

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ

І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

ПОГОДЖЕНО

Декан факультету

ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ

Завідувач кафедри інформаційних
систем і технологій

(підпис) **Ігор БОЛБОТ**

(підпис) **Михайло ШВИДЕНКО**

«__» _____ 2026 р.

«__» _____ 2026 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

**на тему: «Розробка інформаційної системи для фотодіагностики рослин за
допомогою нейронних мереж»**

Спеціальність «122 Комп'ютерні науки»

Освітньо-професійна програма «Комп'ютерні науки»

Гарант освітньої програми

д.е.н., професор

(підпис) **Роман РУДЕНСЬКИЙ**

**Керівник бакалаврської
кваліфікаційної роботи**

д.ф. з к.н.

(підпис) **Роман ЗОЛОТУХА**

Виконав

(підпис) **Микола ЛЕСЬ**

Київ-2026

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ
І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ
Факультет інформаційних технологій**

ЗАТВЕРДЖУЮ

**Завідувач кафедри інформаційних систем і
технологій**

к.е.н., доцент _____ Михайло ШВИДЕНКО
(підпис)

«___» _____ 2026 р.

**ЗАВДАННЯ
ДО ВИКОНАННЯ БАКАЛАВРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ
ЗДОБУВАЧУ**

Лесю Миколі Олександровичу

Спеціальність 122 – «Комп'ютерні науки»

ОП «Комп'ютерні науки»

Тема бакалаврської кваліфікаційної роботи «Розроблення інформаційної системи фотодіагностики рослин на основі нейронних мереж» затверджена наказом від «06» січня 2026 р. № 46 «С»

Термін подання завершеної роботи на кафедру «22» травня 2026 р.

Вихідні дані до бакалаврської кваліфікаційної роботи (проєкту):

Фотозображення рослин; матеріали предметної області; native iPhone-застосунок; локальний Mac-сервер; Gemma 4 E4B; RAG-джерела; результати тестування.

Перелік питань, що підлягають дослідженню:

- Аналіз предметної області; проєктування інформаційного забезпечення; розробка програмного забезпечення; тестування та впровадження системи.

Дата видачі завдання «06» січня 2026 р.

**Керівник бакалаврської
кваліфікаційної роботи** _____
(підпис)

Роман ЗОЛОТУХА

**Завдання взяв до
виконання** _____
(підпис)

Микола ЛЕСЬ

ЗМІСТ

Перелік умовних позначень, символів, одиниць, скорочень і термінів.....	4
Вступ.....	6
1 Системний аналіз предметної області.....	9
1.1 Постановка завдання.....	9
1.2 Огляд інформаційних джерел та існуючих рішень.....	11
1.3 Моделювання предметної області.....	13
1.4 Функціональні та нефункціональні вимоги.....	18
1.5 Ризики предметної області та принцип безпечного завершення.....	21
2 Інформаційне забезпечення.....	23
2.1 Логічна модель даних.....	23
2.2 Вибір системи управління інформаційною базою.....	25
2.3 Створення інформаційної бази.....	26
2.3.1 Фрагмент SQL-структури.....	28
2.4 Аналіз нормалізації інформаційної бази.....	28
2.5 Модель життєвого циклу даних.....	30
2.6 Дані фотодіагностичного контуру.....	33
3 Прикладне програмне забезпечення.....	37
3.1 Організаційна структура програмного забезпечення.....	37
3.2 Вибір інструментарію для створення програмного забезпечення.....	41
3.3 Алгоритмізація та програмування програмних модулів.....	42
3.4 Програмна реалізація iPhone-клієнта.....	53
3.5 Програмна реалізація серверної частини.....	57
3.6 Потокова взаємодія і формування відповіді.....	58
4 Рекомендації щодо впровадження та експлуатації системи.....	60
4.1 Тестування системи.....	60
4.2 Вимоги до апаратного та програмного забезпечення.....	65
4.3 Склад інсталяційного пакету.....	66
4.4 Порядок експлуатації системи.....	67
4.5 Критерії приймання та межі подальшого розвитку.....	69
4.6 Узагальнення проєктних рішень основної частини.....	71
Висновки.....	73
Список використаних джерел.....	75
Додаток А. Графічна частина.....	79
Додаток Б. SQL-лістинг інформаційної бази.....	84
Додаток В. Фрагменти програмних модулів.....	88
Додаток Г. Декларація про академічну доброчесність та використання ШІ90	

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

Таблиця 0.2

Основні скорочення та терміни

Позначення	Пояснення
ACL	Access Control List, правила контролю доступу до ресурсів.
API	Application Programming Interface, програмний інтерфейс взаємодії компонентів.
CV	Computer Vision, комп'ютерний зір.
LLM	Large Language Model, велика мовна модель.
RAG	Retrieval-Augmented Generation, генерація відповіді з використанням знайденого контексту.
SSE	Server-Sent Events, потоковий серверний канал передавання подій.
Організаційний контур	Ізольована область даних, що відповідає холдингу або іншій організаційній одиниці; у схемах може позначатися ідентифікатором tenant.
Рольова участь користувача	Зв'язок користувача з організаційним контуром, підрозділом і роллю доступу; у схемах відповідає сутності membership.
Одиниця пошуку	Мінімальний дозволений фрагмент джерела знань, який може бути переданий у RAG-контекст і використаний для цитування.
Режим відповіді	Тип обробки запиту: звичайний діалог, відповідь на основі джерел або агрономічна фотодіагностика.
Цитата	Структурований доказовий запис, який пов'язує відповідь із конкретним джерелом, локатором і версією матеріалу.
Стан виконання	Перевірений технічний стан запиту, черги, моделі, джерела або відповіді, який не підміняється припущенням.
Журнал дій	Незмінюваний запис службових і користувацьких подій, потрібний для безпеки, супроводу та перевірки.
Ідентифікатор	Стабільне позначення діагностичного класу, за яким добираються

агрономічног о класу	відповідні довідкові матеріали.
Стан роботи системи	Перевірений стан моделі, сегментаційного модуля, черги, джерела або відповіді, який показується користувачу чи оператору.
Інтеграційний шар мовної моделі	Ізольований шар взаємодії серверної частини з локальним середовищем LM Studio та Gemma 4 E4B.
Безпечне завершення	Завершення сценарію без діагнозу або твердження, якщо доказів недостатньо.
Перевірка тверджень	Зіставлення текстових заяв системи з матеріалами перевірки та дозволеними джерелами.

ВСТУП

Фотодіагностика рослин є актуальною задачею для аграрного виробництва, оскільки раннє виявлення ознак хвороб, стресових станів або пошкоджень безпосередньо впливає на врожайність, витрати на захист рослин та швидкість ухвалення управлінських рішень. У великих господарствах візуальний огляд поля або тепличної ділянки часто виконується нерівномірно, а людський фактор призводить до різної якості первинної оцінки. Мобільна інформаційна система може скоротити цей розрив, якщо вона дозволяє швидко зафіксувати фото, передати його в контрольований серверний контур і отримати обґрунтовану відповідь із посиланнями на джерела.

Метою роботи є розробка інформаційної системи AgroChat для фотодіагностики рослин за допомогою нейронних мереж у серверно-орієнтованій (server-first) архітектурі з native iPhone-клієнтом. Такий підхід відокремлює користувацький інтерфейс від серверної логіки, а також дозволяє чесно контролювати можливості локального інференсу, зберігання даних, доступу до знань і аудиту дій.

Об'єктом дослідження є процес отримання, збереження, аналізу та пояснення фотодіагностичних даних рослин у мобільній інформаційній системі. Предметом дослідження є архітектурні, інформаційні та програмні засоби побудови iPhone-first системи, яка поєднує комп'ютерний зір, локальний LLM-інференс, RAG-пошук та рольову модель доступу.

У роботі використано методи системного аналізу, об'єктно-орієнтованого моделювання, проєктування інформаційних баз, клієнт-серверної розробки, потокової взаємодії через SSE, тестування серверних контрактів, аналізу життєвого циклу вкладень і доказового обмеження можливостей системи. Для програмної реалізації застосовано SwiftUI для iPhone-клієнта, Python/FastAPI для серверної частини, SQLite-backed durable stores для локального серверного

контуру, локальне файлове сховище, інтеграційний шар LM Studio для Gemma 4 E4B і окремий агрономічний сегментаційний модуль.

Практична значущість роботи полягає в тому, що система не обмежується демонстраційним класифікатором зображень. Вона включає чат як основну робочу поверхню, автентифікацію, рольовий доступ, RAG-контур знань, цитування, аудит, стан роботи системи і перевірений агрономічний режим. Завдяки цьому результат можна розглядати як основу інформаційної системи для аграрного користувача, а не лише як окрему модель комп'ютерного зору.

Апробація результатів роботи. Основні положення бакалаврської кваліфікаційної роботи апробовано у вигляді тез доповіді «Інформаційна система фотодіагностики рослин на основі нейронних мереж» на VII Всеукраїнській науково-практичній конференції студентів та аспірантів «Теоретичні та прикладні аспекти розробки комп'ютерних систем '2026» (29 квітня 2026 року, НУБіП України, м. Київ). Автор тез: Лесь М.О.; науковий керівник: PhD, старший викладач Золотуха Р.А.

Пояснювальна записка містить 90 сторінок, 34 рисунки, 22 таблиці, 29 використаних джерел та 4 додатки. У першому розділі виконано системний аналіз предметної області. У другому розділі описано інформаційне забезпечення і нормалізацію інформаційної бази. У третьому розділі подано програмне забезпечення, алгоритмічні блок-схеми та фрагменти програмної реалізації. У четвертому розділі наведено рекомендації щодо впровадження, експлуатації, тестування і протоколу виправлення помилок.

Науково-практична новизна роботи полягає в поєднанні трьох контурів, які часто розглядаються окремо: мобільного UX, локального LLM/RAG inference та комп'ютерно-зорової фотодіагностики. У межах AgroChat ці частини зведені в один контрольований процес, де кожна відповідь має стан, джерела, власника і технічне походження.

Важливою методологічною вимогою є відмова від прихованих або маркетингових можливостей. Якщо модельний шлях не підтверджує аудіо,

диплом не описує аудіо як підтриману функцію. Якщо агрономічна модель структурно має багато класів, але доказово прийнято лише частину, у роботі фіксується саме обмежена підтримка, а не універсальна діагностика всіх хвороб.

Такий підхід відповідає інженерному характеру бакалаврської роботи з комп'ютерних наук: результат оцінюється не тільки за наявністю інтерфейсу, а й за якістю архітектурних меж, інформаційної моделі, алгоритмічної дисципліни, тестів і доказів працездатності на цільовому локальному контурі.

