

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «бакалавр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Веб-технології»)

на тему:

«Вебзастосунок для барменів та любителів коктейлів»

Виконав студент групи ВТ-21-1
НЕДАШКІВСЬКИЙ Василь Сергійович

Керівник роботи:
ФАНТ Микола Олександрович

Рецензент:
САВІЦЬКИЙ Роман Святославович

Житомир - 2025

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«ЗАТВЕРДЖУЮ»

Зав. кафедри інженерії
програмного забезпечення

Тетяна Вакалюк

«26» лютого 2025 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **НЕДАШКІВСЬКИЙ Василь Сергійович**

Керівник роботи: **ФАНТ Микола Олександрович**

Тема роботи: **«Вебзастосунок для барменів та любителів коктейлів»**,
затверджена Наказом закладу вищої освіти від **«14» лютого 2025 р., №61/с**

Вихідні дані для роботи: **Використання фреймворку Next.js, бібліотеки React та системи стилізації Tailwind CSS для створення вебзастосунку Bartender Hub - платформи для барменів та любителів коктейлів. Серверна частина реалізована засобами Java Spring Boot, з використанням бази даних MongoDB**

Консультанти з бакалаврської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Фант М.О.	27.02.2025р.	27.02.2025р.
2	Фант М.О.	27.02.2025р.	27.02.2025р.
3	Фант М.О.	27.02.2025р.	27.02.2025р.

РЕФЕРАТ

Випускна робота бакалавра представляє собою «Вебзастосунок для барменів та любителів коктейлів».

Пояснювальна записка до кваліфікаційної роботи складається з переліку умовних скорочень, вступу, трьох розділів, висновків, списку використаної літератури та додатків.

Текстова частина викладена на 69 сторінках друкованого тексту. Пояснювальна записка має 28 додатків. Список використаних джерел містить 36 найменувань і займає 4 сторінки. В роботі наведено 22 рисунки та 15 таблиць. Загальний обсяг роботи - 97 сторінок.

В роботі визначені головні завдання для розробки вебзастосунку Bartender Hub, проаналізовано аналоги продукту та методи розв'язання поставленої задачі, обґрунтовано вибір архітектури системи. Представлено алгоритм роботи програмного комплексу, структуру та взаємодію його компонентів та описана програмна реалізація системи з використанням сучасних веб-технологій. Розроблено функціональність для управління рецептами коктейлів, верифікації професійних барменів, модерації контенту та персоналізованих рекомендацій.

КЛЮЧОВІ СЛОВА: ВЕБЗАСТОСУНОК, БАРМЕН, КОКТЕЙЛЬ, SPRING BOOT, NEXT.JS, REACT, MONGODB, МОДЕРАЦІЯ, ВЕРИФІКАЦІЯ.

					ІПЗ.КР.Б-121-25-ПЗ			
Зм.	Арк.	№ докум.	Підпис	Дата	Вебзастосунок для барменів та любителів коктейлів Пояснювальна записка	Літ.	Арк.	Аркушів
Розроб.		Недашківський В.С.						
Керівник		Фант М.О					3	
н. контр.		Тичина О.П.				Житомирська політехніка, група ВТ-21-1		
Зав. каф.		Вакалюк Т.А.						

ABSTRACT

The bachelor's thesis is a «Web application for bartenders and cocktail lovers.»

The explanatory note to the qualification work consists of a list of conventional abbreviations, an introduction, three sections, conclusions, a list of references, and appendices.

The text part is set out on 69 pages of printed text. The explanatory note has 28 appendices. The list of sources used contains 36 names and takes up 4 pages. The work contains 22 figures and 15 tables. The total volume of the work is 97 pages.

The paper identifies the main tasks for the development of the Bartender Hub web application, analyzes product analogues and methods for solving the task, justifies the choice of system architecture. The algorithm of the software complex, the structure and interaction of its components are presented, and the software implementation of the system using modern web technologies is described. Functionality for managing cocktail recipes, verifying professional bartenders, content moderation, and personalized recommendations is developed.

KEYWORDS: WEB APPLICATION, BARTENDER, COCKTAIL, SPRING BOOT, NEXT.JS, REACT, MONGODB, MODERATION, VERIFICATION.

		Недашківський В.С.			ІІЗ.КР.Б-121-25-ІІЗ	Арк.
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		4

ЗМІСТ

РЕФЕРАТ.....	3
ABSTRACT	4
ПЕРЕЛІК УМОВНИХ ЗНАЧЕНЬ	6
ВСТУП.....	7
РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ РОЗРОБКИ ВЕБ-ДОДАТКУ ДЛЯ БАРМЕНІВ ТА ЛЮБИТЕЛІВ КОКТЕЙЛІВ	9
1.1. Постановка задачі.....	9
1.2. Аналіз аналогів програмного продукту	11
1.3. Вибір архітектури веб-додатку	14
1.4. Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення	16
Висновки до розділу 1.....	18
РОЗДІЛ 2. ПРОЕКТУВАННЯ ВЕБ-ДОДАТКУ ДЛЯ БАРМЕНІВ ТА ЛЮБИТЕЛІВ КОКТЕЙЛІВ	19
2.1. Визначення варіантів використання	19
2.2. Розробка бази даних.....	24
2.3. Визначення варіантів використання Проектування та реалізація алгоритмів роботи системи	33
2.4. Реалізація системи вебзастосунку для створення, оцінювання та обговорення коктейлів.....	38
Висновки до розділу 2.....	46
РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ЗАСТОСУНКОМ	47
3.1. Порядок встановлення та налаштування параметрів системи	47
3.2. Структура інтерфейсу мобільного застосунку.....	49
3.3. Тестування роботи програмного комплексу	58
Висновки до розділу 3.....	62
ВИСНОВОКИ	63
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	65
ДОДАТКИ.....	69

ПЕРЕЛІК УМОВНИХ ЗНАЧЕНЬ

API - прикладний програмний інтерфейс (Application Programming Interface)

CDN - мережа доставки контенту (Content Delivery Network)

CSS - каскадні таблиці стилів (Cascading Style Sheets)

HTML - мова розмітки гіпертексту (HyperText Markup Language)

HTTP - протокол передачі гіпертексту (HyperText Transfer Protocol)

HTTPS - захищений протокол передачі гіпертексту (HyperText Transfer Protocol Secure)

JS - JavaScript

JSON - нотація об'єктів JavaScript (JavaScript Object Notation)

JWT - веб-токен JSON (JSON Web Token)

LTS - довгострокова підтримка (Long Term Support)

MVC - модель-представлення-контролер (Model-View-Controller)

OAuth - відкрита авторизація (Open Authorization)

ПЗ - програмне забезпечення

RAM - оперативна пам'ять (Random Access Memory)

REST - передача репрезентативного стану (Representational State Transfer)

SSD - твердотільний накопичувач (Solid State Drive)

TS - TypeScript

UI - інтерфейс користувача (User Interface)

URL - уніфікований локатор ресурсу (Uniform Resource Locator)

		Недашківський В.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		6

ВСТУП

Актуальність теми. Індустрія барменства та змішування напоїв в Україні демонструє стійку тенденцію до модернізації. Ця зміна зумовлена зростанням інтересу до авторських коктейлів, розвитком барної культури та підвищеними вимогами до якості обслуговування. Паралельно відбувається інтенсивний розвиток веб-технологій, зокрема екосистеми React з впровадженням архітектури Server Components та конкурентного рендерингу, що забезпечує створення високопродуктивних інтерактивних застосунків з оптимізованим користувацьким досвідом.

Професійні бармени потребують ефективних цифрових інструментів для систематизації рецептурної бази знань, документування авторських коктейльних композицій та обміну експертним досвідом з колегами. Традиційні методи зберігання рецептів у паперових форматах або простих електронних документах не відповідають потребам сучасного барного середовища. Відсутність централізованих платформ для верифікації професійних кваліфікацій барменів та структурованого каталогізування коктейльних рецептів створює бар'єри для професійного розвитку галузі. Поціновувачі коктейлів також потребують доступу до достовірної інформації про рецепти, історію напоїв та можливості взаємодії з експертним співтовариством.

Мета роботи полягає у створенні комплексної веб-платформи Bartender Hub, яка поєднує функціональність спеціалізованої соціальної мережі для професійних барменів та ентузіастів коктейлів з технологіями персоналізованих рекомендацій та процесами верифікації професійних навичок.

Завдання роботи охоплюють ключові напрямки технічної реалізації:

1. Проектування та реалізація розподіленої архітектури системи з використанням Spring Boot для серверної частини з підтримкою реактивного програмування через Spring та Next.js з React для клієнтського інтерфейсу.
2. Розробка системи управління користувачами з багаторівневою автентифікацією.

		Недашківський В.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		7

3. Створення функціональності управління рецептами коктейлів з можливістю приватного зберігання та публічного розповсюдження через модерацію.

4. Впровадження системи оцінювання та коментування контенту з використанням реактивних компонентів на базі Radix UI та shadcn/ui.

5. Розробка модуля публікації новин та статей з інтегрованим редакторським інтерфейсом.

6. Реалізація алгоритмів пошуку та фільтрації контенту.

7. Інтеграція системи персоналізованих рекомендацій на основі аналізу користувацької поведінки та смакових профілів.

Об'єкт дослідження - процеси цифровізації професійної діяльності барменів та автоматизації взаємодії спільноти коктейльних ентузіастів засобами сучасних веб-технологій. Дослідження охоплює архітектурні патерни побудови масштабованих веб-застосунків з мікросервісною архітектурою, методики інтеграції алгоритмів рекомендаційних систем у клієнт-серверні застосунки та принципи проектування користувацького досвіду у вузькоспеціалізованих професійних соціальних мережах.

Предмет дослідження - створення високопродуктивної платформи взаємодії професійних барменів з інтегрованою системою рекомендацій за допомогою сучасного технологічного стеку, зокрема фреймворка Next.js та документо орієнтованої бази даних MongoDB.

Методи дослідження включають системний аналіз функціональних вимог до спеціалізованих професійних соціальних мереж, об'єктно-орієнтоване проектування архітектури з використанням патернів MVC та Repository через Spring Data MongoDB, методики проектування RESTful API.

Практичне значення результатів полягає у створенні повнофункціональної платформи, яку можна використовувати в реальному світі барменської спільноти. Архітектурні рішення та програмні модулі можуть бути використані для створення подібних платформ в інших професійних сферах.

		Недашківський В.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		8

РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ РОЗРОБКИ ВЕБ-ДОДАТКУ ДЛЯ БАРМЕНІВ ТА ЛЮБИТЕЛІВ КОКТЕЙЛІВ

1.1. Постановка задачі

Сучасна барменська індустрія переживає значний цифровий зсув. Зміни зумовлені зростаючою потребою фахівців у практичних інструментах для управління знаннями та спілкування з тематичною спільнотою.

Професійні бармени стикаються з кількома технічними труднощами у своїй щоденній роботі. Відсутність централізованих систем зберігання авторських рецептів призводить до втрати цінної інформації та ускладнює процес передачі досвіду між поколіннями фахівців. Традиційні способи зберігання рецептів коктейлів, такі як використання паперових записів або локальних електронних файлів, не забезпечують достатньої організації та доступу до інформації. Любителям коктейлів потрібні надійні джерела інформації про рецепти та можливість поспілкуватися з експертами у спільноті.

Питання перевірки професійної кваліфікації в барменській галузі стає дуже важливим. Існуючі системи сертифікації часто розрізнені. Вони не пропонують централізованого способу перевірки експертних знань. Це ускладнює побудову довіри між професіоналами та ускладнює пошук кваліфікованих фахівців у готельному бізнесі.

Технічний аналіз існуючих рішень виявляв значні недоліки у функціональності наявних платформ. Більшість додатків для приготування коктейлів орієнтовані лише на споживачів. Вони не враховують специфічні потреби професійних барменів. Відсутність систем модерації контенту призводить до поширення неправдивої інформації про рецепти та технології приготування напоїв. Недостатня інтеграція соціальних функцій обмежує можливості професійного спілкування та обміну досвідом.

Архітектурні обмеження наявних рішень також становлять серйозну проблему. Більшість існуючих платформ використовують монолітну

		Недашківський В.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		9

архітектуру. Такий підхід ускладнює масштабування та додавання нових функціональних модулів. Відсутність підтримки сучасних веб-стандартів та обмежені можливості персоналізації користувацького досвіду знижують ефективність використання цих систем професіоналами.

Детальний аналіз функціональних вимог професійної барменської спільноти показав необхідність створення комплексної системи, яка б поєднувала можливості управління рецептами з соціальними функціями та механізмами верифікації експертизи. Така система повинна забезпечувати приватне зберігання авторських рецептів з можливістю подальшої публікації через процес модерації. Реалізація багаторівневої системи ролей дозволить диференціювати доступ до функціональності відповідно до рівня експертизи користувача.

Технічні вимоги до створюваної системи включають підтримку високопродуктивної архітектури з можливістю горизонтального масштабування. Серверна частина повинна базуватися на Spring Boot framework з підтримкою реактивного програмування через WebFlux для забезпечення асинхронної обробки запитів. Використання документ-орієнтованої бази даних MongoDB забезпечить ефективну роботу з великими обсягами структурованих та неструктурованих даних рецептів.

Клієнтська частина системи повинна бути побудована з використанням сучасного технологічного стеку Next.js. Вона повинна підтримувати рендеринг на стороні сервера для підвищення продуктивності. Зв'язок React з TypeScript гарантує безпечну типізацію компонентів користувацького інтерфейсу. Система автентифікації повинна використовувати токени JWT і підтримувати OAuth2 для інтеграції із зовнішніми постачальниками ідентифікаційних даних.

Функціональна архітектура системи повинна включати модуль управління рецептами з підтримкою RecipeIngredient entities та складного workflow публікації через систему статусів. Компонент верифікації професійних барменів має забезпечувати VerificationRequest processing з можливістю модераторського

		Недашківський В.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Фант М.О.				10
Змн.	Арк.	№ докум.	Підпис	Дата		

розгляду. Система персоналізованих рекомендацій потребує імплементації алгоритмів аналізу користувацької поведінки на основі Rating та Favorite entities.

Інтеграція NewsArticle функціональності з EditorJS дозволить створення багатофункціонального контент-менеджменту для публікації тематичних матеріалів. Система Notification має забезпечувати інформування користувачів про зміни статусу контенту.

1.2. Аналіз аналогів програмного продукту

Щоб об'єктивно оцінити необхідність розробленої система, ми детально порівняли її з провідними платформами в галузі. В аналізі розглядаються як масові споживчі рішення, так і професійні платформи. Ці приклади показують різні способи вирішення проблем, з якими стикається барменська спільнота.

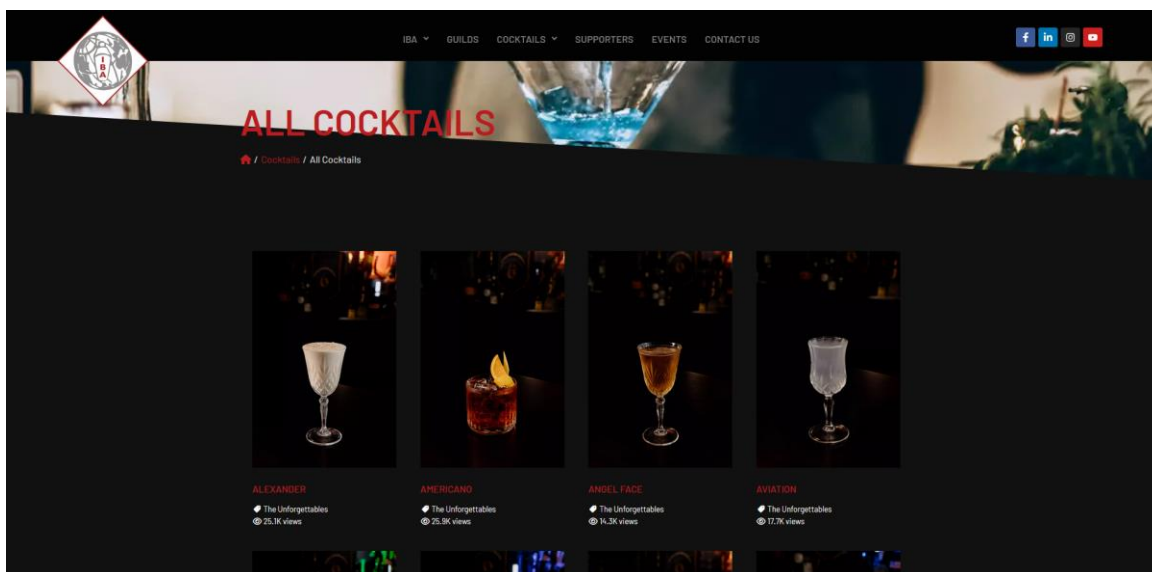


Рис. 1.1 - Головна сторінка IBA World

Платформа IBA World, представлена International Bartenders Association, являє собою еталонний ресурс для професійних барменів з колекцією офіційних рецептів коктейлів. Система забезпечує доступ до понад 90 затверджених рецептів з детальними інструкціями та історичною інформацією. Проте функціонал обмежується переважно інформаційним контентом без можливостей соціальної взаємодії чи власного контенту користувачів.

		Недашківський В.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Фант М.О.				11
Змн.	Арк.	№ докум.	Підпис	Дата		

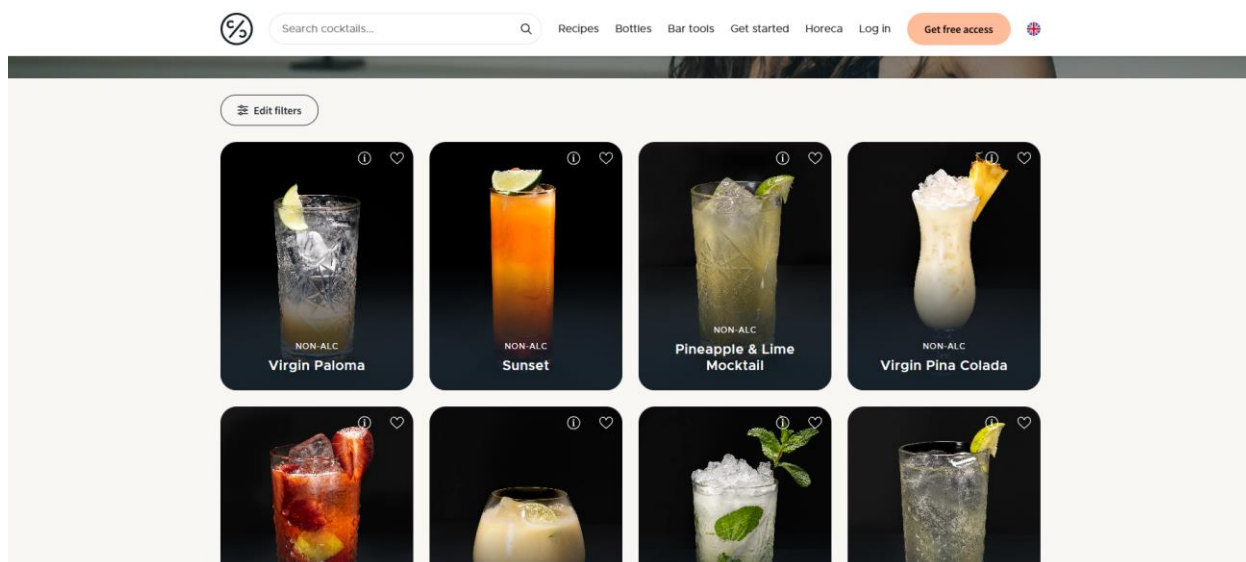


Рис. 1.2 - Інтерфейс додатку Cocktail Club

Cocktail Club демонструє сучасний підхід з елементами гейміфікації та соціальних функцій. Платформа інтегрує систему особистих колекцій, можливості оцінювання рецептів та базовий функціонал обміну досвідом між користувачами.

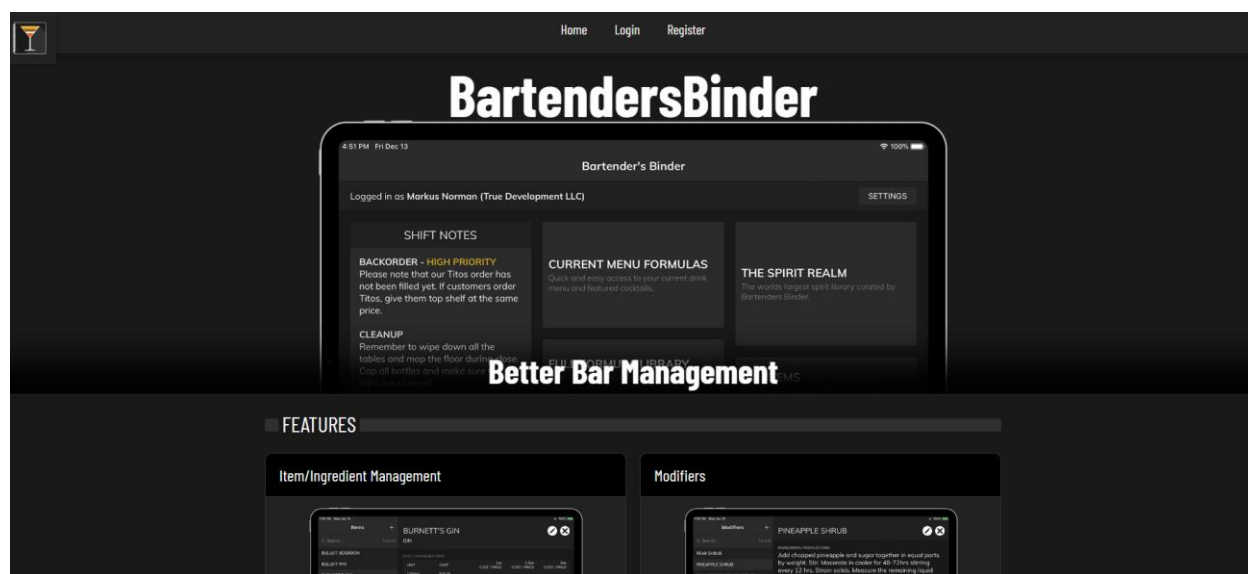


Рис. 1.3 - Інтерфейс платформи Bartenders Binder

Bartenders Binder представляє традиційний підхід до управління барними операціями з фокусом виключно на комерційних аспектах діяльності. Платформа забезпечує базовий функціонал віртуального інвентарю. Критичним недоліком є відсутність соціальних функцій, системи модерації контенту та можливостей професійного розвитку користувачів. Відсутність сучасного користувацького

		Недашківський В.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Фант М.О.				12
Змн.	Арк.	№ докум.	Підпис	Дата		

інтерфейсу знижує привабливість платформи для нового покоління професіоналів.

Таблиця 1.1

Порівняльний аналіз функціональних можливостей наявних платформ

Характеристика	IBA World	Cocktail Club	Bartenders Binder	Bartender Hub
Створення рецептів користувачами	-	-	+	+
Модерація контенту	-	-	-	+
Система ролей	-	+	-	+
Система верифікації	-	-	-	+
Коментарі та обговорення	-	+	-	+
Рекомендаційна система	-	+	-	+
Система сповіщень	-	+	-	+
Створення статей користувачами	-	-	-	+

Ключовою відмінністю розробленої платформи є комплексна система управління життєвим циклом контенту з чотирма статусами рецептів. Механізм DRAFT - SUBMITTED - PUBLISHED/REJECTED забезпечує професійну модерацію контенту з урахуванням експертних оцінок верифікованих барменів, що відсутнє у всіх проаналізованих конкурентах.

