забезпечення
_____ Ірина ЗАМРІЙ
« ____ » _____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Олексієвій Єлизаветі Олегівні

1. Тема кваліфікаційної роботи: «Розробка програмного забезпечення для планування закупівель та обліку витратних матеріалів кав'ярні»
керівник кваліфікаційної роботи старший викладач кафедри Ігор ГАМАНЮК,
затверджені наказом Державного університету інформаційно-комунікаційних технологій від «24» лютого 2025 р. № 56.

2. Строк подання кваліфікаційної роботи «02» червня 2025 р.

3. Вихідні дані до кваліфікаційної роботи: технічна документація системи CoffeeTrack, теоретичні відомості щодо процесів обліку матеріалів у закладах громадського харчування, структура багаторівневої архітектури MVC, технічні специфікації засобів реалізації: ASP.NET Core MVC, Entity Framework Core, SQL Server, бібліотека Chart.js, фреймворк Bootstrap 5, а також інструменти Visual Studio та GitHub.

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

1. Аналіз існуючих програмних рішень для обліку матеріалів у закладах типу кав’ярень.

2. Проектування інформаційної системи для обліку матеріалів у кав’ярні з урахуванням практичних потреб бізнесу..
3. Реалізація програмної системи обліку матеріалів на основі архітектури MVC із використанням ASP.NET Core, EF Core та SQL Server.

5. Перелік графічного матеріалу: *презентація*
1. Аналіз аналогів.
2. Вимоги до програмного забезпечення.
3. Програмні засоби реалізації.
4. Діаграма діяльності.
5. Екранні форми.
6. Апробація результатів дослідження

6. Дата видачі завдання «25» лютого 2025 р.

КАЛЕНДАРНИЙ ПЛАН

	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Підбір та аналіз науково-технічної літератури	25.02.-04.03.2025	
2	Аналіз та дослідження існуючих аналогів	05.03-13.03.2025	
3	Аналіз вимог до обліку витратних матеріалів у закладах типу кав’ярня	14.03-20.03.2025	
4	Проектування архітектури інформаційної системи, бази даних і модулів системи	21.03-15.04.2025	
5	Реалізація функціоналу: закупівлі, списання, звітність, аналітика	16.04-28.04.2025	
7	Оформлення роботи: вступ, висновки, реферат	29.04-11.05.2025	
8	Розробка демонстраційних матеріалів	12.05-18.05.2025	
9	Попередній захист роботи	19.05-30.05.2025	

Здобувачка вищої освіти _____ Єлизавета ОЛЕКСІЄВА
(підпис)

Керівник
кваліфікаційної роботи _____ Ігор ГАМАНЮК
(підпис)

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 52 стор., 6 табл., 22 рис., 16 джерел.

Мета роботи – покращення процесу управління матеріальними запасами у кав'ярнях.

Об'єкт дослідження – процес управління матеріальними запасами у кав'ярнях.

Предмет дослідження – програмна система автоматизації обліку матеріалів, що забезпечує фіксацію закупівель, списання, прогнозування потреб та формування звітів.

Короткий зміст роботи: У кваліфікаційній роботі розглянуто проблему організації ефективного обліку матеріалів. Проаналізовано недоліки традиційних способів обліку (ручні записи, Excel-таблиці) та визначено необхідність у впровадженні спеціалізованого програмного забезпечення. На основі аналізу існуючих рішень було розроблено програмну систему CoffeeTrack, яка автоматизує основні процеси: фіксацію закупівель, списання матеріалів, контроль залишків, прогнозування потреб і формування звітності.

Система реалізована з використанням архітектури MVC на платформі ASP.NET Core, із застосуванням бази даних SQL Server та ORM Entity Framework. Реалізовано функціонал додавання, редагування та перегляду матеріалів, а також систему повідомлень при досягненні критичних порогів запасів.

Особливу увагу приділено зручності інтерфейсу, що дозволяє працівникам кав'ярні швидко адаптуватися до нової системи без спеціальної підготовки.

Крім основних можливостей обліку, у програму впроваджено модулі аналітики та прогнозування, які допомагають у прийнятті управлінських рішень. Також реалізовано механізм експорту звітів у зручному форматі для подальшого аналізу або передачі у бухгалтерію. Практична цінність роботи полягає в можливості впровадження розробленої системи у реальних умовах роботи кав'ярні.

КЛЮЧОВІ СЛОВА: ОБЛІК МАТЕРІАЛІВ, АВТОМАТИЗАЦІЯ, КАВ'ЯРНЯ, COFFEETRACK, ASP.NET CORE, SQL SERVER, ENTITY FRAMEWORK, ПРОГНОЗУВАННЯ, ЗАЛИШКИ, СПИСАННЯ.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

БД – база даних, структуроване сховище інформації, що використовується для зберігання даних про матеріали, закупівлі та списання.

GUI – графічний інтерфейс користувача, спосіб взаємодії користувача із системою за допомогою візуальних елементів.

MVC – Model–View–Controller, архітектурна модель програмного забезпечення, що розділяє логіку даних, інтерфейс користувача та керування подіями.

EF Core – Entity Framework Core, об’єктно-реляційний механізм для доступу до бази даних у середовищі .NET.

CRUD – операції створення (Create), читання (Read), оновлення (Update) та видалення (Delete) даних.

SQL – Structured Query Language, мова структурованих запитів для взаємодії з реляційними базами даних.

ASP.NET Core – фреймворк для розробки веб-додатків, що підтримує кросплатформеність та високу продуктивність.

Razor Views – шаблонна система в ASP.NET, яка дозволяє поєднувати HTML і C# для створення динамічного веб-інтерфейсу.

Chart.js – JavaScript-бібліотека для створення інтерактивних графіків і діаграм.

Git – система керування версіями, яка дозволяє відстежувати зміни в коді та координувати роботу над проєктом.

GitHub – платформа для зберігання та спільного використання репозиторіїв, що працює на основі Git.

ККД – коефіцієнт корисної дії, у контексті дипломної роботи використовується умовно для оцінки ефективності використання ресурсів.

ВСТУП

У сучасних умовах стрімкого розвитку сфери громадського харчування та зростання конкуренції, ефективне управління внутрішніми процесами стає критично важливим навіть для невеликих закладів, зокрема кав'ярень. Якщо раніше облік інгредієнтів здійснювався вручну — у зошитах або з використанням простих електронних таблиць — то на сьогодні такий підхід вже не відповідає сучасним вимогам. Забезпечення безперервності роботи кухні та бару, як і досягнення стабільних фінансових результатів, напряму залежить від впровадження надійних засобів обліку ресурсів.

Працівники закладів щоденно взаємодіють з великою кількістю облікових одиниць, кожна з яких потребує контролю. Вчасно не виявлений дефіцит продукту, невчасна закупівля чи відсутність фіксації списання можуть призвести до порушення ритму роботи, збільшення витрат, зниження якості обслуговування та загального рівня задоволеності клієнтів. На практиці проблема контролю запасів є значно складнішою, ніж може здаватися на перший погляд, і потребує системного підходу.

Особистий практичний досвід показав, що чинник людської помилки, зумовлений втомою, навантаженням або недостатнім контролем, часто призводить до втрати важливої інформації щодо залишків або неточностей у підрахунках. У результаті — виникають екстрені ситуації, зокрема необхідність термінового пошуку постачальників, заміни інгредієнтів чи навіть коригування меню в умовах поточного навантаження. Такі обставини продемонстрували актуальність створення автоматизованого інструменту для впорядкування цих процесів.

Аналізуючи практику ведення обліку у малому бізнесі, було встановлено, що багато підприємців досі покладаються на Excel або інші табличні рішення. Проте зі збільшенням кількості позицій такі методи втрачають ефективність через ризики помилок, дублювання даних і обмеженість аналітичного функціоналу. Ручний

облік або паперові носії втрачають актуальність у зв’язку з недостатньою надійністю та складністю в інтеграції з іншими процесами закладу.

З огляду на зазначене, було прийнято рішення про розробку власної програмної системи, яка б забезпечувала простий, інтуїтивно зрозумілий, але водночас функціональний облік матеріалів. Основна мета — створення рішення, що дозволяє працівникам швидко орієнтуватися у поточних залишках, здійснювати закупівлі та списання, прогнозувати потреби та аналізувати витрати. Розробка орієнтована на практичне використання у малих закладах типу кав’ярні, де критичне значення має простота впровадження та щоденна надійність.

У процесі реалізації проекту була створена інформаційна система, яка надає можливість вести базу матеріалів, контролювати запаси, фіксувати рух товарів, аналізувати витрати та формувати прогнози на основі динаміки споживання. Такий підхід дозволяє мінімізувати ризики людських помилок, знизити адміністративне навантаження на персонал і підвищити загальний рівень організації процесів.

Необхідно підкреслити, що запропонована розробка має не лише навчальний, але й прикладний характер. Вона є відповіддю на конкретні виклики, з якими щодня стикаються представники малого бізнесу у сфері громадського харчування. Подальше вдосконалення системи може передбачати розширення функціоналу, інтеграцію з іншими платформами та адаптацію під індивідуальні потреби користувачів. Уже на поточному етапі програмний продукт демонструє високу практичну цінність та перспективність впровадження у реальному бізнес-середовищі.

1 АНАЛІЗ СУЧАСНИХ ПІДХОДІВ ДО ОБЛІКУ МАТЕРІАЛІВ У ЗАКЛАДАХ ТИПУ КАВ'ЯРНІ

1.1 Актуальність теми

Жоден бізнес, пов'язаний із харчуванням, не обходиться без великої кількості щоденних завдань, де важлива точність і порядок у всьому. Особливо це відчувають власники та персонал кав'ярень, ресторанів і пекарень. У таких закладах, де кожен інгредієнт на рахунку, будь-яка помилка може мати серйозні наслідки. Навіть незначна похибка в облікових розрахунках або втрата даних щодо залишків матеріалів може призвести до суттєвих ускладнень у щоденній роботі закладу, перетворюючи звичні операційні процеси на неконтрольовану та проблемну ситуацію.

Типовою є ситуація, коли в період пікового навантаження, за умов повної заповненості зали та великого потоку замовлень, раптово виявляється відсутність базових інгредієнтів — таких як кава чи молоко. Це призводить до призупинення процесу обслуговування, виникнення вимушених затримок, підвищеного стресу персоналу та необхідності термінового пошуку вирішення проблеми. У результаті подібних збоїв зазнає втрат як фінансова складова діяльності закладу, так і його імідж. Запобігти таким ситуаціям дозволяє лише впровадження системного та автоматизованого підходу до обліку ресурсів. Раціональне управління запасами є одним із ключових факторів успішного функціонування закладу харчування [1].

Довгий час власники закладів трималися за "класичні" способи обліку — блокноти, зошити, Excel-таблиці. Це здавалося зручним і навіть економним рішенням. Але з часом накопичується стільки інформації, що розібратися у власних записах стає справжнім випробуванням. Втома, поспіх або простий людський фактор призводять до помилок у підрахунках, дублювання записів або втрати важливої інформації[2]. Намагання здійснювати облік матеріалів виключно за допомогою пам'яті або зберігати інформацію про залишки у свідомості працівників

є неефективним підходом, який втрачає свою доцільність із зростанням обсягів діяльності та номенклатури ресурсів.

Автоматизація обліку матеріалів у цій сфері — це вже не просто тренд або додаткова “фішка”, а реальна необхідність для стабільної роботи.[3] Сучасні дослідження підтверджують, що комплексний підхід до автоматизації відкриває значні можливості для ефективної організації обліку витрат виробництва та готової продукції. Добре продумана система дозволяє в режимі реального часу бачити, що саме є на складі, скільки було витрачено за день або тиждень, і які позиції вже скоро потрібно поповнювати. Це знімає зайве навантаження з персоналу, дає можливість зосередитись на обслуговуванні клієнтів, а не на нескінченних перевірках і перерахунках.

Варто також наголосити на важливості прогнозування. Впровадження автоматизованої системи дозволяє не лише оперативно реагувати на ситуацію моментально, а й робити висновки на майбутнє. Наприклад, аналізуючи дані про споживання матеріалів, можна спрогнозувати, коли і в яких обсягах потрібно робити нові замовлення, або які продукти варто замінити на більш популярні серед відвідувачів закладу. Це — реальний інструмент для оптимізації витрат і підвищення прибутковості.[4]

Розробка спеціалізованої програми для обліку матеріалів має очевидну практичну користь. Вона вирішує одразу декілька проблем: від мінімізації людського фактору до підвищення прозорості у веденні справ. У результаті впровадження автоматизованого обліку заклад функціонує більш стабільно, демонструє готовність до непередбачуваних обставин і виявляє здатність оперативно адаптуватися до змін зовнішнього та внутрішнього середовища. Саме тому тема автоматизації обліку матеріалів не втрачає своєї актуальності і, навпаки, стає все більш затребуваною в умовах жорсткої конкуренції та постійного розвитку ринку.

Для кав'ярень, де асортимент часто включає швидкопсувні продукти (молоко, десерти, фрукти), а також товари з різними термінами придатності та умовами зберігання (кавові зерна, сиропи, одноразовий посуд), завдання управління

запасами стає ще більш складним, що підтверджується дослідженнями оптимізаційних моделей для ресторанного бізнесу[5].

1.2 Цілі та завдання розробки

Коли берешся за створення будь-якого проекту, особливо якщо він має вирішувати реальні робочі проблеми, перше, що потрібно зробити — чітко визначити, для чого цей продукт узагалі потрібен. Не для “галочки”, не просто щоб був, а щоб дійсно допомагав людям у щоденній роботі. Тому ще до початку розробки програми для обліку матеріалів виникло питання: а що саме вона має спрощувати? Як має змінитися робочий день персоналу з її появою?

Головною метою даного проекту було створення програмного забезпечення, яке поєднувало б функціональність, простоту у використанні та інтуїтивно зрозумілий інтерфейс. Важливим критерієм стала доступність інструменту для користувачів з різним рівнем технічної підготовки — включаючи тих співробітників, які до цього користувалися виключно паперовим документообігом. Основне завдання полягало у розробці системи, що органічно інтегрується у робочі процеси кав’ярні, не викликаючи труднощів у процесі навчання персоналу та не потребуючи додаткової технічної підтримки.

Ключовий акцент було зроблено на **максимальній зручності та прозорості обліку витратних матеріалів**, щоб забезпечити ефективне управління складськими залишками без перевантаження користувача надмірною кількістю функцій або складною навігацією.

Водночас, лише концепція «зручності» не є достатньою для створення повноцінного прикладного рішення. Для реалізації програмного забезпечення, що дійсно відповідає практичним потребам бізнесу, було сформульовано низку конкретних задач, які стали основою функціонального проектування:

- розробка надійної та структурованої бази даних. База повинна зберігати повний перелік витратних матеріалів, їх кількість, інформацію про закупівлі та

історію списань. Особлива увага приділяється стабільності даних, запобіганню втрат інформації внаслідок збоїв або оновлень системи;

- створення зручного інтерфейсу користувача. Система має дозволяти здійснювати базові дії без необхідності вивчення документації — всі основні функції мають бути логічно згруповані та розташовані у доступних елементах керування;

- автоматизація фіксації закупівель. У разі надходження нових партій товарів достатньо декількох дій у системі для актуалізації залишків, без потреби у ручному оновленні кожної позиції;

- реалізація механізму списання матеріалів із зазначенням причини та дати. Така функція дозволяє підтримувати облік у точному стані та швидко аналізувати історію руху матеріалів;

- впровадження аналітичного модуля. Його призначення — надання візуалізованих звітів щодо темпів використання матеріалів, виявлення пікових періодів споживання, виявлення товарів, які не використовуються або накопичуються без потреби;

- засоби прогнозування потреб. На основі статистики витрат система повинна формувати рекомендації щодо майбутніх закупівель, що дозволяє уникати ситуацій дефіциту;

- механізм автоматичних рекомендацій щодо закупівель. Система має повідомляти користувача про критично низькі залишки матеріалів та ініціювати формування замовлень на поповнення;

- формування звітів і можливість їх експорту. Звіти повинні бути доступними у форматі, придатному для друку або передачі керівництву та бухгалтерії.

Усі зазначені функціональні компоненти об’єднуються спільною ідеєю — забезпечити максимальну ефективність, прозорість та зручність управління матеріальними ресурсами кав’ярні. Програмне забезпечення має не лише автоматизувати процеси, а й підвищити загальний рівень організованості в роботі персоналу, сприяти зниженню операційних витрат та покращити якість прийняття

управлінських рішень. Саме на досягнення цих цілей була орієнтована розробка системи *CoffeeTrack*.

1.3 Об’єкт, предмет, мета та постановка завдань кваліфікаційної роботи

Перш ніж перейти до детального опису того, як реалізовувалася система, важливо зрозуміти, що саме стало об'єктом і предметом цього дослідження. Ці поняття часто плутають, але насправді вони мають чіткі межі та відіграють важливу роль у будь-якому науковому чи прикладному проекті.

Отже, об'єктом дослідження в даному випадку є процес управління матеріальними запасами закладів громадського харчування. Це доволі широка сфера, яка охоплює різні аспекти: від надходження продуктів на склад до їх використання в щоденній роботі. У реальному житті цей процес виглядає нескладно, але насправді в ньому чимало нюансів. Необхідно вчасно робити закупівлі, уникати простоїв через брак необхідних матеріалів і, водночас, не допускати перевищення обсягу запасів, щоб уникнути псування продуктів.

Що ж до предмета дослідження — ним стала система автоматизації обліку матеріалів у кав'ярні, яка дозволяє не лише вести запис, а й контролювати рух матеріалів, планувати закупівлі та оптимізувати витрати. Саме ця система є тією частиною об'єкта, на яку було зроблено акцент під час розробки.

Простіше кажучи, об'єкт — це уся сфера управління матеріалами в кав'ярнях, а предмет — конкретний програмний продукт, що був створений для полегшення цієї роботи.

Під час реалізації проекту особлива увага приділялася зручності використання системи. Було важливо зробити її такою, щоб будь-хто з персоналу закладу міг швидко і без зайвих зусиль внести дані про закупівлю чи списання матеріалів. Також предмет дослідження охоплював питання забезпечення надійності зберігання даних, їхньої актуальності та доступності в будь-який момент.

Таким чином, об'єктом є організація обліку матеріалів як процес, а предметом — конкретна програмна реалізація цієї ідеї у вигляді розробленої інформаційної системи.

1.3.1 Опис предметної області

Для обґрунтування розробки програмного забезпечення *CoffeeTrack* необхідно детально проаналізувати предметну область, у межах якої функціонує система. У даному випадку йдеться про діяльність кав’ярні — закладу громадського харчування, в якому щодня використовуються різноманітні витратні матеріали та інгредієнти.

Сфера експлуатації охоплює значно ширший перелік ресурсів, ніж лише базові компоненти, як-от кава чи молоко. До витратних матеріалів належать також одноразові стаканчики, серветки, цукор, сиропи, ароматичні добавки та інші позиції, що є критично важливими для забезпечення належного рівня обслуговування клієнтів. Кожна з таких позицій має власний термін зберігання, динаміку споживання та потребує постійного контролю.