1 СИСТЕМНИЙ АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Постановка завдання

Завдання бакалаврської роботи полягає у створенні інформаційної системи, яка допомагає користувачеві виконувати фотодіагностику рослин за допомогою нейронних мереж і довідкового агрономічного контексту. На практиці це означає, що система повинна приймати зображення рослини з iPhone, передавати його в локальний серверний контур, зберігати як контрольоване вкладення, запускати аналіз, отримувати структурований результат і повертати користувачеві зрозумілий висновок.

На відміну від окремого класифікатора, AgroChat вирішує ширшу інформаційну задачу. Користувач працює не з формою завантаження файлу, а з чат-орієнтованою (chat-first) поверхнею, у якій текстове питання, зображення, джерела знань, цитати і стан виконання об'єднані в один сценарій. Саме тому iPhone-клієнт розглядається як головна поверхня системи, а локальний Mac-сервер - як власник правил, даних і інференсу.

Система повинна бути придатною для багатокористувацького аграрного середовища. Це вимагає моделі організаційного контуру холдингу, ієрархії організаційних вузлів, ролей власника холдингу, агронома й аналітика, приватних нотаток, історії чатів, журналу дій та операторського контролю. Фотодіагностика не може відокремлюватися від цих правил, оскільки зображення, знання і результати можуть належати конкретному холдингу або користувачеві.

Вхідними даними є фотозображення рослин, текстові запити користувача, PDF-джерела знань, службові дані користувачів і організаційної ієрархії, а також технічний стан локального інференсу. Вихідними даними є чатова відповідь, структурований агрономічний результат, джерела з цитуванням, статуси життєвого циклу вкладення і записи аудиту.

Основне обмеження полягає в тому, що система не повинна робити неправдивих заяв про можливості. Якщо аудіоканал не підтримується, він не описується як реалізований. Якщо агрономічний клас не пройшов калібрування або не має доказового RAG-контексту, система повинна завершувати сценарій контрольованою відмовою, а не вигадувати діагноз.

Таким чином, постановка задачі охоплює не лише нейромережеву обробку зображення, а й повний інформаційний контур: автентифікацію, рольовий доступ, файл, модель, знання, відповідь, цитату, журналювання та чесний стан можливостей.

Таблиця 1.1

Вимоги до інформаційної системи фотодіагностики

Група вимог	Зміст вимоги	Реалізація в AgroChat
Клієнтська поверхня	Зручний мобільний інтерфейс для створення запиту і прикріплення фото	Native iPhone app на SwiftUI
Серверна логіка	Обробка прав, вкладень, чатів, знань і інференсу	Python/FastAPI backend на локальному Mac
Діагностика	Аналіз фото через нейромережевий контур	Окремий сегментаційний модуль і перевірка агрономічного класу
Пояснюваність	Наявність джерел і цитат у відповіді	RAG-відповідь із посиланнями на джерела
Безпека	Ізоляція даних за організацією і роллю	Контроль доступу та журналювання
Чесність	Не заявляти непідтверджені можливості	Відображення перевірених, обмежених або недоступних станів

1.2 Огляд інформаційних джерел та існуючих рішень

Огляд предметної області спирається на два типи джерел: наукові роботи з комп'ютерного зору, наборів даних, RAG і довіри до LLM-відповідей [3-13], а також існуючі мобільні продукти для визначення рослин або діагностики проблем рослин [14-17]. Такий поділ дозволяє оцінити не лише алгоритмічні підходи, а й реальні користувацькі сценарії.

Plantix позиціонується як мобільний crop doctor для фермерів і надає діагностику пошкоджень рослин за фото, поради щодо лікування та довідкову інформацію для культур. Його перевага полягає у практичній аграрній спеціалізації, проте для дипломної задачі важливим обмеженням є залежність від зовнішнього сервісного контуру і відсутність локального RAG-пошуку в межах даних конкретного господарства.

PictureThis орієнтований переважно на ідентифікацію рослин і догляд, але також має автоматичну діагностику проблем за фото. Перевагою є широкий споживчий UX і велика база описів, недоліком для аграрного B2B-сценарію є слабша прив'язка до ролей холдингу, внутрішніх PDF-джерел, аудиту та локального інференсу.

Pl@ntNet є citizen-science платформою для ідентифікації видів рослин за фотографіями. Вона цінна як приклад масового мобільного CV-застосування, але її основна мета - ботанічна ідентифікація, а не фотодіагностика хвороб із цитуванням джерел, рольовим доступом і контрольованим RAG-контекстом.

Agrio зосереджується на здоров'ї рослин і взаємодії з агрономічною спільнотою. Такий підхід корисний для консультацій, однак у межах даної роботи ключовою відмінністю AgroChat є локальна серверна частина, операторськи виданий доступ, контрольоване збереження чатів, журнал дій і заборона неперевіраних діагностичних тверджень.

Наукові огляди з виявлення хвороб рослин показують, що CNN, YOLO-подібні детектори, сегментація і пояснюваність моделей ефективні для аналізу

листоків та симптомів [3-10]. Окремі роботи з PlantVillage також показують роль відкритих наборів даних для навчання й порівняння моделей [7; 8]. Водночас ці джерела підкреслюють залежність якості від набору даних, умов зйомки, класового покриття і пояснюваності, тому дипломна система не може обмежитися одним числом впевненості.

Дослідження RAG і галюцинацій у великих мовних моделях показують, що генеративна модель потребує зовнішніх джерел, перевірки тверджень і обмеження контекстом [11-13]. Саме тому AgroChat використовує RAG не як декоративне доповнення, а як механізм, який пов'язує відповідь із дозволеною одиницею пошуку і збереженою цитатою.

AgroChat об'єднує сильні сторони аналогів і усуває частину їхніх обмежень для локального навчально-практичного сценарію: iPhone дає мобільну робочу поверхню, Mac-сервер контролює доступ і дані, сегментаційний модуль виконує CV-частину, а RAG-контур подає мовній моделі агрономічні джерела, прив'язані до визначеного класу. Це зменшує ризик довільної відповіді і робить висновок відтворюванішим.

Таблиця 1.2

Переваги та недоліки існуючих рішень

Рішення	Переваги	Недоліки щодо задачі AgroChat	Висновок для проєкту
Plantix	Аграрна спеціалізація, фото-діагностика, поради	Не орієнтовано на локальну роботу з даними конкретного господарства	Потрібні локальні джерела, контроль доступу і журналювання
PictureThis	Зручний UX, ідентифікація, auto problem diagnosis	Більше consumer-care, ніж робочий процес агропідприємства	Потрібна B2B-модель доступу
Pl@ntNet	Citizen-science база, мобільна ідентифікація	Не є системою хвороб із діагностичним RAG	Корисний приклад mobile CV

	рослин		
Agrio	Фокус на plant health і консультації	Залежність від зовнішньої екосистеми, немає локального контуру мовної моделі	Потрібний контрольований локальний процес
AgroChat	Локальний інференс, iPhone-first UX, рольовий доступ, RAG-циткування	Підтримка класів розширюється лише після перевірки	Пріоритет чесності й доказовості

На відміну від Plantix, PictureThis, Pl@ntNet і Agrio, AgroChat не позиціонується як універсальна публічна база всіх рослинних проблем. Його дипломна цінність полягає у контрольованому локальному контурі: native iPhone-клієнт, локальний Mac-сервер, Gemma 4 E4B без хмарного інференсу, рольовий контроль доступу, RAG-циткування, журналювання і агрономічна підтримка лише для перевірених сценаріїв.

1.3 Моделювання предметної області

Предметна область системи складається з двох взаємопов'язаних частин: агрономічної та інформаційно-технічної. Агрономічна частина описує рослину, зображення, симптоми, клас хвороби або пошкодження, калібрування впевненості й довідкові матеріали. Інформаційно-технічна частина описує користувача, рольову участь, чат, вкладення, джерело знань, цикл відповіді, цитату і подію журналу дій.

Користувачами системи є власник холдингу, агроном і аналітик. Власник холдингу має ширші права видимості в межах організаційного контуру, агроном працює з агрономічними запитами й фотодіагностикою, аналітик працює з даними й аналітичним контекстом. Проте фактичним носієм прав є не

сама особа, а її рольова участь у конкретному вузлі, що дозволяє одному користувачу мати різні ролі в різних організаційних вузлах.

Основний сценарій можна описати як послідовність: автентифікація через операторськи виданий login token, вибір активної ролі користувача, створення або відкриття чату, прикріплення фото, надсилання запиту в режимі Agronomic, серверна перевірка стану вкладення, запуск сегментаційного модуля, RAG-добір агрономічного контексту, генерація відповіді, збереження історії відповіді і цитат.

У навчально-проектних матеріалах уже підготовлено діаграми розгортання та моделі даних для AgroChat. Під час включення в диплом вони використовуються як графічна основа, але текстова інтерпретація приведена до фактичної реалізації: локальний контур базується на durable SQLite-backed stores, а не на PostgreSQL як обов'язковій поточній СУБД.