Система ролей з чотирма рівнями доступу (USER/BARTENDER/AUTHOR/ADMIN) створює унікальну екосистему, де досвідчені бармени можуть ділитися експертними знаннями з починаючими колегами. Процес верифікації включає подання документів про професійний досвід, що забезпечує достовірність експертних оцінок.

Інтеграція EditorJS для створення новинних статей розширює функціонал платформи за межі простого каталогу рецептів, створюючи повноцінне медіа-середовище для професійної спільноти. Система підтримує вбудовування зображень, відео, таблиць та інших мультимедійних елементів з автоматичною оптимізацією контенту.

Система сповіщень допомагає підтримувати активність користувачів і забезпечує швидкий обмін інформацією в професійній спільноті. Алгоритми

персоналізованих рекомендацій враховують вподобання користувачів на основі категорії, інгредієнтів та оцінок авторів.

Детальне дослідження конкурентного середовища виявило неоднорідність якості та функціональної повноти існуючих рішень.

Відсутність єдиних стандартів модерації рецептів коктейлів знижує якість контенту. Технологічна застарілість більшості рішень для платформ обмежує їхню здатність розвиватися та впроваджувати сучасні функції.

Виявлені системні недоліки обумовили необхідність створення інноваційної веб-платформи з комплексним підходом до вирішення галузевих проблем. Розроблена система передбачає реалізацію інтуїтивного користувацького інтерфейсу на базі React з Server Components. Професійна модерація контенту здійснюватиметься через багаторівневу систему ролей.

1.3. Вибір архітектури веб-додатку

Розробка комплексної платформи для професійної барменської спільноти потребує ретельного обґрунтування архітектурних рішень, здатних забезпечити масштабованість, надійність та ефективність системи. Аналіз функціональних вимог виявив необхідність реалізації багаторівневої архітектури з чітким розділенням відповідальностей між компонентами системи.

Класична монолітна архітектура була відкинута на етапі проектування через її обмеження для майбутнього масштабування. Монолітні системи створюють критичні точки відмови, ускладнюють незалежне розгортання компонентів. Враховуючи особливості соціальної платформи з високою взаємодією з користувачами, ми обрали розподілену архітектуру, яка чітко розділяє клієнтські та серверні компоненти.

Серверна архітектура побудована за принципами шарової організації з чітким розмежуванням між шарами презентації, бізнес-логіки та доступу до даних. Контролерний шар відповідає виключно за обробку HTTP-запитів та серіалізацію відповідей, делегуючи бізнес-операції до сервісного рівня. Такий

		Недашківський В.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Фант М.О.				14
Змн.	Арк.	№ докум.	Підпис	Дата		

підхід забезпечує інкапсуляцію доменної логіки та спрощує тестування критичних компонентів системи.

Паттерн Repository абстрагує операції з базою даних, створюючи уніфікований інтерфейс для роботи з доменними сутностями. Реалізація цього паттерна дозволяє ізолювати бізнес-логіку від специфіки конкретної системи управління базами даних та спрощує міграцію між різними постачальниками даних у майбутньому. У складних запитах до рецептів, які фільтрують за категоріями, тегами та рейтингами, використовуються операції агрегації. Такий підхід забезпечує хорошу продуктивність навіть для великих обсягів даних.

Сервісний шар інкапсулює складну бізнес-логіку, включаючи workflow обробки рецептів з переходами між статусами DRAFT-SUBMITTED-PUBLISHED/REJECTED. Окремий сервіс безпеки забезпечує централізовану авторизацію операцій відповідно до системи ролей користувачів. Модульна організація сервісів дозволяє незалежне тестування та рефакторинг компонентів без впливу на суміжні підсистеми.

Клієнтська архітектура базується на композитному патерні з ієрархічною організацією компонентів. Застосування Server-Side Rendering для початкового завантаження сторінок забезпечує оптимальну продуктивність та покращує індексацію контенту пошуковими системами. Подальша навігація здійснюється через Client-Side Rendering з динамічним завантаженням даних, що мінімізує навантаження на сервер та покращує користувацький досвід.

Архітектура маршрутизації реалізована через файлову систему з автоматичною генерацією роутів на основі структури директорій. Middleware-шар забезпечує централізовану обробку автентифікації та авторизації перед доступом до захищених ресурсів. Вкладені layout компоненти створюють модульну структуру інтерфейсу з можливістю повторного використання спільних елементів.

Модульна архітектура frontend компонентів забезпечує високий рівень переосмислення та тестованості. Розділення на презентаційні та контейнерні компоненти створює чітку межу між логікою відображення та управління

		Недашківський В.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Фант М.О.				15
Змн.	Арк.	№ докум.	Підпис	Дата		

даними. Система типізації через TypeScript забезпечує раннє виявлення помилок та покращує рефакторинг кодової бази.

Обрана архітектура дозволяє здійснювати горизонтальне масштабування для серверних компонентів та ефективному кешуванню статичних ресурсів. Модульна організація дозволяє поступово нарощувати функціональність без суттєвих змін базової структури. Розділення відповідальностей між шарами створює основу для майбутнього переходу до мікросервісної архітектури при необхідності.

1.4. Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення

Ефективність розробки та експлуатації складних веб-додатків значною мірою залежить від обґрунтованого вибору технологічного стеку та правильного визначення апаратних вимог. Сучасна екосистема розробки пропонує численні альтернативи для кожного компонента системи, що потребує детального аналізу переваг та обмежень конкретних рішень у контексті специфічних вимог проекту.

Серверна частина системи базується на Java 21 як основній мові програмування через її зрілість, продуктивність та широку екосистему бібліотек. Використання останньої LTS версії забезпечує доступ до сучасних мовних конструкцій, що значно спрощує написання доменної логіки. Virtual threads у Java 21 дозволяють ефективно обробляти тисячі одночасних запитів без значного споживання системних ресурсів.

Spring Boot обрано як фреймворк для побудови серверної архітектури завдяки його конвенційному підходу до конфігурації та широкій інтеграції з екосистемою Spring. Spring Data MongoDB забезпечує елегантну абстракцію для роботи з документною базою даних через repository pattern.

MongoDB вибрано як основну базу даних через її природну відповідність документно-орієнтованій моделі даних рецептів. Складні об'єкти рецептів з вкладеними інгредієнтами, тегами та метаданими ефективно зберігаються.

		Недашківський В.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		16

Клієнтська частина побудована на Next.js 15 як повнофункціональному React фреймворку з підтримкою App Router архітектури. Server-Side Rendering покращує початкову швидкість завантаження, що критично важливо для платформи з публічним контентом. App Router забезпечує файлову маршрутизацію з підтримкою вкладених layout компонентів та middleware обробки.

React 19 з новими Server Components дозволяє виконувати частину логіки на сервері, покращуючи продуктивність на пристроях з обмеженими ресурсами.

TypeScript забезпечує статичну типізацію JavaScript коду, що значно зменшує кількість runtime помилок. Strict режим конфігурації гарантує максимальну безпеку типів, особливо важливу при роботі з API responses та складними об'єктами даних.

EditorJS обрано для реалізації редактора новинних статей завдяки його модульній архітектурі. Блоковий підхід до контенту забезпечує структуровану розмітку з підтримкою різних типів медіа контенту.

Tailwind CSS дозволяє створювати оформлення без написання окремих CSS-файлів, використовуючи класи безпосередньо в HTML-коді. Завдяки вбудованому компілятору стилі створюються лише для тих елементів, які реально використовуються на сторінках, що зменшує обсяг фінального CSS-файлу й пришвидшує завантаження. Адаптивне відображення на різних пристроях забезпечується через вбудовані класи для медіазапитів, без потреби писати окремий адаптивний CSS.

UploadThing інтегровано для безпечного завантаження файлів з автоматичною оптимізацією зображень. Edge-функції забезпечують швидке завантаження медіа контенту незалежно від географічного розташування користувачів.

Розгортання системи вимагає серверного середовища з мінімум 4 ГБ оперативної пам'яті для комфортної роботи JVM та MongoDB процесів. Рекомендовані 8 ГБ RAM забезпечують ефективне кешування часто

		Недашківський В.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Фант М.О.				17
Змн.	Арк.	№ докум.	Підпис	Дата		

використаних даних. SSD storage обов'язковий для швидкого доступу до MongoDB колекцій з compound індексами.

Процесорні вимоги включають мультитядерну архітектуру з мінімум 2 ядрами для паралельної обробки HTTP запитів. Рекомендовані 4 ядра дозволяють ефективно розподіляти навантаження між веб-сервером, базою даних.

Мережеві вимоги передбачають стабільне інтернет-з'єднання з мінімальною латентністю. HTTPS сертифікація обов'язкова для забезпечення безпеки JWT токенів та захисту персональних даних користувачів.

Висновки до розділу 1

Результати аналізу обґрунтовують необхідність створення інноваційної веб-платформи з комплексним підходом до вирішення галузевих проблем. Система, що розробляється, дозволяє здійснювати професійну модерацію контенту. Вона має багаторівневу рольову систему, обробляє рецепти зі зміною статусу та включає спосіб перевірки професійних навичок. Додавання соціальних функцій, персоналізованих рекомендацій та системи сповіщень створить повноцінне медіа-середовище для професійної барменської спільноти. Обрані технологічні рішення та патерни проектування формують технічний фундамент для виконання завдань. Вони також підтримують майбутні покращення функціональності. Запуск цієї платформи сприятиме підвищенню кваліфікації барменів. Це дозволить їм обмінюватися досвідом та отримувати доступ до якісного контенту. Система забезпечить централізоване збереження та систематизацію рецептурних знань, що має важливе значення для розвитку барменської культури. Реалізація проєкту створить передумови для формування активної онлайн-спільноти професіоналів галузі з можливостями подальшого масштабування та комерціалізації.

		Недашківський В.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Фант М.О.				18
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 2. ПРОЕКТУВАННЯ ВЕБ-ДОДАТКУ ДЛЯ БАРМЕНІВ ТА ЛЮБИТЕЛІВ КОКТЕЙЛІВ

2.1. Визначення варіантів використання

Проектування веб-додатку для барменів та любителів коктейлів потребує детального аналізу функціональних можливостей системи через призму взаємодії різних категорій користувачів з платформою. Визначення варіантів використання дозволяє структурувати вимоги до системи та забезпечити повне покриття функціональності відповідно до специфічних потреб кожної ролі користувача.

Архітектура системи передбачає чотири основні ролі користувачів з диференційованими правами доступу: Guest (гість), User (користувач), Bartender (бармен), Author (автор) та Admin (адміністратор). Така багаторівнева структура дозволяє забезпечити гнучке управління контентом з урахуванням професійного досвіду та експертизи учасників платформи.

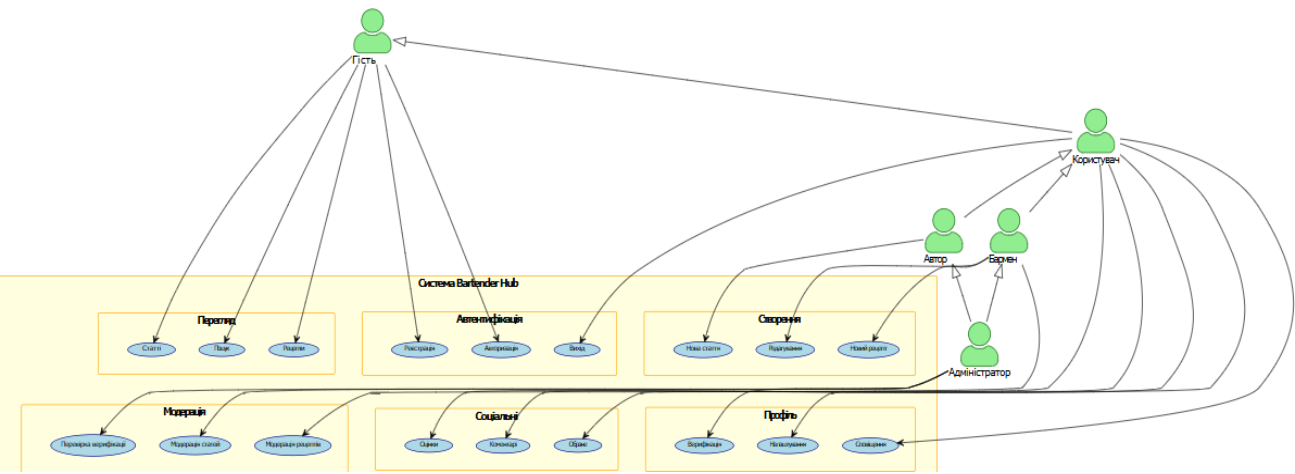


Рис. 2.1 - Діаграма варіантів використання

Функціональні вимоги для ролі Guest (Гість)

Неzareєстровані користувачі мають обмежений доступ до функціональності платформи, що спрямований на ознайомлення з можливостями системи та стимулювання реєстрації. Основні варіанти використання включають перегляд, пошук, фільтрацію опублікованого контенту без можливості взаємодії з ним.

Система автентифікації пропонує два варіанти реєстрації: через локальну форму з email-адресою та паролем або через інтеграцію Google OAuth.

Функціональні вимоги для ролі User (Користувач)

Базова роль зареєстрованого користувача розширює можливості взаємодії з платформою, додаючи соціальні функції та персоналізацію досвіду. Користувачі отримують доступ до повного спектру контенту з можливістю активної участі в спільноті.

Система обраного дозволяє зберігати цікаві рецепти та статті для швидкого доступу. Реалізація базується на сутності Favorite з підтримкою ParentType enum для розрізнення типів контенту.

Функціональність оцінювання контенту реалізована через п'ятибальну систему рейтингів. Компонент Rating зберігає оцінки користувачів з автоматичним розрахунком середнього рейтингу для кожного рецепту або статті.

Коментування забезпечує інтерактивність платформи через можливість залишати відгуки до рецептів та статей. Модель Comment підтримує ієрархічну структуру з полем parentCommentId для реалізації відповідей на коментарі.

Система підписок реалізована через сутність Follow, що дозволяє відстежувати активність обраних авторів та отримувати сповіщення про новий контент. Алгоритм рекомендацій враховує підписки користувача для формування персоналізованої стрічки.

Управління профілем включає редагування особистої інформації, завантаження аватара.

Система верифікації дозволяє подавати заявки на підвищення ролі до Bartender або Author через компонент VerificationRequest. Процес включає завантаження документів, що підтверджують професійний досвід, та опис кваліфікацій.

Функціональні вимоги для ролі Bartender (Бармен)

Роль бармена надає розширені можливості створення та модерації рецептурного контенту. Отримання цієї ролі потребує успішної верифікації професійних компетенцій.

		Недашківський В.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		20

Створення рецептів реалізовано через комплексну форму з підтримкою множинних інгредієнтів, детальних інструкцій та мультимедійного контенту. Модель Recipe включає поля для історичної інформації, категоризації та технічних характеристик напою. Система автоматично визначає наявність алкоголю на основі аналізу інгредієнтів.

Публікації передбачає чотири статуси: DRAFT для роботи над рецептом, SUBMITTED для модерації, PUBLISHED для публічного доступу та REJECTED у разі відхилення. Бармени можуть редагувати власні рецепти зі статусом DRAFT або REJECTED, але опубліковані рецепти потребують повторної модерації після змін.

Компонент RecipeSecurityService забезпечує контроль доступу до операцій з рецептами відповідно до ролі користувача та статусу контенту.

Функціональні вимоги для ролі Author (Автор)

Роль автора фокусується на створенні новинного та освітнього контенту через систему NewsArticle. Верифікація для отримання ролі Author потребує демонстрації навичок написання тематичних матеріалів та знання галузі.

Створення статей реалізовано через інтеграцію EditorJS, що забезпечує блокову структуру контенту з підтримкою різних типів медіа. Редактор дозволяє вбудовувати зображення, відео, таблиці та інші мультимедійні елементи з автоматичною оптимізацією контенту.

Модель NewsArticle використовує ті самі статуси, що й рецепти: DRAFT - SUBMITTED - PUBLISHED/REJECTED. Автори можуть створювати необмежену кількість чернеток та подавати їх на модерацію після завершення редагування.

Функціональні вимоги для ролі Administrator (Адміністратор)

Адміністративна роль забезпечує повний контроль над платформою з правами модерації всіх типів контенту та управління користувачами. Адміністратори отримують доступ до розширеної панелі модерації.

Модерація рецептів включає перегляд контенту зі статусом SUBMITTED через спеціалізований інтерфейс. Адміністратори можуть схвалювати або

		Недашківський В.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Фант М.О.				21
Змн.	Арк.	№ докум.	Підпис	Дата		

відхиляти рецепти з обов'язковим зазначенням коментарів для авторів. Система модерації статей функціонує аналогічно.

Управління системою верифікації реалізовано через компонент VerificationModerationDto, що агрегує інформацію про заявки з деталями користувачів. Адміністратори розглядають документи, перевіряють професійний досвід та приймають рішення про надання відповідних ролей.

Нефункціональні вимоги

Продуктивність системи забезпечується через compound індекси MongoDB для ефективних запитів фільтрації та пошуку. Цільовий час відповіді для основних операцій не повинен перевищувати 200 мілісекунд при навантаженні до 1000 одночасних користувачів.

Безпека реалізована через JWT автентифікацію з HTTP-only cookies та OAuth2 Resource Server конфігурацію. Всі API ендпоінти захищені middleware авторизації з перевіркою ролей користувачів через SecurityService компоненти.

Масштабованість забезпечується безстановою архітектурою серверного додатку та горизонтальним масштабуванням через MongoDB.

Надійність гарантується через комплексне логування операцій, моніторинг системних метрик та автоматичне резервне копіювання даних. Час відновлення після збоїв (RTO) не повинен перевищувати 15 хвилин для критичних компонентів.

Використання системи забезпечується адаптивним дизайном з підтримкою мобільних пристроїв та адаптивним інтерфейсом.

Архітектура системи

Реалізація визначених варіантів використання базується на сучасній двошаровій клієнт-серверній архітектурі з чітким розподілом відповідальності між компонентами. Серверна частина організована за принципами шарової архітектури, що забезпечує високий рівень модульності та можливості масштабування.

Бекенд система побудована на основі трьох основних архітектурних шарів. Шар контролерів (Controller Layer) відповідає за обробку HTTP-запитів та

		Недашківський В.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		22

маршрутизацію, реалізуючи RESTful API endpoints для кожної функціональної області. Контролери забезпечують точки входу для відповідних варіантів використання, здійснюючи валідацію вхідних даних та формування відповідей.

Бізнес-логіка системи концентрується в сервісному шарі (Service Layer), де класи типу сервісів інкапсулюють складні операції предметної області. Сервіси реалізують статусні обробки рецептів з автоматизованими переходами між станами, управління системою ролей з диференційованими правами доступу, алгоритми персоналізованих рекомендацій на основі аналізу поведінки користувачів.

Шар даних (Repository Layer) забезпечує абстракцію доступу до MongoDB через Spring Data repositories. Репозиторії надають типобезпечні методи роботи з документо-орієнтованою базою даних, використовуючи складні агрегаційні запити для аналітичних операцій.

Система безпеки реалізована через інтеграцію Spring Security з OAuth2 Resource Server, що забезпечує автентифікацію через JWT токени. Конфігурація SecurityConfig визначає гранульовані правила авторизації для кожного endpoint.

Фронтенд архітектура базується на Next.js App Router з компонентною організацією коду. Структура додатку відображає принципи сучасної React архітектури з використанням Server Components для статичного контенту та Client Components для інтерактивних елементів.

Комунікація між клієнтом та сервером організована через REST API з використанням Axios для HTTP клієнта.

Архітектурні рішення забезпечують горизонтальне масштабування, ефективне кешування на рівні бази даних через оптимізовані індекси, та мінімізацію затримки мережі. Така архітектура створює надійну основу для реалізації всіх визначених варіантів використання з високою продуктивністю та можливістю розширення.

		Недашківський В.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		23

2.2. Розробка бази даних

Проектування структури даних веб-додатку для барменів та любителів коктейлів базується на документо-орієнтованій MongoDB архітектурі, що забезпечує гнучкість схеми та ефективну обробку складних запитів. Документальна модель даних чітко показує ієрархічні зв'язки між сутностями. Це дозволяє зберігати різні види інформації в одній структурі.

База даних системи налічує тринадцять основних колекцій. Структура даних відображає логіку барменської платформи з урахуванням статусної модерації контенту, системи ролей користувачів та соціальних взаємодій.

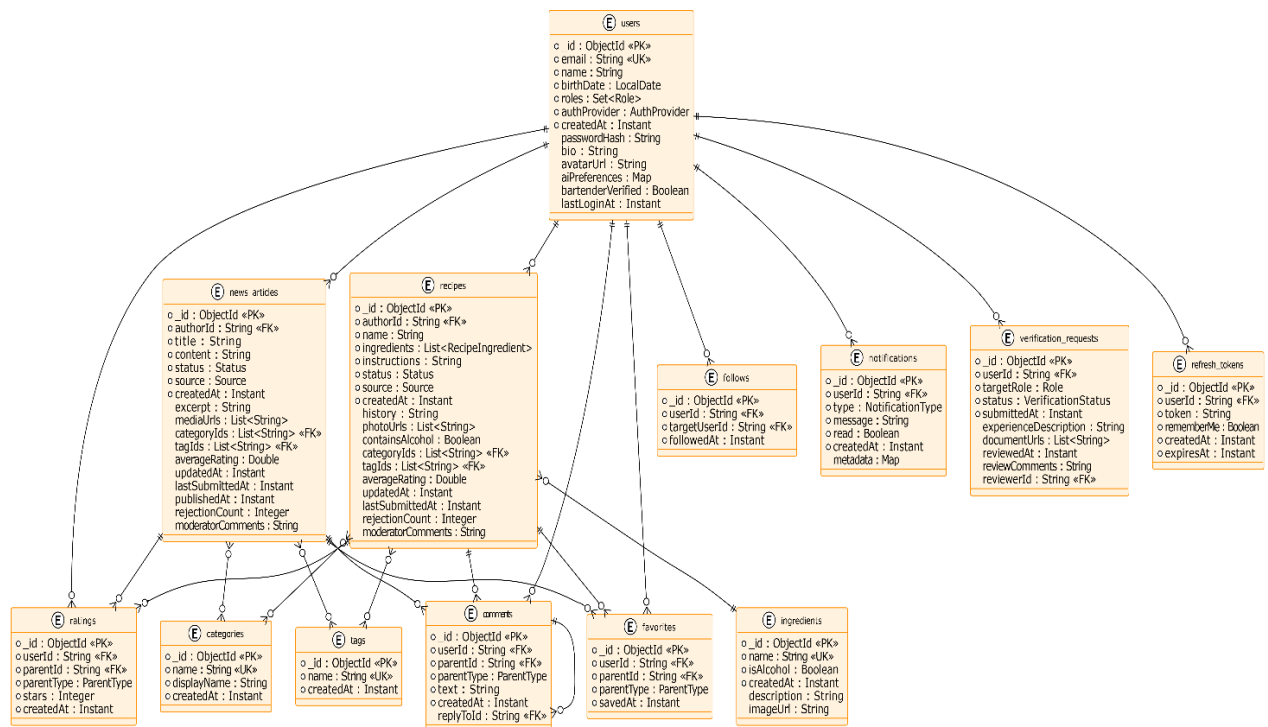


Рис. 2.2 - Діаграма бази даних

Діаграма відображає три основні групи колекцій: контентні (`recipes`, `news_articles`), довідникові (`categories`, `tags`, `ingredients`) та соціальні (`ratings`, `comments`, `favorites`, `follows`). Системні колекції (`notifications`, `verification_requests`, `refresh_tokens`) забезпечують допоміжну функціональність платформи.