Ефективне управління запасами передбачає облік кількісних залишків, своєчасне замовлення нових партій та контроль строків придатності продукції. Крім того, необхідно мати можливість відстеження реального використання матеріалів, що дозволяє виявляти найпопулярніші товари, уникати дефіциту та мінімізувати надлишкові залишки на складі.

До впровадження програмної системи ці процеси реалізовувалися вручну: облік вівся у блокнотах, Excel-таблицях або в усній формі, а контроль залишків здійснювався "на око". Такий підхід був супроводжений численними проблемами — людськими помилками, розрізненістю або втратою даних, а також складністю в отриманні цілісної картини щодо стану складу.

Це часто призводило до практичних труднощів: наприклад, персонал міг не знати про критичні залишки певних матеріалів, а відповідальні особи могли пропустити момент замовлення необхідних ресурсів. Такі ситуації, хоч і здаються

дрібними, у реальних умовах можуть спричинити збої в роботі кав’ярні та фінансові втрати.

У зв’язку з цим виникла потреба у створенні спеціалізованої інформаційної системи, яка забезпечує автоматизацію обліку матеріалів, контроль запасів у реальному часі та підтримку щоденної операційної діяльності. Основним функціональним завданням системи стало надання можливості легко та швидко вносити інформацію про надходження і списання матеріалів, формувати звіти та здійснювати аналіз.

Таким чином, предметна область, яку охоплює розроблене програмне забезпечення, — це організація й оптимізація обліку матеріальних ресурсів у закладах громадського харчування, де критичними є точність, оперативність та надійність у прийнятті рішень. Саме на ці аспекти була орієнтована розробка програмного продукту.

1.4 Аналіз наявних рішень та обґрунтування необхідності розроблення

У процесі ініціалізації розробки інформаційної системи для обліку витратних матеріалів у кав'ярнях одним із першочергових етапів стало проведення ретельного аналізу вже існуючих на ринку програмних рішень. Такий підхід відповідає загальноприйнятій практиці у сфері розробки програмного забезпечення, де доцільність створення нової системи визначається з урахуванням конкурентного середовища, існуючих аналогів, їх функціональних характеристик, рівня адаптованості, масштабованості, а також відповідності конкретним запитам цільового користувача.

Метою даного аналізу було не лише ознайомлення з уже наявними інструментами, а й формулювання висновків щодо доцільності розробки власного програмного продукту з урахуванням виявлених обмежень, недоліків та потенційних точок вдосконалення.

До найпоширеніших інструментів, які використовуються у закладах громадського харчування (особливо у малому бізнесі), належать табличні

редактори, зокрема Microsoft Excel, Google Sheets, LibreOffice Calc тощо[6]. Їхня популярність обумовлена доступністю, відсутністю значних витрат на впровадження, мінімальними вимогами до технічних ресурсів, а також відносною простотою у використанні.

Однак, при всій їх очевидній зручності для простого підрахунку або зберігання даних, ці засоби не здатні виконувати низку ключових завдань, що є критичними для ефективного обліку матеріалів у динамічному середовищі кав'ярні:

- відсутність автоматизації: користувач самостійно повинен оновлювати значення залишків, фіксувати закупівлі та списання вручну, що підвищує ризик помилок;

- відсутність журналу змін: практично неможливо відстежити, хто і коли вносив зміни, що критично для контролю обліку;

- низький рівень візуалізації: побудова графіків і діаграм потребує значних зусиль і технічної підготовки;

- немає попереджень про критичні залишки: система не повідомляє користувача, коли запаси наближаються до мінімуму;

- немає модулів прогнозування чи аналітики: Excel не здатен надати релевантні висновки на основі історичних даних без складного програмування формул або макросів, а застосування сучасної аналітики відіграє важливу роль у прийнятті обґрунтованих управлінських рішень.

Таким чином, навіть при високій гнучкості й загальній зручності, табличні редактори залишаються придатними лише для базового ручного обліку і є неефективними в умовах потреб сучасної автоматизованої моделі управління[7].

На ринку програмного забезпечення для закладів HoReCa також присутні спеціалізовані рішення: Poster POS, R-Keeper, SkyService POS, Keepin CRM, тощо.

Більшість з них є комплексними системами управління закладом — вони охоплюють бухгалтерський облік, роботу персоналу, логістику, управління замовленнями, клієнтський сервіс тощо. Проте, при спробі інтегрувати такі

продукти для завдань обліку витратних матеріалів у невеликому закладі виникає низка труднощів:

- надлишкова складність: система містить безліч модулів, які не мають прямого відношення до управління запасами. Це ускладнює навчання та знижує продуктивність працівників;

- значні фінансові витрати: більшість таких продуктів є платними, з щомісячною або щорічною підпискою, яка не є виправданою для закладів із невеликим обігом;

- низька гнучкість під конкретні задачі: адаптація функціоналу під індивідуальні потреби малого бізнесу часто потребує залучення програмістів або технічних консультантів;

- проблеми з локалізацією: деякі інтерфейси мають обмежену українську локалізацію або взагалі не підтримують мову, що ускладнює адаптацію.

На основі аналізу універсальних інструментів та професійних CRM/ERP-систем можна дійти обґрунтованого висновку: жоден із розглянутих варіантів не забезпечує комплексного та при цьому простого вирішення задач, що постають перед власником або адміністратором кав’ярні[8].

Саме тому постала необхідність у створенні спеціалізованої інформаційної системи, орієнтованої на:

- максимальну простоту інтерфейсу, який не потребує навчання;

- швидке внесення даних про закупівлі та списання;

- автоматичне попередження про критичні залишки;

- наочну аналітику з візуалізацією у вигляді графіків;

- можливість експорту звітів для бухгалтерії;

- прогнозування майбутніх потреб на основі історичних даних.

Розробка власного продукту дозволяє не лише уникнути зайвих витрат, а й створити систему, яка відповідає реальним потребам конкретного типу бізнесу — невеликих кав'ярень з обмеженим штатом і динамічним циклом роботи[9].

Таблиця 1.1

Порівняльна характеристика програм для обліку матеріалів.

Показник	**CoffeeTrack**	**CRM для HoReCa**	**Excel/ Google Sheets**	**SkyService POS**
Доступність	Висока	Середня (потрібні навички)	Висока	Середня
Прогнозування потреб	Присутнє(на основі аналітики витрат)	Відсутнє або реалізовано частково	Відсутнє	Середня (частково реалізоване)
Гнучкість налаштувань	Висока	Висока(але потребує налаштувань)	Висока	Низька
Орієнтованість на кав'ярні	Висока (система спеціалізована)	Середня (універсальні рішення для HoReCa)	Відсутня	Висока (але обмежена в обліку матеріалів)
Вартість	Низька (безкоштовна або умовно безкоштовна)	Висока (підписка, ліцензія)	Низька	Середня (є базовий безкоштовний тариф)

Продовження таблиці 1.1

Порівняльна характеристика програм для обліку матеріалів.

Показник	**CoffeeTrack**	**CRM для HoReCa**	**Excel/ Google Sheets**	**SkyService POS**
Облік залишків	Присутній, з аналітикою і звітами	Частково (залежить від тарифу)	Ручний (немає автоматизації)	Присутній, але без аналітичного модуля

Як свідчить проведений аналіз, універсальні інструменти для ведення обліку, зокрема такі як **Microsoft Excel** або **Google Sheets**, не здатні забезпечити належний рівень автоматизації процесів управління витратними матеріалами у закладах громадського харчування. Їх функціональні можливості обмежені: відсутність механізмів автоматичного попередження про критичний рівень залишків, неможливість оперативного аналізу споживання ресурсів, а також відсутність вбудованих рекомендаційних систем істотно знижують ефективність їх використання в динамічному середовищі щоденної операційної діяльності. Вирішення цього завдання спирається на принципи аналізу даних та інтелектуального аналізу (data mining) для виявлення закономірностей[10].

Водночас, професійні програмні комплекси типу CRM-систем для HoReCa, попри широкий функціонал, часто виявляються надмірними для умов експлуатації в невеликих комерційних одиницях, зокрема кав'ярнях. Їх використання, як правило, передбачає не лише значні фінансові витрати, пов'язані з ліцензуванням та супроводом, а й потребує тривалого процесу навчання персоналу[11]. Це створює додаткове навантаження на бізнес та може негативно впливати на адаптацію співробітників до нових цифрових рішень.

З огляду на зазначене, було прийнято обґрунтоване рішення щодо розробки власного програмного забезпечення, яке б поєднувало доступність та інтуїтивну зрозумілість табличних редакторів з базовими можливостями автоматизації, притаманними промисловим системам. Особливістю такого підходу є орієнтація на реальні потреби кінцевого користувача — працівників невеликих закладів, де пріоритетними є простота, швидкість і мінімальне навчальне навантаження.

Результати аналізу існуючих рішень засвідчили, що жоден із доступних варіантів не забезпечує повної відповідності сформульованим вимогам. Відтак розробка власної інформаційної системи не лише виявилась доцільною, а й необхідною умовою для досягнення бажаного рівня ефективності, зручності та адаптованості до умов конкретного закладу.

2 ТЕХНІЧНЕ ПРОЕКТУВАННЯ СИСТЕМИ ОБЛІКУ МАТЕРІАЛІВ

2.1 Архітектура програмної системи

У процесі розробки прикладного програмного забезпечення, незалежно від галузі застосування або масштабу проекту, надзвичайно важливим є не лише створення робочого функціоналу в межах початкового технічного завдання, а й формування такої архітектури системи, яка забезпечить її гнучкість, підтримуваність, масштабованість і стійкість до змін в умовах реального використання. В сучасних умовах розвитку IT-індустрії питання життєвого циклу програмного продукту набуває особливої ваги, оскільки більшість систем потребують регулярного оновлення, адаптації до нових потреб користувачів, інтеграції з іншими сервісами та подальшого розвитку.

Як свідчить практичний досвід розробників, як індивідуальний, так і у командній роботі, високий рівень структурної організації коду є одним з ключових чинників, що визначає успішність довгострокової підтримки програмного продукту. Система з чіткою архітектурною побудовою:

- спрощує адаптацію нових розробників до коду;
- дозволяє ізолювати зміни в окремих компонентах без негативного впливу на всю систему;
- підвищує якість тестування та спрощує виявлення помилок;
- відкриває можливості для повторного використання окремих модулів у суміжних проектах.

Нерідко можна почути думку, що архітектурне планування доцільне лише у масштабних програмних системах. Однак навіть відносно невеликий прикладний проект, такий як інформаційна система обліку витратних матеріалів у кав’ярні, насправді складається з багатьох взаємопов’язаних компонентів: введення даних, збереження в базі, відображення у вигляді таблиць і графіків, обчислення аналітики та прогнозів, формування звітності, повідомлення користувача, також важливо

забезпечити безперервність контролю над залишками. Якщо реалізовувати подібну систему без логічного поділу на шари — програма швидко перетворюється на «моноліт», в якому будь-яка зміна може призвести до помилок в інших частинах.

У таких умовах виникає об’єктивна потреба у багаторівневій (багатошаровій) архітектурі, яка реалізує принципи розмежування відповідальностей (separation of concerns). Суть цього підходу полягає у тому, що кожен функціональний блок системи реалізується окремо: інтерфейс користувача (UI), бізнес-логіка, доступ до даних тощо. Це дозволяє змінювати або вдосконалювати один рівень без необхідності втручатися в інші.

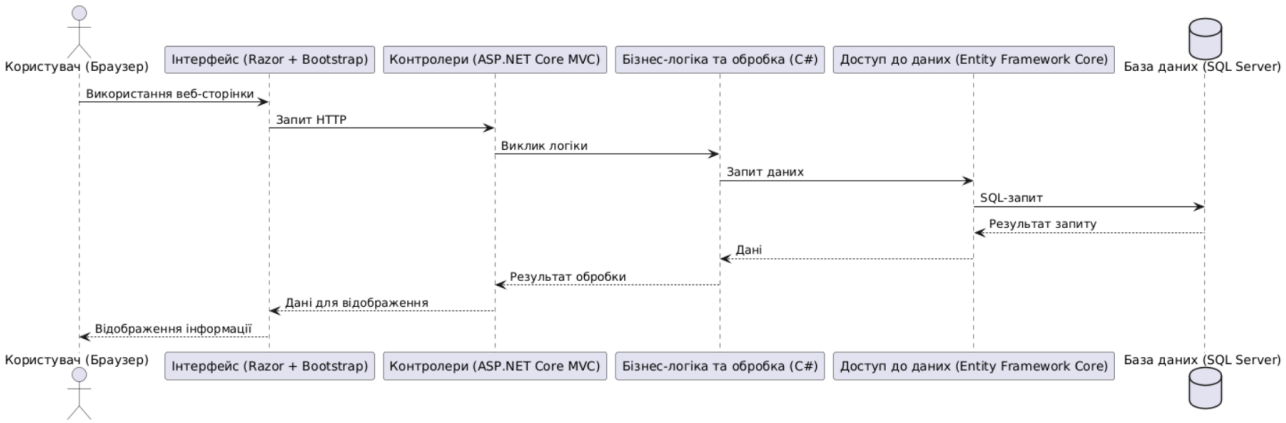


Рис. 2.1 Діаграма взаємодії шарів логічної архітектури при використанні веб-сторінки

2.2 Програма побудована з трьох основних рівнів:

Рівень представлення (Presentation Layer)

Рівень представлення є інтерфейсом взаємодії користувача із системою. Саме тут реалізовано веб-інтерфейс, що забезпечує доступ до функціональних можливостей програми через веб браузер. У розробці використовувалася технологія **ASP.NET Core MVC** у поєднанні з шаблонами **Razor Pages**, що дозволило створити структурований і гнучкий користувацький інтерфейс[12].

На цьому рівні реалізовано основні сторінки програми — зокрема, «Матеріали», «Закупівлі», «Списання», «Звіти» тощо. Користувач має можливість додавати нові записи, наприклад, щодо надходження партії кави, перевіряти

залишки матеріалів, здійснювати списання з відповідним зазначенням причин, а також переглядати історію операцій.

Інтерфейс розроблено з урахуванням принципів мінімізації когнітивного навантаження. Особлива увага приділялася простоті й доступності — система орієнтована на персонал, який може не мати попереднього досвіду роботи зі складними програмами. Завдяки цьому користувачі можуть інтуїтивно зрозуміти логіку навігації та почати роботу з системою без попереднього навчання або залучення технічних спеціалістів.

Загалом, рівень представлення виконує роль «обличчя» системи, забезпечуючи зручний інтерфейс для виконання всіх основних функцій, пов’язаних з обліком і керуванням матеріалами в межах повсякденної діяльності кав’ярні.

Рівень обробки логіки (Business Logic Layer)

Рівень обробки логіки є центральною частиною архітектури програмного забезпечення та виконує ключову роль у функціонуванні всієї системи. Саме тут реалізуються основні механізми обробки даних, що надходять від користувача, здійснюються розрахунки залишків, генеруються рекомендації щодо закупівель, а також оновлюється інформація в базі даних.

Цей рівень виконує функцію зв’язувальної ланки між інтерфейсом користувача та рівнем доступу до даних, реалізуючи бізнес-правила, які визначають поведінку системи відповідно до заданих умов. У рамках даного проєкту бізнес-логіка реалізована за допомогою **контролерів ASP.NET Core** у поєднанні з **моделями**, що забезпечує чітке розмежування представлення даних та їх обробки.

Такий підхід дозволяє легко підтримувати та масштабувати систему, не порушуючи структуру інших рівнів. Наприклад, додавання нового типу матеріалу або впровадження аналітичного модуля може бути здійснене без суттєвих змін до інтерфейсу або логіки доступу до даних. Це забезпечує **гнучкість**, **розширюваність** і **надійність** програмного рішення, що особливо важливо для застосунків, які передбачають тривале використання та подальший розвиток функціональності.

Рівень зберігання даних (Data Access Layer)

Рівень зберігання даних забезпечує надійне збереження, доступ і управління усією інформацією, що використовується в системі. Саме на цьому рівні реалізується взаємодія із базою даних, у якій зберігається повний обсяг записів про матеріали, закупівлі, списання, аналітичні дані та інші сутності.

Для реалізації доступу до даних було використано **Entity Framework Core** — сучасну ORM-технологію (Object-Relational Mapping), яка забезпечує зручне маніпулювання базою даних засобами мови програмування C#, без потреби у створенні складних SQL-запитів[13]. Це суттєво спрощує реалізацію логіки взаємодії з даними, пришвидшує процес розробки та зменшує кількість помилок у коді[14].

Всі запити до бази, включно з пошуком, додаванням, редагуванням або видаленням записів, обробляються саме на цьому рівні, після чого дані передаються в обробленому вигляді на рівень бізнес-логіки або інтерфейсу користувача. Такий підхід дозволяє ізолювати операції з даними від решти системи, що сприяє підвищенню стабільності, тестованості та підтримуваності програмного продукту.

2.3 Опис логіки роботи програмної системи

Розроблена інформаційна система базується на чітко структурованій багаторівневій архітектурі, яка дозволяє організувати внутрішню логіку програми відповідно до принципів інженерії програмного забезпечення. Її ключова особливість полягає у розмежуванні відповідальності між трьома основними рівнями: Presentation Layer (рівень представлення), Business Logic Layer (рівень бізнес-логіки) та Data Access Layer (рівень доступу до даних). Така побудова забезпечує модульність, прозорість і розширюваність системи.

Щоб наочно продемонструвати, як відбувається взаємодія між цими рівнями у реальному сценарії використання системи, розглянемо приклад виконання однієї з ключових функцій — додавання нової закупівлі.

Переваги багаторівневої моделі у контексті реального використання:

- гнучкість розвитку системи;

Завдяки тому, що кожен компонент (UI, логіка, база) існує ізольовано, оновлення або розширення функціоналу не потребує зміни всієї програми. Наприклад, якщо виникає потреба додати новий модуль — “Аналіз витрат за тиждень” або “Управління постачальниками”, його можна реалізувати у вигляді окремого контролера з відповідними моделями. При цьому інші частини системи залишаються стабільними. Це мінімізує ризики, скорочує час розробки і тестування, а також забезпечує безперервність роботи вже впроваджених модулів.

- надійність і відстеження помилок;

Кожен рівень має чітко визначену зону відповідальності. Помилки, які виникають на рівні введення (наприклад, відсутність обов’язкового поля), не впливають на збереження даних. Навпаки — система перехоплює їх ще до того, як запит потрапить до бізнес-логіки чи бази. Якщо ж виникає збій у логіці (наприклад, некоректне оновлення залишків), його можна швидко ізолювати та усунути без загрози загальній стабільності.

- зручність для кінцевого користувача;

Незважаючи на складність внутрішньої структури, інтерфейс залишається простим, інтуїтивним і швидким у використанні. Усі основні дії — у межах 1–2 кліків. Не потрібно вивчати інструкції або витрачати час на пошук потрібної функції. Завдяки багаторівневому підходу всі дії користувача “маскують” складну логіку за простим і зрозумілим інтерфейсом.

- придатність до підтримки та масштабування;

У майбутньому система може бути легко доповнена — новими звітами, аналітичними модулями, API-інтеграціями, мобільною версією тощо. Такий підхід забезпечує готовність проекту до розширення без ризику “зламати” вже наявні функції.