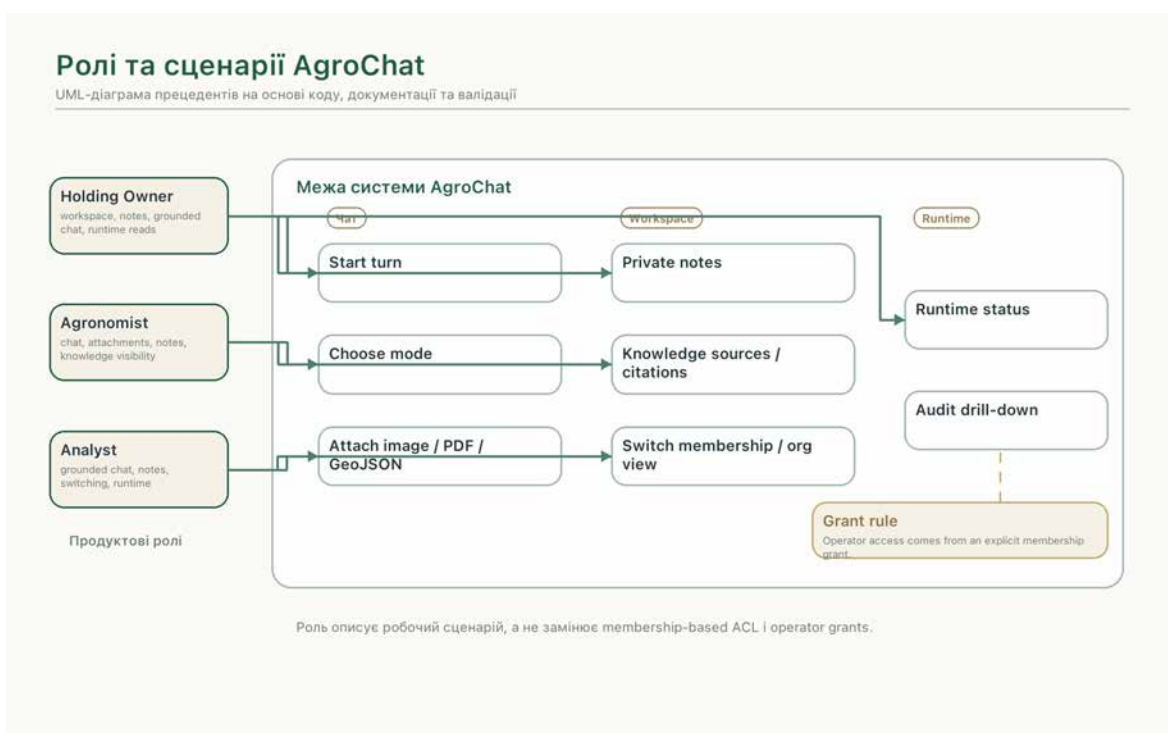


Рисунок 1 - Діаграма варіантів використання AgroChat

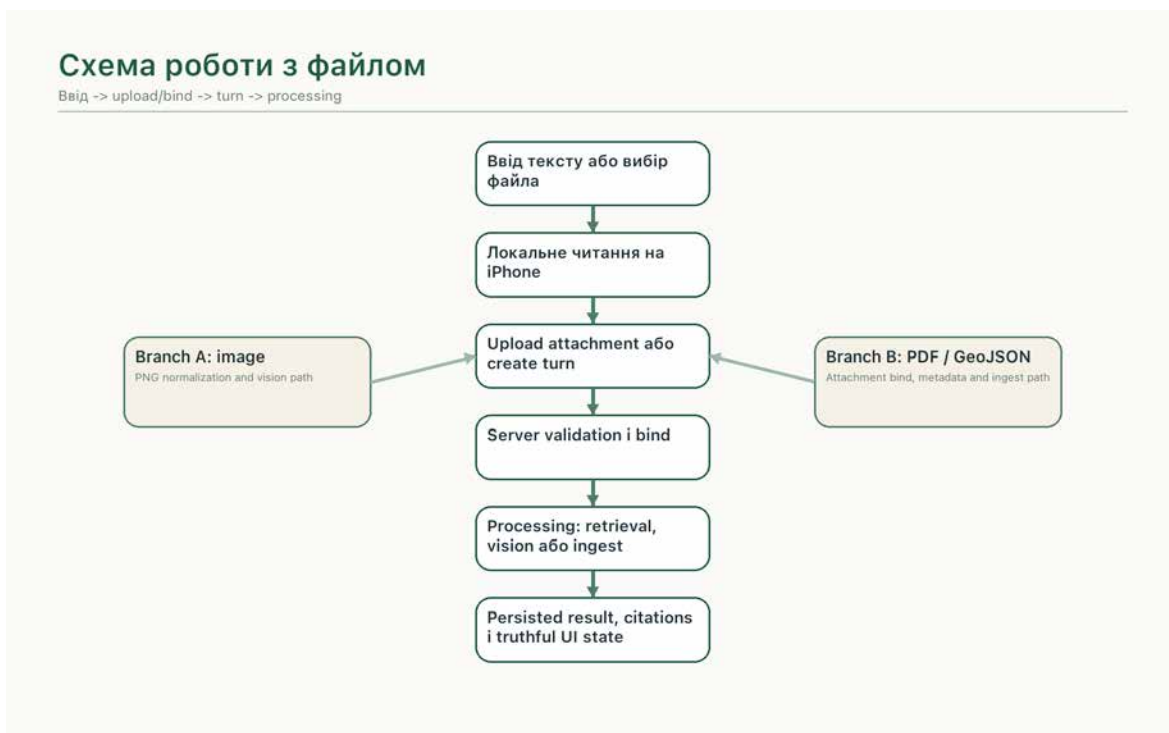


Рисунок 2 - Узагальнений activity-flow для запиту з вкладенням

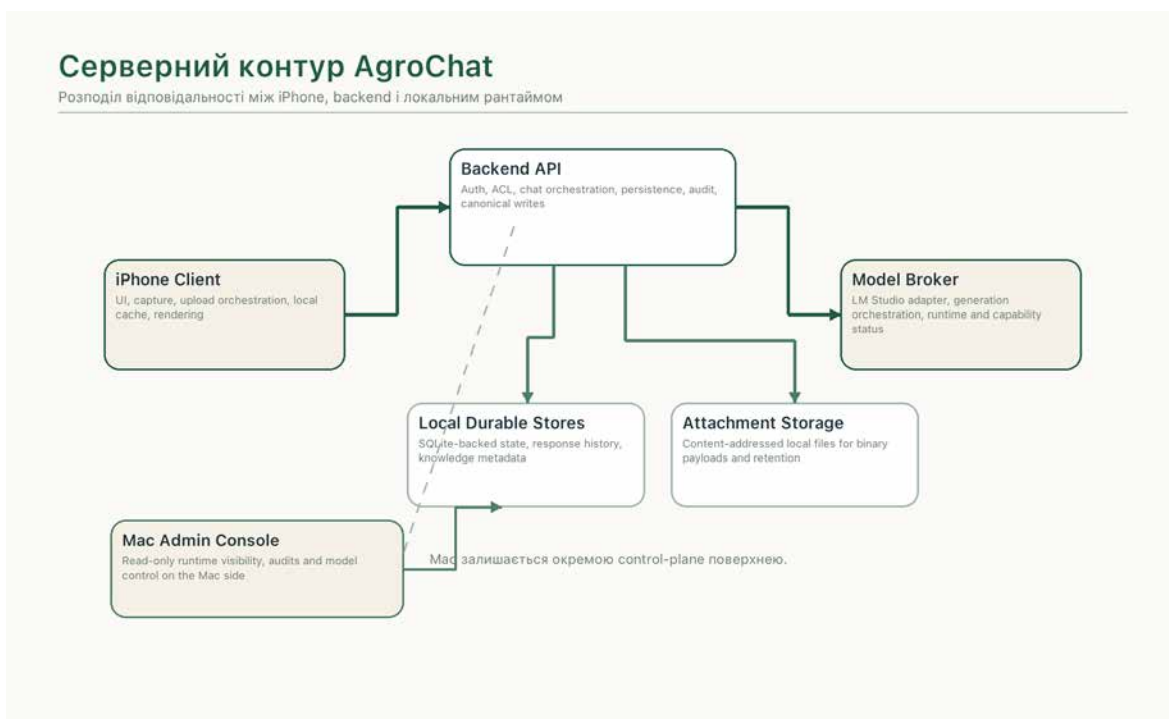


Рисунок 3 - Архітектурна схема iPhone-first AgroChat



Рисунок 4 - Діаграма прецедентів системи фотодіагностики рослин AgroChat

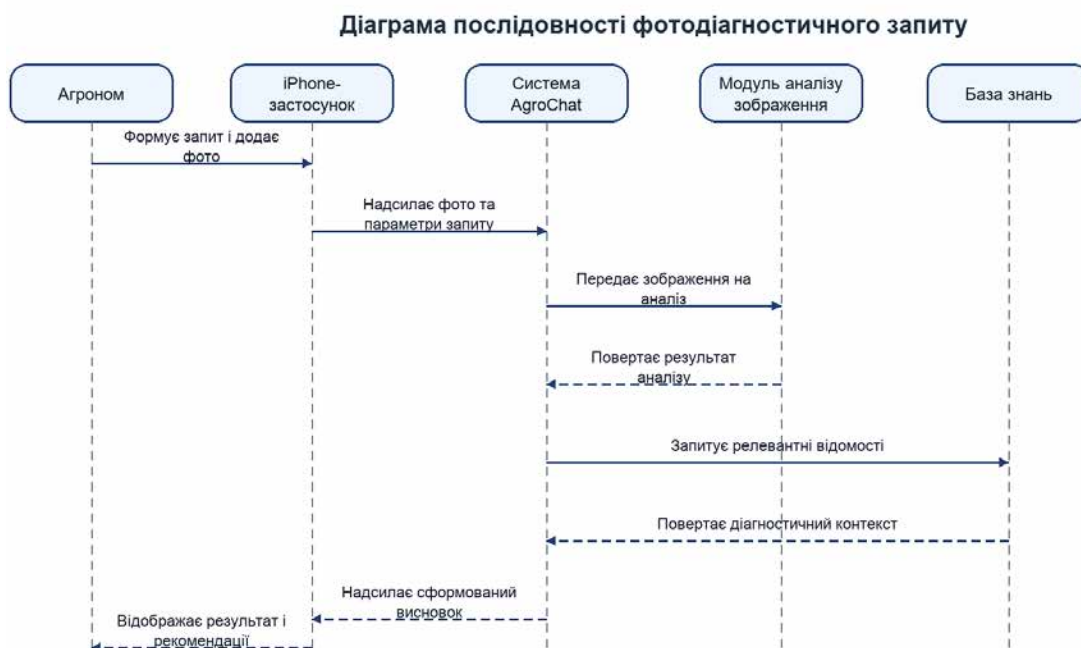


Рисунок 5 - Діаграма послідовності фотодіагностичного запиту AgroChat

Діаграма діяльності фотодіагностичного сценарію

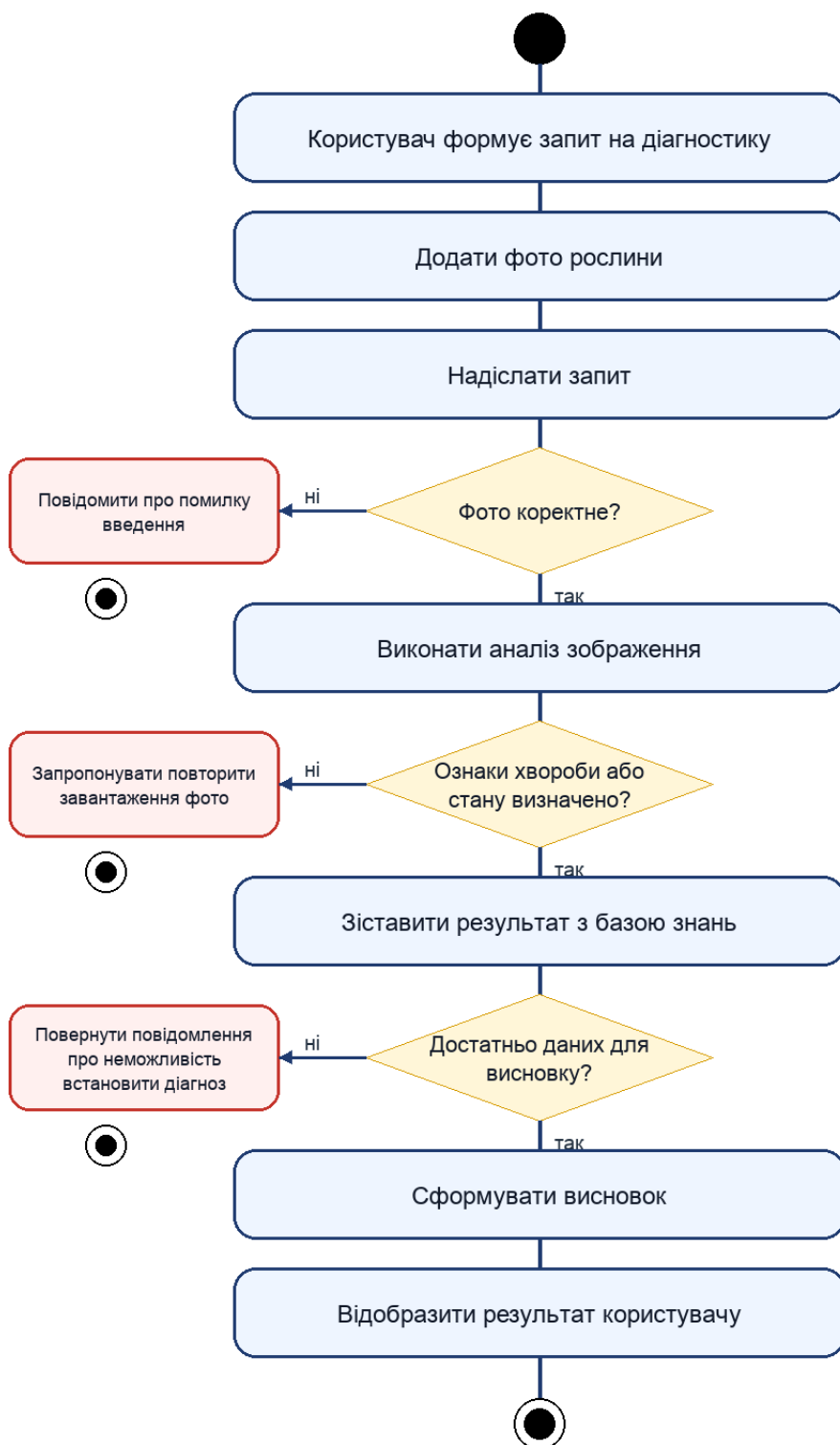


Рисунок 6 - Діаграма діяльності фотодіагностичного сценарію AgroChat

Діаграма варіантів використання демонструє, що система орієнтована на робочий сценарій аграрного користувача, а не на ізольований запуск моделі. Користувач не лише завантажує зображення, а й отримує історію, цитати, стан роботи системи і доступ до знань у межах своїх прав.

Activity-flow показує серверно-орієнтований (server-first) розподіл відповідальності: iPhone готує запит і передає дані, але рішення щодо ACL, життєвого циклу вкладення, діагностичного режиму, RAG-контексту і збереження результату приймає сервер. Це зменшує ризик розходження між UI і канонічним станом.

Архітектурна схема відображає локальну топологію раннього впровадження: iPhone-клієнт, Mac-сервер, інтеграційний шар LM Studio, довговічні локальні сховища і локальний сегментаційний модуль. Хмарні вузли не додаються, оскільки вони суперечили б вимогам роботи.

1.4 Функціональні та нефункціональні вимоги

Функціональні вимоги до AgroChat випливають із цільового сценарію агронома, який працює з рослиною в полі або в іншому виробничому середовищі. Користувач повинен мати можливість увійти в систему, відкрити чат, додати фотозображення, поставити уточнювальне питання, вибрати режим відповіді та отримати результат, який не приховує технічний стан обробки. Кожний крок має бути відображений у UI як зрозумілий стан, а не як неявне очікування.

Першою функціональною вимогою є автентифікація без публічної самореєстрації. У системі прийнято підхід із токеном входу, який видає оператор: оператор на Mac заздалегідь створює або знаходить користувача, прив'язує його до рольової участі і видає токен входу. Така схема краще відповідає аграрному B2B-сценарію, де доступ до даних холдингу не може відкриватися через довільну публічну реєстрацію.

Другою вимогою є робота з активним контекстом ролі користувача. Користувач не просто має профіль; він працює в межах певного організаційного контуру, підрозділу і ролі. Це означає, що той самий користувач у майбутньому може мати різні права у різних підрозділах, а сервер повинен виконувати кожний запит у чітко визначеному контексті.

Третьою вимогою є підтримка чатового сценарію з історією. Фотодіагностика рідко завершується одним повідомленням, бо користувач може уточнювати симптоми, надсилати додатковий опис або питати про подальші дії. Тому chat history є не декоративною функцією, а інформаційною основою повторного аналізу і контролю рішень.

Четверта вимога стосується вкладень. Система повинна розрізняти завантаження, перевірку, прив'язку, обробку і помилку. Якщо файл не пройшов перевірку або не належить активній області доступу, він не може потрапити в модельний запит. Це особливо важливо для зображень, бо неправильне змішування чужого вкладення із новим чатом може призвести до некоректної діагностики або витоку даних.

П'ята вимога - явні режими відповіді. Звичайна відповідь, відповідь на основі джерел і агрономічна фотодіагностика мають різну семантику, різні обмеження і різну відповідальність. Тому в AgroChat використовується контрольований режим відповіді, а не набір неузгоджених прапорців, які могли б випадково ввімкнути інструментальний або зоровий шлях без чіткої політики.

Нефункціональні вимоги включають приватність, локальність, передбачуваність, пояснюваність і відмовостійкість. Локальний Мас-сервер зменшує залежність від хмарного інференсу, але водночас вимагає чесного відображення стану моделі, черги, готовності сегментаційного модуля і помилок. Система не повинна маскувати відмову як успіх.

Вимога до пояснюваності реалізується через відповіді з цитуванням. Користувачеві недостатньо отримати рекомендацію; він повинен бачити, з якого джерела взято твердження. Це не перетворює систему на повноцінну

експертну платформу, але підвищує контрольованість відповіді порівняно з вільною генерацією LLM.

Вимога до продуктивності в локальному середовищі формулюється обережно. Система повинна підтримувати прийнятний інтерактивний цикл для одного користувача або невеликої групи, але не заявляти масштабування хмарного класу. Для локального шляху Gemma особливо важливі стан черги, стан зайнятості, межі скасування і відсутність неправдивого твердження про відновлення або повторення відповіді.