Таблиця 2.1

Призначення основних колекцій бази даних

Назва колекції	Призначення
users	Зберігання інформації про користувачів
recipes	Управління рецептами коктейлів
news_articles	Статті та новини платформи
categories	Категорії для класифікації контенту
tags	Теги для маркування контенту
ingredients	Інгредієнти для рецептів
ratings	Оцінки користувачів
comments	Коментарі до контенту
favorites	Обраний контент користувачів
follows	Підписки користувачів
notifications	Системні сповіщення
verification_requests	Заявки на верифікацію ролей
refresh_tokens	Токени оновлення сесій

Центральна колекція платформи зберігає повну інформацію про користувачів з підтримкою множинних ролей та різних методів автентифікації. Структура документа враховує специфіку барменської галузі з полями для професійної верифікації та персоналізації досвіду.

Таблиця 2.2

Структура колекції users

Поле	Тип	Призначення
id	ObjectId	Унікальний ідентифікатор

email	String	Email адреса для автентифікації
passwordHash	String	Хеш пароля для локальної автентифікації
name	String	Відображуване ім'я користувача
bio	String	Біографічна інформація
birthDate	LocalDate	Дата народження для вікових обмежень
avatarUrl	String	URL аватара користувача
roles	Role	Множина ролей користувача
aiPreferences	Map	Налаштування рекомендацій
authProvider	AuthProvider	Метод автентифікації
bartenderVerified	Boolean	Статус професійної верифікації
createdAt	Instant	Час створення акаунту
lastLoginAt	Instant	Час останнього входу

Колекція users служить основою для системи авторизації та персоналізації контенту. Поле roles реалізує гнучку систему прав доступу, де користувач може мати декілька ролей одночасно.

Найбільша за обсягом колекція системи містить повну інформацію про рецепти коктейлів з багаторівневою категоризацією. Структура оптимізована для швидкого пошуку та фільтрації.

Таблиця 2.3

Структура колекції recipes

Поле	Тип	Призначення
id	ObjectId	Унікальний ідентифікатор рецепту

		Недашківський В.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		26

<i>authorId</i>	String	Ідентифікатор автора рецепту
<i>name</i>	String	Назва коктейлю
<i>ingredients</i>	RecipeIngredient	Список інгредієнтів з кількістю
<i>instructions</i>	String	Інструкції приготування
<i>history</i>	String	Історична інформація про коктейль
<i>photoUrls</i>	String	URL фотографій рецепту
<i>status</i>	Status	Статус модерації
<i>source</i>	Source	Джерело походження
<i>containsAlcohol</i>	Boolean	Наявність алкоголю
<i>categoryIds</i>	String	Ідентифікатори категорій
<i>tagIds</i>	String	Ідентифікатори тегів
<i>averageRating</i>	Double	Середній рейтинг рецепту
<i>createdAt</i>	Instant	Час створення
<i>updatedAt</i>	Instant	Час останнього оновлення
<i>lastSubmittedAt</i>	Instant	Час останньої подачі на модерацію
<i>rejectionCount</i>	Integer	Кількість відхилень
<i>moderatorComments</i>	String	Коментарі модератора

Колекція `recipes` реалізує складний статусний workflow з автоматичними обмеженнями. Структура `RecipeIngredient`, який наведений в додатку А, дозволяє зберігати детальну інформацію про інгредієнти включно з кількістю, одиницями виміру та додатковими нотатками.

Структура колекції news_articles

Поле	Тип	Призначення
<i>id</i>	ObjectId	Унікальний ідентифікатор статті
<i>authorId</i>	String	Ідентифікатор автора
<i>title</i>	String	Заголовок статті
<i>content</i>	String	Контент у форматі EditorJS
<i>excerpt</i>	String	Короткий опис статті
<i>mediaUrls</i>	String	URL медіа файлів
<i>status</i>	Status	Статус модерації
<i>source</i>	Source	Джерело контенту
<i>categoryIds</i>	String	Категорії статті
<i>tagIds</i>	String	Теги статті
<i>averageRating</i>	Double	Середній рейтинг
<i>createdAt</i>	Instant	Час створення
<i>updatedAt</i>	Instant	Час оновлення
<i>lastSubmittedAt</i>	Instant	Час подачі на модерацію
<i>publishedAt</i>	Instant	Час публікації
<i>rejectionCount</i>	Integer	Лічильник відхилень
<i>moderatorComments</i>	String	Коментарі модерації

Особливістю news_articles є інтеграція з EditorJS для створення структурованого контенту. Система індексації забезпечує ефективний повнотекстовий пошук як за заголовками, так і за змістом статей.

Структура колекції categories

Поле	Тип	Призначення
id	ObjectId	Унікальний ідентифікатор
name	String	Системна назва категорії
displayName	String	Відображувана назва
createdAt	Instant	Час створення

Колекція categories забезпечує ієрархічну організацію контенту з підтримкою як рецептів, так і статей. Категорії створюються адміністраторами та служать основою для фільтрації контенту в інтерфейсі користувача.

Таблиця 2.6

Структура колекції tags

Поле	Тип	Призначення
<i>id</i>	ObjectId	Унікальний ідентифікатор
name	String	Назва тегу
createdAt	Instant	Час створення

Колекція tags реалізує маркування для створення крос-категоріальних зв'язків між різними типами контенту.

Таблиця 2.7

Структура колекції ingredients

Поле	Тип	Призначення
id	ObjectId	Унікальний ідентифікатор
name	String	Назва інгредієнту
description	String	Опис інгредієнту
imageUrl	String	URL зображення

isAlcohol	Boolean	Алкогольний інгредієнт
createdAt	Instant	Час створення

Колекція ingredients служить основою для автоматичного визначення алкогольного статусу рецептів та побудови рекомендаційних алгоритмів. Опис інгредієнту надає користувачам додаткову інформацію про походження, смакові характеристики та способи використання компонентів коктейлів.

Таблиця 2.8

Структура колекції ratings

Поле	Тип	Призначення
id	ObjectId	Унікальний ідентифікатор
userId	String	Ідентифікатор користувача
parentId	String	Ідентифікатор оцінюваного об'єкту
parentType	ParentType	Тип об'єкту
stars	Integer	Кількість зірок
createdAt	Instant	Час оцінки

Колекція ratings забезпечує систему оцінювання контенту з підтримкою як рецептів, так і статей через ParentType discrimination. Composite unique constraint на userId+parentId+parentType запобігає множинному оцінюванню одного об'єкту користувачем. Система автоматично розраховує averageRating для батьківських об'єктів при додаванні нових оцінок.

Таблиця 2.9

Структура колекції comments

Поле	Тип	Призначення
id	ObjectId	Унікальний ідентифікатор
userId	String	Автор коментаря

parentId	String	Ідентифікатор батьківського об'єкту
parentType	ParentType	Тип батьківського об'єкту
text	String	Текст коментаря
replyToId	String	Ідентифікатор батьківського коментаря
createdAt	Instant	Час створення

Колекція comments реалізує вкладену систему коментарів з підтримкою глибини вкладеності через replyToId поле. ParentType дозволяє коментувати як рецепти, так і статті в єдиній структурі. Система автоматично відправляє сповіщення авторам батьківських об'єктів та коментарів при створенні відповідей.

Таблиця 2.10

Структура колекції favorites

Поле	Тип	Призначення
id	ObjectId	Унікальний ідентифікатор
userId	String	Користувач що зберіг
parentId	String	Збережений об'єкт
parentType	ParentType	Тип об'єкту
savedAt	Instant	Час збереження

Колекція favorites надає користувачам можливість створювати персональні закладки для швидкого доступу до улюбленого контенту. Система підтримує збереження як рецептів, так і статей через ParentType механізм. UserId забезпечує швидке отримання всього збереженого контенту користувача для відображення в профілі.

Таблиця 2.11

Структура колекції notifications

Поле	Тип	Призначення
<i>id</i>	ObjectId	Унікальний ідентифікатор
userId	String	Отримувач сповіщення
type	NotificationType	Тип сповіщення
message	String	Текст повідомлення
read	Boolean	Статус прочитання
createdAt	Instant	Час створення
metadata	Map	Додаткові дані

Колекція notifications забезпечує централізовану систему сповіщень з підтримкою дванадцяти типів подій від коментарів до результатів модерації. Metadata поле зберігає контекстну інформацію як ID об'єктів, URLs або додаткові параметри залежно від типу сповіщення.

Таблиця 2.12

Структура колекції verification_requests

Поле	Тип	Призначення
id	ObjectId	Унікальний ідентифікатор
userId	String	Заявник
targetRole	Role	Запитувана роль
experienceDescription	String	Опис досвіду
documentUrls	String	Документи підтвердження
status	VerificationStatus	Статус розгляду

submittedAt	Instant	Час подання
reviewedAt	Instant	Час розгляду
reviewComments	String	Коментарі адміністратора
reviewerId	String	Розглядач заявки

Колекція `verification_requests` реалізує повний життєвий цикл заявок на верифікацію з відстежуванням всіх етапів процесу. Система автоматично оновлює ролі користувачів при схваленні заявок та відправляє відповідні сповіщення.

2.3.Визначення варіантів використання Проектування та реалізація алгоритмів роботи системи

Розробка веб-платформи Bartender Hub вимагала створення складних алгоритмів для ефективної роботи системи управління контентом, індивідуальних рекомендацій та соціальних взаємодій.

Система включає кілька основних алгоритмічних частин: модуль індивідуальних рекомендацій на основі аналізу користувацьких уподобань, алгоритм перевірки контенту з кількома етапами схвалення, систему збирання та оновлення оціно, а також механізм швидкого пошуку.

Основна частина системи - персоналізований алгоритм рекомендацій. Він вивчає поведінку користувачів, дивлячись на їхні улюблені рецепти та рейтинги. Це допомагає йому створювати індивідуальні пропозиції контенту. Алгоритм працює за принципом аналізу схожості між користувачами та контентом.

Система рекомендацій починає з вивчення того, що подобається користувачам. Вона збирає дані про елементи, які користувач позначив як улюблені. Процес виявлення вподобань передбачає витягування тегів, категорій та інгредієнтів з обраного користувачем контенту. Це допомагає створити профіль його смаків.

Алгоритм побудови користувацьких уподобань реалізовано у спеціальному методі calculatePreferencesFromFavorites, як наведений у додатку А. Цей метод аналізує улюблені рецепти користувача та створює профіль вподобань на основі міток, інгредієнтів та категорій.

Створення рекомендацій відбувається через пошук схожих рецептів та статей, які використовують можливості бази даних MongoDB для швидкого знаходження подібного контенту. Система будує складні запити для пошуку контенту, що відповідає принаймні одній з користувацьких переваг.

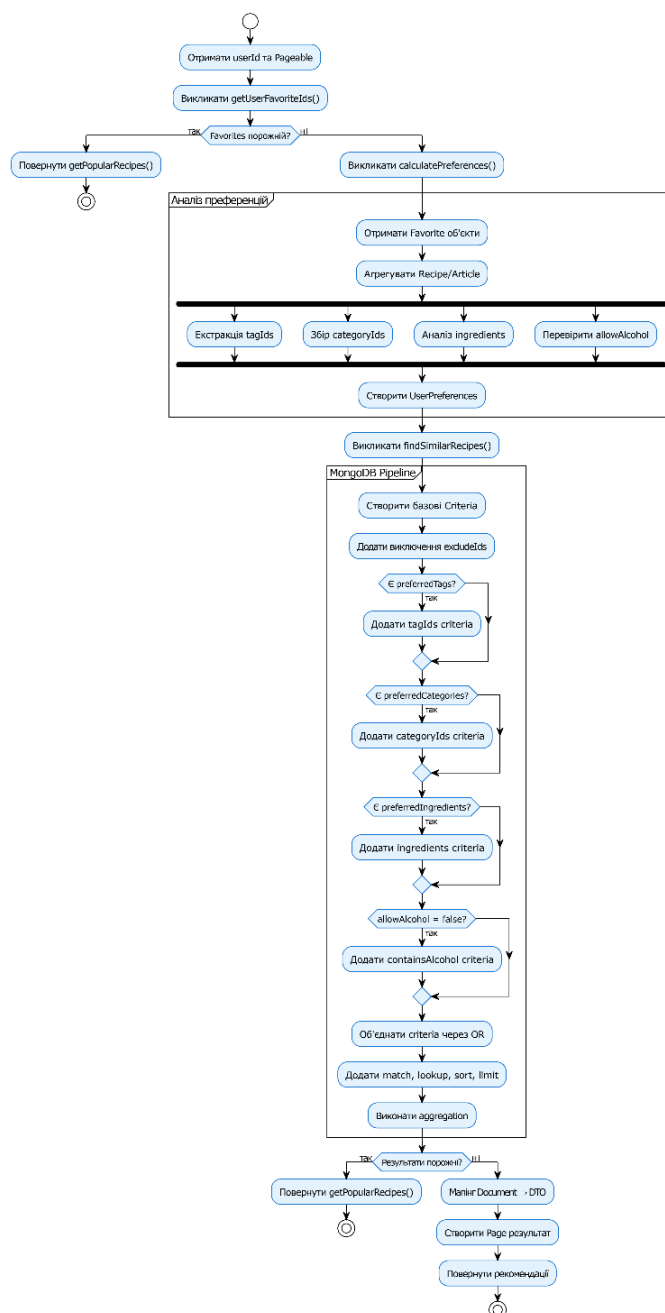


Рис. 2.3 - Діаграма алгоритму персоналізованих рекомендацій

Платформа використовує складну систему перевірки контенту з чотирма станами: чернетка, подано на розгляд, відхилено, опубліковано.

Процес перевірки починається зі стану чернетки, де автор може вільно редагувати свій контент. Коли контент подається на розгляд, редагування блокується і контент потрапляє у чергу до модераторів. Система автоматично запам'ятовує час подачі для відстеження обмежень повторної подачі після відхилення.

Реалізація системи перевірки контенту наведена у додатку А. Основні функції включають перевірку можливості подачі, можливості повторної подачі, саму подачу на публікацію та схвалення рецепту, які забезпечують повний цикл перевірки з автоматичними обмеженнями.

Алгоритм повторної подачі автоматично перевіряє можливість нової спроби через часові обмеження та кількість попередніх відхилень. Система дозволяє максимум три спроби подачі контенту на перевірку.

Безпека перевірки забезпечується через спеціальні сервіси безпеки, які контролюють права доступу на основі ролей користувачів. Модератори з правами адміністратора мають повні права на управління контентом, тоді як звичайні користувачі можуть змінювати лише власний контент у стані редагування.

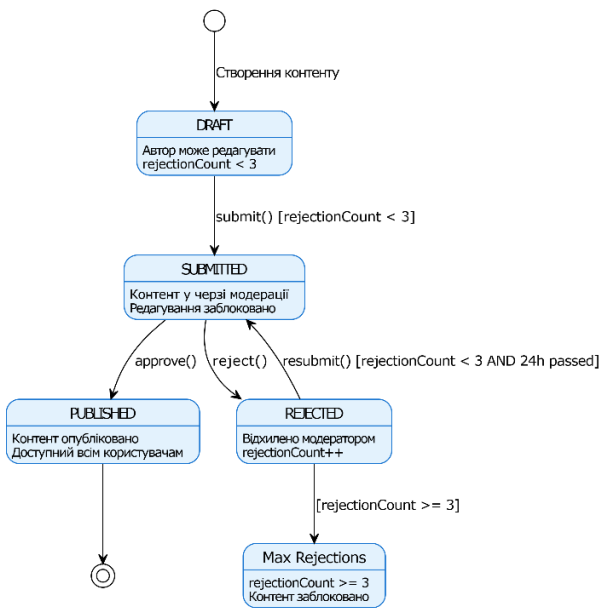


Рисунок 2.4 - Workflow модераторії контенту

Система оцінювання контенту працює на основі збирання оцінок у реальному часі з автоматичним оновленням середніх оцінок у відповідних рецептах та статтях. Алгоритм забезпечує правильність даних через захищені операції та оптимізоване зберігання.

Сервіс оцінок використовує унікальне обмеження через спеціальний індекс бази даних для запобігання багаторазовому оцінюванню одного контенту одним користувачем. Система підтримує як рецепти, так і статті через розрізнення типів контенту.

При створенні нової оцінки система автоматично запускає перерахунок середньої оцінки, який збирає всі наявні оцінки для відповідного рецепту чи статті та оновлює збережене поле середньої оцінки.

Алгоритм збирання оцінок представлено у додатку А. Система включає захищений метод оцінювання з автоматичним перерахунком середніх значень та метод оновлення середньої оцінки для зберігання рейтингових даних у відповідних документах.

Платформа використовує багаторівневу систему пошуку з підтримкою текстового пошуку, фільтрації за категоріями та складних комбінованих запитів. Архітектура пошуку базується на текстових індексах MongoDB з ваговим оцінюванням та комбінованих індексах для оптимізації операцій фільтрації.

Основний пошуковий алгоритм реалізований у методах пошуку опублікованих рецептів та статей через систему обробки MongoDB. Процес включає послідовні операції відбору за статусом, категоріями, мітками та алкогольним вмістом з подальшим текстовим пошуком.

Повна реалізація оптимізованого пошукового алгоритму наведена у додатку А. Метод побудови пошукового процесу демонструє семи етапну оптимізацію обробки MongoDB з комбінованими індексами.

Система підтримує динамічне сортування через параметри напрямку з резервними механізмами. Сортування за рейтингом використовує спеціальний комбінований індекс, сортування за датою використовує часові індекси з підтримкою як дати створення, так і дати публікації.

		Недашківський В.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		36

Платформа включає комплексну систему сповіщень з підтримкою 12 типів повідомлень, визначених у спеціальному переліку зазначеному у додаток А. Сервіс сповіщень використовує архітектуру на основі подій з автоматичним створенням сповіщень при зміні статусів контенту, нових коментарях, рішеннях модераторів про схвалення чи відхилення та соціальних взаємодіях. Кожен тип сповіщення має специфічні правила доставки та персоналізації.

Система автоматично створює сповіщення про схвалення та відхилення рецептів при зміні їх статусу, відповіді на коментарі при відповідях користувачів, нові рецепти від авторів, на яких підписаний користувач. Алгоритм включає розумне групування для об'єднання схожих сповіщень та обмеження частоти для запобігання спаму.

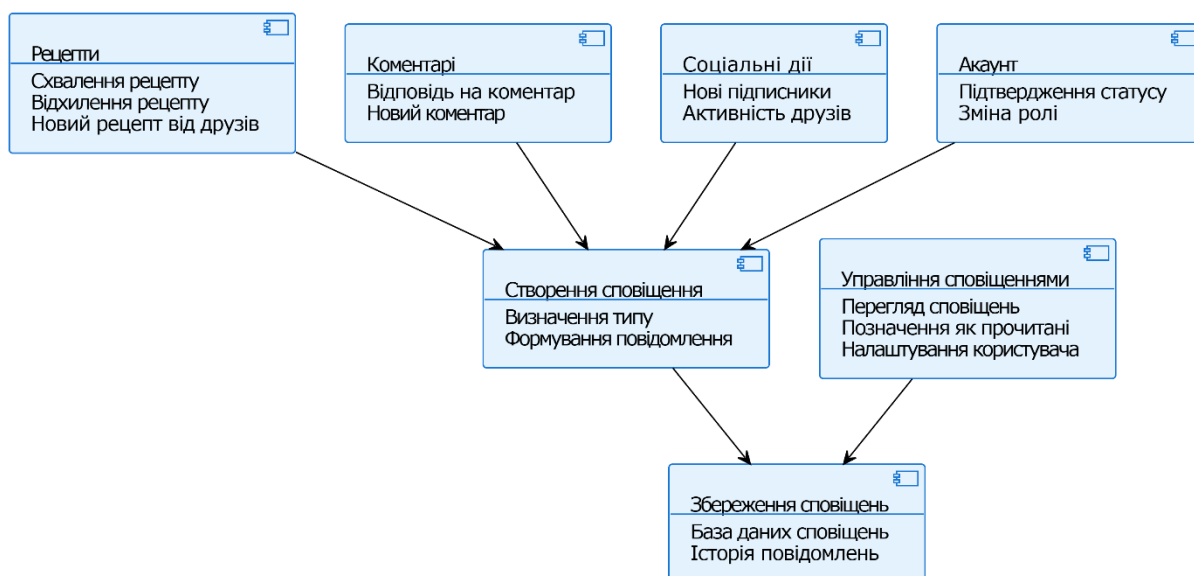


Рис. 2.5 - Архітектура системи нотифікацій

Платформа використовує багатоетапний процес підтвердження професійних барменів через систему заявок з підтримкою завантаження документів та ручного розгляду. Система забезпечує рівень довіри для професійної спільноти через підтверджені значки та підвищення ролей.

Процес підтвердження починається з подачі заявки користувачем через спеціальний контролер з обов'язковим завантаженням документів через сервіс зберігання файлів. Система автоматично створює заявку зі статусом очікування та додає до черги модераторів.

Модератори з правами адміністратора можуть схвалювати або відхиляти заявки з обов'язковими коментарями. При схваленні система автоматично оновлює поле підтвердження бармена у користувача та додає роль бармена до набору ролей. Система відправляє сповіщення з результатом розгляду.

Алгоритм підтвердження користувачів реалізовано через методи подачі та схвалення заявок (див. додаток А.6). Система включає автоматичні перевірки існуючих заявок, підвищення ролей та інтеграцію зі службою сповіщень для повідомлення про результати підтвердження.

2.4.Реалізація системи вебзастосунку для створення, оцінювання та обговорення коктейлів

Реалізація веб-платформи Bartender Hub здійснена на основі розподіленої архітектури з роздільними клієнтським та серверним компонентами. Система включає фронтенд на React з Next.js, бекенд на Spring Boot та MongoDB як основне сховище даних.