2.4 Проектування та структура бази даних

База даних є серцем системи CoffeeTrack, оскільки саме вона зберігає всю ключову інформацію про матеріали, закупівлі та списання. Вона забезпечує узгодженість і доступність даних для всіх функцій додатка.

Щоб максимально ефективно підтримувати всі необхідні бізнес-процеси, база даних побудована відповідно до реляційної моделі. Основними таблицями виступають **Materials**, **Purchases**, **Consumptions** та службова таблиця **EFMigrationsHistory**.

Таблиця 2.1

Materials — Матеріали

Поле	**Тип даних**	**Опис**
Id	int	Унікальний ідентифікатор матеріалу
Name	nvarchar	Назва матеріалу
Unit	nvarchar	Одиниця виміру (кг, л, мм тощо)
Quantity	int	Поточна кількість на складі
CriticalThreshold	int	

Таблиця містить всю інформацію про матеріали, що є в наявності, а також критичні пороги для сповіщень.

Таблиця 2.2

Purchases — Закупівлі

Поле	**Тип даних**	**Опис**
Id	int	Унікальний ідентифікатор матеріалу
MaterialId	int	Посилання на матеріал (зв'язок з Materials)
PurchaseDate	nvarchar	Дата закупівлі
QuantityPurchased	int	Кількість придбаного матеріалу

Закупівлі дозволяють фіксувати надходження нових партій матеріалів на склад.

Таблиця 2.3

Consumptions — Списання

Поле	**Тип даних**	**Опис**
Id	int	Унікальний ідентифікатор матеріалу
MaterialId	int	Посилання на матеріал (зв'язок з Materials)
DateUsed	datetime	Дата списання

Продовження таблиці 2.3

Consumptions — Списання

Поле	**Тип даних**	**Опис**
QuantityUsed	int	Кількість списаного матеріалу

Тут зберігаються записи про використання матеріалів у процесі виробництва або обслуговування.

Таблиця 2.4

EFMigrationsHistory — Історія міграцій

Поле	**Тип даних**	**Опис**
MigrationId	nvarchar	Ідентифікатор міграції
ProductVersion	nvarchar	Версія Entity Framework

Службова таблиця для автоматичного ведення історії змін бази даних. Використовується лише розробниками.

2.5 Діаграма зв'язків бази даних

На діаграмі зображено взаємозв’язки між таблицями. Таблиця **Materials** виступає головною сутністю, з якою пов’язані **Purchases** та **Consumptions**. Це дозволяє швидко і зручно отримувати інформацію про запаси, закупівлі та списання. Дані про списання і закупівлі не дублюються, а пов’язуються через зовнішні ключі, що гарантує цілісність інформації та спрощує її обробку.

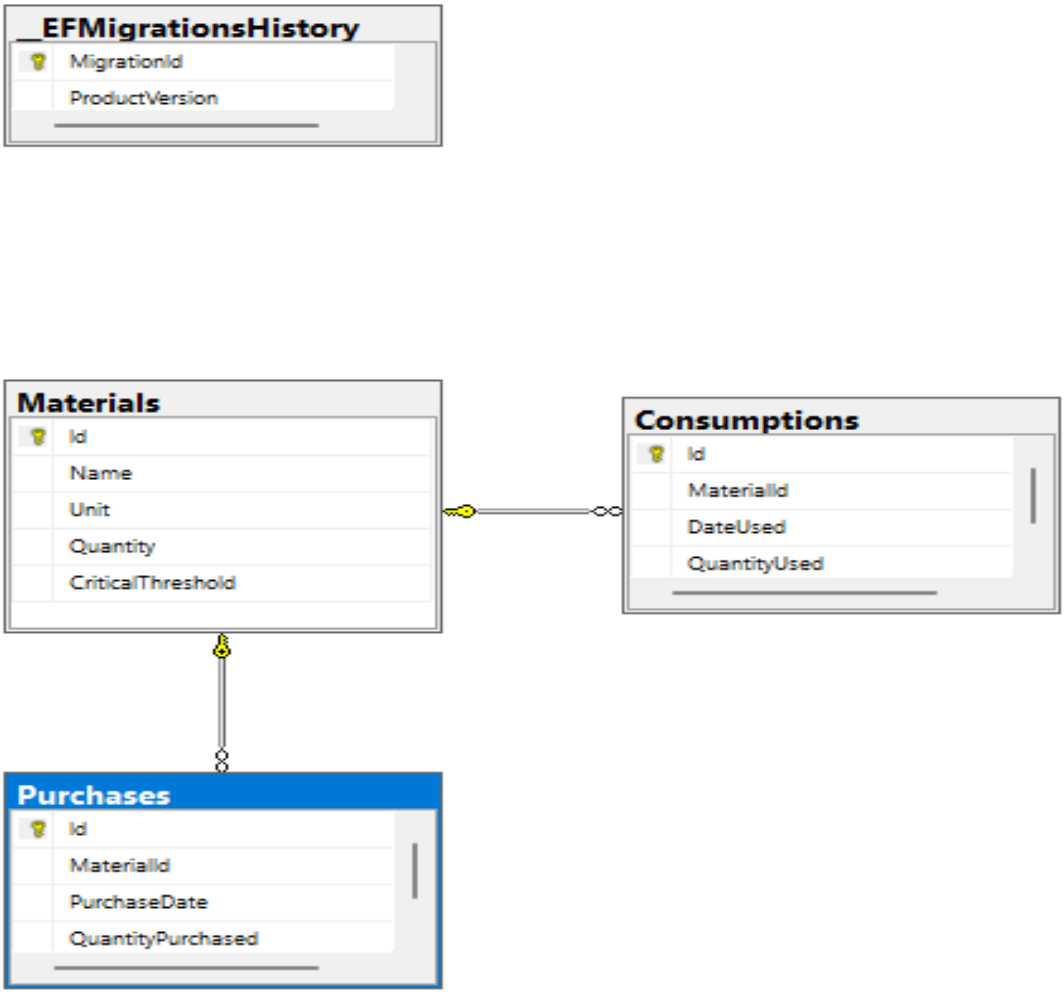


Рис 2.1 Структура бази даних системи CoffeeTrack

Побудова надійної й ефективної бази даних є фундаментом будь-якої інформаційної системи. Саме база даних відповідає за збереження, структурування, логічну цілісність і подальше використання накопичених даних. У контексті системи обліку витратних матеріалів для кав'ярні особливе значення має чітке проектування зв'язків між основними сутностями, що формують бізнес-логіку роботи закладу.

На розробленій **діаграмі структури бази даних** (див. Рис. 2.1) візуально зображено ключові таблиці системи та взаємозв'язки між ними. Архітектура спроектована відповідно до принципів **реляційної моделі даних**, що забезпечує гнучкість, масштабованість та логічну цілісність.

2.5.1 Центральна роль таблиці Materials

У центрі моделі знаходиться таблиця **Materials**, яка виконує роль основної (референтної) сутності. Саме в цій таблиці зберігається інформація про всі витратні матеріали, які використовуються у кав’ярні. Для кожного матеріалу фіксується:

- назва;
- одиниця виміру;
- поточна кількість на складі;
- критичний поріг, нижче якого необхідно здійснити закупівлю.

Ця таблиця є основою для формування всіх облікових та аналітичних процесів у системі. Вона виконує функцію своєрідного "центру тяжіння" для інших сутностей бази.

2.5.2 Зв’язок з таблицею Purchases

Таблиця **Purchases** відповідає за облік усіх закупівель матеріалів. Кожен запис у цій таблиці містить:

- ідентифікатор матеріалу (MaterialId) — зовнішній ключ, який посилається на таблицю Materials;
- дату закупівлі;
- кількість закупленого матеріалу.

Завдяки використанню **зовнішнього ключа**, забезпечується референційна цілісність: не можна додати запис про закупівлю без попереднього існування відповідного матеріалу. Таким чином, система завжди оперує тільки валідними сутностями, що виключає можливість помилок або дублювання даних.

2.5.3 Зв’язок з таблицею Consumptions

Аналогічним чином реалізовано таблицю **Consumptions**, яка фіксує факти списання матеріалів. У кожному записі вказується:

- матеріал, що був використаний;
- дата списання;
- кількість списаного ресурсу.

І тут також використовується зовнішній ключ (MaterialId), який забезпечує цілісність записів і виключає випадки “висячих” даних (тобто посилання на неіснуючі об’єкти).

2.5.4 Переваги такої структури

Запропонована модель бази даних є результатом аналізу функціональних вимог системи та врахування реальних бізнес-процесів, що відбуваються в кав’ярні. Вона дозволяє досягнути таких переваг:

- ефективне управління матеріальними залишками;

Завдяки зв’язку між матеріалами та закупівлями/списаннями можна у будь-який момент визначити актуальні залишки, а також виявити найчастіше використовувані позиції.

- прогнозування потреб у закупівлях;

Історія змін кількості матеріалів дозволяє зафіксувати закономірності — наприклад, зростання споживання у святкові дні або вихідні. Ці дані використовуються у модулі прогнозування для підвищення ефективності планування.

- формування аналітичної звітності;

Чітка структура зв’язків між таблицями дозволяє безперешкодно формувати звіти за різними періодами, фільтрувати дані за матеріалами, аналізувати динаміку витрат. Це реалізовано за допомогою запитів до об’єднаних таблиць через Entity Framework Core.

- ведення повної історії дій;

Кожне списання або закупівля зберігається окремим записом, що дозволяє сформувати повну хронологію подій. Це важливо для звітності, внутрішніх перевірок, а також для обґрунтування управлінських рішень.

- масштабованість;

Модель спроектована з урахуванням можливості подальшого розширення. За потреби до неї можна легко додати нові таблиці — наприклад, “Постачальники”,

“Користувачі”, “Категорії матеріалів”, не порушуючи цілісності вже існуючих даних.

2.6 Технології та інструментів розробки

У процесі проектування та реалізації програмного забезпечення CoffeeTrack, ключовою складовою технічного успіху виступає правильний вибір технологій, інструментів і платформ. Саме від цього етапу значною мірою залежить не лише працездатність готової системи, але й її подальша доля — здатність до еволюції, масштабування, підтримки та адаптації до нових вимог, що можуть з’являтися з часом унаслідок змін бізнес-середовища.

З урахуванням специфіки об’єкта автоматизації — а саме, закладів типу кав’ярень, які працюють у сфері малого бізнесу — постала потреба обрати рішення, що будуть одночасно простими у впровадженні, стабільними у функціонуванні, ефективними з точки зору продуктивності та достатньо гнучкими для можливого розширення.

Варто зазначити, що сучасний ринок IT-технологій надзвичайно насичений і динамічний. Програмісту, що стоїть перед вибором технологічного стеку, доводиться аналізувати десятки (а іноді й сотні) фреймворків, бібліотек, мов програмування, інструментів управління даними, засобів візуалізації та середовищ розробки. При цьому кожен інструмент має власний спектр можливостей, сумісностей, переваг і недоліків, що ускладнює вибір і вимагає серйозного аналітичного підходу.

У випадку CoffeeTrack технологічний вибір не був здійснений з позиції “слідування моді” або орієнтації виключно на популярність того чи іншого рішення. Навпаки, пріоритетом стали практичні міркування: стабільність, документація, підтримка, легкість інтеграції, а також сумісність між компонентами.

Особливої уваги було приділено питанню довгострокової підтримки. Тобто, обрані рішення мали залишатися актуальними та підтримуваними не лише на

момент реалізації проекту, а й у перспективі кількох років — без ризику, що технологія втратить підтримку або стане застарілою. Крім того, розглядалася можливість масштабування проекту — наприклад, адаптація до декількох точок продажу, розширення аналітичного модуля, інтеграція з бухгалтерськими системами або зовнішніми API-сервісами.

Окремо варто наголосити на тому, що одним із ключових факторів була доступність рішень для кінцевих користувачів, які здебільшого не мають глибокої технічної підготовки. Це передбачає не лише простий та інтуїтивний інтерфейс, але й стабільну роботу навіть на середньостатистичних пристроях, які використовуються у кав'ярнях. У зв'язку з цим обирались лише ті технології, які дозволяють створити надійне та водночас легке у використанні програмне забезпечення, що не потребує глибоких знань від персоналу.

Ще одним важливим критерієм стало дотримання принципів багаторівневої архітектури (зокрема, MVC), що забезпечує логічну структуризацію програмного коду та дозволяє незалежно розвивати окремі модулі програми (інтерфейс, логіку, базу даних). Вибрані інструменти повинні були органічно вписуватись у цю архітектуру та не створювати конфліктів на етапі реалізації або при розгортанні системи.

У результаті проведеного аналізу, було сформовано технологічний стек, до якого увійшли перевірені та добре документовані рішення: ASP.NET Core MVC, Entity Framework Core, SQL Server, Bootstrap 5, Razor Views, Chart.js, Git та GitHub для контролю версій, а також середовище розробки Visual Studio. Кожен із цих інструментів був обраний не випадково — він відповідав специфічному завданню, забезпечував оптимальну ефективність і взаємодіяв з іншими компонентами системи.

У подальших підрозділах розглянуто кожен із обраних інструментів окремо, з поясненням причин його вибору, прикладами реалізації та оцінкою ефективності в рамках виконаного проекту.

2.6.1 ASP.NET Core MVC

Фреймворк ASP.NET Core MVC був обраний як основа для реалізації системи CoffeeTrack, оскільки він нативно підтримує архітектурний патерн Model-View-Controller (MVC), описаний в розділі 2.2. Використання ASP.NET Core MVC дозволило структурувати додаток, розділивши відповідальність між моделями даних, їх представленням (Razor Views) та контролерами, що обробляють логіку запитів. Це сприяло модульній розробці та полегшило подальшу підтримку системи.

2.6.2 Entity Framework Core (EF Core)

З метою забезпечення зручного, безпечного та продуктивного доступу до бази даних у межах розробки програмного забезпечення CoffeeTrack було обрано технологію Entity Framework Core (EF Core) — сучасну об’єктно-реляційну систему відображення (ORM), яка забезпечує інтеграцію з базою даних на рівні C#-коду без необхідності прямої роботи з мовою SQL.

EF Core дозволяє розробнику оперувати з базою даних як із наборами об’єктів, що значно спрощує реалізацію CRUD-операцій (Create, Read, Update, Delete) та логіку взаємодії з даними загалом. Однією з ключових переваг є наявність механізму міграцій, що дає змогу швидко вносити зміни до структури бази даних без втручання у вже реалізовану бізнес-логіку, а детальна документація доступна на офіційних ресурсах Microsoft[15]. Усі зміни структури застосовуються через декілька команд у консолі, що мінімізує ризики помилок і пришвидшує процес розробки.

2.6.3 SQL Server

Для зберігання даних у проекті було використано **Microsoft SQL Server**, який зарекомендував себе як стабільна, масштабована та високопродуктивна реляційна система керування базами даних (СКБД).

Цей інструмент забезпечує високу швидкість обробки запитів навіть за умови збільшення обсягів даних, підтримує транзакції, забезпечує цілісність і

консистентність даних. Окремо слід зазначити відмінну інтеграцію SQL Server з EF Core, що значно спростило налаштування доступу до даних і реалізацію відповідних запитів у рамках архітектури додатка. Також варто відзначити доступність безкоштовних редакцій (наприклад, SQL Server Express), що робить цю СКБД оптимальною для реалізації проектів малого та середнього масштабу.

2.6.4 Bootstrap 5 + Razor Views

Для реалізації адаптивного та сучасного інтерфейсу користувача було використано фреймворк Bootstrap 5 у поєднанні з Razor Views.

Bootstrap 5 забезпечує візуальну привабливість і коректне відображення сторінок на пристроях з різною роздільною здатністю, що є критично важливим у випадку застосування системи на мобільних пристроях або планшетах. Завдяки наперед визначеним стилям, компоненти інтерфейсу мають охайний вигляд «із коробки», що дозволяє зосередитися на функціональній частині проєкту без потреби у складному дизайні.

Razor Views забезпечують тісну інтеграцію між HTML-кодом та серверною логікою на C#, дозволяючи реалізовувати динамічні шаблони сторінок із високою гнучкістю, необхідною для побудови інтерактивного веб-інтерфейсу.

2.6.5 Chart.js (або аналог для побудови графіків)

З метою забезпечення візуалізації аналітичних даних було обрано бібліотеку **Chart.js** як зручний інструмент для побудови графіків і діаграм.

Аналітична інформація у вигляді числових таблиць не завжди дозволяє швидко оцінити поточний стан справ, особливо у сфері обліку матеріалів. Графічні елементи значно покращують сприйняття динаміки: дозволяють візуально відстежити, які ресурси використовуються найінтенсивніше, де відбуваються перевитрати, а які матеріали залишаються невикористаними. **Chart.js** легко інтегрується з Bootstrap, має простий у використанні API та підтримує кастомізацію зовнішнього вигляду графіків відповідно до вимог користувача.

2.6.6 Git + GitHub

Для ефективного управління версіями коду та забезпечення спільної роботи над проектом використовувалася система контролю версій Git. Хмарна платформа GitHub була обрана для хостингу репозиторію, що забезпечило надійне зберігання історії змін та легкий доступ до кодової бази.

2.6.7 Visual Studio

Розробка програмного забезпечення CoffeeTrack велася в інтегрованому середовищі розробки (IDE) Visual Studio. Її вибір зумовлений широким набором інструментів для написання, налагодження та тестування коду, а також тісною інтеграцією з платформою .NET та фреймворком ASP.NET Core, що значно прискорило процес розробки.

3 РЕАЛІЗАЦІЯ ПРОГРАМНОЇ СИСТЕМИ

3.1 Функціональні та нефункціональні вимоги

Ефективність будь-якого прикладного програмного забезпечення визначається не лише його загальним призначенням, а й здатністю враховувати практичні потреби користувачів у реальних умовах експлуатації. У цьому контексті ключову роль відіграють функціональні та нефункціональні вимоги, які закладаються ще на етапі проектування системи.

Під час розробки програмного забезпечення *CoffeeTrack* основною метою було створення прикладного рішення, здатного оптимізувати щоденну роботу кав'ярні. Основний акцент зроблено на деталях, які, хоча й можуть здаватися другорядними, суттєво впливають на загальну зручність, швидкість та надійність роботи.

У межах реалізації програмного забезпечення *CoffeeTrack* передбачено комплекс функціональних можливостей, що забезпечують повноцінний облік матеріалів у кав'ярні. Основні з них включають:

- додавання нових матеріалів. Система дозволяє швидко реєструвати нові позиції матеріалів (наприклад, каву, стаканчики, серветки) із зазначенням назви, одиниці виміру та початкової кількості. Це усуває потребу в неструктурованому веденні обліку та мінімізує залежність від пам'яті персоналу;

- контроль мінімальних залишків. Коли залишки певного матеріалу досягають критичного рівня, система автоматично формує відповідне повідомлення. Це дозволяє своєчасно реагувати на потребу в закупівлях, особливо у періоди підвищеного попиту;

- фіксація закупівель. Надходження нових партій матеріалів реєструється за кілька кліків. Для кожного запису вказуються дата, кількість та відповідна позиція, що забезпечує точність і хронологічність обліку;

- облік списання матеріалів. Система дозволяє фіксувати витрати, псування або втрати матеріалів із зазначенням причини. Це дає змогу відстежувати динаміку використання ресурсів та виявляти потенційні джерела нераціональних витрат;

- аналітична обробка даних. Вбудовані таблиці та графіки візуалізують динаміку використання матеріалів за обраний період (день, тиждень, місяць). Аналітичні інструменти допомагають приймати обґрунтовані управлінські рішення щодо оптимізації закупівель та складу;

- формування прогнозів. На основі накопиченої статистики система прогнозує майбутні потреби в матеріалах на мові C#[16]. Це дозволяє уникати як дефіциту, так і надмірного накопичення запасів, що економить ресурси підприємства;

- зберігання та експорт звітів. Програма підтримує формування структурованих звітів, придатних для подальшого аналізу, друку або передачі адміністрації й бухгалтерії. Це забезпечує прозорість документообігу та підвищує ефективність управління.