Вимога до супроводу означає, що архітектура повинна залишатися зрозумілою. Компоненти розділені за відповідальністю: iPhone-інтерфейс, серверні контракти, сховище, інтеграційний шар мовної моделі, сегментаційний модуль і ресурси. Такий розподіл полегшує тестування і дозволяє замінювати окремі частини без переписування всієї системи.

Таблиця 1.3

Функціональні та нефункціональні вимоги

Категорія	Вимога	Проектне рішення
Функціональна	Вхід через виданий оператором token	Login-token exchange і bearer session
Функціональна	Чат з історією	Chat/message/response run контур
Функціональна	Фото як контрольоване вкладення	Upload -> validate -> bind -> process
Функціональна	Діагностика за зображенням	Сегментаційний модуль і режим Agronomic
Нефункціональна	Локальність інференсу	LM Studio на локальному Mac-сервері
Нефункціональна	Пояснюваність	RAG із цитуванням джерел
Нефункціональна	Безпека	Рольовий контроль доступу

Нефункціонал ьна	Чесність можливостей	Явні стани успіху, обмеження або відмови
---------------------	----------------------	---

1.5 Ризики предметної області та принцип безпечного завершення

Фотодіагностика рослин належить до сценаріїв, у яких помилка може мати практичні наслідки. Неправильна порада щодо хвороби або стресового стану може спричинити невчасну обробку, зайві витрати або втрату часу. Тому система не повинна поводитися як безумовний діагностичний авторитет.

Перший ризик полягає в якості зображення. Фото може бути темним, розмитим, занадто далеким, містити кілька рослин або фонові об'єкти. Якщо сегментаційний модуль не досягає порогу впевненості, система повинна повідомити про невизначеність і попросити якісніше фото, а не створювати слабкий діагноз.

Другий ризик - класова невідповідність. Навіть якщо модель має структурний список класів, не кожний клас може бути достатньо перевіраним для практичного висновку. Для частини класів можуть бракувати калібрування, якісних прикладів або підготовленого RAG-контексту. У таких випадках відповідь повинна завершуватися повідомленням про недоступність класу або відсутність відповідного джерела.

Третій ризик - галюцинація мовної моделі. LLM може добре формулювати текст, але це не гарантує фактичної правильності. У AgroChat ця небезпека зменшується через RAG за агрономічним класом і правила цитування: модель отримує обмежений контекст, а відповідь має посилатися на дозволені джерела.

Четвертий ризик пов'язаний із витоком даних між організаційними контурами. Аграрні холдинги можуть мати власні фотографії, документи, нотатки й операційні дані. Якщо пошук джерел відфільтрувати після генерації,

витік уже відбувся. Тому перевірка доступу є обов'язковою до добору джерел і формування відповіді.

П'ятий ризик - змішування продуктової архітектури та середовища перевірки. Матеріали тестування корисні для підтвердження сценаріїв, але не повинні перетворюватися на другий сервер або альтернативну архітектуру. Диплом описує один основний сервер застосунку плюс окремий агрономічний сегментаційний модуль.

Шостий ризик - завищені маркетингові твердження. Якщо в документах або презентації написати, що система підтримує всі 116 класів, аудіо або загальне розуміння будь-яких документів, це створить неправдиве очікування. У дипломі тому використовується перевірка тверджень: кожне твердження має відповідати фактичним матеріалам перевірки.

Принцип безпечного завершення означає, що при невизначеності система обирає контрольовану відмову, а не приблизний успіх. Для користувача це може виглядати менш ефектно, але для аграрної діагностики така поведінка професійніша, ніж впевнена неправда.

З погляду проектування безпечне завершення реалізується через завершальні стани, типізовані стани даних і чіткі контракти. Наприклад, невизначена сегментація не є помилкою інтерфейсу, а нормальним завершальним станом, який повідомляє користувачу, що достатніх підстав для діагнозу немає.

Таким чином, системний аналіз показує, що предметна область потребує не лише моделі комп'ютерного зору, а й дисципліни доступу, доказовості, відмов і обмеження заяв про можливості. Саме ці вимоги визначають архітектуру AgroChat.

2 ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ

2.1 Логічна модель даних

Інформаційне забезпечення AgroChat має підтримувати не лише збереження зображень і результатів діагностики, а й повний контур безпеки, чатів, знань і доказовості. Тому логічна модель побудована навколо ресурсів, прив'язаних до організаційного контуру, і ролі користувача в цьому контурі.

На верхньому рівні організаційний контур відповідає холдингу. Кожний контур має кореневий організаційний вузол типу holding, а нижчі вузли можуть відповідати підрозділам, регіонам, фермам або іншим операційним одиницям. Така модель не є схемою фіксованої глибини, тому вона придатна для подальшого розширення організаційної структури.