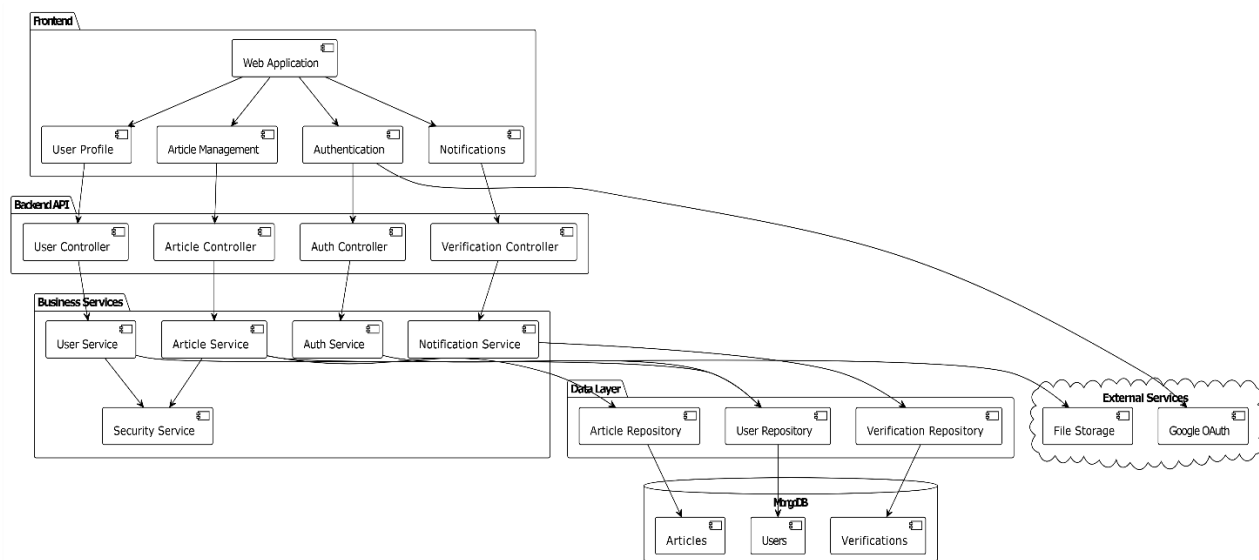


Рис. 2.6 - Діаграма компонентів системи

одуль автентифікації забезпечує безпечний вхід користувачів та управління їх сесіями. Клієнтська частина реалізована через React Context API з підтримкою двох способів автентифікації: традиційного через email/пароль та Google OAuth.

Компонент AuthContext забезпечує глобальний стан автентифікації для всього додатку:

```
javascript
const fetchUserProfile = useCallback(async () => {
  try {
    setLoading(true);
    const { data } = await getProfile();
    setUser(data);
    setLoading(false);
  } catch (error: unknown) {
    const axiosError = error as AxiosError;
    if (axiosError.response?.status === 401) {
      setUser(null);
    }
    setLoading(false);
  }
}, []);
```

Лістинг 2.1 - Автоматичне відновлення сесії

Цей код демонструє автоматичне відновлення сесії користувача при завантаженні додатку. Функція `fetchUserProfile` виконується асинхронно та встановлює стан завантаження під час запиту. Ключовою особливістю є обробка 401 помилки, яка сигналізує про недійсний або прострочений токен доступу. У такому випадку користувач автоматично виходить із системи шляхом встановлення `setUser(null)`, що запобігає несанкціонованому доступу до захищених ресурсів.

Серверна частина автентифікації включає валідацію та Google OAuth інтеграцію:

```
java
if (userService.existsByEmail(req.getEmail())) {
  return ResponseEntity.status(HttpStatus.CONFLICT)
    .body("Користувач з таким email вже існує");
}

User u = new User();
u.setEmail(req.getEmail().trim().toLowerCase());
u.setPasswordHash(passwordEncoder.encode(req.getPassword()));
u.setAuthProvider(AuthProvider.LOCAL);
```

		Недашківський В.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		39

```

u.setRoles(Set.of(Role.USER));
User saved = userService.create(u);
issueTokens(saved, res);

```

Лістинг 2.2 - Реєстрація користувача

Процес реєстрації користувача демонструє багаторівневий підхід до безпеки. Спочатку система перевіряє унікальність email адреси, запобігаючи створенню дублікатів акаунтів. Email автоматично нормалізується через `trim().toLowerCase()`, що усуває проблеми з регістром та зайвими пробілами. Пароль хешується за допомогою BCrypt алгоритму через `passwordEncoder.encode()`, забезпечуючи необоротне шифрування. Кожен новий користувач отримує базову роль USER та провайдер автентифікації LOCAL, або GOOGLE. Після успішного створення акаунта функція `issueTokens()` генерує JWT токени для автоматичного входу.

Система управління рецептами включає створення, модерацію та публікацію рецептів коктейлів. Сервіс автоматично визначає наявність алкоголю та управляє станами модерації:

```

java
recipe.setAuthorId(authorId);
recipe.setCreatedAt(Instant.now());
recipe.setAverageRating(0.0);
recipe.setStatus(Status.DRAFT);
recipe.setContainsAlcohol(detectAlcohol(recipe));

List<String> errors = validateRecipeContent(recipe);
if (!errors.isEmpty()) {
    throw new IllegalArgumentException("Recipe validation failed");
}

```

Лістинг 2.3 - Створення рецепта

Створення рецепта розпочинається з встановлення базових метаданих. Автор рецепта зберігається через `setAuthorId()`, що дозволяє системі контролювати права доступу. Часова мітка `Instant.now()` забезпечує точне відстеження моменту створення. Початковий рейтинг встановлюється на 0.0,

		Недашківський В.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		40

оскільки нові рецепти ще не оцінювались користувачами. Статус DRAFT дозволяє авторам редагувати контент без публікації.

Особливо важливою є функція `detectAlcohol()`, яка автоматично аналізує інгредієнти рецепта та визначає наявність алкогольних компонентів. Це критично для дотримання правильної категоризації контенту. Валідація контенту через `validateRecipeContent()` перевіряє обов'язкові поля, обмеження довжини та структурну цілісність даних перед збереженням.

Модерація рецептів включає автоматичне надсилання сповіщень:

```
java
recipe.setStatus(Status.PUBLISHED);
recipe.setModeratorComments(comments);
Recipe saved = recipeRepo.save(recipe);

notificationService.sendRecipeApprovedNotification(
    author.getId(), recipe.getName(), recipe.getId());
notificationService.sendNewRecipeNotification(saved, author);
```

Лістинг 2.4 - Затвердження рецепта

Процес схвалення рецепта модератором включає зміну статусу на PUBLISHED, що робить контент доступним для всіх користувачів. Коментарі модератора зберігаються для зворотного зв'язку з автором. Після збереження змін система автоматично надсилає два типи сповіщень: персональне повідомлення автору про схвалення та загальні сповіщення підписникам про новий опублікований контент. Це забезпечує швидке розповсюдження інформації та підтримує активність спільноти.

Модуль оцінювання забезпечує валідацію оцінок та автоматичне оновлення середніх рейтингів:

```
java
if (value < 1 || value > 5) {
    throw new IllegalArgumentException("Rating must be between 1 and 5");
}

Optional<Rating> existingRating = ratingRepo.findByUserIdAndRecipeId(userId, recipeId);
if (existingRating.isPresent()) {
    rating = existingRating.get();
```

		Недашківський В.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Фант М.О.				41
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        rating.setValue(value);
        rating.setUpdatedAt(Instant.now());
    } else {
        rating = new Rating();
        rating.setUserId(userId);
        rating.setRecipeId(recipeId);
        rating.setValue(value);
    }

    Rating saved = ratingRepo.save(rating);
    recipeService.updateAverageRating(recipeId);

```

Лістинг 2.5 - Встановлення оцінки

Система оцінювання використовує п'ятибальну шкалу з жорсткою валідацією діапазону значень. Перевірка `value < 1 || value > 5` запобігає некоректним оцінкам, які могли б спотворити статистику. Ключовою особливістю є запобігання множинним оцінкам від одного користувача - система шукає існуючу оцінку через `findByUserIdAndRecipeId()`. Якщо оцінка вже існує, вона оновлюється замість створення нової, що запобігає маніпуляціям рейтингом.

Автоматичне оновлення середнього рейтингу через `updateAverageRating()` відбувається асинхронно після кожної зміни оцінки. Це забезпечує актуальність рейтингів без блокування користувацького інтерфейсу. Часова мітка `setUpdatedAt()` дозволяє відстежувати останні зміни оцінок для аналітики.

Інтелектуальна система сповіщень

Система сповіщень підтримує дванадцять типів повідомлень з розумним групуванням:

```

java
public enum NotificationType {
    COMMENT_REPLY, RECIPE_APPROVED, RECIPE_REJECTED,
    ARTICLE_APPROVED, ARTICLE_REJECTED, VERIFICATION_RESULT,
    NEW_RECIPE_BY_FOLLOWED, NEW_ARTICLE_BY_FOLLOWED,
    NEW_RECIPE_SUBMITTED, NEW_ARTICLE_SUBMITTED,
    NEW_VERIFICATION_REQUEST, GENERIC
}

```

Лістинг 2.6 - Типи сповіщень

		Недашківський В.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Фант М.О.				42
Змн.	Арк.	№ докум.	Підпис	Дата		

Enum типів сповіщень покриває всі важливі події в життєвому циклі платформи. Персональні сповіщення як COMMENT_REPLY та RECIPE_APPROVED надсилаються конкретним користувачам, тоді як NEW_RECIPE_SUBMITTED призначені для адміністраторів. Соціальні сповіщення NEW_RECIPE_BY_FOLLOWED підтримують систему підписок. Тип GENERIC служить для нестандартних повідомлень, забезпечуючи гнучкість системи.

Розумне групування запобігає спаму сповіщень для адміністраторів:

```
java
if (existingNotifications.isEmpty()) {
    notification.setMessage("Новий рецепт подано на модерацию");
    notification.setMetadata(Map.of("count", 1));
    send(notification);
} else {
    Notification existing = existingNotifications.get(0);
    int currentCount = (Integer) metadata.getOrDefault("count", 1);

    existing.setMessage(String.format("Нові рецепти подано на модерацию (%d)",
        currentCount + 1));
    existing.setMetadata(Map.of("count", currentCount + 1));
    repo.save(existing);
}
```

Лістинг 2.7 - Групування сповіщень

Алгоритм групування сповіщень демонструє продуманий підхід до користувацького досвіду. Замість створення окремого сповіщення для кожного нового рецепта на модерацию, система перевіряє наявність існуючих непрочитаних сповіщень того ж типу. Для першого рецепта створюється нове сповіщення з лічильником 1. Наступні рецепти оновлюють існуюче сповіщення, збільшуючи лічильник та змінюючи текст повідомлення. Метадані зберігають додаткову інформацію як лічильник, дозволяючи гнучко управляти груповими сповіщеннями. Це значно зменшує навантаження на адміністраторів при високій активності користувачів.

Система верифікації дозволяє підвищення ролей користувачів з автоматичним оновленням прав:

		Недашківський В.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		43

```

java
Set<Role> userRoles = user.getRoles();
userRoles.add(request.getTargetRole());
user.setRoles(userRoles);

if (request.getTargetRole() == Role.BARTENDER) {
    user.setBartenderVerified(true);
}

userService.update(user);
request.setStatus(VerificationStatus.APPROVED);
request.setReviewedAt(Instant.now());

notificationService.sendVerificationResultNotification(
    user.getId(), true, request.getTargetRole(), null);

```

Лістинг 2.8 - Затвердження верифікації

Процес схвалення верифікації демонструє адитивний підхід до ролей - нові ролі додаються до існуючих замість заміни. Це дозволяє користувачам накопичувати права по мірі прогресу в системі.

Оновлення статусу заявки на APPROVED та встановлення часової мітки reviewedAt забезпечують аудит процесу верифікації. Автоматичне сповіщення користувача про результат через sendVerificationResultNotification() підтримує прозорість процесу та покращує користувацький досвід.

Система рекомендацій аналізує вподобання користувачів та використовує MongoDB агрегаційні пайплайни:

```

java
if (!preferences.preferredTags().isEmpty()) {
    preferencesCriteria.add(Criteria.where("tagIds").in(preferences.preferredTags()));
}

if (!preferences.preferredCategories().isEmpty()) {
    preferencesCriteria.add(Criteria.where("categoryIds").in(preferences.preferredCategories()));
}

if (!preferences.allowAlcohol()) {
    criteria.and("containsAlcohol").ne(true);
}

```

		Недашківський В.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Фант М.О.				44
Змн.	Арк.	№ докум.	Підпис	Дата		