Таблиця 3.1

Основні модулі CoffeeTrack та можливості в кожному з них

Розділ	**Що дозволяє зробити**
Матеріали	Переглядати залишки, додавати нові матеріали, слідкувати за критичними порогами
Закупівлі	Додавати записи про надходження матеріалів
Списання	Фіксувати використання матеріалів під час роботи

Продовження таблиці 3.1

Основні модулі CoffeeTrack та можливості в кожному з них

Розділ	**Що дозволяє зробити**
Аналітика	Аналізувати споживання матеріалів по днях, тижнях або місяцях
Прогноз	Отримувати підказки про майбутні закупівлі на основі статистики
Звіти	Формувати та завантажувати звіти в зручному форматі

Система *CoffeeTrack* не претендує на конкуренцію з комплексними корпоративними рішеннями, орієнтованими на великі мережі. Її головна перевага полягає у сфокусованості на основних завданнях обліку матеріалів у закладах малого бізнесу. Інтерфейс є інтуїтивно зрозумілим, функціонал — чітко окресленим, а використання — простим і ефективним. Завдяки цьому повсякденна робота персоналу, зокрема бариста або адміністратора, стає більш впорядкованою та передбачуваною, а процес управління залишками перетворюється з рутинної задачі на зручний і швидкий інструмент, що потребує мінімум зусиль.

Нефункціональні можливості CoffeeTrack:

- адаптивність інтерфейсу. Графічний інтерфейс користувача повинен бути оптимізований для використання на персональних комп’ютерах під управлінням операційної системи Windows. Передбачається підтримка стандартних елементів керування та роздільної здатності екрана не нижче 1366×768 пікселів;

- зручність використання. Навігація системою повинна мати можливість виконання основних дій у межах 1–2 кліків. Це забезпечує мінімальний час на навчання персоналу та зменшує ймовірність помилок при введенні даних;

- швидкий запуск. Система повинна запускатися повністю протягом не більше ніж 5 секунд на стандартному робочому комп’ютері з сучасними характеристиками;

- продуктивність при роботі з даними. Час обробки та відповіді на запити (наприклад, перегляд таблиць або аналітики) не повинен перевищувати 3 секунд при обсязі до 1000 записів у базі даних;

3.2 Опис користувацького інтерфейсу

Користувацький інтерфейс (UI, від англ. *User Interface*) є одним із найважливіших компонентів будь-якої програмної системи, орієнтованої на щоденне практичне використання. У випадку системи CoffeeTrack, яка розроблялася для автоматизації обліку матеріалів у кав’ярнях, саме інтерфейс визначає загальну якість взаємодії між людиною та програмою, впливає на ефективність роботи персоналу, швидкість адаптації до нового інструменту та загальний рівень задоволеності користувачів.

На етапі проектування інтерфейсу було сформульовано низку ключових вимог, які базувалися як на загальних принципах UX/UI-дизайну, так і на специфіці функціонування закладів громадського харчування малого формату. Основною метою було створення максимально доступного, логічного, зручного та передбачуваного інтерфейсу, який не потребує додаткового навчання або спеціальної технічної підготовки з боку персоналу.

Інтерфейс системи побудований за принципом модульності та функціональної сегментації. Усі основні дії згруповані у чітко розмежовані логічні блоки, що дозволяє швидко знаходити потрібну функцію — незалежно від того, чи користувач є баристою, адміністратором чи власником закладу. Основне навігаційне меню включає розділи:

- «Матеріали» — перегляд, додавання, редагування залишків;

- «Закупівлі» — фіксація нових поставок із деталями щодо дати, кількості та номенклатури;

- «Списання» — контроль списаних ресурсів з поясненням причин;
- «Аналітика» — відображення динаміки витрат у вигляді таблиць і графіків;
- «Прогноз» — підказки щодо майбутніх потреб;
- «Звіти» — генерація та експорт даних у форматах для подальшого аналізу.

Завдяки впровадженню зрозумілих форм та зрозумілих текстових підказок, процес взаємодії з кожним із цих розділів є максимально прямолінійним і логічним. Наприклад, додавання закупівлі відбувається через просту форму із випадаючими списками, полем для дати та кількісним інпутом — без зайвих етапів або технічних складнощів.

Значну увагу також було приділено відповідності інтерфейсу сучасним стандартам адаптивного дизайну. Система повноцінно функціонує як на стаціонарних ПК, так і на мобільних пристроях (планшети, смартфони), що особливо важливо для власників, які можуть контролювати ситуацію віддалено. Завдяки використанню фреймворку Bootstrap 5 інтерфейс автоматично підлаштовується під ширину екрану, зберігаючи зручність читання та керування.

Додатково варто відзначити такі ключові властивості інтерфейсу:

- мінімізація когнітивного навантаження — зменшення кількості інформації, яку користувач повинен одночасно обробити;
- простота навігації — відсутність вкладених меню, логічна послідовність дій, зрозумілі іконки;
- системна уніфікація — всі форми побудовані за єдиним шаблоном, що знижує час на освоєння нових розділів;
- зворотний зв’язок — кожна дія супроводжується повідомленням (успіх, помилка, підтвердження), що підвищує впевненість у роботі.

Таким чином, інтерфейс CoffeeTrack було розроблено відповідно до принципів доступності, ергономічності та функціональної чистоти. Він не лише виконує роль точки входу до системи, а й створює комфортне середовище для виконання рутинних завдань персоналом кав’ярні — швидко, точно та без зайвих ускладнень. Такий підхід дозволяє не лише підвищити ефективність роботи, а й

мінімізувати ймовірність помилок, що особливо важливо у динамічному середовищі закладів громадського харчування.

3.2 Проектування та реалізація користувацького інтерфейсу

Розробка користувацького інтерфейсу інформаційної системи CoffeeTrack здійснювалася з урахуванням ключових принципів сучасного UX/UI-дизайну (User Experience / User Interface), які базуються на забезпеченні високого рівня зручності, зрозумілості, візуальної узгодженості та ефективності взаємодії з програмним продуктом.

Зважаючи на специфіку цільової аудиторії — персонал закладів громадського харчування (бариста, адміністратори, менеджери), які, як правило, не мають технічної підготовки — до дизайну інтерфейсу були поставлені жорсткі вимоги щодо простоти освоєння, швидкості доступу до функцій і мінімізації дій, необхідних для досягнення результату.

- мінімалізм та візуальна чистота;

Одним із базових принципів, що ліг в основу дизайну системи, є мінімалізм. Було прийнято рішення відмовитися від надмірної графіки, складних елементів керування, спливаючих підказок і контекстних меню, які лише ускладнюють роботу користувача. Натомість перевага надавалася лаконічному представленню інформації, з акцентом на найважливіших даних, що потребують дій або уваги. Наприклад, система автоматично підсвічує критичні залишки матеріалів червоним кольором, аби уникнути їх непомітного вичерпання.

- принцип «одна дія — один результат»;

Усі форми та сторінки інтерфейсу побудовані за логікою "одна дія — один результат". Це означає, що кожен екран реалізує лише одну ключову функцію — наприклад, додавання закупівлі або перегляд звітів — без перевантаження інтерфейсу додатковими опціями. Таким чином, користувач завжди чітко розуміє, що саме від нього очікується, і не відволікається на зайві елементи.

- принцип передбачуваності;

Інтерфейс системи побудовано таким чином, щоб усі елементи розміщувалися у стандартних та очікуваних місцях. Кнопки типу “Додати”, “Редагувати”, “Зберегти” завжди розташовані у нижній частині форм, і мають однаковий зовнішній вигляд на всіх сторінках. Це дозволяє сформувати у користувача звичку і зменшити час на пошук елементів керування.

- доступність;

Завдяки використанню простих форм, піктограм, контрастних кольорів та зручних полів введення, інтерфейс CoffeeTrack є інтуїтивно зрозумілим навіть для нових користувачів. Система розрахована на те, що працівник зможе самостійно почати роботу, не витрачаючи час на навчання або ознайомлення з інструкціями. Це особливо важливо для кав’ярень з високою плинністю персоналу, де критичною є низька вартість входу у використання програмного продукту.

- адаптивність і мобільність;

В умовах сучасного бізнесу, особливо у сфері обслуговування, зростає потреба в мобільності інформаційних систем. У зв’язку з цим CoffeeTrack підтримує адаптивний дизайн, реалізований на основі фреймворку Bootstrap 5, що дозволяє відображати інтерфейс коректно на екранах будь-якого розміру — від смартфонів до великих моніторів. Таким чином, як бариста, так і власник кав’ярні може працювати з програмою незалежно від фізичного місця перебування.

- візуальна ієрархія та акценти;

Ще одним важливим принципом стало використання візуальної ієрархії — розміщення найважливішої інформації в зоні першого погляду, підсвічування пріоритетних дій і контрастне оформлення повідомлень. Наприклад, сповіщення про помилки мають червоне тло, тоді як успішне виконання операції — зелене. Заголовки, підрозділи та кнопки згруповано у відповідні блоки, що покращує сприйняття і знижує навантаження на користувача.

- уніфікованість і повторюваність шаблонів;

Одним із завдань під час дизайну було забезпечення уніфікованого досвіду — користувач, який навчився додавати матеріал, зможе з такою ж легкістю додавати закупівлю, формувати звіт тощо. Повторення шаблонів у компонуванні

сторінок, структурі форм та логіці кнопок створює впевненість у діях і зменшує вірогідність помилок.

Таким чином, принципи дизайну CoffeeTrack спрямовані на досягнення максимальної зрозумілості, ефективності, простоти та надійності у використанні. Вони відображають сучасні стандарти побудови UI/UX-систем для малого бізнесу, особливо у галузі громадського харчування, де кожна секунда операційного часу має значення, а стабільність і швидкість прийняття рішень безпосередньо впливають на якість обслуговування клієнтів.

3.3Огляд основних сторінок

- сторінка "Матеріали"(Рисунок 3.1);

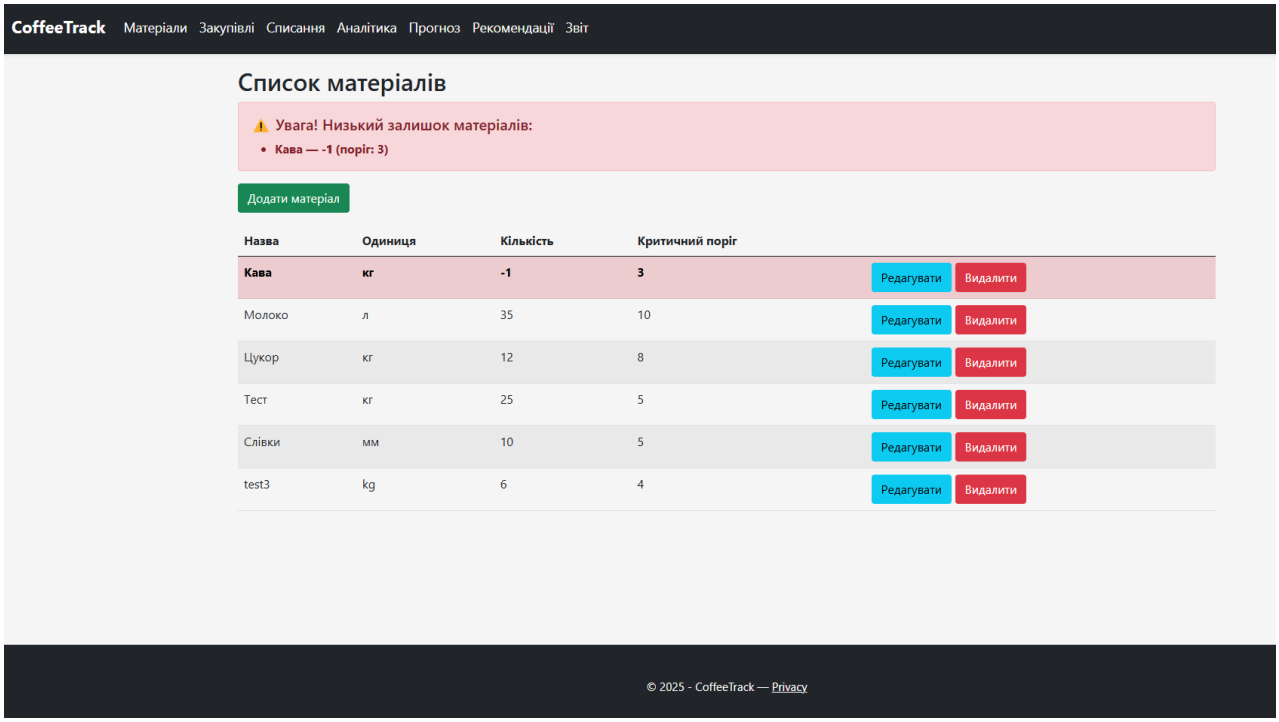


Рис. 3.1 Приклад екранних форм «Список матеріалів»

На екрані представлена таблиця обліку залишків, у якій відображається актуальна інформація щодо кожного матеріалу: назва, кількість, одиниця виміру, а також стан щодо досягнення мінімального порогу. Система автоматично виділяє критичні залишки (наприклад, червоним кольором), що дозволяє своєчасно

реагувати на потребу у поповненні. Функції додавання та редагування реалізовані у зручній формі — доступ до них здійснюється у декілька кліків без зайвої навігації.

- сторінка "Закупівлі"(Рисунок 3.2);

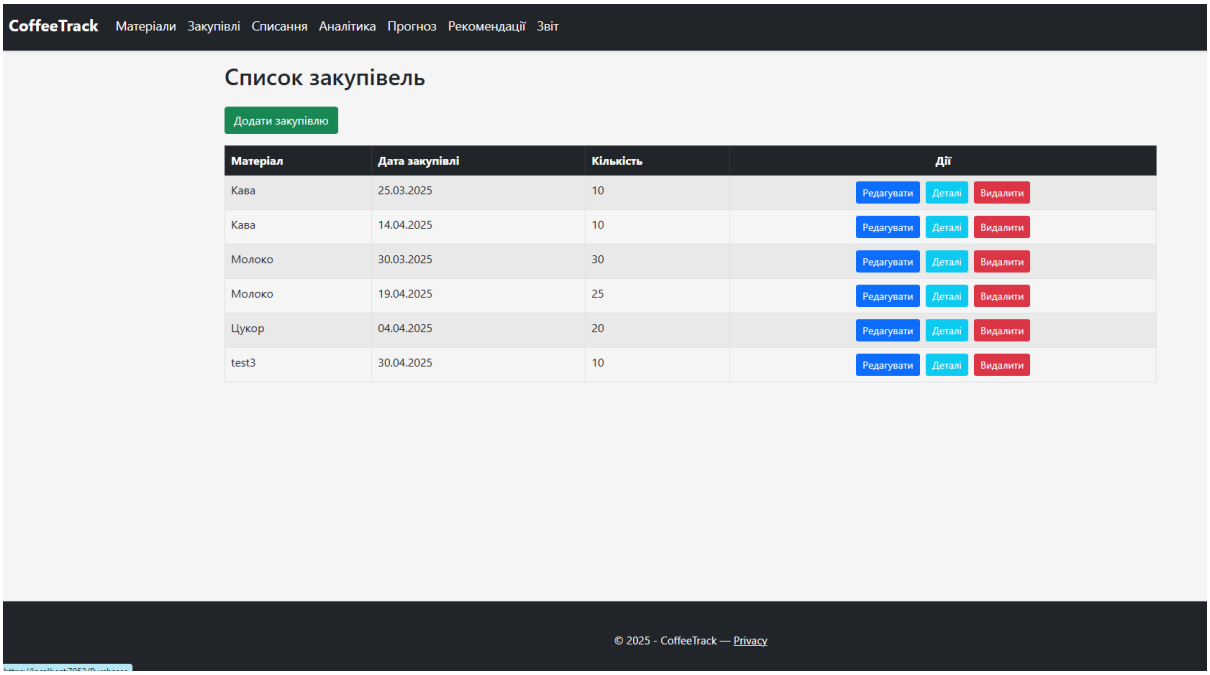


Рис 3.2 Приклад екранних форм «Список закупівель»;

Інтерфейс форми призначений для реєстрації нових надходжень матеріалів. Користувач обирає найменування матеріалу, зазначає кількість і дату закупівлі. У нижній частині форми відображається хронологічна історія попередніх постачань, що дозволяє швидко отримати доступ до даних про минулі закупівлі без необхідності додаткового пошуку.

- сторінка "Списання"(Рисунок 3.3);

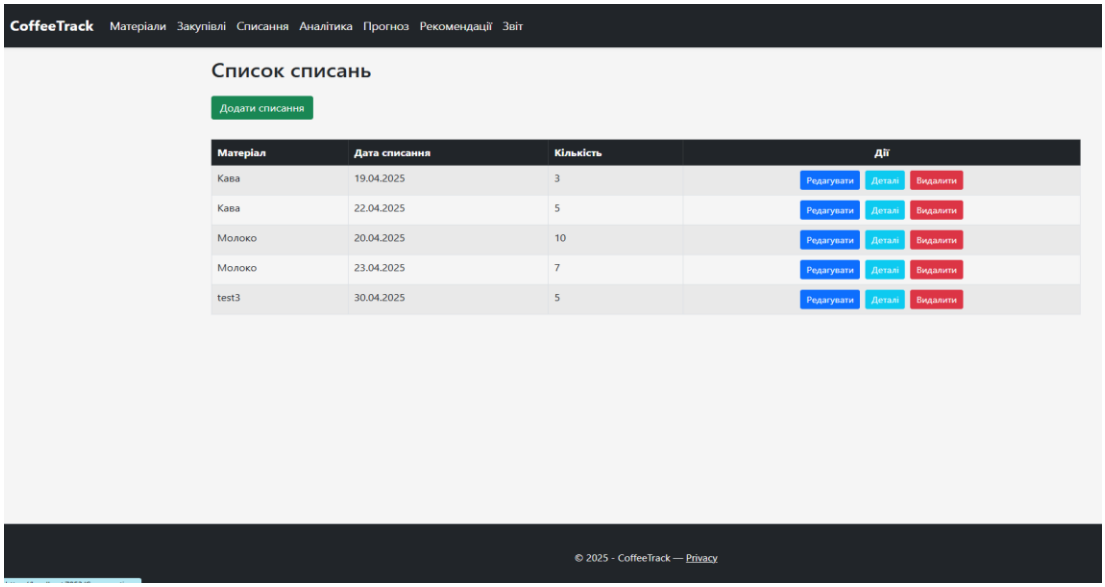


Рис 3.3 Приклад екранних форм «Список списань»

Форма призначена для обліку витратних матеріалів, що були використані, зіпсовані або підлягають списанню з інших причин. Інтерфейс забезпечує швидке та інтуїтивно зрозуміле внесення записів із зазначенням матеріалу, дати та причини списання. Це дозволяє підтримувати актуальність даних про залишки та контролювати джерела витрат.