Користувач описує особу, але права виконання запиту визначає її рольова участь у конкретному організаційному контурі. Це важливо для системи фотодіагностики, оскільки один агроном може працювати в різних областях доступу, а доступ до фото, чатів, знань і нотаток повинен залежати від активної ролі.

Фотодіагностичний контур формують вкладення із зображенням рослини, результат сегментації, таксономічний клас, правило калібрування, агрономічний результат, джерело агрономічних знань, одиниця пошуку і цитата. Разом вони описують шлях від зображення до доказового висновку.

Цитата є ключовою сутністю для пояснюваності. Вона не просто зберігає посилання на документ, а пов'язує конкретний цикл відповіді з джерелом, яке було дозволене для активної ролі користувача і використане у відповіді.

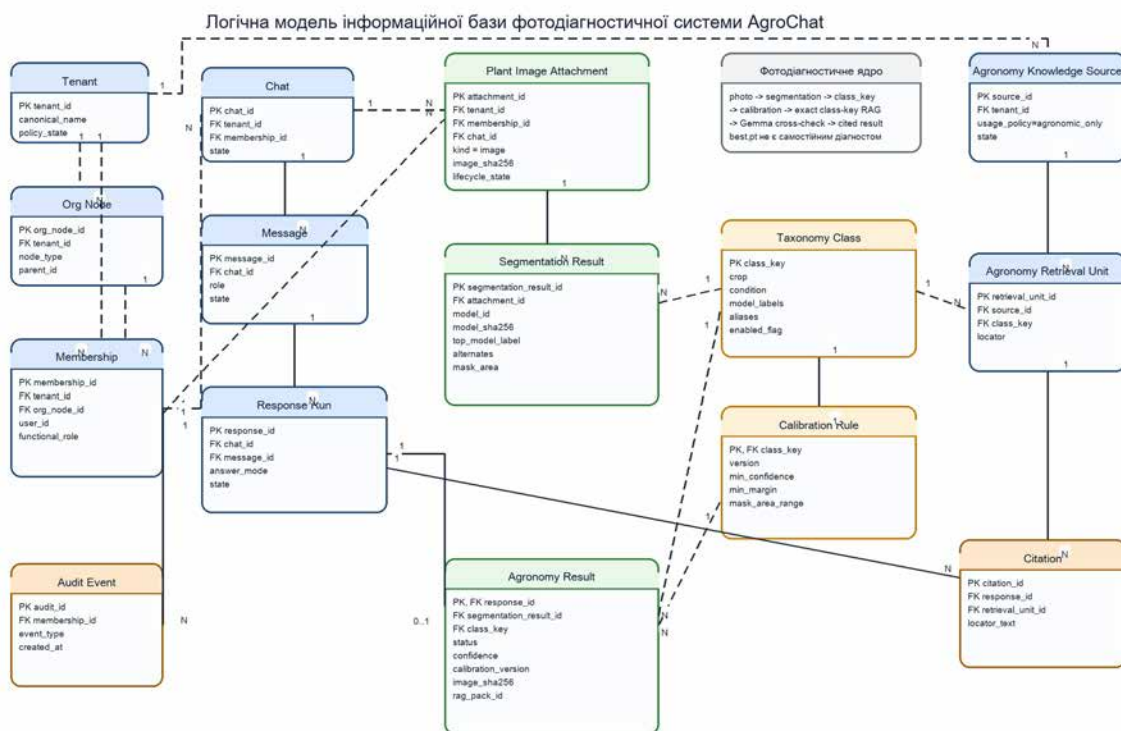


Рисунок 7 - Логічна модель інформаційної бази фотодіагностичної системи AgroChat

Діаграма розгортання AgroChat з акцентом на комп'ютерний зір

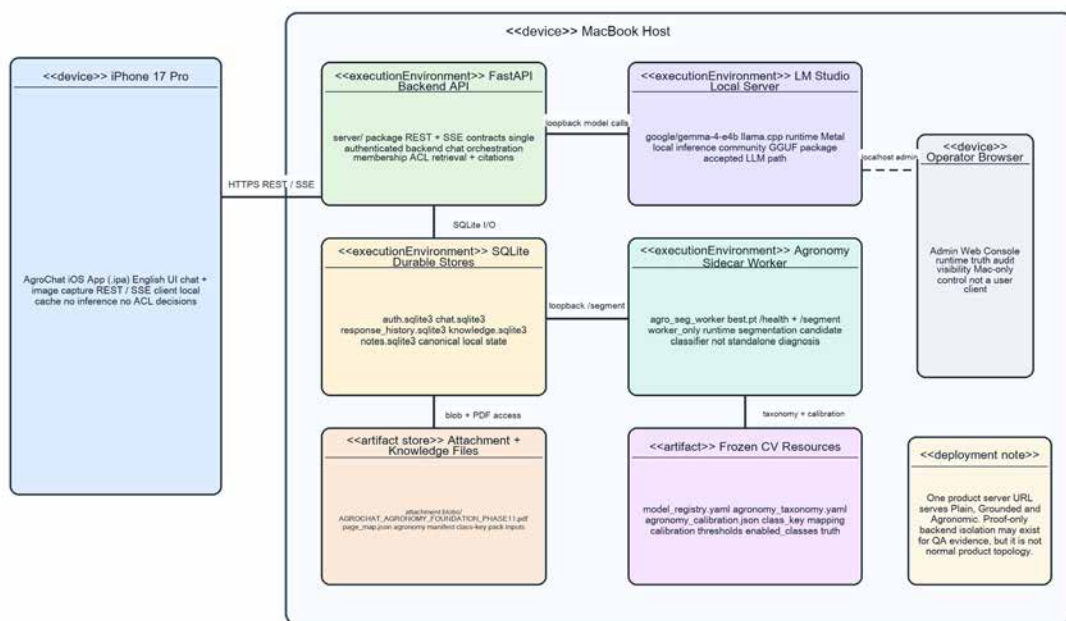


Рисунок 8 - Діаграма розгортання AgroChat у локальному середовищі з native iPhone-клієнтом і локальний Мас-сервер

Таблиця 2.1

Основні сутності логічної моделі

Сутність	Призначення	Ключова умова цілісності
tenant	Верхній контур ізоляції даних	Ресурс не змінює організаційний контур після створення
org_node	Ієрархічна організаційна структура	Дерево є ациклічним і має один кореневий вузол
membership	Рольова участь користувача	Кожний запит виконується під однією активною роллю
attachment	Зображення, PDF або geodata-файл	Не використовується до валідованого стану
agronomy_result	Підсумковий результат фотодіагностики	Посилається на результат сегментації та агрономічний клас
citation	Доказове посилання відповіді	Може вказувати лише на дозволене джерело

2.2 Вибір системи управління інформаційною базою

Для поточного локального серверного контуру системи використано durable SQLite-backed stores. Такий вибір відповідає фактичній архітектурі роботи: локальний Mac-сервер на етапі розробки та раннього тестування зберігає узгоджений стан у локальних довговічних сховищах, а не потребує зовнішнього кластеру бази даних.

SQLite є доречною основою для цього етапу, оскільки система працює в одному локальному серверному вузлі, потребує транзакційності, простого розгортання, WAL-поведінки, збереження історії, станів і журналу дій, а також мінімального операційного навантаження. Для бакалаврського проєкту це також зменшує складність інсталяційного пакету.

Водночас архітектура не замикає систему на SQLite як незмінну майбутню інфраструктуру. Контракти серверної частини, клієнта і сховища розділені так, щоб пізніший перехід на іншу основу даних не ламав iPhone-клієнт і не змінював семантику організаційного контуру, ролі користувача, чату або цитати.

У ранніх навчально-проектних матеріалах частина схем описувала PostgreSQL/pgvector як цільову модель. У цій пояснювальній записці ці фрагменти використовуються лише як навчальні SQL/ER-заготовки або як додатки, а основний текст приведено до фактичної реалізації: локальна версія спирається на SQLite-backed stores і детермінований локальний індекс для PDF-джерел.

Таблиця 2.2

Обґрунтування вибору SQLite-backed stores

Критерій	Вимога системи	Відповідність SQLite-backed сховищ
Локальність	Робота на Mac-сервері без хмари	Не потребує окремого серверу БД
Транзакційність	Збереження чатів, сесій, журналу дій і задач	Підтримка ACID і WAL
Операційна простота	Швидке локальне розгортання	Мінімальний інсталяційний контур
Еволюційність	Можливість майбутньої зміни сховища	Контракти відокремлені від клієнта
Доказовість	Фіксація стану роботи та можливостей	Локальні довговічні записи

2.3 Створення інформаційної бази

Створення інформаційної бази в AgroChat відбувається не як один монолітний SQL-файл, а як набір серверних компонентів зберігання, кожний з

яких відповідає за конкретний контур даних. Auth store обслуговує користувачів, ролі, сесії і login tokens; chat store зберігає чати й повідомлення; attachment store фіксує файли та їхній життєвий цикл; knowledge store веде джерела, фрагменти, задачі і метадані пошуку.

Для двійкових файлів використовується content-addressed local file storage. Це відокремлює великі об'єкти від табличних метаданих і дозволяє перевіряти власника, hash, media type, стан перевірки і прив'язку до чату або джерела знань.

PDF-джерело проходить явний життєвий цикл: реєстрація, витягування тексту, нормалізований вміст, сторінкові фрагменти, локальний індекс і стан готовності. Такий процес дозволяє отримати RAG-пошук із цитуванням без заяв про підтримку DOCX, CSV, URL або OCR.

Для агрономічного режиму інформаційна база містить окремі ресурси: таксономію, калібрування, реєстр моделі, сформований агрономічний PDF-пакет і карту сторінок. Ці ресурси підтримують безпечну поведінку: якщо клас не ввімкнений або для нього немає відповідного розділу джерела, система не створює діагноз.

Таблиця 2.3

Локальні сховища серверної частини

Сховище	Дані	Роль у системі
Auth store	users, memberships, sessions, login tokens, auth audit	Контроль доступу й авторизація
Chat/history store	chats, messages, response runs	Канонічна історія діалогу
Attachment store	file metadata, hash, state, bind records	Життєвий цикл вкладень
Knowledge store	sources, chunks, vectors, jobs	PDF/RAG і походження цитат
State/capability records	ідентичність моделі, контекст, позначки стану	Чесна поверхня стану системи

	можливостей (capability labels)	
File storage	binary blobs	Зберігання фото, PDF, GeoJSON

2.3.1 Фрагмент SQL-структури

```
-- AgroChat CV installation package.
-- Physical target: SQLite 3 durable local store for the photodiagnostic contour.
-- Актуальна реалізація використовує кілька локальних SQLite-сховищ за межами
сервісів;
-- цей фрагмент узагальнює фізичні сутності для документації розгортання.

PRAGMA foreign_keys = ON;
PRAGMA journal_mode = WAL;

CREATE TABLE IF NOT EXISTS ref_membership_roles (
-- ... повний навчальний SQL-лістинг інформаційної бази наведено у Додатку Б ...
```

Наведений фрагмент використовується як коротка навчальна ілюстрація створення об'єктів бази даних. У Додатку Б подано повний релевантний SQL-лістинг фізичної схеми, який підтверджує структуру таблиць і ключові обмеження цілісності. У тексті диплома цей фрагмент інтерпретується через актуальний локальний контур SQLite-сховищ, а не як твердження про обов'язковий PostgreSQL-runtime для поточної реалізації.