```
pipeline.add(Aggregation.match(criteria));
pipeline.add(Aggregation.sort(Sort.Direction.DESC, "averageRating", "createdAt"));
```

Лістинг 2.9 - Побудова рекомендацій

Алгоритм рекомендацій будує критерії пошуку на основі аналізу поведінкових патернів користувача. Система витягує теги та категорії з улюблених рецептів користувача, формуючи профіль вподобань. Використання `in` операторів дозволяє знаходити контент, що містить будь-який з цікавих користувачу тегів або категорій.

Особливо важливою є перевірка `allowAlcohol()`, яка автоматично фільтрує алкогольні рецепти для неповнолітніх користувачів. Система визначає вік на основі дати народження та дотримується законодавчих вимог щодо вікових обмежень. Сортування за рейтингом та датою створення забезпечує пріоритетність якісного та свіжого контенту.

MongoDB агрегаційні пайплайни виконують складні запити на рівні бази даних, що значно покращує продуктивність порівняно з обробкою даних в додатку. Операція `Aggregation.match()` фільтрує документи, а `Aggregation.sort()` впорядковує результати ефективно використовуючи індекси.

Реалізація системи демонструє ефективне поєднання сучасних веб-технологій з продуманою архітектурою. Використання React з Next.js для фронтенду забезпечує швидкий та інтуїтивний користувацький інтерфейс завдяки серверному рендерингу та оптимізації продуктивності. Spring Boot з MongoDB створює потужну та масштабовану серверну платформу з автоматичною конфігурацією та багатими можливостями інтеграції.

Модульна організація коду дозволяє незалежно розвивати різні функціональні області системи. Використання спеціалізованих сервісів безпеки як `RecipeSecurityService` забезпечує централізований контроль доступу та запобігає дублюванню логіки авторизації.

		Недашківський В.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		45

Висновки до розділу 2

Проектування веб-додатку для барменів та любителів коктейлів передбачало вирішення комплексу технічних завдань, пов'язаних з оптимізацією бази даних, алгоритмічними рішеннями та архітектурною реалізацією системи. Розроблена платформа демонструє ефективне поєднання сучасних технологій зберігання даних з алгоритмами персоналізації контенту.

Структура MongoDB бази даних з тринадцятьма колекціями забезпечує гнучкість документо-орієнтованого підходу при збереженні високої продуктивності завдяки compound індексам та текстовим пошуковим механізмам. Центральна роль колекції users у схемі даних створює логічну основу для інтеграції всіх функціональних модулів платформи.

Алгоритмічна частина системи базується на персоналізованих рекомендаціях, що аналізують користувацькі вподобання через улюблені рецепти та створюють індивідуальні профілі смаків. Механізм збирання оцінок у реальному часі підтримує актуальність рейтингової інформації завдяки транзакційним операціям та денормалізованому зберіганню середніх значень.

Пошукова система поєднує текстові індекси з ваговим оцінюванням та багаторівневу фільтрацію за категоріями, тегами та алкогольним вмістом.

Реалізація веб-застосунку через модульну архітектуру з роздільними функціональними компонентами забезпечує масштабованість та підтримуваність системи. Інтеграція React компонентів з Spring Boot сервісами через RESTful API створює ефективний канал обміну даними між клієнтською та серверною частинами.

Результати проектування демонструють можливість створення складних веб-платформ з використанням сучасного технологічного стеку. Розроблена архітектура забезпечує основу для подальшого розвитку функціональності та інтеграції додаткових можливостей персоналізації користувацького досвіду в галузі барменського мистецтва.

		Недашківський В.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		46

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ЗАСТОСУНКОМ

3.1. Порядок встановлення та налаштування параметрів системи

Експлуатація веб-платформи Bartender Hub вимагає підготовки програмно-апаратного середовища з урахуванням специфіки розподіленої архітектури та технологічних залежностей системи. Мінімальні системні вимоги визначаються потребами виконавчого середовища для клієнтської та серверної частин додатку.

Клієнтський компонент функціонує у веб-браузері користувача та потребує сучасного браузерного двигуна з підтримкою стандартів ECMAScript 2017 та новітніх веб-технологій.

Мінімальні вимоги до робочої станції користувача включають операційну систему Windows 10 (або новіша), macOS 10.15 Catalina або Ubuntu 20.04 LTS з встановленою підтримкою сучасних веб-браузерів.

Апаратні вимоги передбачають наявність мінімум 4 ГБ оперативної пам'яті для комфортної роботи з інтерактивними компонентами та Editor.js для створення статей. Стабільне підключення до мережі Інтернет зі швидкістю не менше 5 Мбіт/с забезпечує належну швидкість завантаження медіа контенту через UploadThing та взаємодії з REST API сервісами.

Серверний компонент базується на Java 21 LTS та Spring Boot 3.4.4, що визначає мінімальні вимоги до хостингового середовища. Виконавче середовище повинно підтримувати OpenJDK 21 або новішу версію з встановленим Maven 3.9.9 для збирання та управління залежностями проекту.

Операційна система сервера може бути Linux Ubuntu 22.04 LTS або Windows Server 2019 з підтримкою контейнеризації. Рекомендована конфігурація включає мінімум 8 ГБ оперативної пам'яті для ефективної роботи Spring Boot застосунку та обробки паралельних запитів користувачів.

MongoDB версії 6.0 або новішої служить основним сховищем даних і потребує окремого сервера або хмарного сервісу з підтримкою compound індексів та текстового пошуку. Рекомендований обсяг дискового простору для

		Недашківський В.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Фант М.О.				47
Змн.	Арк.	№ докум.	Підпис	Дата		

бази даних становить мінімум 100 ГБ з урахуванням зростання контенту та медіа файлів.

Платформа інтегрується з декількома зовнішніми службами, що потребують попереднього налаштування API ключів та конфігураційних параметрів. Google OAuth 2.0 забезпечує альтернативний метод автентифікації через Google Cloud Console з налаштованими redirect URI для production та development середовищ.

UploadThing сервіс обробляє завантаження файлів користувачів і потребує реєстрації API ключів та налаштування storage policies для контролю типів та розмірів файлів.

Конфігурація відбувається через environment variables, визначені у файлі application.properties серверної частини. Критично важливі параметри включають MONGO_URI для підключення до бази даних, JWT_SECRET для підпису токенів автентифікації та GOOGLE_CLIENT_ID для OAuth інтеграції.

Розгортання веб-платформи відбувається за принципом роздільного хостингу компонентів з централізованою базою даних та зовнішніми інтеграціями. Архітектурна схема демонструє взаємодію між користувацькими пристроями, веб-серверами та backend сервісами через стандартизовані протоколи.

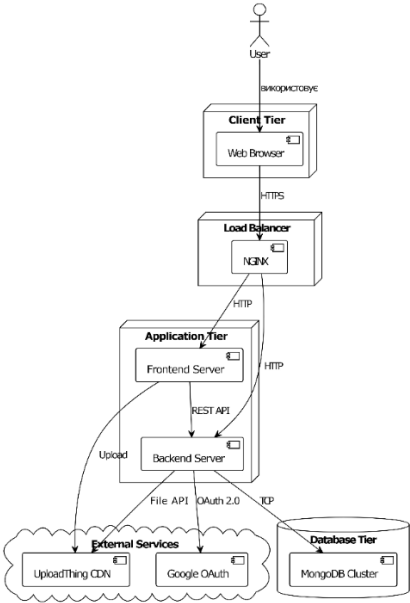


Рис. 3.1 - Діаграма розгортання системи

Клієнтські пристрої взаємодіють з Next.js фронтом через HTTPS протокол, забезпечуючи шифрування трафіку та безпечну передачу користувацьких даних.

Backend сервер на Spring Boot обробляє бізнес-логіку та взаємодіє з MongoDB через зашифровані з'єднання. Система підтримує горизонтальне масштабування через додавання додаткових backend інстансів для обробки зростаючого навантаження користувачів.

Зовнішні сервіси інтегруються через захищені API endpoints з автентифікацією за допомогою API ключів та OAuth tokenів.

3.2. Структура інтерфейсу мобільного застосунку

Архітектура користувацького інтерфейсу веб-платформи Bartender Hub побудована на принципах компонентної організації. Інтерфейс структурований у вигляді ієрархічної системи компонентів, що забезпечує інтуїтивність взаємодії користувачів з функціональними модулями системи.

Глобальна навігаційна структура реалізована через фіксований заголовок сторінки з адаптивним меню, що автоматично пристосовується до різних розмірів екранів пристроїв. Основні розділи платформи доступні через головне меню навігації, включаючи головну сторінку, каталог рецептів, блог-секцію та адміністративні інструменти модерації контенту.

Центральна сторінка застосунку демонструє багатосекційну компоновку з динамічним завантаженням контенту та персоналізованими рекомендаціями. Інтерфейс автоматично адаптується до статусу автентифікації користувача, відображаючи різний набір секцій для зареєстрованих та анонімних відвідувачів.

Система візуального зворотного зв'язку забезпечує миттєву індикацію стану завантаження та результатів користувацьких дій через анімації та повідомлення.

		Недашківський В.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		49

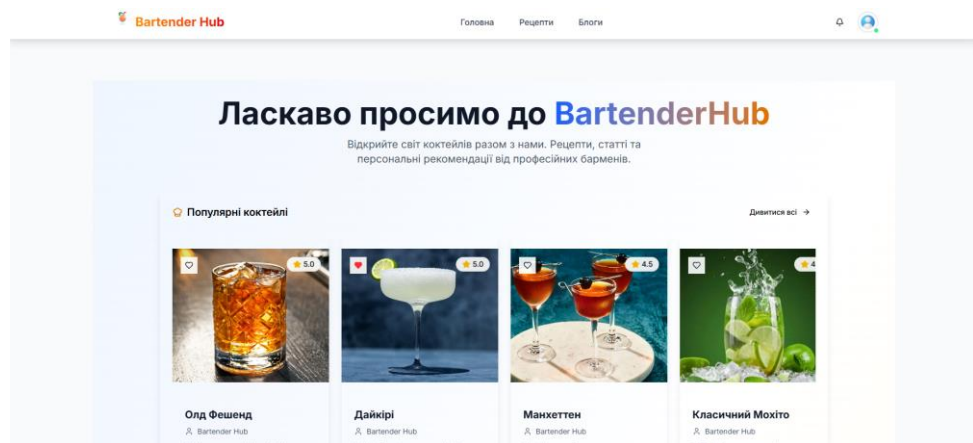


Рис. 3.2 - Головний інтерфейс веб-платформи

Верхня частина сторінки містить секцію популярних рецептів з горизонтальним скролінгом карток, що дозволяє ефективно відображати великий обсяг контенту без перевантаження вертикального простору. Кожна картка рецепту включає прев'ю зображення, базову інформацію про коктейль та рейтингову систему з візуальними індикаторами популярності.

Персоналізований блок рекомендацій активується для автентифікованих користувачів та відображає контент, підібраний на основі алгоритмічного аналізу їхніх уподобань. Система використовує дані про збережені рецепти та поставлені оцінки для формування індивідуальних пропозицій, що підвищує релевантність контенту та залученість аудиторії.

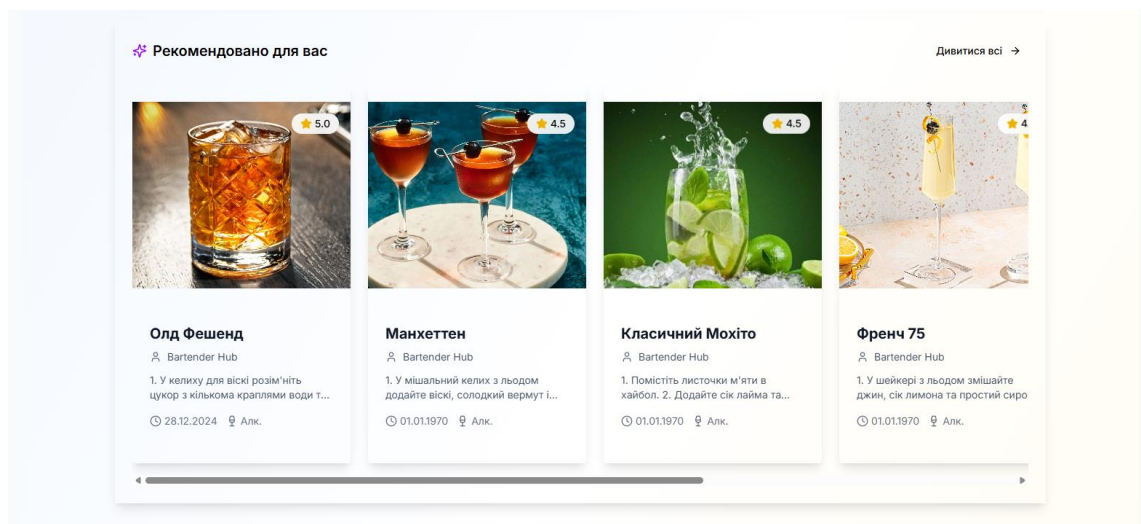


Рис. 3.3 - Блок рекомендацій рецептів

Нижня частина головної сторінки присвячена блог-секції з останніми статтями та трендинговими публікаціями. Структура відображення автоматично

		Недашківський В.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Фант М.О.				50
Змн.	Арк.	№ докум.	Підпис	Дата		

змінюється залежно від кількості доступного контенту та розміру екрану користувача, забезпечуючи оптимальний досвід перегляду на всіх типах пристроїв.

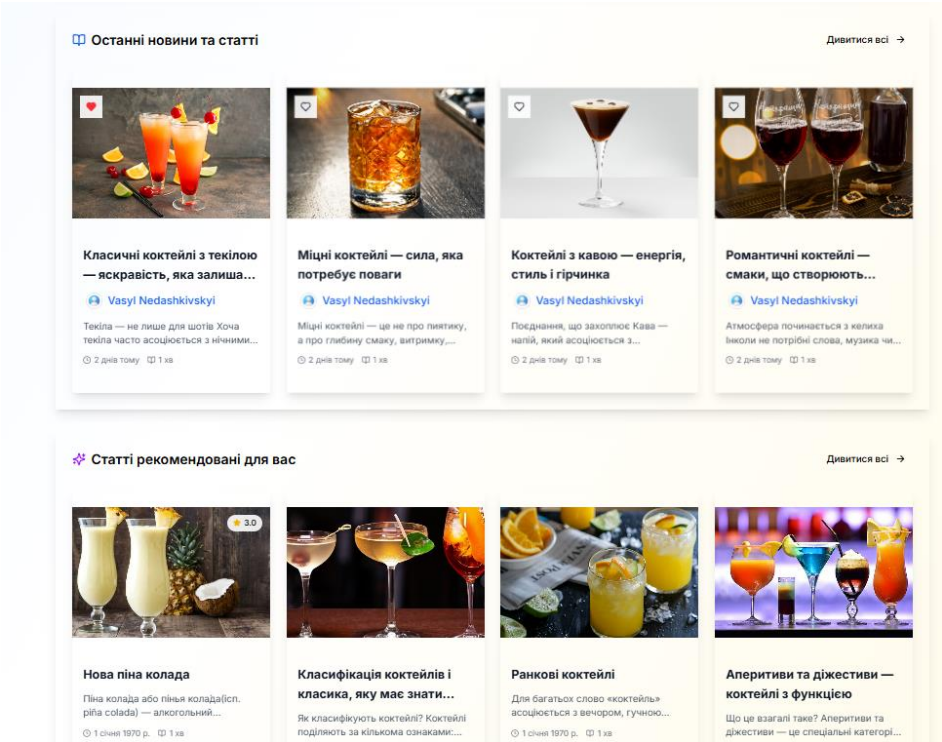


Рис. 3.4 Блок рекомендацій статей та останні статті

Навігаційна панель реалізована як фіксований елемент з напівпрозорим фоном та backdrop-blur ефектом, що створює сучасний візуальний стиль при скролінгу сторінки.

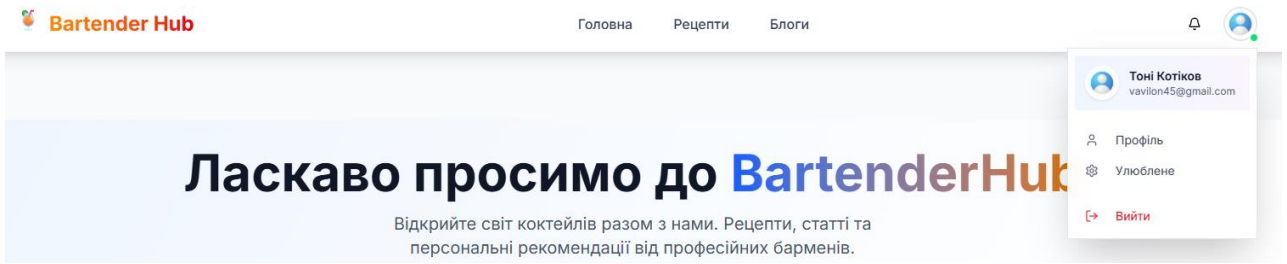


Рис 3.5 - Навігаційна панель з dropdown меню користувача

Основне меню включає прямі посилання на ключові розділи: головну сторінку, каталог рецептів та блог-секцію. Користувачі з адміністративними правами отримують доступ до розширеного dropdown меню модерації з можливістю управління заявками на публікацію рецептів, верифікацію користувачів та модерацію статей.

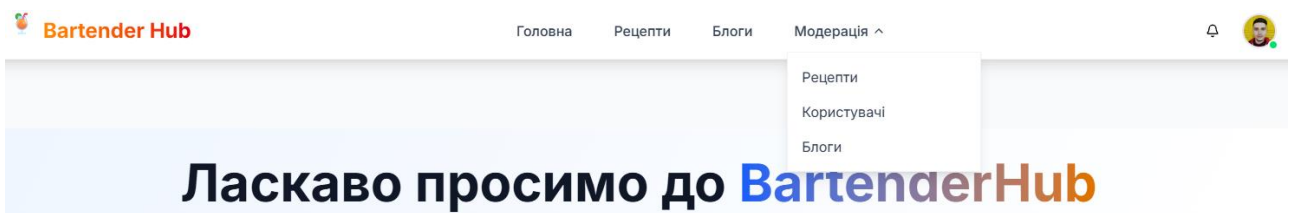


Рис. 3.6 - Dropdown меню модерации

Правий сегмент навігації адаптується до статусу автентифікації користувача. Для неавторизованих відвідувачів відображається кнопка входу з виразним стилізуванням. Зареєстровані користувачі бачать dropdown меню з аватаром, системою сповіщень та швидкими посиланнями на профіль, налаштування та функцію виходу з акаунту.

Мобільна версія навігації трансформується у компактне меню-гамбургер з плавними анімаціями розгортання та повноекранним overlay для покращення зручності взаємодії на сенсорних пристроях.

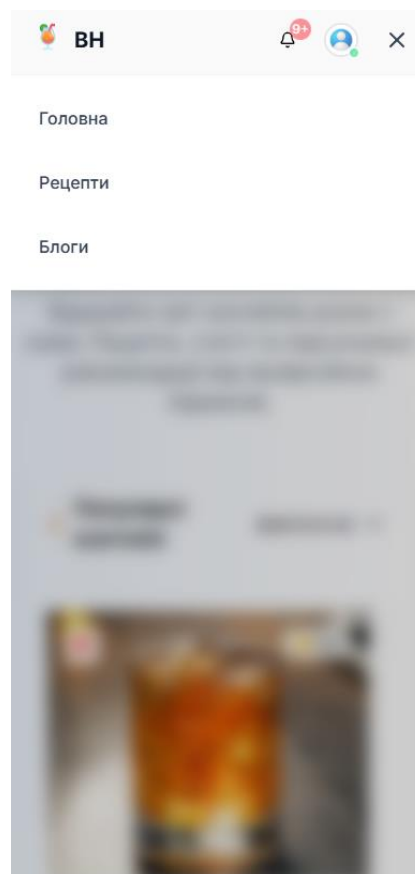


Рис. 3.7 - Мобільна версія навігаційного меню

Система форм платформи базується на принципах progressive enhancement з валідацією в реальному часі та інтуїтивними індикаторами помилок. Форма

		Недашківський В.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		52

створення рецепту демонструє складну багатосекційну структуру з динамічним управлінням інгредієнтами та інтегрованим завантаженням медіа файлів.

← Назад

Створити рецепт

Поділіться своїм унікальним рецептом коктейлю

Основна інформація

Назва рецепту *

Наприклад: Мохіто з м'ятою

0/100

Інструкції приготування *

Детально опишіть процес приготування...

0/5000

Історія рецепту (необов'язково)

Розкажіть історію створення цього рецепту...

0/1000

Фото рецепту

Виберіть файл або перетягніть сюди

Зображення (до 4MB)

Вибрати файл

Інгредієнти

Інгредієнт	Кількість	Одиниця	Примітки
Оберіть інгредієнт	50	мл	свіжий
+ Додати інгредієнт			

Налаштування

Видимість рецепту

Приватний

Публічний

Приватні рецепти доступні тільки вам

Категорія

Оберіть категорію

Теги

з ромом

елітний

фруктовий

з віскі

з горілкою

святковий

з джином

ранковий

ягідний

цитрусовий

низькокалорійний

вечірній

кавовий

кремовий

зимовий

заспокійливий

пряний

всипільний

енергетичний

романтичний

ментовий

літній

з теплого

класичний

шоколадний

Створити рецепт

Рис. 3.8 - Інтерфейс створення рецепту коктейлю

Основна секція форми містить поля для назви рецепту з валідацією унікальності в реальному часі, детальні інструкції приготування через багаторядкове текстове поле та опціональну історичну довідку про походження коктейлю. Система автоматично відстежує кількість введених символів та надає користувачам зворотній зв'язок щодо обмежень довжини тексту.

Секція інгредієнтів реалізована як динамічний список з можливістю додавання та видалення позицій.

Кожен інгредієнт має структуровані поля для вибору з попередньо визначеного каталогу, кількості з одиницями виміру та додаткових примітів щодо специфіки використання. Dropdown селектори забезпечують швидкий пошук серед сотень доступних інгредієнтів через автокомплітну функціональність.

Боковий блок налаштувань включає селектори категорій та тегів з множинним вибором, що дозволяє детальну категоризацію контенту для покращення його знаходимості через пошукові фільтри. Система відображення статусу рецепту автоматично адаптується до ролей користувача, показуючи різні опції публікації для звичайних авторів та адміністраторів.

Форми входу та реєстрації демонструють мінімалістичний дизайн з акцентом на зручність заповнення та безпечність даних. Інтерфейс підтримує як традиційну автентифікацію через email та пароль, так і інтеграцію з Google OAuth для спрощеного доступу.

Рис. 3.9 - Форма входу до системи

Форма реєстрації структурована у логічній послідовності полів з прогресивною валідацією та інформативними підказками. Поле дати народження

		Недашківський В.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		54

включає автоматичні обмеження для забезпечення вікових вимог платформи, а поле біографії надає користувачам можливість персоналізації профілю.

Система сповіщень відображається через спеціалізований dropdown з групуванням повідомлень за типами та автоматичним маркуванням прочитаних елементів.

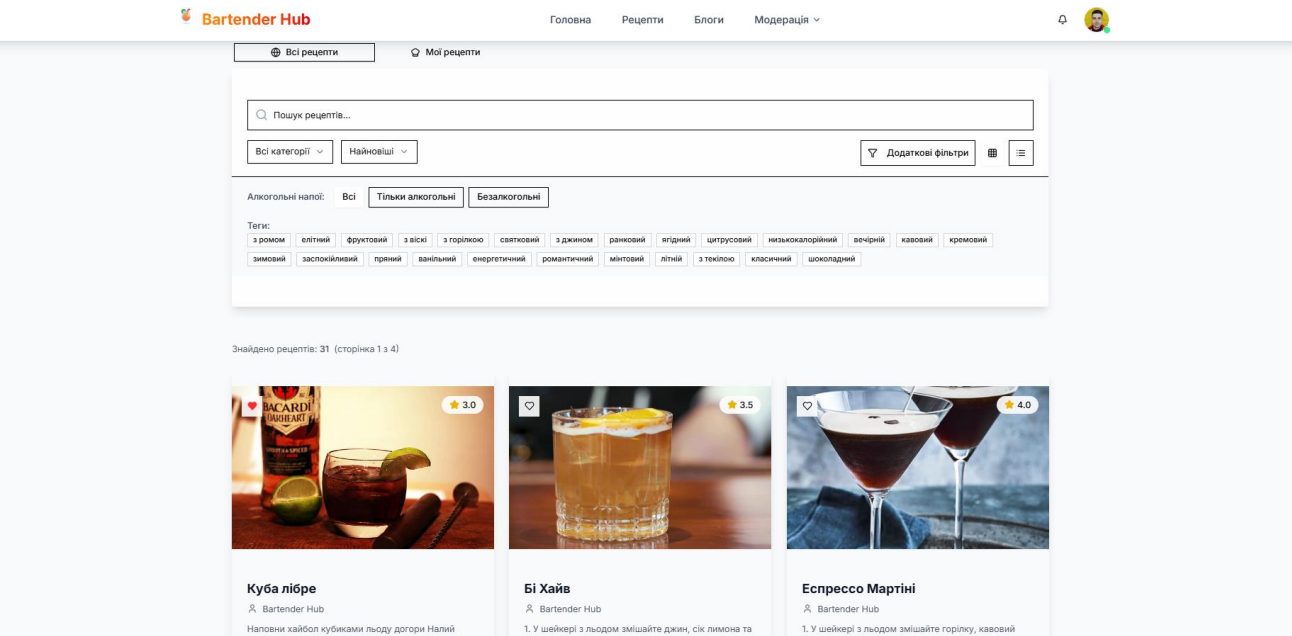


Рис. 3.10 - Каталог рецептів з фільтрами

Каталожна сторінка рецептів демонструє адаптивну сітку карток з можливістю перемикання між режимами відображення. Система фільтрації дозволяє користувачам звужувати пошук за категоріями, інгредієнтами та рейтингом, забезпечуючи швидкий доступ до релевантного контенту серед великої бази рецептів.

Редакційна система платформи реалізована через спеціалізований блог-модуль з розширеними можливостями створення та публікації тематичного контенту. Інтерфейс створення статті інтегрує потужний редактор з підтримкою медіа-контенту.

Пагінація результатів забезпечує ефективне відображення великих обсягів контенту без перевантаження інтерфейсу користувача.

Головна

Рецепти

Блоги

Модерація

← Назад

Створити статтю

Поділіться своїми знаннями з іншими користувачами

Основна інформація

Заголовок статті *

Наприклад: Історія коктейлю Mojito

0/200

Контент статті *

Почніть писати свою статтю. Використовуйте панель інструментів для форматування тексту, звань, списків та інших елементів...

Фільтр

Текст

Заголовок

Маркований список

Нумерований список

Чек-лист

Зображення

Категорія

Оберіть категорію

Теги

з ромом

спітний

фруктовий

з віскі

з горілкою

святковий

з джином

ранковий

ягідний

цитрусовий

низькокалорійний

вечірній

казовий

кремовий

зимовий

заспокійливий

пряний

вагітний

енергетичний

романтичний

мінтовий

літній

з теплотою

класичний

шоколадний

Зберегти як чернетку

Подати на розгляд

Головне зображення статті

Виберіть файл або перетягніть сюди

Зображення (до 4MB)

Вибрати файл

Рис. 3.11 - Редактор статей з інтегрованими медіа-інструментами

Система категоризації статей дублює логіку рецептів але адаптована під специфіку публіцистичного контенту. Автори можуть призначати множинні теги, обирати рівень складності матеріалу та встановлювати статус публікації через інтуїтивний боковий блок налаштувань.

Каталог блогів структурований як адаптивна сітка з превью-картками, що містять заголовок, анотацію, зображення автора та мета-інформацію про дату публікації. Система пагінації забезпечує ефективну навігацію через великі обсяги контенту без перевантаження сторінки.

		Недашківський В.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Фант М.О.				56
Змн.	Арк.	№ докум.	Підпис	Дата		

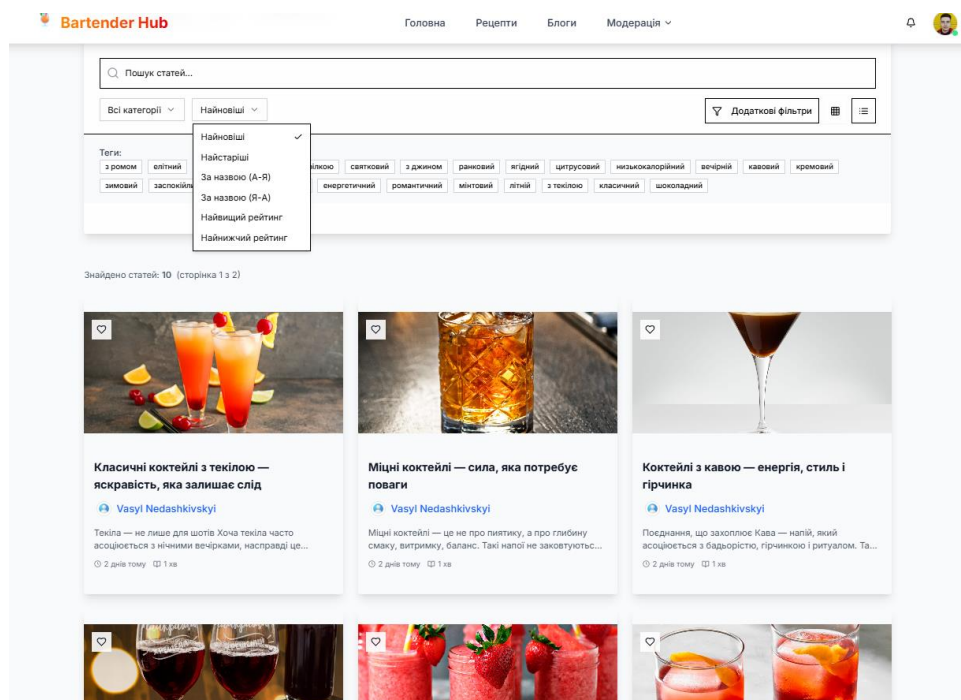


Рис. 3.12 - Каталог статей з фільтрацією

Детальна сторінка статті демонструє типографічно оптимізований макет з боковою панеллю навігації між розділами та інтегрованою системою коментарів. Читачі можуть залишати відгуки, оцінювати корисність матеріалу та зберігати статті до персональної колекції улюблених публікацій.

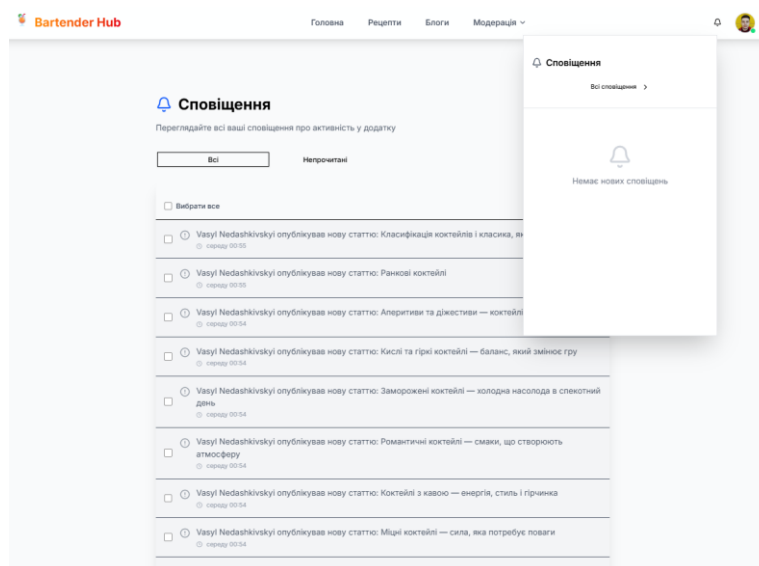


Рис. 3.13 - Інтерфейс системи сповіщень

Система сповіщень функціонує через спеціалізований dropdown компонент з реальним часом оновлення статусу повідомлень. Інтерфейс автоматично групує сповіщення за типами активності - коментарі до рецептів,

		Недашківський В.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		57

лайки статей, підписки на авторів та адміністративні повідомлення про статус модерації контенту.

Кожне сповіщення містить контекстну інформацію з аватаром ініціатора дії, часовою міткою та прямим посиланням на відповідний контент. Непрочитані повідомлення виділяються візуально через колірне кодування та лічильник у верхній панелі навігації, що забезпечує негайне інформування користувачів про нову активність.

3.3.Тестування роботи програмного комплексу

Тестування веб-додатків - важлива частина розробки. Воно допомагає забезпечити стабільну роботу та хороший користувацький досвід. Ми провели ретельне тестування всіх функцій системи. Це було зроблено для перевірки коректної роботи основних модулів платформи.

Тест 1: Реєстрація нового користувача

Передумови: користувач не зареєстрований у системі;

Кроки: перехід на сторінку реєстрації, введення імені "Олексій Іванов", email "test@example.com", пароля "TestPass123", дати народження "1995-05-15", короткого опису "Початківець бармен", натискання кнопки "Зареєструватися";

Очікуваний результат: користувач успішно реєструється та перенаправляється на сторінку входу;

Фактичний результат: реєстрація пройшла успішно з перевіркою валідності email.

Тест 2: Автентифікація через Google OAuth

Передумови: користувач має Google акаунт та знаходиться на сторінці входу;

Кроки: натискання кнопки "Увійти через Google", вибір Google акаунту, підтвердження дозволів;

Очікуваний результат: користувач успішно авторизується та потрапляє на головну сторінку;

		Недашківський В.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		58

Фактичний результат: автентифікація через Google пройшла успішно з автоматичним створенням профілю користувача.

Тест 3: Створення рецепта коктейлю з інгредієнтами

Передумови: користувач авторизований та знаходиться на сторінці створення рецепта;

Кроки: введення назви "Мохіто айс", інструкцій приготування, додавання інгредієнтів з бази (ром білий 50мл, м'ята свіжа 6 листочків, лайм 1/2 шт), встановлення категорії "Коктейлі", завантаження фото, збереження рецепта;

Очікуваний результат: рецепт створюється зі статусом DRAFT та зберігається в особистій колекції;

Фактичний результат: рецепт успішно створений з автоматичним визначенням вмісту алкоголю та збереженням усіх інгредієнтів.

Тест 4: Редагування існуючого рецепта

Передумови: користувач є автором рецепта зі статусом DRAFT;

Кроки: перехід на сторінку деталей рецепта, натискання кнопки "Редагувати", зміна інструкцій приготування, додавання нового інгредієнту, збереження змін;

Очікуваний результат: зміни зберігаються без зміни статусу рецепта;

Фактичний результат: рецепт успішно оновлений з новими інгредієнтами та інструкціями.

Тест 5: Створення статті в блозі

Передумови: користувач має роль AUTHOR та знаходиться на сторінці створення статті;

Кроки: введення заголовку "Історія коктейлю Мохіто", написання контенту за допомогою редактора, додавання медіа-файлів, встановлення категорії "Класичні коктейлі", збереження статті;

Очікуваний результат: стаття створюється зі статусом DRAFT;

Фактичний результат: стаття успішно створена з автоматичним генеруванням витягу з контенту.

		Недашківський В.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Фант М.О.				59
Змн.	Арк.	№ докум.	Підпис	Дата		

Тест 6: Додавання рецепта до улюблених

Передумови: користувач переглядає опублікований рецепт іншого автора;

Кроки: натискання іконки серця біля назви рецепта;

Очікуваний результат: рецепт додається до списку улюблених, іконка змінює стан на активний;

Фактичний результат: рецепт успішно доданий до улюблених з відображенням у розділі "Улюблене".

Тест 7: Оцінювання рецепта

Передумови: користувач переглядає опублікований рецепт;

Кроки: перехід на вкладку "Відгуки", натискання на 5 зірок для оцінки;

Очікуваний результат: рейтинг зберігається та оновлюється середня оцінка рецепта;

Фактичний результат: оцінка успішно збережена з автоматичним перерахунком середнього рейтингу.

Тест 8: Написання коментаря до рецепта

Передумови: користувач переглядає опублікований рецепт;

Кроки: перехід на вкладку "Коментарі", введення тексту "Чудовий рецепт, обов'язково спробую", натискання кнопки "Опублікувати";

Очікуваний результат: коментар з'являється у списку коментарів з інформацією про автора;

Фактичний результат: коментар успішно опублікований з відображенням часу створення.

Тест 9: Модерація рецепта адміністратором

Передумови: адміністратор авторизований, є рецепт зі статусом SUBMITTED;

Кроки: перехід до розділу "Модерація рецептів", введення коментаря модератора, натискання кнопки "Схвалити";

Очікуваний результат: рецепт отримує статус PUBLISHED та стає доступним у каталозі;

		Недашківський В.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Фант М.О.				60
Змн.	Арк.	№ докум.	Підпис	Дата		

Фактичний результат: рецепт успішно схвалений з відправленням сповіщення автору.

Тест 10: Пошук рецептів за фільтрами

Передумови: користувач знаходиться на сторінці каталогу рецептів;

Кроки: введення в пошуковому полі "мохіто", встановлення фільтра "Містить алкоголь", натискання кнопки пошуку;

Очікуваний результат: відображається відфільтрований список рецептів;

Фактичний результат: знайдено рецепт коктейлю з мохіто, що містять алкоголь.

Тест 11: Швидкість завантаження веб-сторінок

Передумови: користувач відкриває веб-застосунок у браузері;

Кроки: навігація на головну сторінку, вимірювання часу завантаження;

Очікуваний результат: сторінка завантажується менш ніж за 3 секунди;

Фактичний результат: середній час завантаження головної сторінки склав 2.1 секунди завдяки оптимізації Next.js.

Тест 12: Завантаження зображень рецептів

Передумови: користувач створює рецепт та завантажує фото;

Кроки: вибір зображення розміром 3МБ, завантаження через ImageUpload компонент;

Очікуваний результат: зображення успішно завантажується на UploadThing CDN;

Фактичний результат: файл завантажений та оптимізований з отриманням URL для збереження у рецепті.

Тест 13: Захист адміністративних функцій

Передумови: звичайний користувач (роль USER) авторизований у системі;

Кроки: спроба прямого доступу до URL модерації рецептів "/admin/request/recipes";

Очікуваний результат: доступ заборонений, користувач не бачить вміст сторінки;

		Недашківський В.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		61

Фактичний результат: система заблокувала доступ відповідно до ролей користувача, сторінка не відображається.

Проведене комплексне тестування підтвердило високу функціональну готовність веб-платформи Bartender Hub для експлуатації в продуктивному середовищі. Всі ключові функціональні модулі системи продемонстрували стабільну роботу з показником успішного виконання операцій при стандартних навантаженнях.

Висновки до розділу 3

Третій розділ дипломної роботи розкрив практичні аспекти впровадження та експлуатації веб-платформи Bartender Hub, охопивши ключові етапи від встановлення системи до комплексного тестування функціональності.

Розроблена процедура встановлення та налаштування параметрів системи забезпечує ефективне розгортання платформи в різних середовищах. Визначені мінімальні системні вимоги гарантують стабільну роботу клієнтської частини на сучасних пристроях, тоді як серверні компоненти оптимізовані для роботи з Java та Spring Boot.

Структура інтерфейсу користувача побудована з урахуванням адаптивним відображенням контенту. Реалізована навігаційна система забезпечує інтуїтивний доступ до основних функціональних модулів платформи через логічно організовані розділи. Система сповіщень та відображення статусів модерації забезпечує прозорість робочих процесів для всіх категорій користувачів.

Результати третього розділу засвідчують технічну готовність веб-платформи Bartender Hub до промислової експлуатації з дотриманням сучасних стандартів веб-розробки та вимог інформаційної безпеки. Розроблена документація по встановленню що забезпечує ефективне впровадження системи в різних організаційних контекстах.

		Недашківський В.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		62

ВИСНОВОКИ

Виконана дипломна робота присвячена розробці та реалізації комплексної веб-платформи Bartender Hub для професійних барменів та любителів коктейлів. Дослідження охопило повний цикл створення програмного продукту від аналізу вимог до практичного впровадження з комплексним тестуванням функціональності.

Результати теоретичного дослідження підтвердили актуальність створення спеціалізованої платформи для барменської індустрії. Аналіз існуючих аналогів виявив значні функціональні обмеження наявних рішень, що не враховують специфічні потреби професійних фахівців галузі. Обґрунтовано вибір розподіленої архітектури з шаровою організацією компонентів, що забезпечує ефективне розмежування бізнес-логіки та представлення даних. Використання технологічного стеку Java з Spring Boot для серверної частини та React з Next.js для клієнтського інтерфейсу створило надійну основу для реалізації складних інтерактивних функцій.

Проектна частина роботи охопила розробку комплексного рішення з тринадцятьма колекціями MongoDB, що забезпечують ефективне зберігання даних користувачів, рецептів, статей та соціальних взаємодій. Алгоритмічні рішення включають персоналізовані рекомендації на основі аналізу користувацьких уподобань, чотиристанову систему модерації контенту та систему збирання рейтингів у реальному часі. Інтеграція дванадцяти типів сповіщень які забезпечують ефективну комунікацію між користувачами платформи.

Практична реалізація веб-застосунку демонструє ефективне поєднання сучасних фронтенд-технологій з надійною серверною архітектурою. Використання модульної структури React компонентів з TypeScript забезпечує типобезпечність та підтримуваність клієнтського коду. Інтеграція з зовнішніми сервісами UploadThing для медіа-контенту розширює функціональні можливості платформи. Адаптивний користувацький інтерфейс з дев'ятьма ключовими

		Недашківський В.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		63

візуальними компонентами забезпечує інтуїтивну взаємодію на різних пристроях.

Розроблена система вирішує ключові проблеми барменської індустрії через централізоване зберігання рецептурних баз знань, верифікацію професійних кваліфікацій та створення спеціалізованої соціальної мережі для обміну експертним досвідом. Персоналізовані рекомендації на основі аналізу користувацької поведінки підвищують ефективність пошуку релевантного контенту. Система модерації забезпечує високу якість публікованої інформації через багаторівневий контроль та експертну перевірку.

Перспективи подальшого розвитку включають впровадження машинного навчання для покращення алгоритмів рекомендацій, інтеграцію з зовнішніми API барних закладів для покращення списку рецептів та розширення мобільної підтримки через прогресивні веб-додатки. Система готова до горизонтального масштабування та інтеграції додаткових модулів без значних архітектурних змін.

		Недашківський В.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		64

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Walls C. Spring Boot in Action / C. Walls. - 2nd ed. - Greenwich: Manning Publications, 2023. - 416 p.
2. Gutierrez F. Learning Spring Boot 3.0: Simplify the development of production-grade applications using Java and Spring / F. Gutierrez. - 3rd ed. - Birmingham: Packt Publishing, 2023. - 270 p.
3. Spilca L. Spring Security in Action / L. Spilca. - Shelter Island: Manning Publications, 2020. - 560 p.
4. Banks A., Porcello E. Learning React: Modern Patterns for Developing React Apps / A. Banks, E. Porcello. - 2nd ed. - Sebastopol: O'Reilly Media, 2020. - 310 p.
5. Larsen R. Real-World Next.js: Build scalable, high-performance, and modern web applications using Next.js / R. Larsen. - Birmingham: Packt Publishing, 2022. - 366 p.
6. Chodorow K. MongoDB: The Definitive Guide / K. Chodorow. - 3rd ed. - Sebastopol: O'Reilly Media, 2019. - 514 p.
7. Wilson S. TypeScript Quickly: Learn TypeScript programming fundamentals / S. Wilson, Y. Fain. - Shelter Island: Manning Publications, 2020. - 488 p.
8. Martin R. C. Clean Code: A Handbook of Agile Software Craftsmanship / R. C. Martin. - Upper Saddle River: Prentice Hall, 2008. - 464 p.
9. Evans E. Domain-Driven Design: Tackling Complexity in the Heart of Software / E. Evans. - Boston: Addison-Wesley, 2003. - 560 p.
10. Fowler M. Patterns of Enterprise Application Architecture / M. Fowler. - Boston: Addison-Wesley, 2002. - 560 p.

		Недашківський В.С.			ІІЗ.КР.Б-121-25-ІІЗ	Арк.
		Файт М.О.				65
Змн.	Арк.	№ докум.	Підпис	Дата		

11. Kleppmann M. Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems / M. Kleppmann. - Sebastopol: O'Reilly Media, 2017. - 616 p.
12. Ricci F., Rokach L., Shapira B. Recommender Systems Handbook / F. Ricci, L. Rokach, B. Shapira. - 2nd ed. - New York: Springer, 2015. - 1010 p.
13. DeGroff D. The Craft of the Cocktail: Everything You Need to Know to Be a Master Bartender / D. DeGroff. - New York: Clarkson Potter, 2002. - 240 p.
14. Morgenthaler J. The Bar Book: Elements of Cocktail Technique / J. Morgenthaler. - San Francisco: Chronicle Books, 2014. - 320 p.
15. Wondrich D. Imbibe!: From Absinthe Cocktail to Whiskey Smash / D. Wondrich. - Updated ed. - New York: Perigee Books, 2015. - 384 p.
16. Kaplan D. Liquid Intelligence: The Art and Science of the Perfect Cocktail / D. Kaplan. - New York: W. W. Norton, 2014. - 416 p.
17. Aleksendrić M., Batra S., Palmer R., Ranjan S. Full Stack FastAPI, React, and MongoDB: Fast-paced web app development with the FARM stack / M. Aleksendrić et al. - Birmingham: Packt Publishing, 2024. - 336 p.
18. Grinberg M. Flask Web Development: Developing Web Applications with Python / M. Grinberg. - 2nd ed. - Sebastopol: O'Reilly Media, 2018. - 316 p.
19. Hussam H. Spring Boot & MongoDB Full Project Tutorial [Електронний ресурс] / Hussam Hussien Abd El-Aziz // Medium. - 2025. - Режим доступу: <https://medium.com/@hussam7ussien/spring-boot-mongodb-full-project-tutorial-017ae9b7689f>
20. Ayoob S. Simplifying Spring Boot Development with React and MongoDB [Електронний ресурс] / Sarah Ayoob // Medium SLIIT Women In FOSS Community. - 2023. - Режим доступу:

		Недашківський В.С.			ІІЗ.КР.Б-121-25-ІІЗ	Арк.
		Фант М.О.				66
Змн.	Арк.	№ докум.	Підпис	Дата		

<https://medium.com/sliitwif/simplifying-spring-boot-development-with-react-and-mongodb-215fb8dbde05>

21.Devlink Tips. Spring Boot + React/Next.js: How to Build a Full-Stack App in 2025 [Електронний ресурс] / Devlink Tips // Medium. - 2025. - Режим доступу: <https://medium.com/@devlink/spring-boot-react-next-js-how-to-build-a-full-stack-app-in-2025-49666fc10d59>

22.Mahad. Building a Reactive Spring Boot Application with MongoDB, Batch Processing, and Testcontainers [Електронний ресурс] / Mahad // Medium. - 2023. - Режим доступу: <https://medium.com/@mbanaee61/building-a-reactive-spring-boot-application-with-mongodb-batch-processing-and-testcontainers-a-22ff13b42a86>

23.GeeksforGeeks. Full Stack Developer Roadmap 2025 Updated [Електронний ресурс]. - 2024. - Режим доступу: <https://www.geeksforgeeks.org/blogs/full-stack-developer-roadmap/>

24.Shameer A. Building Modern Web Applications with Spring Boot and React [Електронний ресурс] / Ahmad Shameer // Dev.to. - 2024. - Режим доступу: <https://dev.to/ahmadshameer/building-modern-web-applications-with-spring-boot-and-react-4h2k>

25.Spring Framework Team. Spring Framework Reference Documentation [Електронний ресурс]. - Режим доступу: <https://docs.spring.io/spring-framework/docs/current/reference/html/>

26.Spring Boot Team. Spring Boot Reference Guide [Електронний ресурс]. - Режим доступу: <https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/>

27.MongoDB Inc. MongoDB Manual [Електронний ресурс]. - Режим доступу: <https://docs.mongodb.com/manual/>

		Недашківський В.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Фант М.О.				67
Змн.	Арк.	№ докум.	Підпис	Дата		

28. Meta. React Documentation [Електронний ресурс]. - Режим доступу: <https://react.dev/>
29. Vercel. Next.js Documentation [Електронний ресурс]. - Режим доступу: <https://nextjs.org/docs>
30. Microsoft. TypeScript Documentation [Електронний ресурс]. - Режим доступу: <https://www.typescriptlang.org/docs/>
31. Oracle. Java SE Documentation [Електронний ресурс]. - Режим доступу: <https://docs.oracle.com/en/java/javase/>
32. JSON Web Token. JWT.io [Електронний ресурс]. - Режим доступу: <https://jwt.io/>
33. Google Developers. OAuth 2.0 Documentation [Електронний ресурс]. - Режим доступу: <https://developers.google.com/identity/protocols/oauth2>
34. Baeldung. Spring Boot and React Tutorial [Електронний ресурс]. - 2024. - Режим доступу: <https://www.baeldung.com/spring-boot-react-crud>
35. Mozilla Developer Network. Progressive Web Apps Guide [Електронний ресурс]. - Режим доступу: https://developer.mozilla.org/en-US/docs/Web/Progressive_web_apps
36. International Bartenders Association. Official IBA Cocktails [Електронний ресурс]. - Режим доступу: <https://iba-world.com/cocktails/>

		Недашківський В.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		68

ДОДАТКИ

		Недашківський В.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		69

Лістинг коду сервісу відповідального за рекомендації користувачів:

@Service