- сторінка "Аналітика"(Рисунок 3.4);

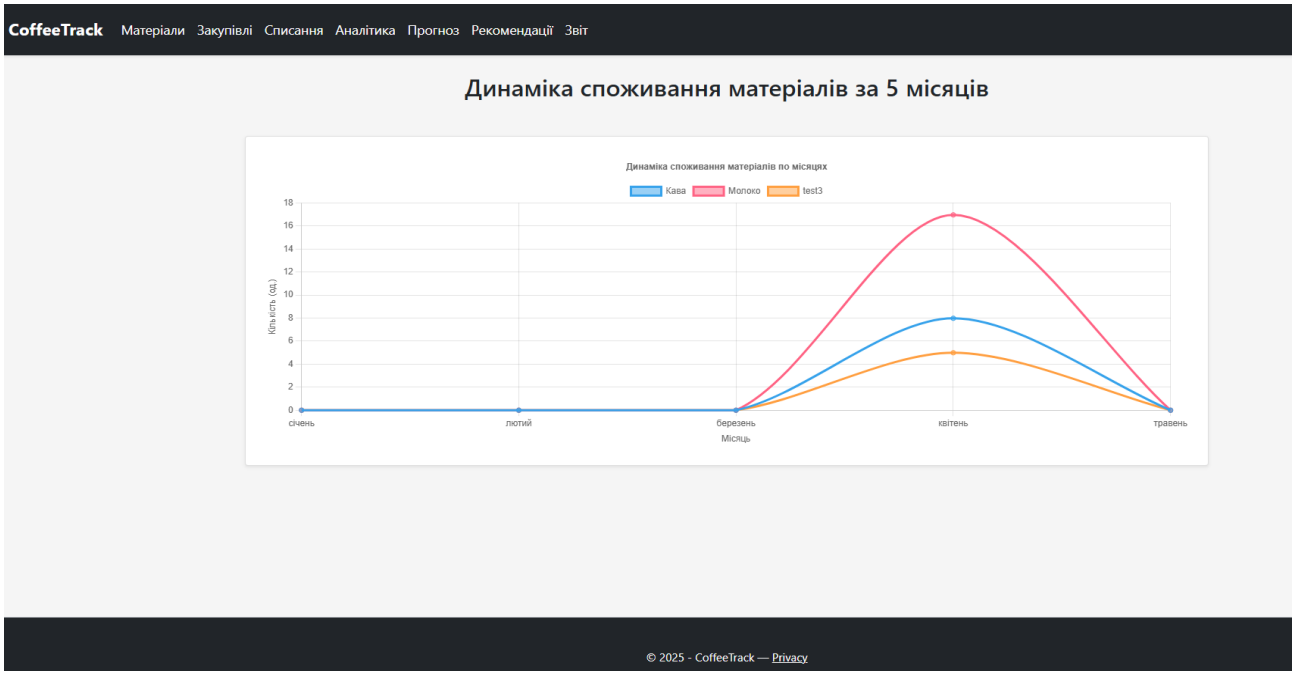


Рис 3.4 Приклад екранних форм «Аналітика»

На даній формі представлено інструменти аналітичної обробки даних щодо використання матеріалів. Система візуалізує динаміку споживання у вигляді графіків і діаграм, що дозволяє власникам та адміністраторам відстежувати тренди, виявляти пікові періоди та порівнювати витрати за різні часові інтервали. Такий підхід сприяє прийняттю обґрунтованих управлінських рішень.

- сторінка "Прогноз"(Рисунок 3.5);

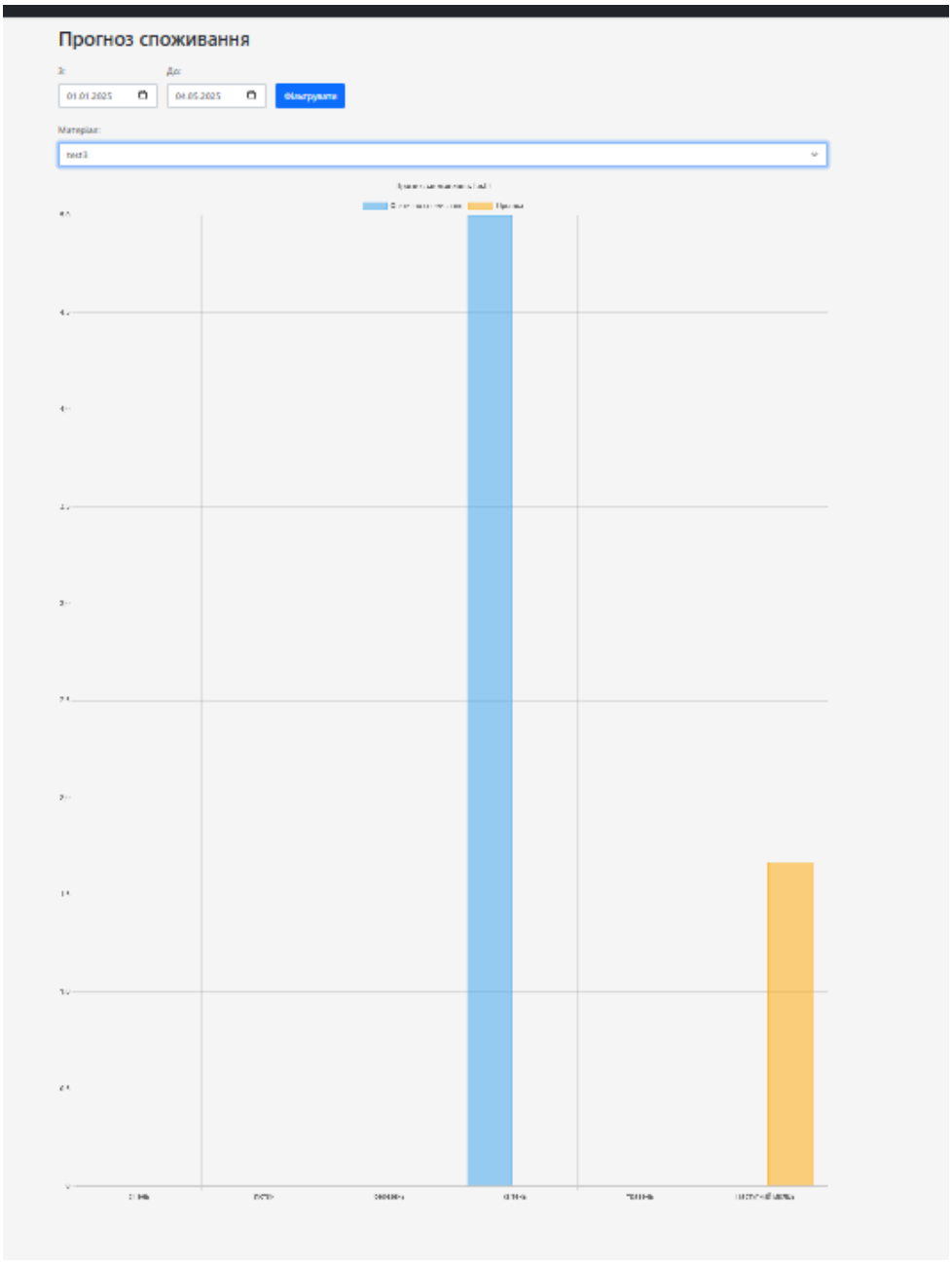


Рис 3.5 Приклад екранних форм «Прогноз»

Форма відображає результати прогнозного аналізу залишків матеріалів. На основі накопиченої статистики система автоматично визначає позиції, що наближаються до критичного рівня, та формує відповідні рекомендації. Це дозволяє завчасно планувати закупівлі та запобігати дефіциту ресурсів у критичні моменти роботи закладу.

- сторінка "Звіти"(Рисунок 3.6);

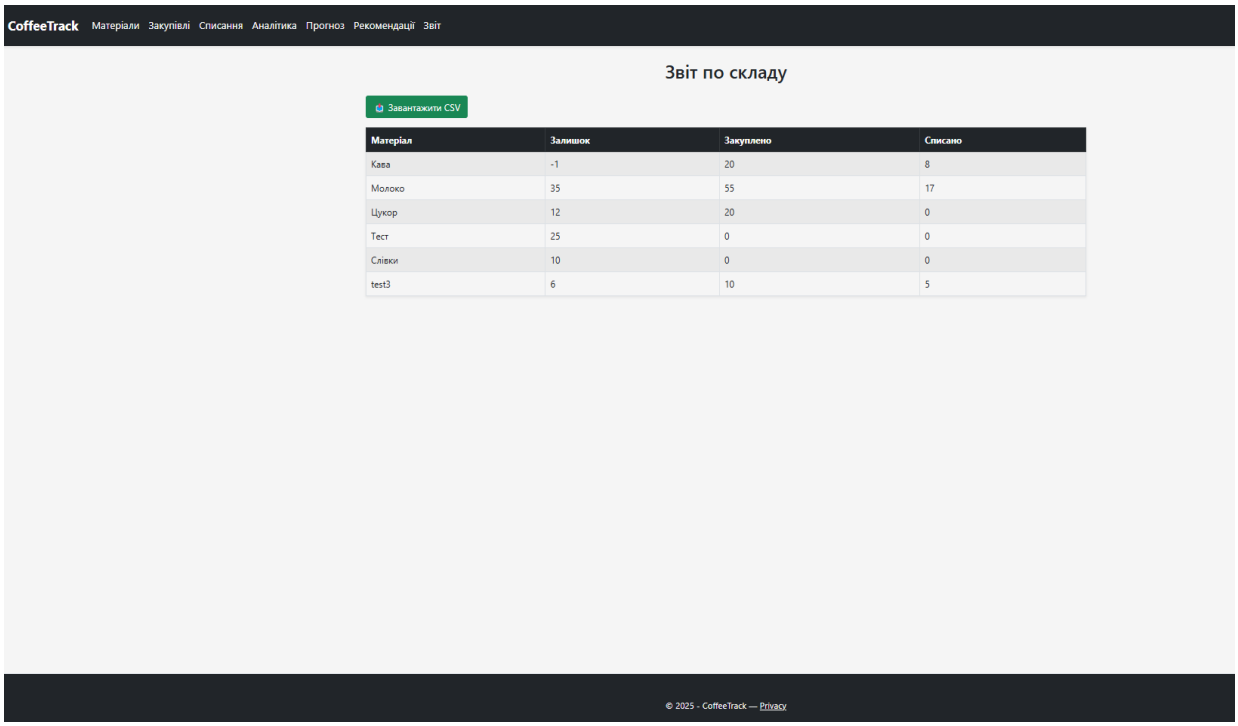


Рис 3.6 Приклад екранних форм «Звіти»

Форма призначена для формування структурованих звітів про закупівлі, списання та залишки матеріалів. Дані представлені у форматі, придатному для експорту (наприклад, у CSV), що дозволяє зберігати документи, передавати їх керівництву або використовувати в бухгалтерській звітності. Інтерфейс забезпечує зручність перегляду та простоту навігації по даних.

3.4Кольори та стилі

- основа інтерфейсу — світлі кольори з акцентами на важливих елементах (червоні для критичних залишків, зелені для позитивних значень) ;

- всі кнопки підписані та мають іконки;

- форматування таблиць робить інформацію зручною для сприйняття;

Інтерфейс системи *CoffeeTrack* не обмежується сукупністю форм і таблиць. Його призначення — слугувати інструментом підтримки користувача в щоденній роботі, забезпечуючи зручність, оперативність і наочність у взаємодії з даними. Завдяки інтуїтивному дизайну та логічній структурі, інтерфейс сприяє зменшенню навантаження на персонал і забезпечує прийняття більш обґрунтованих управлінських рішень на основі актуальної інформації.

3.5 Опис основних сценаріїв роботи

У будь-якому програмному забезпеченні головним завданням є не просто наявність кнопок і форм — а зрозумілі, продумані сценарії використання. Саме через ці сценарії і формується взаємодія користувача з системою. У випадку з нашим додатком **CoffeeTrack** усе побудовано так, щоб будь-який співробітник міг легко й без зусиль виконати всі основні операції. Розглянемо детально кожен з можливих сценаріїв.

3.5.1 Додавання нового матеріалу

У процесі роботи з закладом часто з’являються нові продукти чи інгредієнти. Для цього існує сторінка **"Додавання матеріалу"**. Користувач натискає відповідну кнопку і бачить просту форму. Йому потрібно вказати:

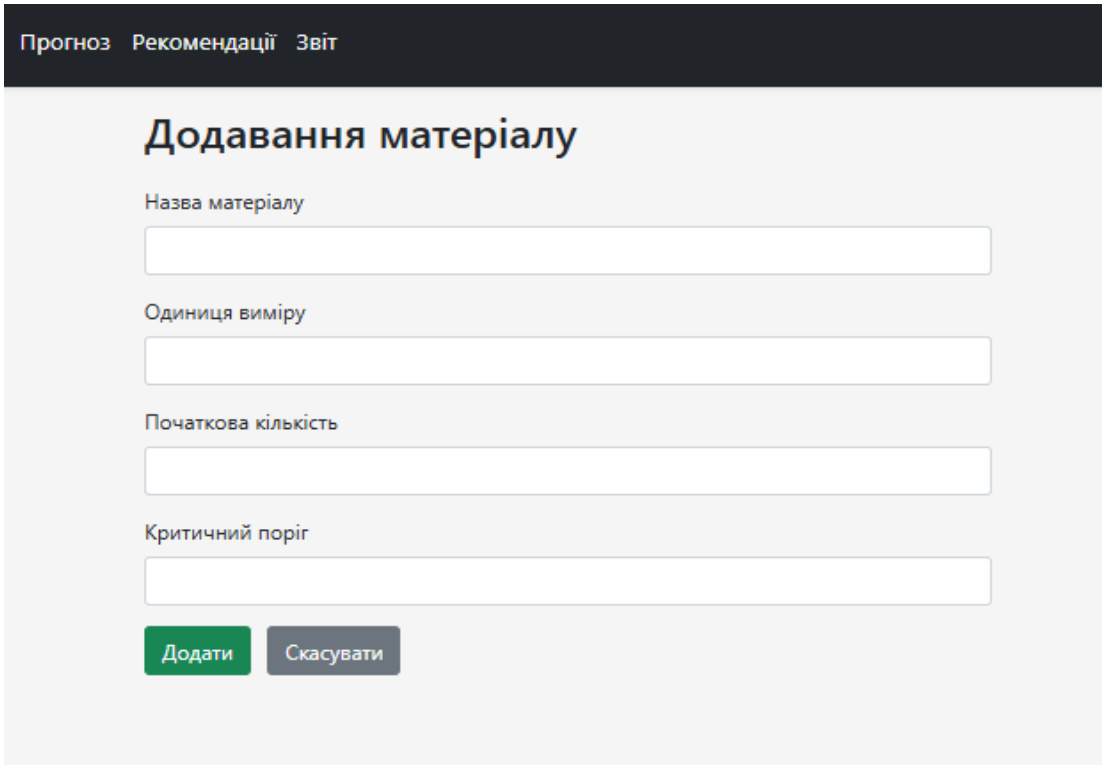


Рис 3.7 Приклад екранних форм «Додавання матеріалу»

- назву матеріалу — наприклад, «Цукор» ;

- одиницю виміру — кг, л, шт та ін;

- початкову кількість — з якої починається облік;

- критичний поріг — мінімально допустимий залишок, після якого система рекомендуватиме закупівлю;

Ця дія дозволяє ефективно додавати нові елементи в систему та одразу встановлювати важливі параметри контролю.

3.5.2 Проведення списання

Щодня частина запасів витрачається. Для цього передбачена сторінка **"Списання"**. Тут співробітник вказує:

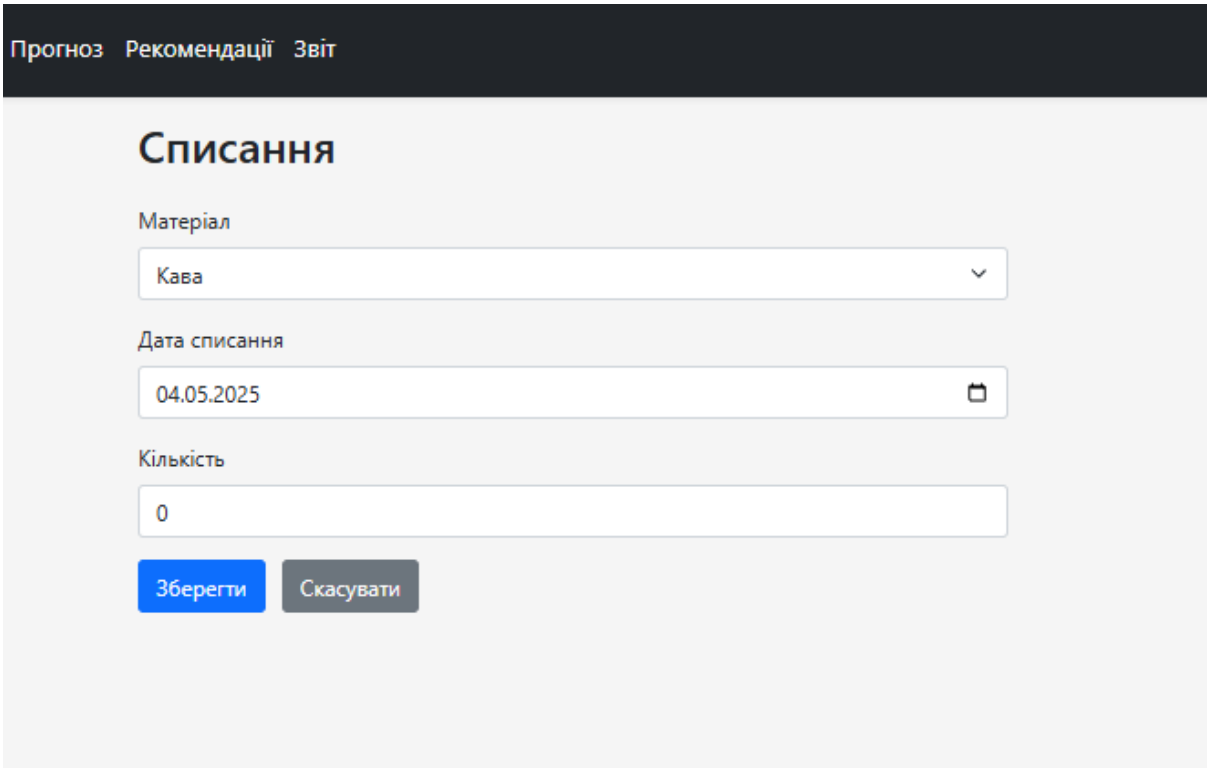


Рис 3.8 Приклад екранних форм «Списання»

- матеріал, який списується (вибирається зі списку) ;
- дату списання — день, коли була витрата;
- кількість списаного матеріалу.