2.4 Аналіз нормалізації інформаційної бази

Логічна модель інформаційної бази AgroChat відповідає вимогам реляційності та третій нормальній формі в межах тих сутностей, які зберігаються у табличних довговічних сховищах. Кожна основна сутність має первинний ключ: `tenant\_id`, `org\_node\_id`, `membership\_id`, `chat\_id`, `message\_id`, `attachment\_id`, `source\_id`, `retrieval\_unit\_id`, `response\_id` або `citation\_id`.

Перша нормальна форма забезпечується атомарністю атрибутів. Наприклад, рольова участь, стан вкладення, тип організаційного вузла, ідентифікатор агрономічного класу, локатор цитати і hash файлу зберігаються як окремі значення, а не як змішані текстові описи. Повторювані групи не зберігаються в одному полі: повідомлення винесені в `message`, цитати - в `citation`, рольові участі - в окрему таблицю від users.

Друга нормальна форма дотримується через відсутність залежності неключових атрибутів від частини складеного ключа. У моделі переважно використовуються surrogate primary keys, тому атрибути сутності залежать від повного ідентифікатора запису. Зв'язки many-to-one або one-to-many оформлюються через foreign-key-like поля, а не через дублювання описових даних.

Третя нормальна форма забезпечується винесенням довідкових і повторно використовуваних понять в окремі сутності або закриті state/type catalogs. Полі, lifecycle states, taxonomy class keys, usage policies і завершальні статуси відповіді (response terminal statuses) не дублюються як довільні описи в різних таблицях, а використовуються як контрольовані значення, які інтерпретуються серверними контрактами.

Транзитивні залежності мінімізовано. Наприклад, користувач не є безпосереднім власником ресурсу організаційного контуру; права визначаються через рольову участь, яка поєднує користувача, організаційний контур, вузол і функціональну роль. Цитата не зберігає довільний текст джерела, а посилається на одиницю пошуку, джерело, локатор і hash версії. Це дозволяє перевіряти джерело без дублювання повного документа в кожній відповіді.

Довідкові таблиці і контрольовані стани логічно відокремлені: ролі, стани, статуси, таксономічні класи, правила калібрування і політики використання знань не змішуються з діловими записами. Завдяки цьому зміна назви або стану не вимагає переписувати історичні записи чатів, відповідей і цитат та не порушує аудит.

У підсумку нормалізацію перевірено за такими ознаками: усі сутності мають первинні ключі; атрибути атомарні; повторюваних груп немає; неключові атрибути залежать від повного ключа; транзитивних залежностей немає; reference tables винесені окремо; roles/states/statuses нормалізовані через довідники.

Таблиця 2.4

Перевірка нормалізації інформаційної бази

Вимога	Реалізація в AgroChat	Результат
Первинні ключі	Кожна основна сутність має стабільний id	Ідентифікація записів однозначна
Атомарність	Ролі, стани, hash, class_key, locator зберігаються окремо	Виконується 1НФ
Відсутність повторюваних груп	Повідомлення, цитати й рольові участі винесені в окремі сутності	Виконується 1НФ
Повна залежність від ключа	Атрибути належать конкретній сутності, не частині складеного ключа	Виконується 2НФ
Немає транзитивних залежностей	Права через рольову участь, джерела через одиницю пошуку і джерело знань	Виконується 3НФ
Довідники й стани	Roles, statuses, taxonomy, calibration відокремлені	Модель підтримує цілісність

2.5 Модель життєвого циклу даних

Життєвий цикл даних у AgroChat починається до моменту звернення до нейронної мережі. Користувач повинен мати дійсну сесію, активну рольову участь і доступ до організаційного контуру. Це дозволяє серверу одразу прив'язати майбутній чат, вкладення і цикл відповіді до правильного власника.

Коли користувач додає зображення, iPhone читає файл або фото з системного джерела, готує корисне навантаження і передає його через API. На цьому етапі клієнт може показувати попередній перегляд, але він не є канонічним доказом того, що файл прийнято сервером. Канонічний стан виникає тільки після серверного запису завантаження.

Після завантаження сервер зберігає метадані, hash, тип медіа і первинний стан. Якщо файл пошкоджений, має неприпустимий тип або не відповідає очікуваному сценарію, він не переходить у стан, придатний для прив'язки. Така модель запобігає ситуації, коли модель отримує неперевірений вхід.

Прив'язку відокремлено від завантаження, бо один файл може бути завантажений, але ще не пов'язаний із конкретним чатом або джерелом знань. Для фотодіагностики прив'язка до чату означає, що саме це зображення використовується в конкретному ході діалогу. Для PDF прив'язка до джерела знань означає, що документ стає частиною процесу пошуку.

Цикл відповіді створюється після повідомлення користувача. Він має власний стан, номер спроби і зв'язок із початковим повідомленням. Це дозволяє підтримувати повторне формування без перезаписування попередньої відповіді. У дипломній системі така модель важлива, бо історія діагностичних спроб не повинна зникати.

Джерело знань має інший життєвий цикл. Після реєстрації воно може перейти у витягування тексту, поділ на фрагменти, індексацію і готовий стан. Кожний стан повинен бути видимим, тому що користувач не повинен отримувати відповідь на основі джерел із документа, який ще не пройшов обробку.

Агрономічні ресурси мають напівстатичний життєвий цикл. Таксономія, калібрування, реєстр моделі і сформований агрономічний PDF-пакет не змінюються довільно під час одного запиту. Вони повинні бути заморожені або явно версіоновані, інакше повторне формування могло б отримати іншу інтерпретацію того самого зображення.

Життєвий цикл цитати завершує доказовий контур. Коли відповідь сформована, вона має зберегти не лише текст, а й посилання на одиниці пошуку. Це дозволяє пізніше відкрити деталі джерела і перевірити, що твердження не з'явилося без підтримки.

Життєвий цикл журналу дій охоплює операції, які важливі для безпеки та супроводу: видачу токена входу, обмін, відкликання, перемикання ролі, дії керування моделлю, операції зі знаннями і критичні завершальні стани. Аудит не є UI-функцією, але він робить систему відповідальною.

Усі ці життєві цикли проєктуються як типізовані стани. Це зменшує кількість неявних проміжних станів і спрощує тестування: кожний endpoint або UI-компонент може перевіряти не довільний текст, а закритий набір станів.

Таблиця 2.5

Життєві цикли основних ресурсів

Ресурс	Початковий стан	Проміжні стани	Термінальний або робочий стан
Session	issued	active / revoked / expired checks	active або безпечне завершення
Attachment	uploaded	validated, bound, processing	ready або failed
Response run	queued	running, cancelling, failed	completed або terminal error
Knowledge source	registered	extracting, chunking, indexing	ready або failed
Agronomy result	pending	segmented, calibrated, retrieved	ok або завершальний статус (terminal status)
Audit event	created	immutable append-only	persisted

2.6 Дані фотодіагностичного контуру

Фотодіагностичний контур має власну інформаційну структуру, яка не зводиться до загального повідомлення чату. Центральним вхідним об'єктом є вкладення із зображенням. Воно описує не лише файл, а й походження: хто завантажив, у якому організаційному контурі, з яким hash і в якому стані перевірки.

Результат сегментації зберігає технічний результат комп'ютерно-зорового модуля. У ньому важливі мітка класу, ідентифікатор класу, впевненість, версія моделі, hash зображення і технічні метадані виконання. Без цих полів неможливо пояснити, чому система обрала саме такий агрономічний шлях.

Таксономічний клас відокремлює сиру мітку моделі від стабільного доменного ідентифікатора класу. Це критично, тому що сирі мітки можуть змінюватися при зміні ваг моделі або назв у наборі даних. Стабільний ідентифікатор класу дозволяє зв'язати модельний результат із RAG-документом і правилами калібрування.

Правило калібрування визначає, чи можна вважати клас достатньо надійним для діагнозу, видимого користувачеві. Якщо впевненість нижче порогу або клас не входить до дозволеної підмножини, система не повинна генерувати діагноз. Це один із найважливіших елементів чесного опису можливостей.

Джерело агрономічних знань містить розділи за ідентифікаторами класів. У поточній реалізації таким джерелом є сформований агрономічний PDF-пакет із картою сторінок. Система не шукає агрономічний контекст у загальному корпусі для відповідей на основі джерел, щоб не змішувати звичайний RAG і спеціалізовані агрономічні знання.

Агрономічна одиниця пошуку відповідає конкретному фрагменту джерела для ідентифікатора класу. Вона є авторизованим контекстом, який

можна передати у складання відповіді. Якщо одиниця пошуку відсутня, відповідь повинна завершитися станом ``rag_section_missing``.

Агрономічний результат з'єднує всі попередні елементи: зображення, сегментацію, ідентифікатор класу, версію калібрування, RAG-джерело і фінальний стан. Це дає змогу відтворити, чому користувач отримав певну відповідь або чому система відмовилася від діагнозу.

Усі фотодіагностичні дані залишаються прив'язаними до дозволеного організаційного контуру. Навіть якщо агрономічний knowledge pack є системним, конкретні фото, чати й результати належать активній області доступу. Це запобігає витоку виробничих зображень між організаційними одиницями.

Модель даних також підтримує мовну поведінку. UI залишається англomовним, але відповідь може слідувати мові питання користувача. Це не змінює структуру даних, але впливає на збережену фінальну відповідь і на очікування тестів.

Таким чином, інформаційна база не просто зберігає результат класифікації. Вона зберігає повний ланцюг доказів від вхідного зображення до відповіді за агрономічним класом із цитуванням джерел.