```

public class RecommendationService {
    private final RecipeRepository recipeRepo;
    private final NewsArticleRepository articleRepo;
    private final FavoriteRepository favoriteRepo;
    private final UserRepository userRepo;
    private final MongoTemplate mongoTemplate;
    private final RecipeService recipeService;
    private final NewsArticleService articleService;

    public RecommendationService(
        RecipeRepository recipeRepo,
        NewsArticleRepository articleRepo,
        FavoriteRepository favoriteRepo,
        UserRepository userRepo,
        MongoTemplate mongoTemplate,
        RecipeService recipeService,
        NewsArticleService articleService) {
        this.recipeRepo = recipeRepo;
        this.articleRepo = articleRepo;
        this.favoriteRepo = favoriteRepo;
        this.userRepo = userRepo;
        this.mongoTemplate = mongoTemplate;
        this.recipeService = recipeService;
        this.articleService = articleService;
    }

    public Page<RecipeWithAuthorDto> recommendRecipes(String userId, Pageable pageable) {
        Set<String> favoriteRecipeIds = getUserFavoriteIds(userId, ParentType.RECIPE);

        if (favoriteRecipeIds.isEmpty()) {
            return getPopularRecipes(pageable);
        }

        UserPreferences preferences = calculatePreferencesFromFavorites(userId, ParentType.RECIPE);
        List<RecipeWithAuthorDto> recommendations = findSimilarRecipes(preferences, favoriteRecipeIds, pageable.getPageSize());

        if (recommendations.isEmpty()) {
            return getPopularRecipes(pageable);
        }

        return createPage(recommendations, pageable);
    }
}

```

		Недашківський В.С.			ІПЗ.КР.Б-121-25-ІЗ	Арк.
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		70

```

public Page<NewsArticleWithAuthorDto> recommendArticles(String userId, Pageable pageable) {
    Set<String> favoriteArticleIds = getUserFavoriteIds(userId, ParentType.NEWS);

    if (favoriteArticleIds.isEmpty()) {
        return getPopularArticles(pageable);
    }

    UserPreferences preferences = calculatePreferencesFromFavorites(userId, ParentType.NEWS);
    List<NewsArticleWithAuthorDto> recommendations = findSimilarArticles(preferences, favoriteArticleIds,
pageable.getPageSize());

    if (recommendations.isEmpty()) {
        return getPopularArticles(pageable);
    }

    return createPage(recommendations, pageable);
}

private Set<String> getUserFavoriteIds(String userId, ParentType parentType) {
    return favoriteRepo.findByIdAndParentType(userId, parentType)
        .stream()
        .map(Favorite::getParentId)
        .collect(Collectors.toSet());
}

private UserPreferences calculatePreferencesFromFavorites(String userId, ParentType parentType) {
    User user = userRepo.findById(userId).orElse(null);
    boolean allowAlcohol = UserUtils.getAllowAlcohol(user);

    List<Favorite> favorites = favoriteRepo.findByIdAndParentType(userId, parentType);
    if (favorites.isEmpty()) {
        return new UserPreferences(Set.of(), Set.of(), Set.of(), allowAlcohol);
    }

    Set<String> tagIds = new HashSet<>();
    Set<String> ingredientIds = new HashSet<>();
    Set<String> categoryIds = new HashSet<>();

    if (parentType == ParentType.RECIPE) {
        List<Recipe> recipes = recipeRepo.findAllById(
            favorites.stream().map(Favorite::getParentId).toList());

        tagIds = recipes.stream()
            .filter(r -> r.getTagIds() != null)
            .flatMap(r -> r.getTagIds().stream())
            .collect(Collectors.toSet());
    }
}

```

		Недашківський В.С.			ІІЗ.КР.Б-121-25-ІІЗ	Арк.
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		71

```

ingredientIds = recipes.stream()
    .filter(r -> r.getIngredients() != null)
    .flatMap(r -> r.getIngredients().stream())
    .map(RecipeIngredient::getIngredientId)
    .collect(Collectors.toSet());

categoryIds = recipes.stream()
    .filter(r -> r.getCategoryIds() != null)
    .flatMap(r -> r.getCategoryIds().stream())
    .collect(Collectors.toSet());
} else {
    List<NewsArticle> articles = articleRepo.findAllById(
        favorites.stream().map(Favorite::getParentId).toList());

    tagIds = articles.stream()
        .filter(a -> a.getTagIds() != null)
        .flatMap(a -> a.getTagIds().stream())
        .collect(Collectors.toSet());

    categoryIds = articles.stream()
        .filter(a -> a.getCategoryIds() != null)
        .flatMap(a -> a.getCategoryIds().stream())
        .collect(Collectors.toSet());
}

return new UserPreferences(tagIds, categoryIds, ingredientIds, allowAlcohol);
}

private List<RecipeWithAuthorDto> findSimilarRecipes(UserPreferences preferences, Set<String> excludeIds, int limit) {
    List<AggregationOperation> pipeline = new ArrayList<>();

    Criteria criteria = Criteria.where("status").is(Status.PUBLISHED)
        .and("_id").nin(excludeIds);

    List<Criteria> preferencesCriteria = new ArrayList<>();

    if (!preferences.preferredTags().isEmpty()) {
        preferencesCriteria.add(Criteria.where("tagIds").in(preferences.preferredTags()));
    }

    if (!preferences.preferredCategories().isEmpty()) {
        preferencesCriteria.add(Criteria.where("categoryIds").in(preferences.preferredCategories()));
    }

    if (!preferences.preferredIngredients().isEmpty()) {
        preferencesCriteria.add(Criteria.where("ingredients.ingredientId").in(preferences.preferredIngredients()));
    }
}

```

		Недашківський В.С.			ІІЗ.КР.Б-121-25-ІІЗ	Арк.
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		72

```

if (!preferencesCriteria.isEmpty()) {
    criteria.orOperator(preferencesCriteria.toArray(new Criteria[0]));
}

if (!preferences.allowAlcohol()) {
    criteria.and("containsAlcohol").ne(true);
}

pipeline.add(Aggregation.match(criteria));
pipeline.add(Aggregation.lookup("users", "authorId", "_id", "authorInfo"));
pipeline.add(Aggregation.unwind("authorInfo", true));
pipeline.add(Aggregation.sort(Sort.Direction.DESC, "averageRating", "createdAt"));
pipeline.add(Aggregation.limit(limit));

try {
    AggregationResults<org.bson.Document> results = mongoTemplate.aggregate(
        Aggregation.newAggregation(pipeline), "recipes", org.bson.Document.class);

    return results.getMappedResults().stream()
        .map(this::mapDocumentToRecipeDto)
        .collect(Collectors.toList());
} catch (Exception e) {
    return Collections.emptyList();
}
}

private List<NewsArticleWithAuthorDto> findSimilarArticles(UserPreferences preferences, Set<String> excludeIds, int limit) {
    List<AggregationOperation> pipeline = new ArrayList<>();

    Criteria criteria = Criteria.where("status").is(Status.PUBLISHED)
        .and("_id").nin(excludeIds);

    List<Criteria> preferencesCriteria = new ArrayList<>();

    if (!preferences.preferredTags().isEmpty()) {
        preferencesCriteria.add(Criteria.where("tagIds").in(preferences.preferredTags()));
    }

    if (!preferences.preferredCategories().isEmpty()) {
        preferencesCriteria.add(Criteria.where("categoryIds").in(preferences.preferredCategories()));
    }

    if (!preferencesCriteria.isEmpty()) {
        criteria.orOperator(preferencesCriteria.toArray(new Criteria[0]));
    }
}

```

		Недашківський В.С.			ІІЗ.КР.Б-121-25-ІІЗ	Арк.
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		73