Після натискання кнопки "Зберегти" інформація фіксується в системі, а залишок на складі автоматично зменшується.

3.5.3 Закупівля матеріалів

Коли запаси досягають критичного рівня, система рекомендує зробити закупівлю. Через розділ **"Закупівлі"** користувач може:

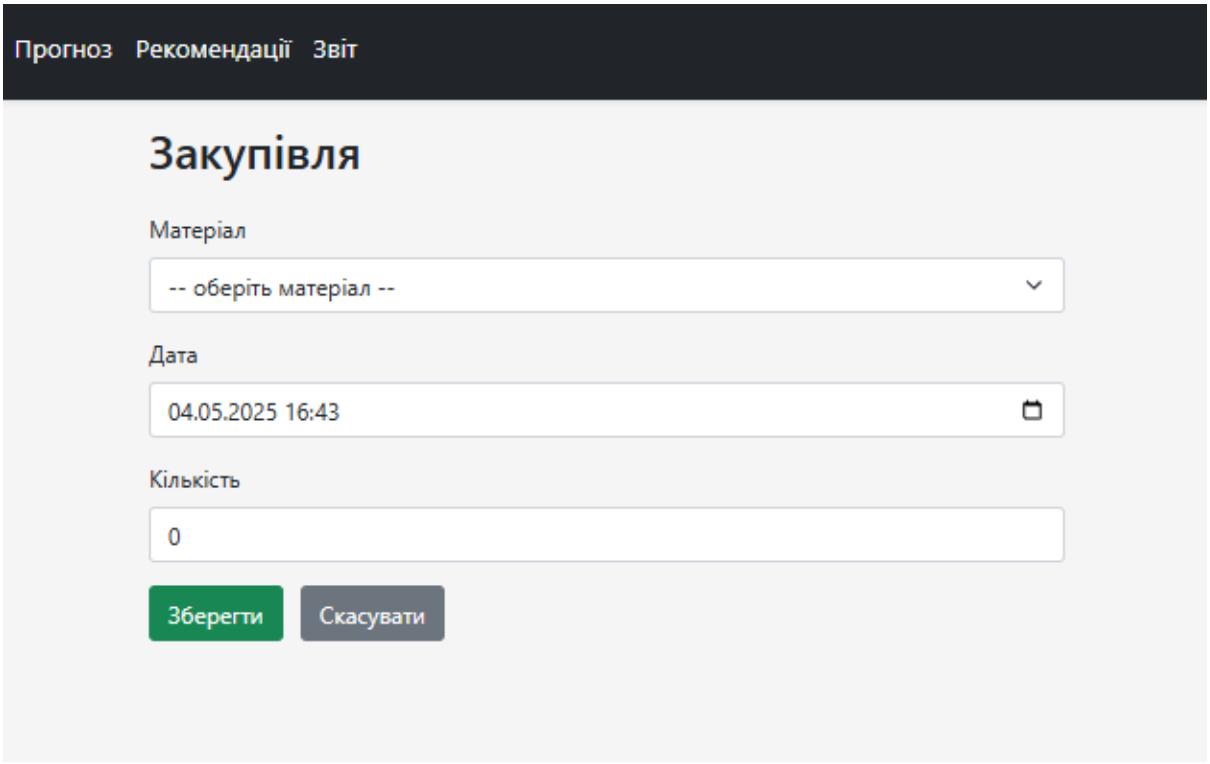


Рис 3.9 Приклад екранних форм «Закупівля»

Додати нову закупівлю, вказавши:

- матеріал;

- дату закупівлі;

- кількість закупленого.

Це допомагає постійно підтримувати склад у належному стані та запобігати дефіциту витратних матеріалів.

3.5.4 Прогноз споживання

Унікальною особливістю системи є можливість переглядати **"Прогноз споживання"**. Завдяки цьому користувач може:

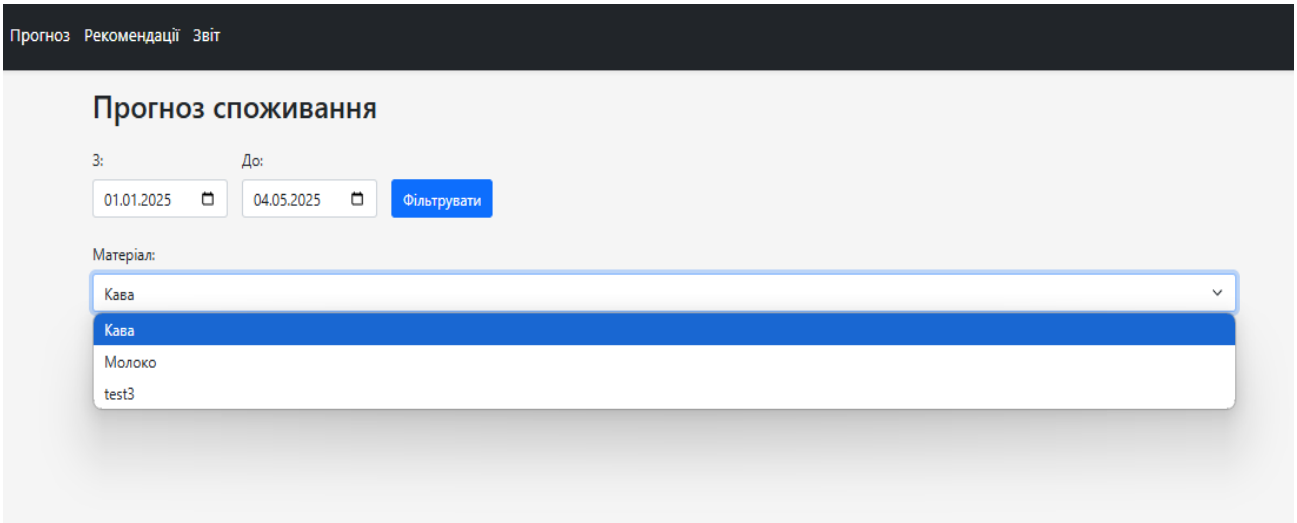


Рис 3.10 Приклад екранних форм «Прогноз»

- обрати певний матеріал;
- вказати часовий період;
- отримати графік, що показує тенденції витрати.

Це корисно для планування майбутніх закупівель, а також для виявлення пікових періодів споживання. В основі механізму прогнозування системи CoffeeTrack лежить аналіз динаміки споживання матеріалів за попередні періоди. Зокрема, для розрахунку прогнозованої потреби на майбутній період система використовує усереднені дані про фактичне споживання конкретного матеріалу за декілька останніх облікових відрізків. Такий підхід, хоч і є відносно простим, обраний свідомо, оскільки він забезпечує достатню точність для умов невеликої кав'ярні, не потребує складних налаштувань з боку користувача та дозволяє швидко отримати зрозумілі рекомендації. Головна перевага полягає у наданні персоналу орієнтиру для своєчасного поповнення запасів, мінімізуючи ризик раптового дефіциту ключових інгредієнтів.

3.5.5 Формування звітності та експорт

Коли наближається період звітності, працівникам потрібно мати під рукою зведену інформацію. На сторінці **"Звіт"**:

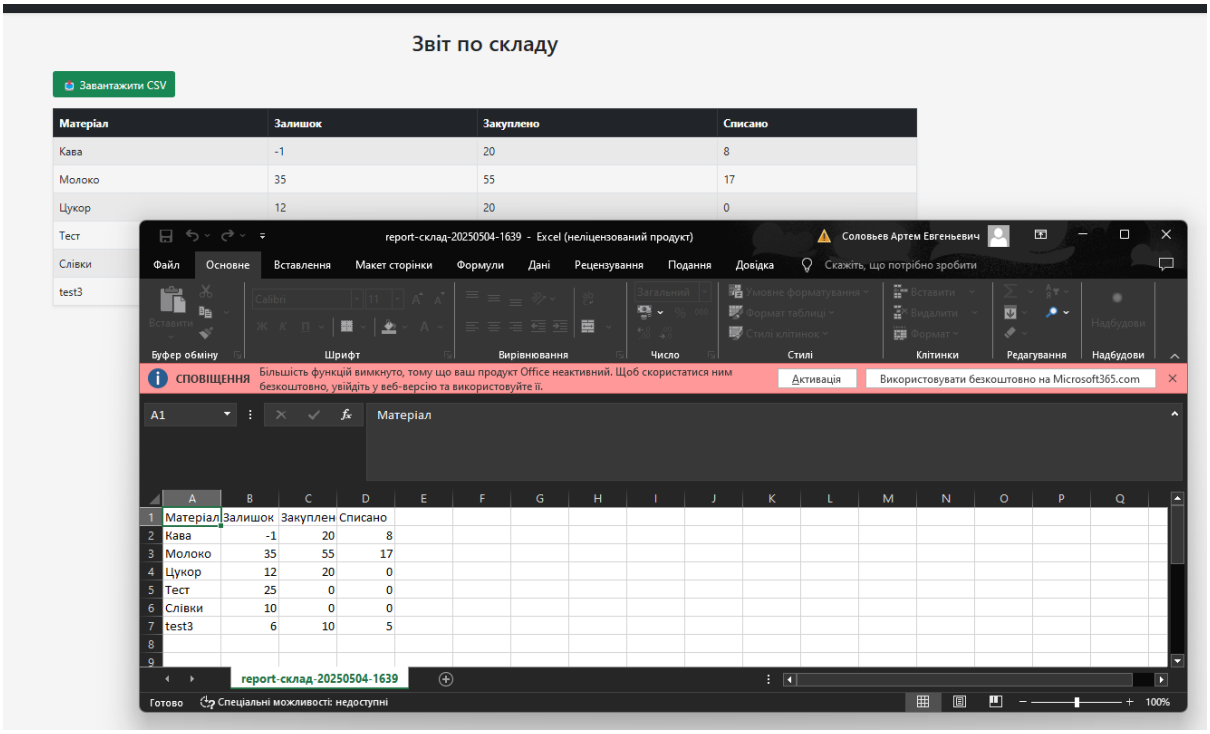


Рис 3.11 Приклад екранних форм «Звіт по складу»

- показано таблицю з матеріалами, їх залишками, закупленим та списаним обсягами;

- є кнопка для завантаження CSV-файлу для подальшої обробки в Excel або інших програмах;

- функція експорту дозволяє швидко надати всю необхідну інформацію для бухгалтерії або керівництва.

.

3.5.6 Редагування даних

Іноді трапляється, що дані введені помилково або потребують актуалізації. Для цього:

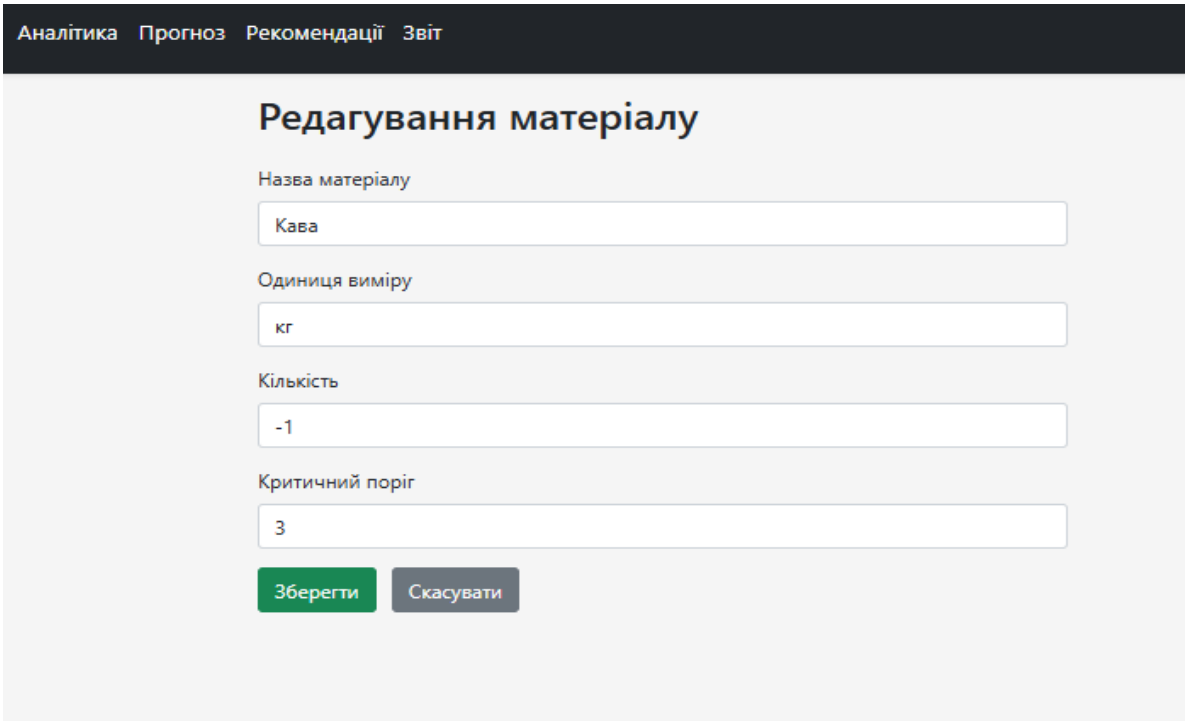


Рис 3.12 Приклад екранних форм «Редагування»

- на сторінці списку матеріалів чи списань є кнопка **"Редагувати"**;
- після натискання відкривається форма з уже внесеними даними, які можна змінити та зберегти.

Це дозволяє легко виправляти неточності без необхідності видаляти дані.

3.6 Демонстрація роботи та архітектурної побудови системи

Для повного розуміння принципів функціонування програмного забезпечення *CoffeeTrack* важливо не лише ознайомитися з інтерфейсною частиною системи, а й простежити логіку її внутрішніх процесів. Візуальна складова є лише зовнішнім відображенням складної взаємодії між різними рівнями архітектури.

Зокрема, демонстрація роботи системи дає змогу проаналізувати повний цикл обробки даних: від моменту ініціювання дії користувачем (наприклад, натискання кнопки «Додати матеріал») до збереження інформації у базі даних і подальшого відображення оновлених даних у користувацькому інтерфейсі. Така послідовність операцій реалізується через взаємодію між рівнем представлення,

бізнес-логікою та рівнем доступу до даних, що підтверджує ефективність обраної багаторівневої архітектури.

Злагоджена побудова архітектурної моделі сприяє стабільній роботі системи, її масштабованості та зручності у підтримці. Це особливо важливо для прикладних рішень, що використовуються в динамічному середовищі малого бізнесу, зокрема у закладах громадського харчування.

3.6.1 Архітектурна модель (MVC)

Як було зазначено раніше, програмне забезпечення CoffeeTrack реалізовано з використанням архітектурного патерну Model-View-Controller. Нижче наведено конкретні приклади реалізації кожного компоненту в системі: Модель MVC умовно поділяє логіку програми на три взаємопов’язані складові:

- **Model (Модель)** — У CoffeeTrack моделі представлені класами, що описують основні сутності системи та логіку роботи з ними. Наприклад, клас Material інкапсулює дані про матеріали (назва, кількість, критичний поріг), а взаємодія з базою даних для збереження та отримання цих даних здійснюється через Entity Framework Core. (Рисунок 3.13)

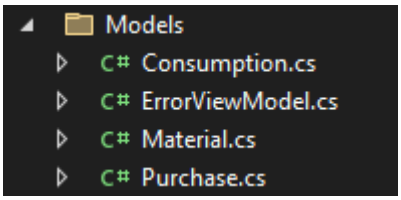


Рис 3.13 Приклад «Models»

- **View (Вигляд)** — Представлення реалізовані за допомогою технології Razor Views. Вони відповідають за відображення даних користувачеві та надання інтерфейсу для взаємодії (Рисунок 3.14)

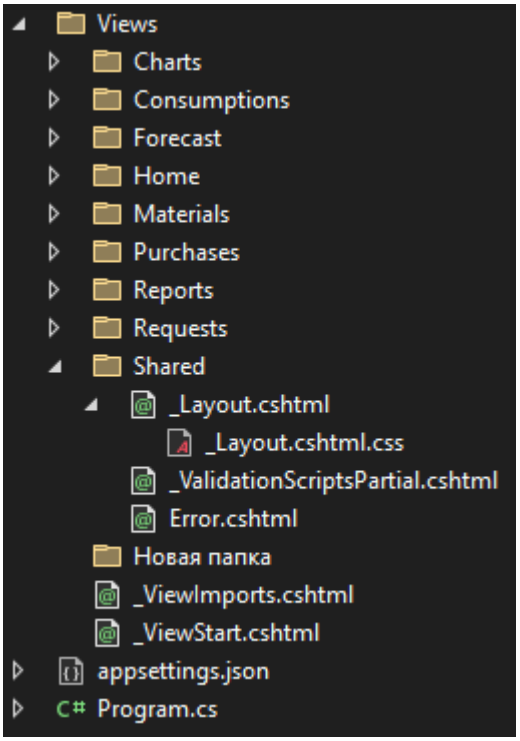


Рис 3.14 Приклад «Views»

- **Controller (Контролер)** — Контролери в ASP.NET Core MVC обробляють HTTP-запити від користувача, взаємодіють з моделями для виконання бізнес-логіки та повертають відповідні View. Наприклад, MaterialsController містить методи для відображення списку матеріалів, додавання нових матеріалів та їх редагування. Приклад взаємодії: користувач заповнює форму додавання нового матеріалу (View), дані надходять до відповідного методу контролера, контролер передає ці дані моделі для збереження в БД, після чого повертає оновлений список матеріалів (View)."(Рисунок 3.15)

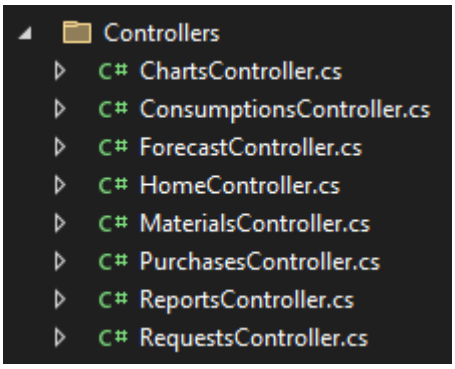


Рис 3.15 Приклад «Controllers»

3.6.2 Схема взаємодії представлена на рисунку 3.16.