Таблиця 2.6

Дані агрономічного режиму

Сутність	Ключові поля	Призначення
Image attachment	attachment_id, hash, owner, state	Контрольований вхідний файл
Segmentation result	model, class_key, confidence, image_sha	Технічний CV-висновок
Taxonomy class	class_key, label, enabled flag	Стабільна доменна ідентичність
Calibration rule	threshold, version, status	Рішення про допустимість діагнозу

Agronomy source	source_id, usage_policy, page_map	Джерело class-key знань
Citation	response_id, retrieval_unit_id, locator	Доказ відповіді

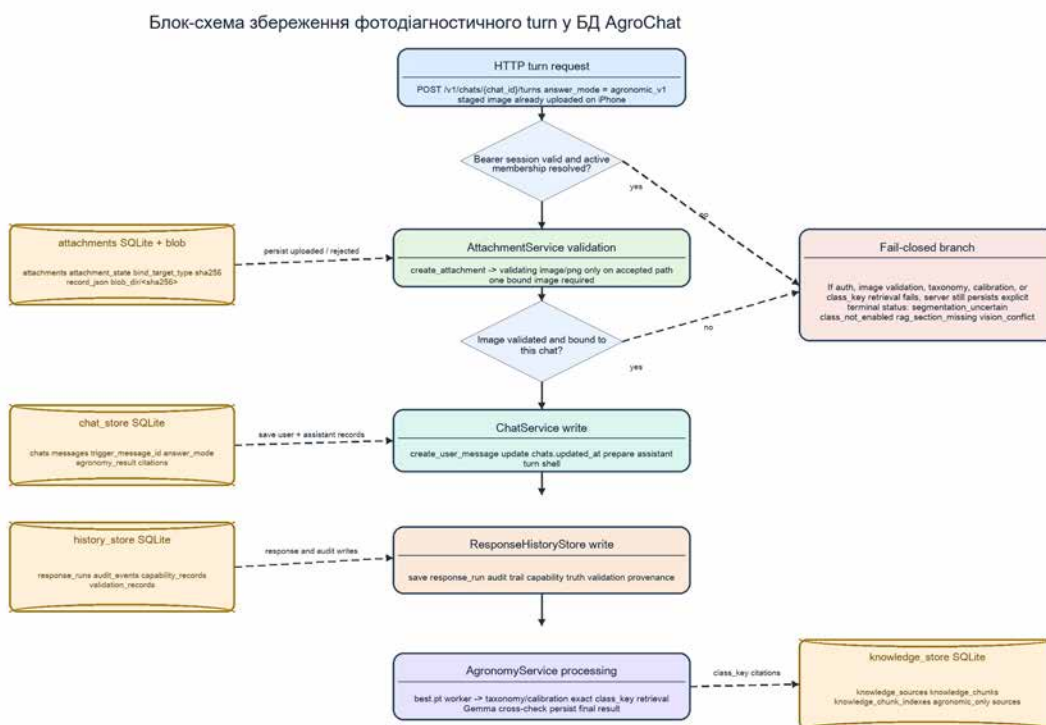


Рисунок 9 - Блок-схема збереження фотодіагностичного turn у БД AgroChat

Блок-схема інгестії agronomy knowledge source і підготовки DB context

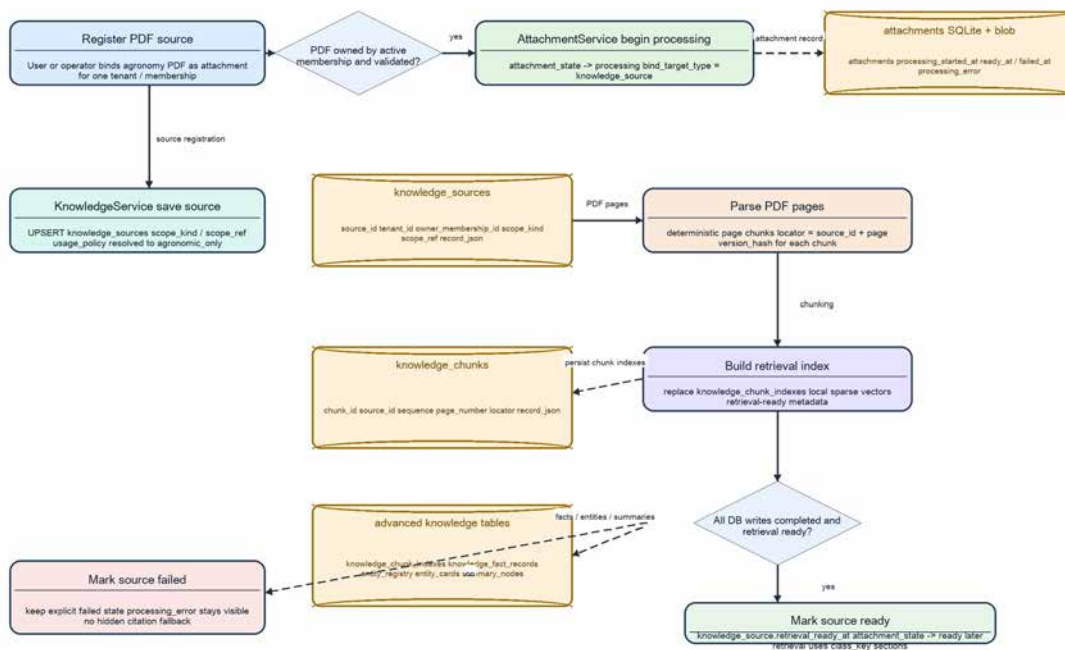


Рисунок 10 - Блок-схема інгестії agronomy knowledge source і підготовки DB context

3 ПРИКЛАДНЕ ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

3.1 Організаційна структура програмного забезпечення

Прикладне програмне забезпечення AgroChat складається з native iPhone app, FastAPI-сервера, інтеграційного шару локальної Gemma 4 E4B, агрономічного сегментаційного модуля, локальних довговічних сховищ і матеріалів перевірки. Таке розділення відповідає серверно-орієнтованій (server-first) доктрині: iPhone не володіє правилами доступу, інференсом або канонічними даними.

iPhone-застосунок реалізовано на SwiftUI. Його основні поверхні - Auth, Chat, Workspace, Runtime і Settings. Auth відповідає за вхід і збереження bearer session; Chat є головним сценарієм; Workspace показує роль користувача, нотатки і джерела знань; Runtime показує автентифікований стан роботи системи та можливостей; Settings містить адресу сервера і локальні параметри.

Серверний API реалізовано на Python/FastAPI з Pydantic-контрактами. Сервер містить маршрути для auth, chats, attachments, knowledge, citations, notes, runtime, capabilities і адміністративних переглядів. Кожний endpoint повинен працювати під активною роллю користувача і не розкривати дані поза дозволеною областю доступу.

Інтеграційний шар мовної моделі ізолює роботу з LM Studio. Сервер не робить вигляд, що керує будь-якою моделлю: у межах роботи використано Gemma 4 E4B, а екрани стану системи показують ідентичність моделі, серверний контур і контекстні параметри тоді, коли середовище їх надає.

Агрономічний сегментаційний модуль є окремим локальним процесом. Він завантажує затверджені ваги, перевіряє реєстр моделі, таксономію і калібрування, виконує запит сегментації і повертає структурований результат. Сервер не дозволяє Gemma самотійно вигадувати діагноз без перевірки агрономічного класу.

Діаграма станів

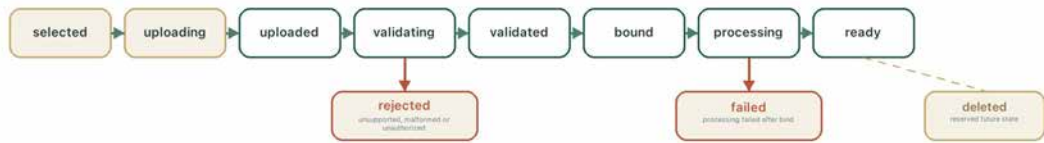
Attachment lifecycle та підтверджений PDF-ingestion шлях

1. Attachment lifecycle

Клієнтський вхід плюс server-side validation, bind, processing і terminal truth.

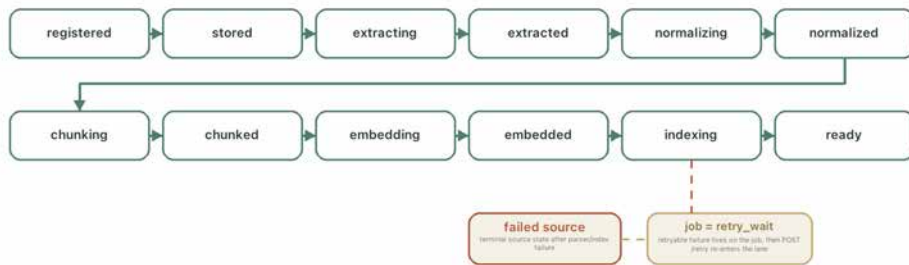
selected / uploading = client-only transient

deleted = reserved future state, не частина current v1 gate



2. Attachment-backed PDF source

Поточний repo path: Phase 7 PDF-only ingestion з explicit source truth і vector-index-ready фіналом.



Примітка: "cancelled" існує в загальному lifecycle-model, але не зафіксований як exercised state на поточному PDF lane. Саме тому на схемі він показаний як текстова обводка, а не як рівноправний крок основного реалізованого контуру.