```

pipeline.add(Aggregation.match(criteria));
pipeline.add(Aggregation.lookup("users", "authorId", "_id", "authorInfo"));
pipeline.add(Aggregation.unwind("authorInfo", true));
pipeline.add(Aggregation.sort(Sort.Direction.DESC, "averageRating", "publishedAt"));
pipeline.add(Aggregation.limit(limit));

try {
    AggregationResults<org.bson.Document> results = mongoTemplate.aggregate(
        Aggregation.newAggregation(pipeline), "news_articles", org.bson.Document.class);

    return results.getMappedResults().stream()
        .map(this::mapDocumentToArticleDto)
        .collect(Collectors.toList());
} catch (Exception e) {
    return Collections.emptyList();
}
}

private Page<RecipeWithAuthorDto> getPopularRecipes(Pageable pageable) {
    return recipeService.findPublishedRecipes(null, null, null, null, null,
        PageRequest.of(pageable.getPageNumber(), pageable.getPageSize(),
            Sort.by(Sort.Direction.DESC, "averageRating", "createdAt")));
}

private Page<NewsArticleWithAuthorDto> getPopularArticles(Pageable pageable) {
    return articleService.findPublishedArticles(null, null, null, null,
        PageRequest.of(pageable.getPageNumber(), pageable.getPageSize(),
            Sort.by(Sort.Direction.DESC, "averageRating", "publishedAt")));
}

private <T> Page<T> createPage(List<T> content, Pageable pageable) {
    int start = (int) pageable.getOffset();
    int end = Math.min(start + pageable.getPageSize(), content.size());

    if (start >= content.size()) {
        return new PageImpl<>(Collections.emptyList(), pageable, content.size());
    }

    List<T> pageContent = content.subList(start, end);
    return new PageImpl<>(pageContent, pageable, content.size());
}

private RecipeWithAuthorDto mapDocumentToRecipeDto(org.bson.Document doc) {
    RecipeWithAuthorDto dto = new RecipeWithAuthorDto();

    Object idObj = doc.get("_id");
    if (idObj instanceof org.bson.types.ObjectId) {

```

		Недашківський В.С.			ІІЗ.КР.Б-121-25-ІІЗ	Арк.
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		74

```

        dto.setId(((org.bson.types.ObjectId) idObj).toHexString());
    } else if (idObj instanceof String) {
        dto.setId((String) idObj);
    }

    dto.setName(doc.getString("name"));
    dto.setInstructions(doc.getString("instructions"));
    dto.setHistory(doc.getString("history"));

    List<?> photoUrls = doc.getList("photoUrls", String.class);
    dto.setPhotoUrls(photoUrls != null ? (List<String>) photoUrls : new ArrayList<>());

    String sourceStr = doc.getString("source");
    if (sourceStr != null) {
        try {
            dto.setSource(Source.valueOf(sourceStr));
        } catch (IllegalArgumentException e) {
            dto.setSource(Source.USER);
        }
    }

    Boolean containsAlcohol = doc.getBoolean("containsAlcohol");
    dto.setContainsAlcohol(containsAlcohol != null ? containsAlcohol : false);

    Object avgRating = doc.get("averageRating");
    dto.setAverageRating(avgRating instanceof Number ? ((Number) avgRating).doubleValue() : 0.0);

    Object createdAtObj = doc.get("createdAt");
    if (createdAtObj instanceof Date) {
        dto.setCreatedAt(((Date) createdAtObj).toInstant());
    }

    org.bson.Document authorDoc = doc.get("authorInfo", org.bson.Document.class);
    if (authorDoc != null) {
        RecipeWithAuthorDto.AuthorInfo authorInfo = new RecipeWithAuthorDto.AuthorInfo();

        Object authorIdObj = authorDoc.get("_id");
        if (authorIdObj instanceof org.bson.types.ObjectId) {
            authorInfo.setId(((org.bson.types.ObjectId) authorIdObj).toHexString());
        } else if (authorIdObj instanceof String) {
            authorInfo.setId((String) authorIdObj);
        }

        authorInfo.setName(authorDoc.getString("name"));
        authorInfo.setAvatarUrl(authorDoc.getString("avatarUrl"));
        Boolean verified = authorDoc.getBoolean("bartenderVerified");
        authorInfo.setBartenderVerified(verified != null ? verified : false);
    }

```

		Недашківський В.С.			ІПЗ.КР.Б-121-25-ІІЗ	Арк.
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		75

```

        dto.setAuthor(authorInfo);
    }

    return dto;
}

private NewsArticleWithAuthorDto mapDocumentToArticleDto(org.bson.Document doc) {
    NewsArticleWithAuthorDto dto = new NewsArticleWithAuthorDto();

    Object idObj = doc.get("_id");
    if (idObj instanceof org.bson.types.ObjectId) {
        dto.setId(((org.bson.types.ObjectId) idObj).toHexString());
    } else if (idObj instanceof String) {
        dto.setId((String) idObj);
    }

    dto.setTitle(doc.getString("title"));
    dto.setContent(doc.getString("content"));
    dto.setExcerpt(doc.getString("excerpt"));

    List<> mediaUrls = doc.getList("mediaUrls", String.class);
    dto.setMediaUrls(mediaUrls != null ? (List<String>) mediaUrls : new ArrayList<>());

    String sourceStr = doc.getString("source");
    if (sourceStr != null) {
        try {
            dto.setSource(Source.valueOf(sourceStr));
        } catch (IllegalArgumentException e) {
            dto.setSource(Source.USER);
        }
    }

    Object avgRating = doc.get("averageRating");
    dto.setAverageRating(avgRating instanceof Number ? ((Number) avgRating).doubleValue() : 0.0);

    Object publishedAtObj = doc.get("publishedAt");
    if (publishedAtObj instanceof Date) {
        dto.setPublishedAt(((Date) publishedAtObj).toInstant());
    }

    org.bson.Document authorDoc = doc.get("authorInfo", org.bson.Document.class);
    if (authorDoc != null) {
        NewsArticleWithAuthorDto.AuthorInfo authorInfo = new NewsArticleWithAuthorDto.AuthorInfo();

        Object authorIdObj = authorDoc.get("_id");
        if (authorIdObj instanceof org.bson.types.ObjectId) {

```

		Недашківський В.С.			ІІЗ.КР.Б-121-25-ІІЗ	Арк.
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		76

```

        authorInfo.setId(((org.bson.types.ObjectId) authorIdObj).toHexString());
    } else if (authorIdObj instanceof String) {
        authorInfo.setId((String) authorIdObj);
    }

    authorInfo.setName(doc.getString("name"));
    authorInfo.setAvatarUrl(doc.getString("avatarUrl"));
    Boolean verified = doc.getBoolean("bartenderVerified");
    authorInfo.setBartenderVerified(verified != null ? verified : false);

    dto.setAuthor(authorInfo);
}

return dto;
}

private record UserPreferences(
    Set<String> preferredTags,
    Set<String> preferredCategories,
    Set<String> preferredIngredients,
    boolean allowAlcohol
) {}
}

```

Лістинг коду сервісу відповідального за рецепти:

```

@Service
@Transactional
public class RecipeService {

    private final RecipeRepository recipeRepo;
    private final IngredientRepository ingredientRepo;
    private final UserRepository userRepo;
    private final RatingRepository ratingRepo;
    private final FavoriteRepository favoriteRepo;
    private final CommentRepository commentRepo;
    private final MongoTemplate mongoTemplate;
    private final RecipeSecurityService securityService;
    private final UserService userService;
    private final NotificationService notificationService;

    public RecipeService(RecipeRepository recipeRepo,
        IngredientRepository ingredientRepo,
        UserRepository userRepo,
        RatingRepository ratingRepo,
        FavoriteRepository favoriteRepo,
        CommentRepository commentRepo,
        MongoTemplate mongoTemplate,
        RecipeSecurityService securityService,
        UserService userService,
        NotificationService notificationService) {
    }
}

```

		Недашківський В.С.			ІПЗ.КР.Б-121-25-ІЗ	Арк.
		Фант М.О.				77
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        NotificationService notificationService) {
    this.recipeRepo = recipeRepo;
    this.ingredientRepo = ingredientRepo;
    this.userRepo = userRepo;
    this.ratingRepo = ratingRepo;
    this.favoriteRepo = favoriteRepo;
    this.commentRepo = commentRepo;
    this.mongoTemplate = mongoTemplate;
    this.securityService = securityService;
    this.userService = userService;
    this.notificationService = notificationService;
}

public Recipe createRecipe(Recipe recipe, String authorId) {
    User author = userService.findById(authorId)
        .orElseThrow(() -> new IllegalArgumentException("Author not found"));

    recipe.setAuthorId(authorId);
    recipe.setCreatedAt(Instant.now());
    recipe.setUpdatedAt(Instant.now());
    recipe.setAverageRating(0.0);
    recipe.setRejectionCount(0);

    if (recipe.getStatus() == null) {
        recipe.setStatus(Status.DRAFT);
    }

    if (recipe.getSource() == null) {
        recipe.setSource(Source.USER);
    }

    if (recipe.getIngredients() == null) {
        recipe.setIngredients(new ArrayList<>());
    }

    if (recipe.getPhotoUrls() == null) {
        recipe.setPhotoUrls(new ArrayList<>());
    }

    if (recipe.getCategoryIds() == null) {
        recipe.setCategoryIds(new ArrayList<>());
    }

    if (recipe.getTagIds() == null) {
        recipe.setTagIds(new ArrayList<>());
    }

    List<String> errors = validateRecipeContent(recipe);
    if (!errors.isEmpty()) {
        throw new IllegalArgumentException("Recipe validation failed: " + String.join(", ", errors));
    }

    recipe.setContainsAlcohol(detectAlcohol(recipe));
}

```

		Недашківський В.С.			ІІЗ.КР.Б-121-25-ІІЗ	Арк.
		Фант М.О.				78
Змн.	Арк.	№ докум.	Підпис	Дата		

```

Recipe saved = recipeRepo.save(recipe);

return saved;
}

public Recipe updateRecipe(String recipeId, Recipe updatedRecipe, String userId) {
    Recipe existingRecipe = recipeRepo.findById(recipeId)
        .orElseThrow(() -> new IllegalArgumentException("Recipe not found"));

    User user = userService.findById(userId)
        .orElseThrow(() -> new IllegalArgumentException("User not found"));

    if (!securityService.canEdit(existingRecipe, user)) {
        throw new SecurityException("No permission to edit this recipe");
    }

    updatedRecipe.setId(recipeId);
    updatedRecipe.setAuthorId(existingRecipe.getAuthorId());
    updatedRecipe.setCreatedAt(existingRecipe.getCreatedAt());
    updatedRecipe.setAverageRating(existingRecipe.getAverageRating());
    updatedRecipe.setRejectionCount(existingRecipe.getRejectionCount());
    updatedRecipe.setLastSubmittedAt(existingRecipe.getLastSubmittedAt());
    updatedRecipe.setUpdatedAt(Instant.now());

    if (existingRecipe.getStatus() == Status.REJECTED) {
        updatedRecipe.setStatus(Status.DRAFT);
        updatedRecipe.setModeratorComments(null);
    } else if (updatedRecipe.getStatus() == null) {
        updatedRecipe.setStatus(existingRecipe.getStatus() != null
            ? existingRecipe.getStatus()
            : Status.DRAFT);
    }

    List<String> errors = validateRecipeContent(updatedRecipe);
    if (!errors.isEmpty()) {
        throw new IllegalArgumentException("Recipe validation failed: " + String.join(", ", errors));
    }

    handleStatusOnUpdate(existingRecipe, updatedRecipe, user);

    updatedRecipe.setContainsAlcohol(detectAlcohol(updatedRecipe));

    Recipe saved = recipeRepo.save(updatedRecipe);

    return saved;
}

private boolean detectAlcohol(Recipe recipe) {
    if (recipe.getIngredients() == null || recipe.getIngredients().isEmpty()) {
        return false;
    }
}

```

		Недашківський В.С.			ІПЗ.КР.Б-121-25-ІПЗ	Арк.
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		79

```

Set<String> ingredientIds = recipe.getIngredients().stream()
    .map(RecipeIngredient::getIngredientId)
    .filter(Objects::nonNull)
    .filter(id -> !id.trim().isEmpty())
    .collect(Collectors.toSet());

if (ingredientIds.isEmpty()) {
    return false;
}

List<Ingredient> ingredients = ingredientRepo.findAllById(ingredientIds);
boolean hasAlcohol = ingredients.stream().anyMatch(Ingredient::isAlcohol);

return hasAlcohol;
}

public Recipe submitForPublication(String recipeId, String userId, String note) {
    Recipe recipe = recipeRepo.findById(recipeId)
        .orElseThrow(() -> new IllegalArgumentException("Recipe not found"));

    User user = userService.findById(userId)
        .orElseThrow(() -> new IllegalArgumentException("User not found"));

    if (!securityService.canSubmitForPublication(recipe, user)) {
        throw new SecurityException("Cannot submit this recipe for publication");
    }

    recipe.setStatus(Status.SUBMITTED);
    recipe.setLastSubmittedAt(Instant.now());
    recipe.setUpdatedAt(Instant.now());

    Recipe saved = recipeRepo.save(recipe);
    notificationService.sendNewRecipeSubmissionNotification(saved, user);

    return saved;
}

public Recipe approveRecipe(String recipeId, String moderatorId, String comments) {
    Recipe recipe = recipeRepo.findById(recipeId)
        .orElseThrow(() -> new IllegalArgumentException("Recipe not found"));

    User moderator = userService.findById(moderatorId)
        .orElseThrow(() -> new IllegalArgumentException("Moderator not found"));

    if (!securityService.canApprove(recipe, moderator)) {
        throw new SecurityException("No permission to approve this recipe");
    }

    recipe.setStatus(Status.PUBLISHED);
    recipe.setModeratorComments(comments);
    recipe.setUpdatedAt(Instant.now());

    Recipe saved = recipeRepo.save(recipe);

```

		Недашківський В.С.			ІПЗ.КР.Б-121-25-ІІЗ	Арк.
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		80

```

User author = userService.findById(recipe.getAuthorId()).orElse(null);
if (author != null) {
    notificationService.sendRecipeApprovedNotification(
        author.getId(), recipe.getName(), recipe.getId());

    notificationService.sendNewRecipeNotification(saved, author);
}

return saved;
}

public Recipe rejectRecipe(String recipeId, String moderatorId, String reason) {
    Recipe recipe = recipeRepo.findById(recipeId)
        .orElseThrow(() -> new IllegalArgumentException("Recipe not found"));

    User moderator = userService.findById(moderatorId)
        .orElseThrow(() -> new IllegalArgumentException("Moderator not found"));

    if (!securityService.canReject(recipe, moderator)) {
        throw new SecurityException("No permission to reject this recipe");
    }

    recipe.setStatus(Status.REJECTED);
    recipe.setModeratorComments(reason);
    recipe.setRejectionCount(recipe.getRejectionCount() + 1);
    recipe.setUpdatedAt(Instant.now());

    Recipe saved = recipeRepo.save(recipe);

    User author = userService.findById(recipe.getAuthorId()).orElse(null);
    if (author != null) {
        notificationService.sendRecipeRejectedNotification(
            author.getId(), recipe.getName(), recipe.getId(), reason);
    }

    return saved;
}

public Page<RecipeWithAuthorDto> findPublishedRecipes(
    Source source,
    List<String> categoryIds,
    List<String> tagIds,
    String search,
    Boolean containsAlcohol,
    Pageable pageable) {

    List<AggregationOperation> pipeline = new ArrayList<>();

    Criteria criteria = Criteria.where("status").is(Status.PUBLISHED);

    if (source != null) {
        criteria.and("source").is(source);
    }

```

		Недашківський В.С.			ІІЗ.КР.Б-121-25-ІІЗ	Арк.
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		81

```

    }

    if (categoryIds != null && !categoryIds.isEmpty()) {
        criteria.and("categoryIds").in(categoryIds);
    }

    if (tagIds != null && !tagIds.isEmpty()) {
        criteria.and("tagIds").in(tagIds);
    }

    if (search != null && !search.trim().isEmpty()) {
        criteria.orOperator(
            Criteria.where("name").regex(search, "i"),
            Criteria.where("instructions").regex(search, "i")
        );
    }

    if (containsAlcohol != null) {
        criteria.and("containsAlcohol").is(containsAlcohol);
    }

    pipeline.add(Aggregation.match(criteria));
    pipeline.add(Aggregation.lookup("users", "authorId", "_id", "authorInfo"));
    pipeline.add(Aggregation.unwind("authorInfo", true));

    String sortField = pageable.getSort().iterator().hasNext()
        ? pageable.getSort().iterator().next().getProperty()
        : "createdAt";

    Sort.Direction direction = pageable.getSort().iterator().hasNext()
        ? pageable.getSort().iterator().next().getDirection()
        : Sort.Direction.DESC;

    pipeline.add(Aggregation.sort(direction, sortField));

    pipeline.add(Aggregation.facet()
        .and(Aggregation.skip(pageable.getOffset()),
            Aggregation.limit(pageable.getPageSize()))
        .as("recipes")
        .and(Aggregation.count().as("totalCount"))
        .as("metadata"));

    return executeAggregation(pipeline, pageable);
}

public Page<RecipeWithAuthorDto> findUserRecipes(
    String userId,
    Status status,
    Pageable pageable) {

    List<AggregationOperation> pipeline = new ArrayList<>();

    Criteria criteria = Criteria.where("authorId").is(userId);

```

		Недашківський В.С.			ІПЗ.КР.Б-121-25-ІЗ	Арк.
		Фант М.О.				82
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        if (status != null) {
            criteria.and("status").is(status);
        }

        pipeline.add(Aggregation.match(criteria));
        pipeline.add(Aggregation.lookup("users", "authorId", "_id", "authorInfo"));
        pipeline.add(Aggregation.unwind("authorInfo", true));

        if (pageable.getSort().isSorted()) {
            pipeline.add(Aggregation.sort(pageable.getSort()));
        } else {
            pipeline.add(Aggregation.sort(Sort.Direction.DESC, "createdAt"));
        }

        pipeline.add(Aggregation.facet()
            .and(Aggregation.skip(pageable.getOffset()),
                Aggregation.limit(pageable.getPageSize()))
            .as("recipes")
            .and(Aggregation.count().as("totalCount"))
            .as("metadata"));

        return executeAggregation(pipeline, pageable);
    }

    public Optional<RecipeWithAuthorDto> findRecipeByIdSecure(String recipeId, String userId) {
        Optional<Recipe> recipeOpt = recipeRepo.findById(recipeId);
        if (recipeOpt.isEmpty()) {
            return Optional.empty();
        }

        Recipe recipe = recipeOpt.get();

        if (!securityService.canView(recipe, userId)) {
            return Optional.empty();
        }

        return Optional.of(mapToDto(recipe));
    }

    public Page<RecipeWithAuthorDto> getRecipesForModeration(Pageable pageable) {
        List<AggregationOperation> pipeline = new ArrayList<>();

        Criteria criteria = Criteria.where("status").is(Status.SUBMITTED);
        pipeline.add(Aggregation.match(criteria));

        pipeline.add(new AggregationOperation() {
            @Override
            public Document toDocument(AggregationOperationContext context) {
                return new Document("$lookup", new Document()
                    .append("from", "users")
                    .append("let", new Document("authorIdStr", "$authorId"))
                    .append("pipeline", Arrays.asList(
                        new Document("$match", new Document("$expr",

```

		Недашківський В.С.			ІПЗ.КР.Б-121-25-ІПЗ	Арк.
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		83

```

        new Document("$eq", Arrays.asList("$_id",
            new Document("$objectId", "$$authorIdStr"))))
    ))
    .append("as", "authorInfo"));
    }
    });

pipeline.add(Aggregation.unwind("authorInfo", true));

pipeline.add(Aggregation.sort(Sort.Direction.ASC, "lastSubmittedAt"));

pipeline.add(Aggregation.facet()
    .and(Aggregation.skip(pageable.getOffset()),
        Aggregation.limit(pageable.getPageSize()))
    .as("recipes")
    .and(Aggregation.count().as("totalCount"))
    .as("metadata"));

try {
    AggregationResults<Document> results = mongoTemplate.aggregate(
        Aggregation.newAggregation(pipeline), "recipes", Document.class);

    Document result = results.getUniqueMappedResult();
    if (result == null) {
        return new PageImpl<>(Collections.emptyList(), pageable, 0);
    }

    List<Document> recipeDocuments = result.getList("recipes", Document.class);
    List<Document> metadata = result.getList("metadata", Document.class);

    long totalCount = metadata.isEmpty() ? 0 :
        metadata.get(0).getInteger("totalCount", 0);

    List<RecipeWithAuthorDto> recipes = recipeDocuments.stream()
        .map(this::mapDocumentToDto)
        .collect(Collectors.toList());

    return new PageImpl<>(recipes, pageable, totalCount);

} catch (Exception e) {
    e.printStackTrace();
    return new PageImpl<>(Collections.emptyList(), pageable, 0);
}
}

@Transactional
public void deleteRecipe(String recipeId, String userId) {
    Recipe recipe = recipeRepo.findById(recipeId)
        .orElseThrow(() -> new IllegalArgumentException("Recipe not found"));

    User user = userService.findById(userId)
        .orElseThrow(() -> new IllegalArgumentException("User not found"));

```

		Недашківський В.С.			ІПЗ.КР.Б-121-25-ІІЗ	Арк.
		Фамт М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		84

```

    if (!securityService.canDelete(recipe, user)) {
        throw new SecurityException("No permission to delete this recipe");
    }

    deleteRelatedRecipeData(recipeId);
    recipeRepo.deleteById(recipeId);
}

private void deleteRelatedRecipeData(String recipeId) {
    List<Favorite> recipeFavorites = favoriteRepo.findByParentIdAndParentType(recipeId, ParentType.RECIPE);
    if (!recipeFavorites.isEmpty()) {
        favoriteRepo.deleteAll(recipeFavorites);
    }

    List<Rating> recipeRatings = ratingRepo.findByParentIdAndParentType(recipeId, ParentType.RECIPE);
    if (!recipeRatings.isEmpty()) {
        ratingRepo.deleteAll(recipeRatings);
    }

    deleteRecipeComments(recipeId);
}

private void deleteRecipeComments(String recipeId) {
    List<Comment> recipeComments = commentRepo.findByParentIdAndParentType(recipeId, ParentType.RECIPE);

    for (Comment comment : recipeComments) {
        deleteCommentAndReplies(comment.getId());
    }
}

private void deleteCommentAndReplies(String commentId) {
    List<Comment> replies = commentRepo.findByReplyTold(commentId);

    for (Comment reply : replies) {
        deleteCommentAndReplies(reply.getId());
    }

    commentRepo.deleteById(commentId);
}

private List<String> validateRecipeContent(Recipe recipe) {
    List<String> errors = new ArrayList<>();

    if (recipe.getName() == null || recipe.getName().trim().isEmpty()) {
        errors.add("Recipe name is required");
    }

    if (recipe.getName() != null && recipe.getName().length() > 100) {
        errors.add("Recipe name cannot exceed 100 characters");
    }

    if (recipe.getInstructions() == null || recipe.getInstructions().trim().isEmpty()) {

```

		Недашківський В.С.			ІПЗ.КР.Б-121-25-ІЗ	Арк.
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		85