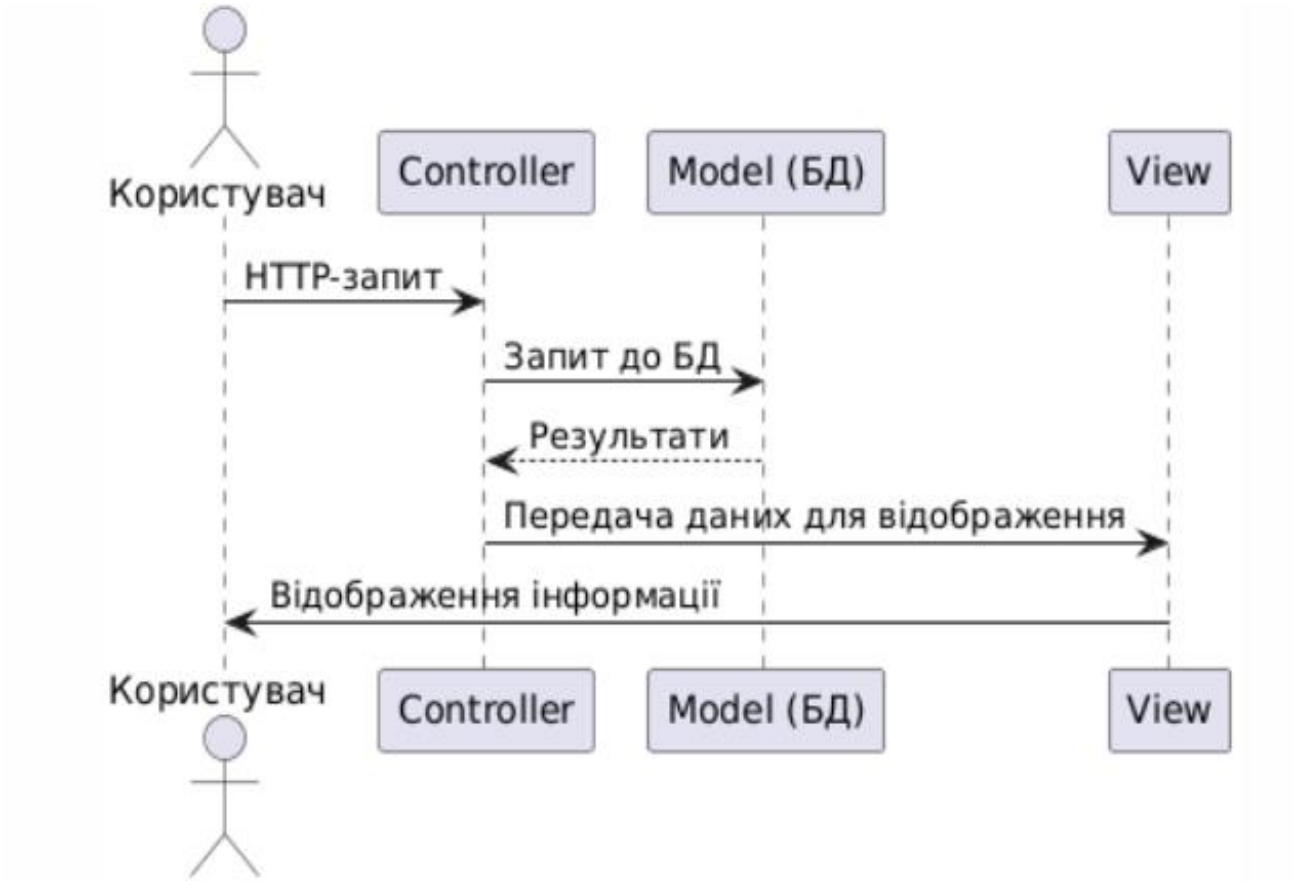


Рис 3.16 «Схема взаємодії»

3.7 Тестування програмного забезпечення

Після завершення розробки основного функціоналу системи CoffeeTrack було проведено її тестування з метою перевірки працездатності та відповідності поставленим вимогам. Тестування включало такі основні етапи:

- функціональне тестування: Перевірка коректності виконання всіх ключових функцій системи, зокрема, тестувалися сценарії додавання та редагування матеріалів, фіксації закупівель та списань, генерації звітів та відображення аналітичних даних і прогнозів;

- тестування користувацького інтерфейсу (UI Testing): Оцінка зручності навігації, коректності відображення елементів на різних сторінках, а також перевірка адаптивності інтерфейсу на екранах з різною роздільною здатністю;

- перевірка валідації даних: Тестування реакції системи на введення некоректних або неповних даних у формах.

В результаті тестування було підтверджено, що розроблена система CoffeeTrack функціонує відповідно до визначених вимог. Виявлені незначні недоліки були виправлені.

Основні сценарії роботи виконуються коректно, а інтерфейс є інтуїтивно зрозумілим для користувача.

ВИСНОВКИ

1. Проведено аналіз предметної області у сфері управління витратними матеріалами в кав'ярнях. Визначено основні проблеми, пов’язані з ручним обліком та відсутністю спеціалізованих інструментів автоматизації.

2. Розглянуто наявні програмні рішення, проаналізовано їх функціональні обмеження у контексті потреб малих закладів громадського харчування. Обґрунтовано доцільність створення власного програмного продукту, адаптованого до специфіки діяльності кав’ярень.

3. Сформульовано функціональні та нефункціональні вимоги до системи. Обрано відповідні технології реалізації: ASP.NET Core, Entity Framework, SQL Server, Razor Views, Bootstrap. Розроблено архітектуру програмного забезпечення на основі моделі MVC.

4. Створено інформаційну систему *CoffeeTrack*, яка забезпечує фіксацію закупівель і списань, контроль залишків, прогнозування потреб та формування звітності. Реалізовано користувацький інтерфейс, адаптований для повсякденної роботи персоналу кав’ярні.

5. Проведено тестування основних сценаріїв роботи системи, підтверджено її працездатність, зручність та відповідність визначеним вимогам.

6. Система має прикладну цінність та може бути впроваджена у діяльність реальних закладів малого бізнесу. У подальшому можливий розвиток функціональності, зокрема — інтеграція з обліковими системами, розширення аналітичних можливостей та підтримка мобільних платформ.

7. Робота пройшла апробацію:

1. Олексієва Є.О., Гаманюк І.М. Моделювання основної функціональності обліку витратних матеріалів та впорядкування процесу закупівель у щоденній діяльності кав’ярні. VI Всеукраїнська науково-технічна конференція «Застосування програмного забезпечення в інформаційно-комунікаційних технологіях», 24.04.2025, Київ, Державний університет

інформаційно-телекомунікаційних технологій. Збірник тез К.: ДУІКТ, 2025. С.522-524.

2. Олексієва Є.О., Гаманюк І.М. Розробка модуля аналітики та прогнозування потреб у витратних матеріалах на основі даних споживання кав’ярні. VI Всеукраїнська науково-технічна конференція «Застосування програмного забезпечення в інформаційно-комунікаційних технологіях», 24.04.2025, Київ, Державний університет інформаційно-телекомунікаційних технологій. Збірник тез К.: ДУІКТ, 2025. С. 618-620.

ПЕРЕЛІК ПОСИЛАНЬ

1. Захарін С.В. Управління товарними запасами. Економіка підприємств. 2006. (дата звернення: 23.05.2025).

2. Кудлаєва Н., Андрицька В. Формування обліково-аналітичного забезпечення управління запасами. 2020. URL: https://doi.org/10.32839/2304-5809/2020-12-88-29 (дата звернення: 28.05.2025)

3. Пушкарь І., Шишкіна А. Особливості обліково-аналітичного забезпечення управління запасами. Економіка та суспільство. 2022. URL: https://doi.org/10.32782/2524-0072/2022-44-74 (дата звернення: 27.05.2025).

4. Муравський В. Автоматизація інвентаризації в комп"ютерно-комунікаційній формі обліку. *Вісник Тернопільського національного економічного університету. Економічні науки*. 2017.

5. Bustamante Sosa J. C., Salazar Pérez F. J., Meza Ortiz R. Inventory management optimization model in a restaurant applying inventory control and lean manufacturing. 5th south american international conference on industrial engineering and operations management. 2024. URL: https://doi.org/10.46254/sa05.20240148 (date of access: 27.05.2025).

6. Шепель І. Сучасні можливості організації обліку витрат виробництва і готової продукції в умовах комплексної автоматизації. Економіка та суспільство . 2023. DOI: https://doi.org/10.32782/2524-0072/2023-55-7 (дата звернення: 19.05.2025)..

7. Пеняк Ю. С., Сергієнко О. А. Інструментарій обліково-аналітичного управління витратами. Проблеми сучасних трансформацій. Серія: економіка та управління. 2023. URL: https://doi.org/10.54929/2786-5738-2023-9-09-03 (дата звернення: 27.05.2025).

8. Sutton S. G., Arnold V. Enterprise resource planning systems. Thomson South-Western. 2001.

9. Висоцький Ю. Б. Процес формування обліково-аналітичного забезпечення управління економічною безпекою підприємства. Проблеми сучасних трансформацій. Серія: економіка та управління . 2025. URL: https://doi.org/10.54929/2786-5738-2025-17-09-02 (дата звернення: 28.05.2025).

10. Fawcett T., Provost F. Data science for business: what you need to know about data mining and data-analytic thinking. O'Reilly Media, Incorporated, 2013.

11. Rostami-Tabar B. Demand forecasting for executives and professionals. CRC Press LLC, 2023.

12. Overview of ASP.NET Core MVC. Microsoft Learn: Build skills that open doors in your career. URL: https://learn.microsoft.com/aspnet/core/mvc/overview (date of access: 19.05.2025).

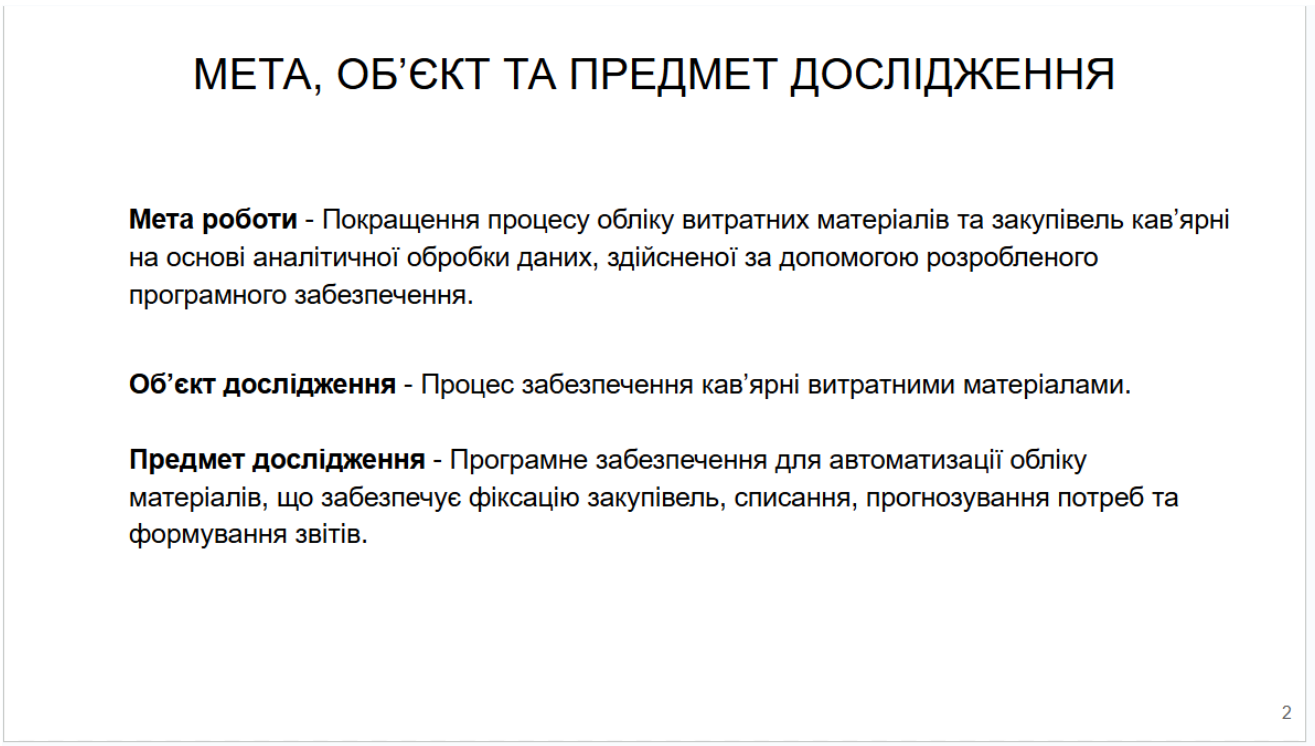
13. C# 11 and . NET 7 - modern cross-platform development fundamentals: start building websites and services with ASP. NET core 7, blazor, and EF core 7, 7th edition. Packt Publishing, Limited, 2022.

14. Overview of entity framework core - EF core. Microsoft Learn: Build skills that open doors in your career. URL: https://learn.microsoft.com/en-us/ef/core/ (date of access: 27.05.2025).

15. Microsoft Docs. Entity Framework Core documentation [Електронний ресурс]. URL: https://learn.microsoft.com/ef/core (дата звернення: 19.05.2025).

16. 8. Stack Overflow. How to implement forecasting in C# for sales/usage prediction. *Stack Overflow*. URL: https://stackoverflow.com (date of access: 25.05.2025).

ДОДАТОК А. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ



ЗАДАЧІ ДИПЛОМНОЇ РОБОТИ

1. Проаналізувати особливості процесу обліку витратних матеріалів у закладах типу кав'ярні.
2. Визначити основні проблеми, пов'язані з плануванням закупівель у кав'ярнях.
3. Дослідити сучасні підходи до автоматизації обліку та аналітики витрат.
4. Сформувати вимоги до програмного забезпечення для обліку та прогнозування потреб у витратних матеріалах.
5. Розробити програмне забезпечення, що забезпечує аналіз та візуалізацію даних про споживання.
6. Провести тестування розробленого застосунку на відповідність визначеним вимогам.

3

АНАЛІЗ АНАЛОГІВ

Характеристика	Облік залишків	Ведення історії витрат	Формування звітів	Прогнозування потреб	Орієнтація на кав'ярні	Гнучкість налаштувань
Poster POS	+	+	+	-	+	-
SkyService POS	+	+	-	-	+	-
Excel / Google Таблиці	-	+	-	-	-	+
CoffeeTrack	+	+	+	+	+	+

4

ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Функціональні вимоги:
1) Ведення обліку витратних матеріалів: додавання, редагування, списання, перегляд залишків.
2) Збереження інформації про закупівлі (дата, тип матеріалу, кількість).
3) Відображення списку витрат за обраний період.
4) Формування звіту (у вигляді таблиці з можливістю друку або експорту в Excel).
5) Статистичне прогнозування потреби на основі середнього споживання за попередні місяці.

Нефункціональні вимоги:
1) Інтерфейс, адаптований для ПК (Windows).
2) Простота використання: навігація та дії в 1–2 кліки.
3) Запуск не довше ніж (5 секунд)
4) Час відповіді на запити не більше ніж 3 секунди при обсязі 1000 записів.

5

ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ

ASP.NET MVC

Entity Framework Core

GitHub

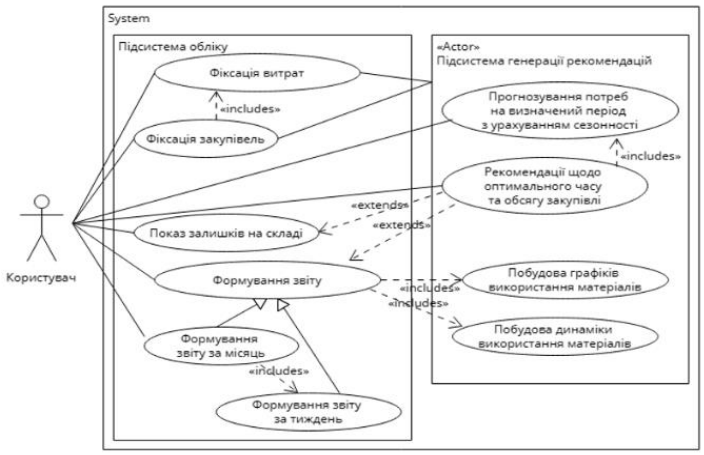
Visual Studio

Microsoft SQL Server License

Chart.js

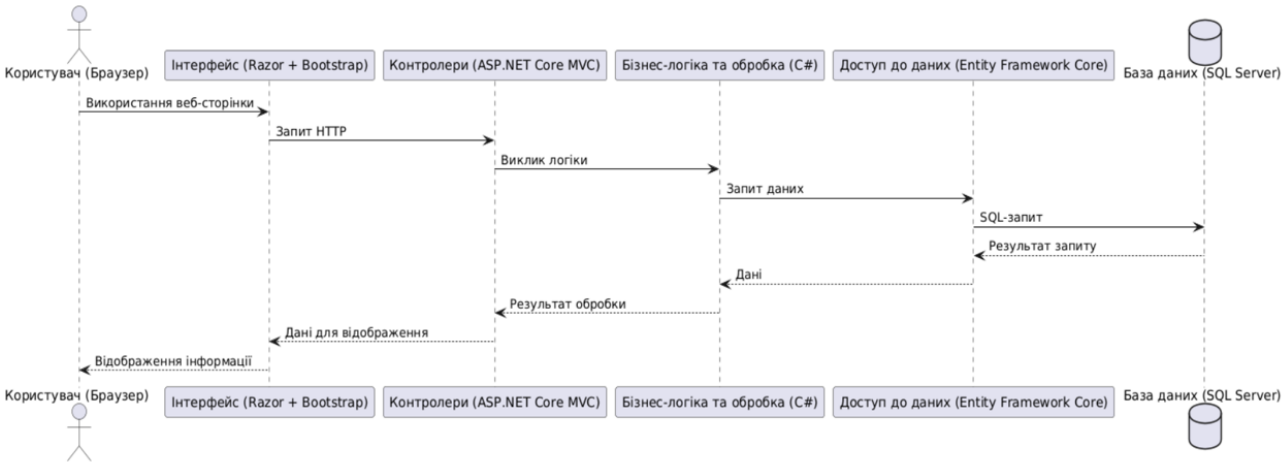
6

МОДЕЛЮВАННЯ ФУНКЦІОНАЛЬНОСТІ СИСТЕМИ

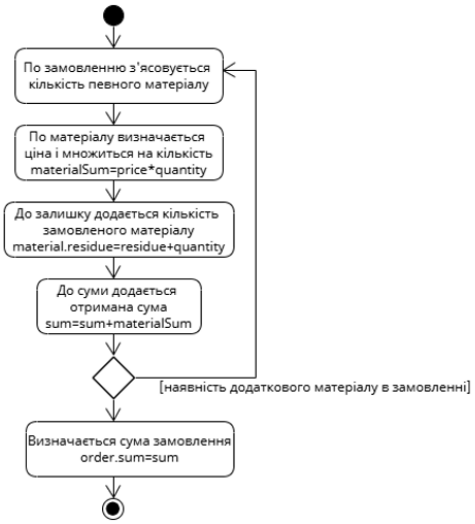


Діаграма варіантів використання

ДІАГРАМА ПОСЛІДОВНОСТІ



Діаграми діяльності

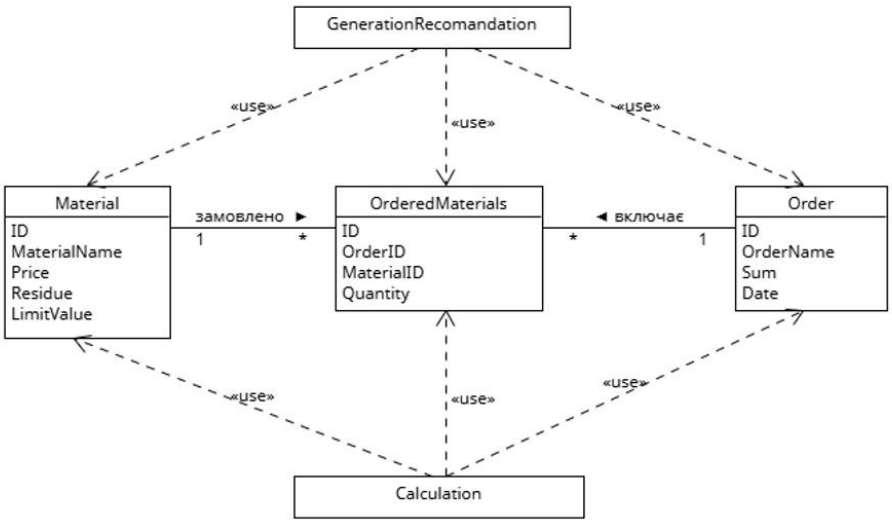


діаграма діяльності моделювання роботи генерації рекомендацій щодо закупівлі матеріалів

діаграма діяльності при калькуляції суми замовлення.