```

        errors.add("Instructions are required");
    }

    if (recipe.getInstructions() != null && recipe.getInstructions().length() > 5000) {
        errors.add("Instructions cannot exceed 5000 characters");
    }

    if (recipe.getIngredients() == null || recipe.getIngredients().isEmpty()) {
        errors.add("At least one ingredient is required");
    }

    return errors;
}

private void handleStatusOnUpdate(Recipe existing, Recipe updated, User user) {
    if (existing.getStatus() == Status.PUBLISHED &&
        (!Objects.equals(existing.getInstructions(), updated.getInstructions()) ||
         !Objects.equals(existing.getIngredients(), updated.getIngredients()))) {

        updated.setStatus(Status.DRAFT);
        updated.setModeratorComments(null);
    }
}

private Page<RecipeWithAuthorDto> executeAggregation(List<AggregationOperation> pipeline, Pageable pageable) {
    try {
        AggregationResults<Document> results = mongoTemplate.aggregate(
            Aggregation.newAggregation(pipeline), "recipes", Document.class);

        Document result = results.getUniqueMappedResult();
        if (result == null) {
            return new PageImpl<>(Collections.emptyList(), pageable, 0);
        }

        List<Document> recipeDocuments = result.getList("recipes", Document.class);
        List<Document> metadata = result.getList("metadata", Document.class);

        long totalCount = metadata.isEmpty() ? 0 :
            metadata.get(0).getInteger("totalCount", 0);

        List<RecipeWithAuthorDto> recipeDtos = recipeDocuments.stream()
            .map(this::mapDocumentToDto)
            .collect(Collectors.toList());

        return new PageImpl<>(recipeDtos, pageable, totalCount);

    } catch (Exception e) {
        e.printStackTrace();
        return new PageImpl<>(Collections.emptyList(), pageable, 0);
    }
}

private RecipeWithAuthorDto mapDocumentToDto(Document doc) {

```

		Недашківський В.С.			ІІЗ.КР.Б-121-25-ІІЗ	Арк.
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		86

```

RecipeWithAuthorDto recipe = new RecipeWithAuthorDto();

Object idObj = doc.get("_id");
if (idObj instanceof ObjectId) {
    recipe.setId(((ObjectId) idObj).toHexString());
} else if (idObj instanceof String) {
    recipe.setId((String) idObj);
}

recipe.setName(doc.getString("name"));
recipe.setInstructions(doc.getString("instructions"));
recipe.setHistory(doc.getString("history"));

List<?> ingredientsList = doc.getList("ingredients", Object.class);
if (ingredientsList != null) {
    List<RecipeIngredient> ingredients = ingredientsList.stream()
        .filter(item -> item instanceof Document)
        .map(item -> {
            Document ingredientDoc = (Document) item;
            RecipeIngredient ingredient = new RecipeIngredient();
            ingredient.setIngredientId(ingredientDoc.getString("ingredientId"));
            ingredient.setAmount(ingredientDoc.getString("amount"));
            ingredient.setUnit(ingredientDoc.getString("unit"));
            ingredient.setNotes(ingredientDoc.getString("notes"));
            ingredient.setOptional(ingredientDoc.getBoolean("optional", false));
            return ingredient;
        })
        .collect(Collectors.toList());
    recipe.setIngredients(ingredients);
} else {
    recipe.setIngredients(new ArrayList<>());
}

List<?> photoUrls = doc.getList("photoUrls", String.class);
if (photoUrls != null) {
    recipe.setPhotoUrls((List<String>) photoUrls);
} else {
    recipe.setPhotoUrls(new ArrayList<>());
}

String sourceStr = doc.getString("source");
if (sourceStr != null) {
    try {
        recipe.setSource(Source.valueOf(sourceStr));
    } catch (IllegalArgumentException e) {
        recipe.setSource(Source.USER);
    }
} else {
    recipe.setSource(Source.USER);
}

String statusStr = doc.getString("status");
if (statusStr != null) {

```

		Недашківський В.С.			ІІЗ.КР.Б-121-25-ІІЗ	Арк.
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		87

```

try {
    recipe.setStatus(Status.valueOf(statusStr));
} catch (IllegalArgumentException e) {
    recipe.setStatus(Status.DRAFT);
}
} else {
    recipe.setStatus(Status.DRAFT);
}

Boolean containsAlcohol = doc.getBoolean("containsAlcohol");
recipe.setContainsAlcohol(containsAlcohol != null ? containsAlcohol : false);

List<?> categoryIds = doc.getList("categoryIds", Object.class);
if (categoryIds != null) {
    recipe.setCategoryIds(categoryIds.stream()
        .map(Object::toString)
        .collect(Collectors.toList()));
} else {
    recipe.setCategoryIds(new ArrayList<>());
}

List<?> tagIds = doc.getList("tagIds", Object.class);
if (tagIds != null) {
    recipe.setTagIds(tagIds.stream()
        .map(Object::toString)
        .collect(Collectors.toList()));
} else {
    recipe.setTagIds(new ArrayList<>());
}

Object createdAtObj = doc.get("createdAt");
if (createdAtObj instanceof Date) {
    recipe.setCreatedAt(((Date) createdAtObj).toInstant());
}

Object updatedAtObj = doc.get("updatedAt");
if (updatedAtObj instanceof Date) {
    recipe.setUpdatedAt(((Date) updatedAtObj).toInstant());
}

Object avgRating = doc.get("averageRating");
if (avgRating instanceof Number) {
    recipe.setAverageRating(((Number) avgRating).doubleValue());
} else {
    recipe.setAverageRating(0.0);
}

recipe.setIsPrivate(recipe.getStatus() == Status.DRAFT || recipe.getStatus() == Status.REJECTED);

Document authorDoc = doc.get("authorInfo", Document.class);
if (authorDoc != null) {
    RecipeWithAuthorDto.AuthorInfo authorInfo = new RecipeWithAuthorDto.AuthorInfo();

```

		Недашківський В.С.			ІІЗ.КР.Б-121-25-ІІЗ	Арк.
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		88

```

Object authorIdInDoc = authorDoc.get("_id");
if (authorIdInDoc instanceof ObjectId) {
    authorInfo.setId(((ObjectId) authorIdInDoc).toHexString());
} else if (authorIdInDoc instanceof String) {
    authorInfo.setId((String) authorIdInDoc);
}

String authorName = authorDoc.getString("name");
if (authorName != null && !authorName.trim().isEmpty()) {
    authorInfo.setName(authorName);
} else {
    authorInfo.setName("Невідомий автор");
}

String avatarUrl = authorDoc.getString("avatarUrl");
authorInfo.setAvatarUrl(avatarUrl != null ? avatarUrl : "");

Boolean bartenderVerified = authorDoc.getBoolean("bartenderVerified");
authorInfo.setBartenderVerified(bartenderVerified != null ? bartenderVerified : false);

recipe.setAuthor(authorInfo);

} else {

Object authorIdObj = doc.get("authorId");
if (authorIdObj != null) {
    try {
        String authorIdStr = authorIdObj.toString();
        Optional<User> userOpt = userService.findById(authorIdStr);
        if (userOpt.isPresent()) {
            User user = userOpt.get();

            RecipeWithAuthorDto.AuthorInfo authorInfo = new RecipeWithAuthorDto.AuthorInfo();
            authorInfo.setId(user.getId());
            authorInfo.setName(user.getName() != null ? user.getName() : "Невідомий автор");
            authorInfo.setAvatarUrl(user.getAvatarUrl() != null ? user.getAvatarUrl() : "");
            authorInfo.setBartenderVerified(user.isBartenderVerified());
            recipe.setAuthor(authorInfo);

        } else {
            RecipeWithAuthorDto.AuthorInfo fallbackAuthor = new RecipeWithAuthorDto.AuthorInfo();
            fallbackAuthor.setName("Невідомий автор");
            fallbackAuthor.setBartenderVerified(false);
            fallbackAuthor.setAvatarUrl("");
            recipe.setAuthor(fallbackAuthor);
        }
    } catch (Exception e) {
        System.out.println("Error in recipe direct user lookup: " + e.getMessage());
    }
}
}

```

		Недашківський В.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		89

```

        return recipe;
    }

    private RecipeWithAuthorDto mapToDto(Recipe recipe) {
        Optional<User> authorOpt = userRepo.findById(recipe.getAuthorId());
        return mapToDto(recipe, authorOpt.orElse(null));
    }

    private RecipeWithAuthorDto mapToDto(Recipe recipe, User author) {
        RecipeWithAuthorDto dto = new RecipeWithAuthorDto();
        dto.setId(recipe.getId());
        dto.setAuthorId(recipe.getAuthorId());
        dto.setName(recipe.getName());
        dto.setIngredients(recipe.getIngredients() != null ? recipe.getIngredients() : new ArrayList<>());
        dto.setInstructions(recipe.getInstructions());
        dto.setHistory(recipe.getHistory());
        dto.setPhotoUrls(recipe.getPhotoUrls() != null ? recipe.getPhotoUrls() : new ArrayList<>());
        dto.setSource(recipe.getSource());
        dto.setContainsAlcohol(recipe.getContainsAlcohol());
        dto.setCategoryIds(recipe.getCategoryIds() != null ? recipe.getCategoryIds() : new ArrayList<>());
        dto.setTagIds(recipe.getTagIds() != null ? recipe.getTagIds() : new ArrayList<>());
        dto.setAverageRating(recipe.getAverageRating() != null ? recipe.getAverageRating() : 0.0);
        dto.setCreatedAt(recipe.getCreatedAt());
        dto.setUpdatedAt(recipe.getUpdatedAt());

        switch (recipe.getStatus()) {
            case DRAFT:
                dto.setStatus(Status.DRAFT);
                break;
            case SUBMITTED:
                dto.setStatus(Status.SUBMITTED);
                break;
            case PUBLISHED:
                dto.setStatus(Status.PUBLISHED);
                break;
            case REJECTED:
                dto.setStatus(Status.REJECTED);
                break;
            default:
                dto.setStatus(Status.DRAFT);
        }

        dto.setIsPrivate(dto.getStatus() == Status.DRAFT || dto.getStatus() == Status.REJECTED);
        if (author != null) {
            RecipeWithAuthorDto.AuthorInfo authorInfo = new RecipeWithAuthorDto.AuthorInfo();
            authorInfo.setId(author.getId());
            authorInfo.setName(author.getName());
            authorInfo.setAvatarUrl(author.getAvatarUrl());
            authorInfo.setBartenderVerified(author.isBartenderVerified());
            dto.setAuthor(authorInfo);
        }

        return dto;
    }

    public void updateAverageRating(String recipeId) {

```

		Недашківський В.С.			ІІЗ.КР.Б-121-25-ІІЗ	Арк.
		Фант М.О.				90
Змн.	Арк.	№ докум.	Підпис	Дата		

```

List<Rating> ratings = ratingRepo.findByIdAndParentType(
    recipeId, ParentType.RECIPE);
if (!ratings.isEmpty()) {
    double average = ratings.stream()
        .mapToInt(Rating::getStars)
        .average()
        .orElse(0.0);

    Optional<Recipe> recipeOpt = recipeRepo.findById(recipeId);
    if (recipeOpt.isPresent()) {
        Recipe recipe = recipeOpt.get();
        recipe.setAverageRating(Math.round(average * 10.0) / 10.0);
        recipeRepo.save(recipe);
    }
}
}
}

```

Лістинг коду типу сповіщень:

```

public enum NotificationType {
    COMMENT_REPLY,
    RECIPE_APPROVED,
    RECIPE_REJECTED,
    ARTICLE_APPROVED,
    ARTICLE_REJECTED,
    VERIFICATION_RESULT,
    NEW_RECIPE_BY_FOLLOWED,
    NEW_ARTICLE_BY_FOLLOWED,
    NEW_RECIPE_SUBMITTED,
    NEW_ARTICLE_SUBMITTED,
    NEW_VERIFICATION_REQUEST,
    GENERIC
}

```

Лістинг коду сервісу сповіщень:

```

@Service
public class NotificationService {
    private final NotificationRepository repo;
    private final FollowRepository followRepo;
    private final UserRepository userRepo;

    public NotificationService(NotificationRepository repo, FollowRepository followRepo, UserRepository userRepo) {
        this.repo = repo;
        this.followRepo = followRepo;
        this.userRepo = userRepo;
    }

    public Notification send(Notification n) {
        n.setCreatedAt(Instant.now());
        n.setRead(false);
    }
}

```

		Недашківський В.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Фант М.О.				91
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        return repo.save(n);
    }

    public Optional<Notification> findById(String id) {
        return repo.findById(id);
    }

    public List<Notification> findUnreadByUserId(String uid) {
        return repo.findByUserIdAndReadFalseOrderByCreatedAtDesc(uid);
    }

    public List<Notification> findAllByUserId(String uid, int limit) {
        return repo.findByUserIdOrderByCreatedAtDesc(uid).stream()
            .limit(limit)
            .toList();
    }

    public Notification markRead(Notification n) {
        n.setRead(true);
        return repo.save(n);
    }

    public void markAllAsRead(String userId) {
        List<Notification> unreadNotifications = repo.findByUserIdAndReadFalse(userId);
        unreadNotifications.forEach(n -> n.setRead(true));
        repo.saveAll(unreadNotifications);
    }

    public long getUnreadCount(String userId) {
        return repo.countByUserIdAndReadFalse(userId);
    }

    public void sendFollowNotification(String targetUserId, User follower) {
        Notification notification = new Notification();
        notification.setUserId(targetUserId);
        notification.setType(NotificationType.NEW_RECIPE_BY_FOLLOWED);
        notification.setMessage(String.format("%s підписався на вас", follower.getName()));
        send(notification);
    }

    public void sendNewRecipeNotification(Recipe recipe, User author) {
        List<String> followerIds = followRepo.findByTargetUserId(author.getId())
            .stream()
            .map(follow -> follow.getUserId())
            .toList();

        for (String followerId : followerIds) {

```

		Недашківський В.С.			ІІЗ.КР.Б-121-25-ІІЗ	Арк.
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		92

```

        Notification notification = new Notification();
        notification.setUserId(followerId);
        notification.setType(NotificationType.NEW_RECIPE_BY_FOLLOWED);
        notification.setMessage(String.format("%s опублікував новий рецепт: %s",
            author.getName(), recipe.getName()));
        notification.setMetadata(Map.of(
            "recipeId", recipe.getId(),
            "recipeName", recipe.getName(),
            "userName", author.getName()
        ));
        send(notification);
    }
}

public void sendRecipeApprovedNotification(String userId, String recipeName, String recipeId) {
    Notification notification = new Notification();
    notification.setUserId(userId);
    notification.setType(NotificationType.RECIPE_APPROVED);
    notification.setMessage(String.format("Ваш рецепт '%s' було схвалено та опубліковано", recipeName));
    notification.setMetadata(Map.of(
        "recipeId", recipeId,
        "recipeName", recipeName
    ));
    send(notification);
}

public void sendRecipeRejectedNotification(String userId, String recipeName, String recipeId, String reason) {
    Notification notification = new Notification();
    notification.setUserId(userId);
    notification.setType(NotificationType.RECIPE_REJECTED);
    notification.setMessage(String.format("Ваш рецепт '%s' було відхилено. Причина: %s", recipeName, reason));
    notification.setMetadata(Map.of(
        "recipeId", recipeId,
        "recipeName", recipeName,
        "rejectionReason", reason
    ));
    send(notification);
}

public void sendNewRecipeSubmissionNotification(Recipe recipe, User author) {
    List<User> admins = userRepo.findAll().stream()
        .filter(user -> user.getRoles() != null &&
            user.getRoles().contains(com.nedavasyt.app.bartenderhubbackend.model.enums.Role.ADMIN))
        .toList();

    for (User admin : admins) {
        List<Notification> existingNotifications = repo.findByUserIdAndTypeAndReadFalse(

```

		Недашківський В.С.			ІІЗ.КР.Б-121-25-ІІЗ	Арк.
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		93

```

        admin.getId(), NotificationType.NEW_RECIPE_SUBMITTED);

if (existingNotifications.isEmpty()) {
    Notification notification = new Notification();
    notification.setUserId(admin.getId());
    notification.setType(NotificationType.NEW_RECIPE_SUBMITTED);
    notification.setMessage("Новий рецепт подано на модерацию");
    notification.setMetadata(Map.of(
        "recipeId", recipe.getId(),
        "recipeName", recipe.getName(),
        "authorName", author.getName(),
        "count", 1
    ));
    send(notification);
} else {
    Notification existing = existingNotifications.get(0);
    Map<String, Object> metadata = existing.getMetadata();
    int currentCount = metadata != null ? (Integer) metadata.getOrDefault("count", 1) : 1;

    existing.setMessage(String.format("Нові рецепти подано на модерацию (%d)", currentCount + 1));
    existing.setCreatedAt(Instant.now());
    existing.setMetadata(Map.of(
        "count", currentCount + 1
    ));
    repo.save(existing);
}
}

public void sendNewVerificationRequestNotification(VerificationRequest request, User user) {
    List<User> admins = userRepo.findAll().stream()
        .filter(adminUser -> adminUser.getRoles() != null &&
            adminUser.getRoles().contains(Role.ADMIN))
        .toList();

    String roleDisplayName = getRoleDisplayName(request.getTargetRole());

    for (User admin : admins) {
        List<Notification> existingNotifications = repo.findByUserIdAndTypeAndReadFalse(
            admin.getId(), NotificationType.NEW_VERIFICATION_REQUEST);

        if (existingNotifications.isEmpty()) {
            Notification notification = new Notification();
            notification.setUserId(admin.getId());
            notification.setType(NotificationType.NEW_VERIFICATION_REQUEST);
            notification.setMessage(String.format("Нова заявка на підвищення до ролі %s", roleDisplayName));
            notification.setMetadata(Map.of(

```

		Недашківський В.С.			ІІЗ.КР.Б-121-25-ІІЗ	Арк.
		Фант М.О.				94
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        "requestId", request.getId(),
        "targetRole", request.getTargetRole().toString(),
        "userName", user.getName(),
        "count", 1
    ));
    send(notification);
} else {
    Notification existing = existingNotifications.get(0);
    Map<String, Object> metadata = existing.getMetadata();
    int currentCount = metadata != null ? (Integer) metadata.getOrDefault("count", 1) : 1;

    existing.setMessage(String.format("Нові заявки на підвищення ролі (%d)", currentCount + 1));
    existing.setCreatedAt(Instant.now());
    existing.setMetadata(Map.of(
        "count", currentCount + 1
    ));
    repo.save(existing);
}
}
}

public void sendVerificationResultNotification(String userId, boolean approved, Role targetRole, String reason) {
    Notification notification = new Notification();
    notification.setUserId(userId);
    notification.setType(NotificationType.VERIFICATION_RESULT);

    String roleDisplayName = getRoleDisplayName(targetRole);

    if (approved) {
        notification.setMessage(String.format("Вашу заявку на підвищення до ролі %s схвалено!", roleDisplayName));
        notification.setMetadata(Map.of(
            "verificationStatus", "APPROVED",
            "targetRole", targetRole.toString(),
            "roleDisplayName", roleDisplayName
        ));
    } else {
        notification.setMessage(String.format("Вашу заявку на підвищення до ролі %s відхилено. Причина: %s",
            roleDisplayName, reason));
        notification.setMetadata(Map.of(
            "verificationStatus", "REJECTED",
            "targetRole", targetRole.toString(),
            "roleDisplayName", roleDisplayName,
            "rejectionReason", reason
        ));
    }

    send(notification);
}

```

		Недашківський В.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Фант М.О.				95
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    }

    public void delete(String notificationId) {
        repo.deleteById(notificationId);
    }

    public void deleteBulk(List<String> notificationIds, String userId) {
        List<Notification> userNotifications = notificationIds.stream()
            .map(id -> repo.findById(id))
            .filter(Optional::isPresent)
            .map(Optional::get)
            .filter(n -> n.getUserId().equals(userId))
            .toList();

        userNotifications.forEach(n -> repo.deleteById(n.getId()));
    }

    public void sendCommentReplyNotification(String userId, String commenterName, String recipeOrArticleTitle) {
        Notification notification = new Notification();
        notification.setUserId(userId);
        notification.setType(NotificationType.COMMENT_REPLY);
        notification.setMessage(String.format("%s відповів на ваш коментар до %s",
            commenterName, recipeOrArticleTitle));
        send(notification);
    }

    public void sendNewArticleNotification(NewsArticle article, User author) {
        List<String> followerIds = followRepo.findByTargetUserId(author.getId())
            .stream()
            .map(follow -> follow.getUserId())
            .toList();

        for (String followerId : followerIds) {
            Notification notification = new Notification();
            notification.setUserId(followerId);
            notification.setType(NotificationType.NEW_ARTICLE_BY_FOLLOWED);
            notification.setMessage(String.format("%s опублікував нову статтю: %s",
                author.getName(), article.getTitle()));
            notification.setMetadata(Map.of(
                "articleId", article.getId(),
                "articleTitle", article.getTitle(),
                "authorName", author.getName()
            ));
            send(notification);
        }
    }
}

```

```

public void sendArticleApprovedNotification(String userId, String articleTitle, String articleId) {
    Notification notification = new Notification();
    notification.setUserId(userId);
    notification.setType(NotificationType.ARTICLE_APPROVED);
    notification.setMessage(String.format("Вашу статтю '%s' було схвалено та опубліковано", articleTitle));
    notification.setMetadata(Map.of(
        "articleId", articleId,
        "articleTitle", articleTitle
    ));
    send(notification);
}

public void sendArticleRejectedNotification(String userId, String articleTitle, String articleId, String reason) {
    Notification notification = new Notification();
    notification.setUserId(userId);
    notification.setType(NotificationType.ARTICLE_REJECTED);
    notification.setMessage(String.format("Вашу статтю '%s' було відхилено. Причина: %s", articleTitle, reason));
    notification.setMetadata(Map.of(
        "articleId", articleId,
        "articleTitle", articleTitle,
        "rejectionReason", reason
    ));
    send(notification);
}

public void sendNewArticleSubmissionNotification(NewsArticle article, User author) {
    List<User> admins = userRepo.findAll().stream()
        .filter(user -> user.getRoles() != null &&
            user.getRoles().contains(Role.ADMIN))
        .toList();

    for (User admin : admins) {
        List<Notification> existingNotifications = repo.findByUserIdAndTypeAndReadFalse(
            admin.getId(), NotificationType.NEW_ARTICLE_SUBMITTED);

        if (existingNotifications.isEmpty()) {
            Notification notification = new Notification();
            notification.setUserId(admin.getId());
            notification.setType(NotificationType.NEW_ARTICLE_SUBMITTED);
            notification.setMessage("Нова стаття подана на модерацію");
            notification.setMetadata(Map.of(
                "articleId", article.getId(),
                "articleTitle", article.getTitle(),
                "authorName", author.getName(),
                "count", 1
            ));
            send(notification);
        }
    }
}

```

		Недашківський В.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Фант М.О.				97
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    } else {
        Notification existing = existingNotifications.get(0);
        Map<String, Object> metadata = existing.getMetadata();
        int currentCount = metadata != null ? (Integer) metadata.getOrDefault("count", 1) : 1;

        existing.setMessage(String.format("Нові статті подано на модерацію (%d)", currentCount + 1));
        existing.setCreatedAt(Instant.now());
        existing.setMetadata(Map.of(
            "count", currentCount + 1
        ));
        repo.save(existing);
    }
}

private String getRoleDisplayName(Role role) {
    switch (role) {
        case BARTENDER:
            return "Бармен";
        case AUTHOR:
            return "Автор";
        case ADMIN:
            return "Адміністратор";
        default:
            return "Користувач";
    }
}
}

```

		Недашківський В.С.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Фант М.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		98