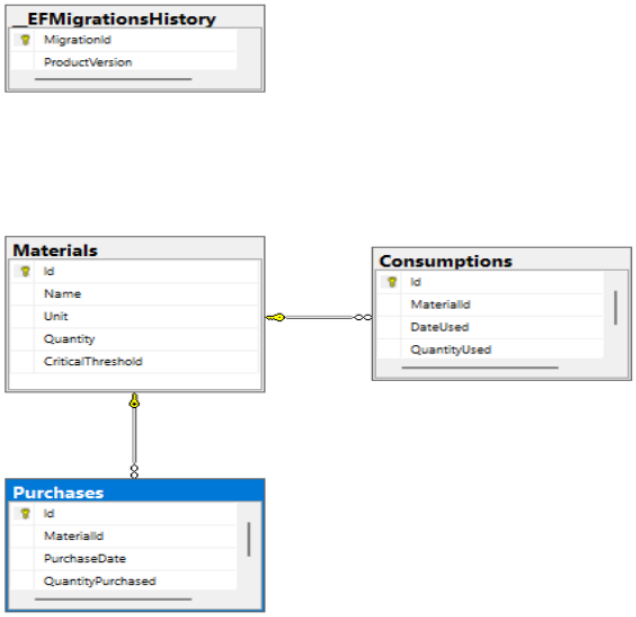
9

Діаграма предметної галузі

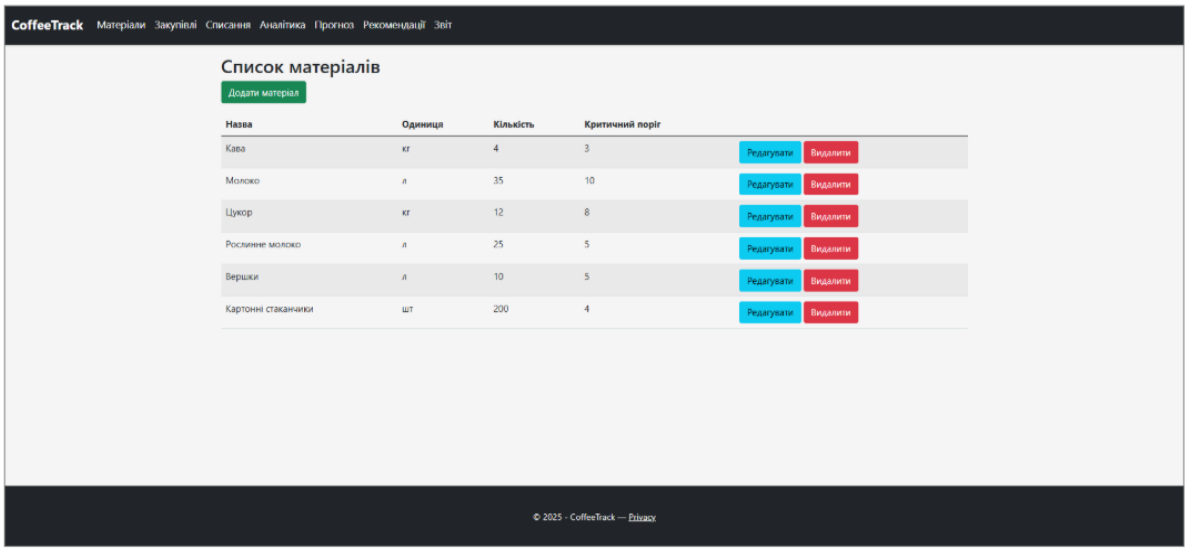


10

ДІАГРАМА КЛАСІВ

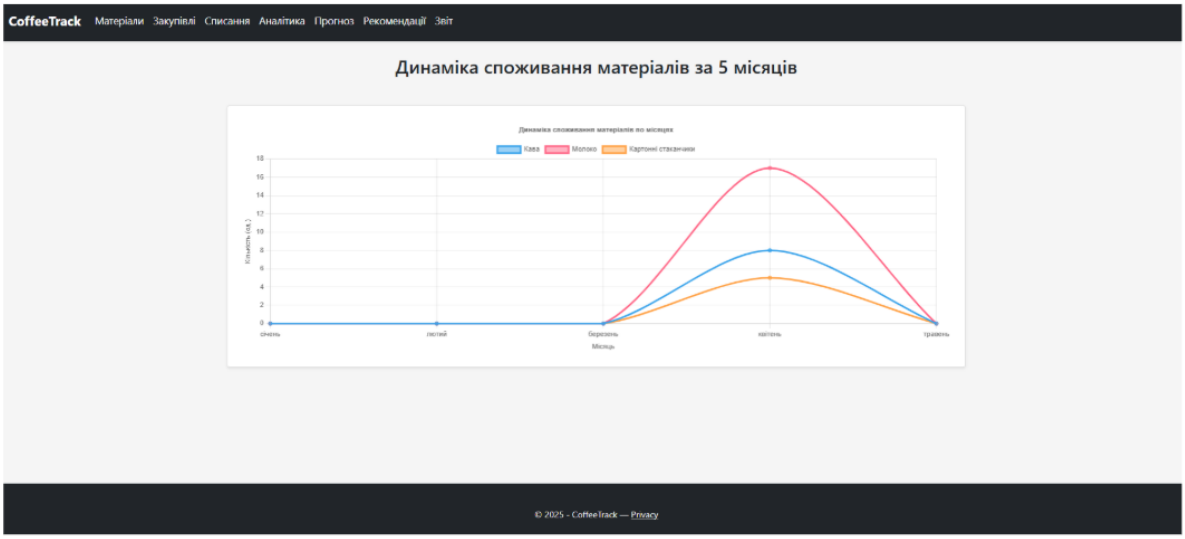


ЕКРАННІ ФОРМИ



Головна сторінка. Редагування, додавання та видалення витратних матеріалів.

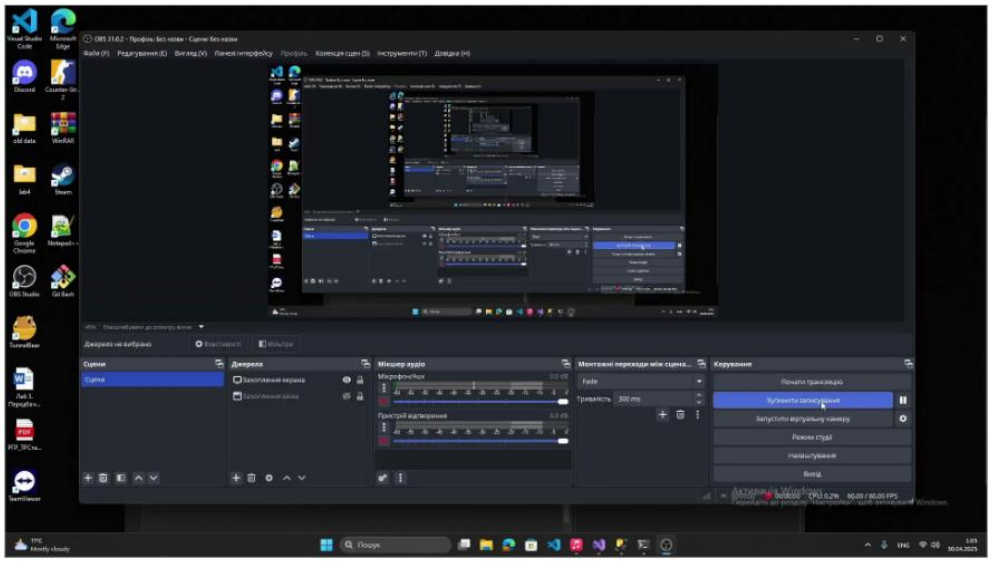
ЕКРАННІ ФОРМИ



Аналітика споживань.

12

ДЕМОНСТРАЦІЯ РОБОТИ ПРОГРАМИ



15

АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

1. Олексієва Є.О., Гаманюк І.М. Моделювання основної функціональності обліку витратних матеріалів та впорядкування процесу закупівель у щоденній діяльності кав'ярні. VI Всеукраїнська науково-технічна конференція «Застосування програмного забезпечення в інформаційно-комунікаційних технологіях», 24.04.2025, Київ, Державний університет інформаційно-телекомунікаційних технологій. Збірник тез К.: ДУІКТ, 2025. С.522-524.
2. Олексієва Є.О., Гаманюк І.М. Розробка модуля аналітики та прогнозування потреб у витратних матеріалах на основі даних споживання кав'ярні. VI Всеукраїнська науково-технічна конференція «Застосування програмного забезпечення в інформаційно-комунікаційних технологіях», 24.04.2025, Київ, Державний університет інформаційно-телекомунікаційних технологій. Збірник тез К.: ДУІКТ, 2025. С. 618-620.

ВИСНОВКИ

1. Проведено аналіз предметної області, зокрема вивчено специфіку організації обліку витратних матеріалів у кав'ярнях. Виокремлено характерні особливості та типові труднощі, що виникають у процесі планування закупівель, зокрема: залежність від людського фактора, відсутність оперативного доступу до даних та складність прогнозування потреб.

2. Визначено ключові проблеми, пов'язані з неефективністю ручного обліку: відсутність аналітики, ймовірність втрати інформації, труднощі у своєчасному формуванні замовлень. Ці фактори підтвердили актуальність потреби в автоматизованому рішенні.

3. Досліджено сучасні підходи до автоматизації облікових процесів, розглянуто існуючі системи управління (CRM, табличні редактори тощо), виявлено їх переваги та обмеження в контексті потреб малого закладу громадського харчування.

4. Сформульовано функціональні та нефункціональні вимоги до програмного забезпечення, що охоплюють процеси обліку, аналізу, прогнозування та звітності. Особливу увагу приділено зручності інтерфейсу, швидкодії та адаптивності системи.

5. Розроблено програмне забезпечення *CoffeeTrack*, яке реалізує функції ведення обліку залишків, фіксації закупівель і списань, візуалізації споживання ресурсів, а також формування прогнозів майбутніх потреб на основі накопичених даних.

6. Проведено тестування створеного застосунку щодо відповідності вимогам. Результати підтвердили стабільність роботи системи, коректність розрахунків та зручність користування інтерфейсом як на стаціонарних, так і на мобільних пристроях.

ДОДАТОК Б. ЛІСТИНГИ ПРОГРАМНИХ МОДУЛІВ

```
public async Task<IActionResult> MaterialsUsage()
        {
        var today = DateTime.Today;
        var startDate = new DateTime(today.Year,
today.Month, 1).AddMonths(-4); // останні 5 місяців
        var months = Enumerable.Range(0, 5)
        .Select(i => startDate.AddMonths(i))
        .ToList();
        var labels = months
        .Select(m => m.ToString("MMMM", new
CultureInfo("uk-UA")))
        .ToList();
        var consumptions = await
_context.Consumptions
        .Include(c => c.Material)
        .Where(c => c.DateUsed >= startDate)
        .ToListAsync();
        var grouped = consumptions
        .GroupBy(c => new { c.Material.Name,
c.DateUsed.Year, c.DateUsed.Month })
        .ToList();
        var usageDict = new Dictionary<string,
List<int>>();
        foreach (var material in consumptions.Select(c
=> c.Material.Name).Distinct())
        {
        var usage = new List<int>();
        foreach (var m in months)
        {
        var sum = grouped
        .Where(g => g.Key.Name == material &&
g.Key.Year == m.Year && g.Key.Month == m.Month)
        .Sum(g => g.Sum(x => x.QuantityUsed));
        usage.Add(sum);
        }
        usageDict[material] = usage;
        }
        ViewBag.Labels = labels;
        ViewBag.Data = usageDict;
        return View();
        }
        public async Task<IActionResult>
Index(DateTime? from, DateTime? to)
        {
        var today = DateTime.Today;
        var start = from ?? new DateTime(today.Year,
today.Month, 1).AddMonths(-4);
        var end = to ?? today;
        var consumptions = await
_context.Consumptions
        .Include(c => c.Material)
        .Where(c => c.DateUsed >= start &&
c.DateUsed <= end)
        .ToListAsync();
        var grouped = consumptions
        .GroupBy(c => new { c.Material.Name,
c.DateUsed.Year, c.DateUsed.Month })
        .ToList();
        var labels = Enumerable.Range(0, 5)
        .Select(i =>
start.AddMonths(i).ToString("MMMM", new
CultureInfo("uk-UA")))
        .ToList();
        var forecastData = new
List<ForecastChartViewModel>();
        foreach (var material in consumptions.Select(c
=> c.Material.Name).Distinct())
        {
        var dataPoints = labels.Select(label =>
        grouped.Where(g =>
        g.Key.Name == material &&
        new DateTime(g.Key.Year, g.Key.Month,
1).ToString("MMMM", new CultureInfo("uk-UA")) ==
label)
        .Sum(g => g.Sum(x =>
x.QuantityUsed))).ToList();
        double forecast =
dataPoints.TakeLast(3).DefaultIfEmpty(0).Average();
        forecastData.Add(new
ForecastChartViewModel
        {
        Material = material,
        Labels = labels,
        Actual = dataPoints,
        Forecast = Math.Round(forecast, 2)
        });
        }
        ViewBag.From = start.ToString("yyyy-MM-
dd");
        ViewBag.To = end.ToString("yyyy-MM-dd");
        return View(forecastData);
        }
        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult>
Create([Bind("Id,MaterialId,DateUsed,QuantityUsed")]
Consumption consumption)
        {
        if (ModelState.IsValid)
        {
        var material = await
_context.Materials.FindAsync(consumption.MaterialId);
        if (material != null)
        {
        material.Quantity -=
consumption.QuantityUsed;
        }
        _context.Consumptions.Add(consumption);
        await _context.SaveChangesAsync();
        return RedirectToAction(nameof(Index));
```

```
        }
        ViewData["MaterialId"] = new
SelectList(_context.Materials, "Id", "Name",
consumption.MaterialId);
        return View(consumption);
        }
        public class ApplicationDbContext : DbContext
        {
        public
ApplicationDbContext(DbContextOptions<ApplicationD
bContext> options) : base(options) { }
        public DbSet<Material> Materials =>
Set<Material>();
        public DbSet<Consumption> Consumptions =>
Set<Consumption>();
        public DbSet<Purchase> Purchases =>
Set<Purchase>();
        }
        public class Purchase
        {
        public int Id { get; set; }
        public int MaterialId { get; set; }
        [ValidateNever]
        public Material Material { get; set; } = null!;

        public DateTime PurchaseDate { get; set; }
        public int QuantityPurchased { get; set; }
        }

        public class Purchase
        {
        public int Id { get; set; }
        public int MaterialId { get; set; }
        [ValidateNever]
        public Material Material { get; set; } = null!;
        public DateTime PurchaseDate { get; set; }
        public int QuantityPurchased { get; set; }
        }
        @model
WebApplication1.Models.Consumption
        @{
        ViewData["Title"] = "Списання";
        var materialSelectList =
ViewData["MaterialId"] as SelectList ?? new
SelectList(Enumerable.Empty<SelectListItem>());
        }

        <h2 class="mb-4">@ViewData["Title"]</h2>
        <div class="row">
        <div class="col-md-6">
        <form asp-action="Create" asp-
controller="Consumptions" method="post">
        <div class="mb-3">
        <label asp-for="MaterialId" class="form-
label">Матеріал</label>
        <select asp-for="MaterialId" class="form-
select" asp-items="materialSelectList"></select>
        </div>
        <div class="mb-3">
        <label asp-for="DateUsed" class="form-
label">Дата списання</label>
        <input asp-for="DateUsed" type="date"
class="form-control" />
        </div>
        <div class="mb-3">
        <label asp-for="QuantityUsed" class="form-
label">Кількість</label>
        <input asp-for="QuantityUsed" class="form-
control" />
        </div>
        <button type="submit" class="btn btn-
primary">Зберегти</button>
        <a asp-action="Index" class="btn btn-secondary
ms-2">Скасувати</a>
        </form>
        </div>
        </div>
        @model
WebApplication1.Models.Consumption
        @{
        ViewData["Title"] = "Видалення списання";
        }
        <h2 class="mb-4">@ViewData["Title"]</h2>
        <h4 class="text-danger">Ви впевнені, що
хочете видалити це списання?</h4>
        <div class="card mt-3">
        <div class="card-body">
        <dl class="row mb-0">
        <dt class="col-sm-4">Матеріал</dt>
        <dd class="col-sm-
8">@Model.Material.Name</dd>
        <dt class="col-sm-4">Дата списання</dt>
        <dd class="col-sm-
8">@Model.DateUsed.ToString("dd.MM.yyyy")</dd>
        <dt class="col-sm-4">Кількість списано</dt>
        <dd class="col-sm-
8">@Model.QuantityUsed</dd>
        </dl>
        </div>
        </div>
        <form asp-action="Delete" class="mt-4">
        <input type="hidden" asp-for="Id" />
        <button type="submit" class="btn btn-
danger">Так, видалити</button>
        <a asp-action="Index" class="btn btn-secondary
ms-2">Скасувати</a>
        </form>
        using Microsoft.EntityFrameworkCore;
        using webApplication1.Data;
        var builder =
WebApplication.CreateBuilder(args);
        // Підключення Razor + контролерів
        builder.Services.AddControllersWithViews();
        // Підключення EF Core
        builder.Services.AddDbContext<ApplicationDb
Context>(options =>
        options.UseSqlServer(builder.Configuration.Get
ConnectionString("DefaultConnection")));
        var app = builder.Build();
        // Міграції + початкові дані
        using (var scope = app.Services.CreateScope())
        {
        var services = scope.ServiceProvider;
```

```
        var context =
services.GetRequiredService<ApplicationDbContext>();
        context.Database.Migrate();
        SeedData.Initialize(context);
        }
        app.UseHttpsRedirection();
        app.UseStaticFiles();
        app.UseRouting();
        public class HomeController : Controller
        {
        private readonly ILogger<HomeController>
_logger;
        public
HomeController(ILogger<HomeController> logger)
        {
        _logger = logger;
        }
        public IActionResult Index() => View();
        public IActionResult Privacy() => View();
        app.UseAuthorization();
        // Роутинг
        app.MapControllerRoute(
        name: "default",
        pattern:
"{controller=Materials}/{action=Index}/{id?}");
        app.Run();

        public IActionResult Error() =>
        View(new ErrorViewModel { RequestId =
Activity.Current?.Id ?? HttpContext.TraceIdentifier });
        }
        public class ForecastChartViewModel
        {
        public string Material { get; set; } = "";
        public List<string> Labels { get; set; } = new();
        public List<double> Actual { get; set; } =
new();
        public double Forecast { get; set; }
```