Рисунок 11 - Життєвий цикл запиту і формування відповіді

Пакетна діаграма AgroChat з акцентом на комп'ютерний зір

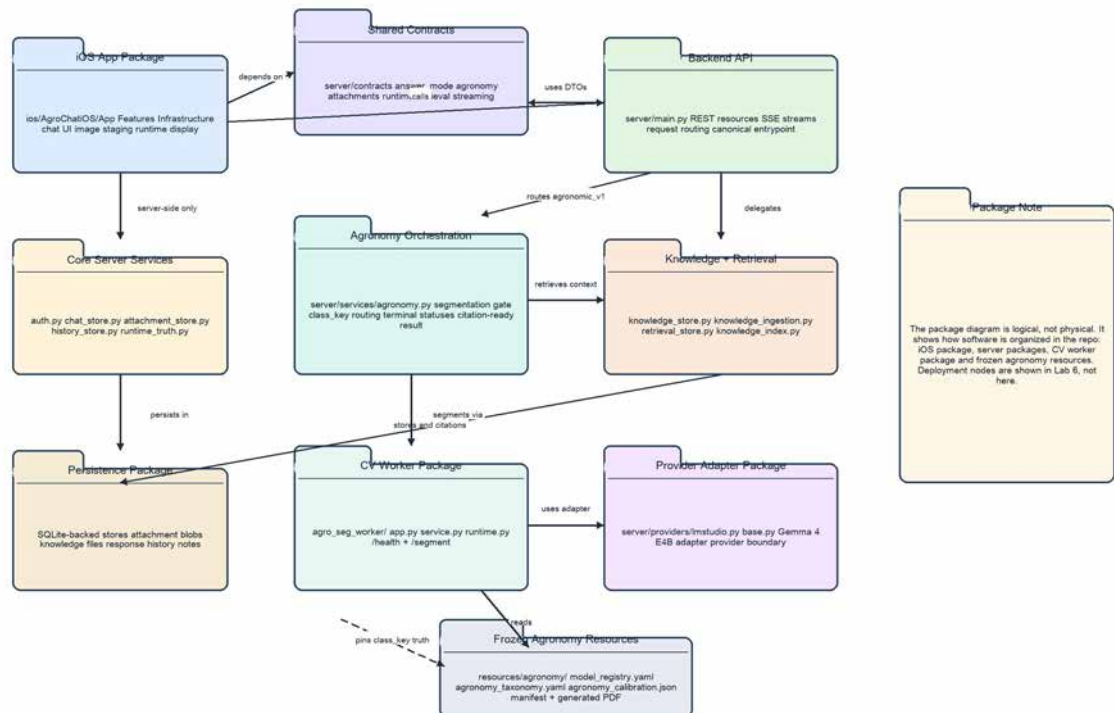


Рисунок 12 - Пакетна діаграма AgroChat з акцентом на фотодіагностичний контур

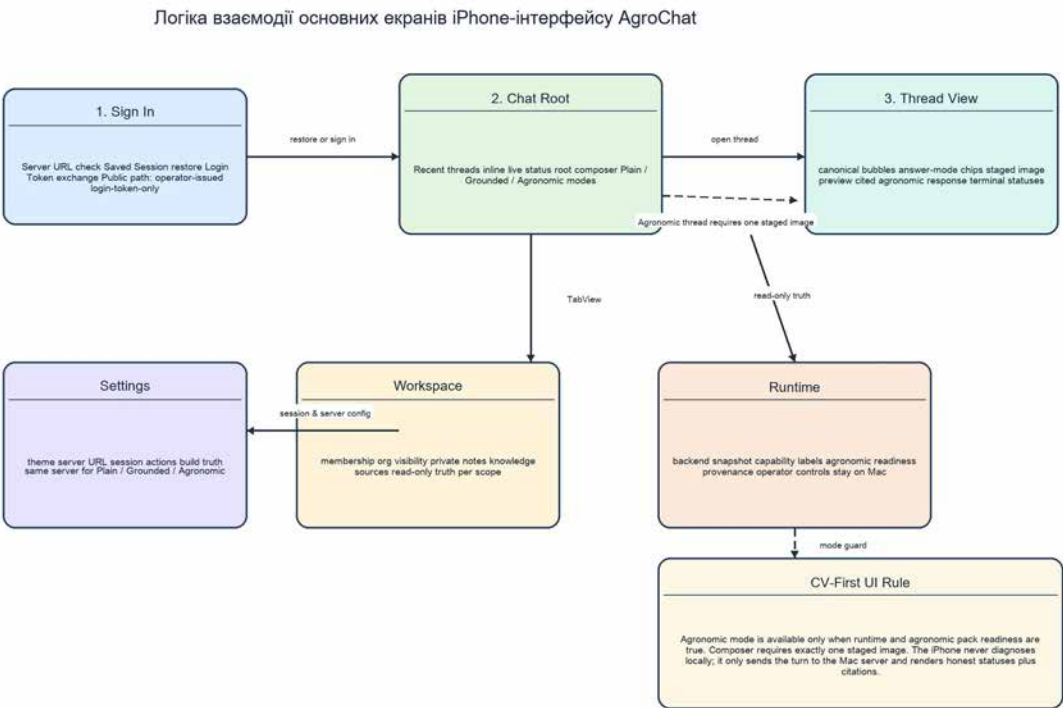


Рисунок 13 - Логіка взаємодії основних екранів iPhone-інтерфейсу AgroChat

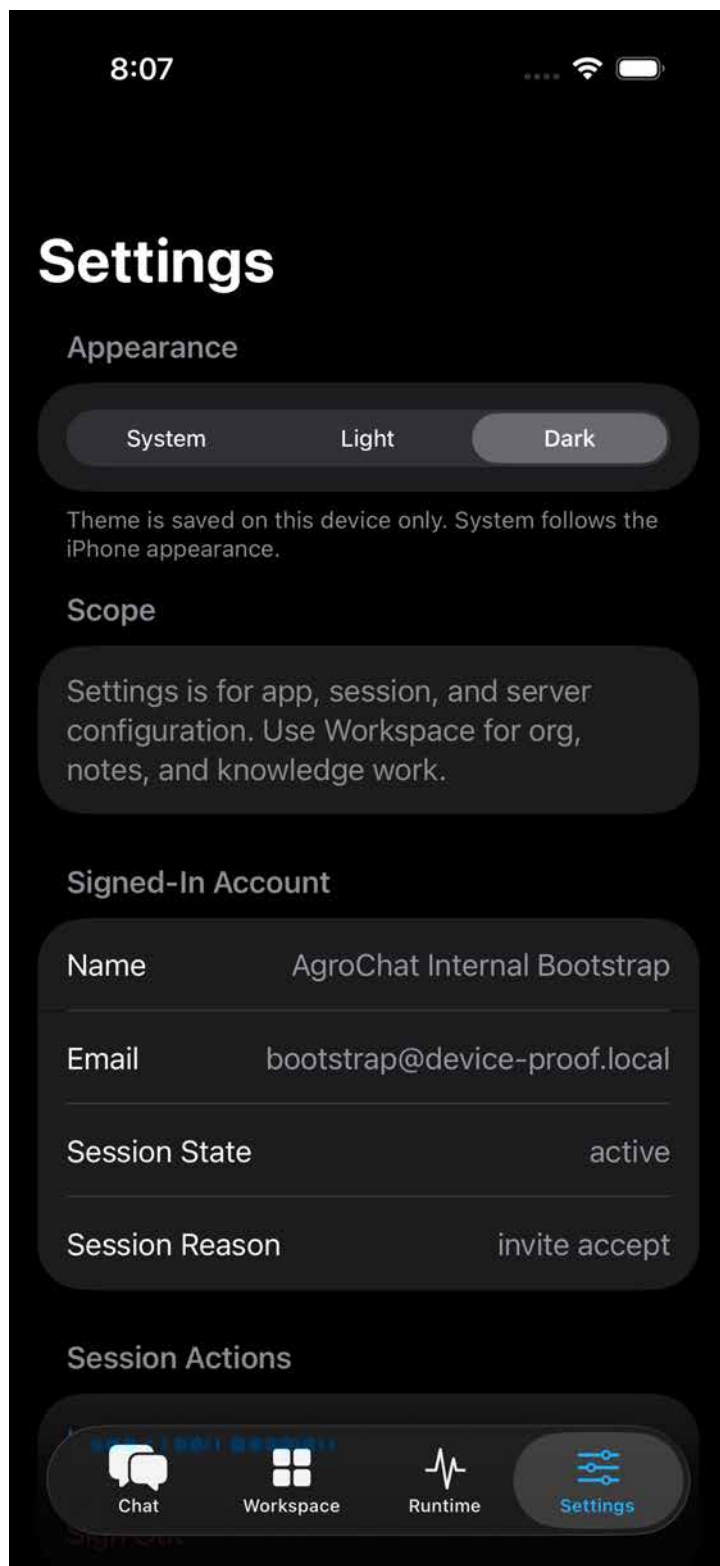


Рисунок 14 - Екран Settings із локальною сесією і межами застосунку

Джерело: знімок екрана симулятора iPhone; показано локальну сесію і налаштування адреси сервера.

Таблиця 3.1

Компоненти прикладного програмного забезпечення

Компонент	Технологія	Відповідальність
AgroChat iOS	Swift 6, SwiftUI, URLSession	UI, capture, streaming display, local preview cache
Серверний API	Python 3.12, FastAPI, Pydantic	Auth, контроль доступу, chat, attachments, RAG, стан системи
Інтеграційний шар мовної моделі	LM Studio OpenAI-like API	Gemma 4 E4B text/image generation
Сегментаційний модуль	Python process	Image segmentation, taxonomy/calibration checks
Durable stores	SQLite-backed stores	Узгоджений локальний стан і матеріали перевірки
Admin console	операторська поверхня на Mac (операторський інтерфейс на Mac)	Перегляд стану і керування моделлю на Mac

3.2 Вибір інструментарію для створення програмного забезпечення

SwiftUI обрано для iPhone-клієнта, оскільки він дозволяє реалізувати native touch-first інтерфейс без web-shell компромісів. Для AgroChat це важливо не лише естетично, а й архітектурно: клієнт повинен інтегрувати системні можливості iOS, роботу з фото, файловий імпорт, безпечне збереження session token і плавну потокову презентацію.

FastAPI обрано для серверної частини через прозорий опис REST endpoints, підтримку Pydantic-моделей і швидку побудову contract-first API. У системі з багатьма станами - життєвий цикл вкладення, стан відповіді, стан задачі, перелік можливостей - явні DTO-контракти знижують ризик неузгодженості між клієнтом і сервером.

SQLite-backed stores відповідають локальній deployment-моделі. Вони спрощують інсталяцію, зменшують кількість зовнішніх залежностей і достатні для локального серверного контуру. File storage відокремлює blobs від метаданих, а детермінований індекс підтримує PDF retrieval без хмарного vector service.

LM Studio використовується як локальне середовище для Gemma 4 E4B. Система не ототожнює технічний ідентифікатор середовища з офіційною точною Hugging Face identity і не заявляє GPU-only execution. Це приклад чесного опису можливостей: у дипломі описується перевірений локальний шлях, а не маркетингова назва моделі.

Python-сегментаційний модуль виділено окремо, бо комп'ютерно-зоровий runtime має інші залежності й ризики, ніж chat backend. Окремий процес дозволяє перевіряти готовність, стартову працездатність, ваги моделі, таксономію і повторне використання кешу незалежно від основного API.

Таблиця 3.2

Обрані інструменти розробки

Засіб	Призначення	Причина вибору
SwiftUI	Native iPhone UI	Touch-first UX, інтеграція з iOS, без web-shell
FastAPI	Серверний API	Contract-first REST/SSE поверхня
Pydantic	DTO/contract models	Типізовані payloads і стани
SQLite	Durable local stores	Простий local Mac deployment
LM Studio	Local Gemma serving	Локальний інференс без cloud
Python module	Segmentation runtime	Ізоляція CV-залежностей

3.3 Алгоритмізація та програмування програмних модулів

Алгоритм обробки звичайного чатового запиту починається з перевірки session token і активної ролі користувача. Після цього сервер створює

повідомлення користувача і цикл відповіді, застосовує політику черги, запускає запит до мовної моделі, передає події через SSE і завершує цикл збереженим повідомленням асистента або чесною помилкою завершення.

Алгоритм обробки вкладки розділено на `upload` і `bind`. Під час `upload` сервер фіксує `hash`, `kind`, `media type`, власника і стан перевірки. Під час `bind` перевіряється, чи сумісне вкладки з чатом або джерелом знань. Такий поділ запобігає використанню сирого файлу до перевірки.

Алгоритм `grounded retrieval` виконує перевірку доступу до добору джерел. Система спочатку відкидає одиниці пошуку, які не доступні активній ролі користувача, і лише потім виконує `scoring` або складання контексту. Це принципово для ізоляції даних: ранжування не може виправити витік, якщо контроль доступу застосовано після генерації.

Агрономічний алгоритм має жорсткіший контур. Користувач надсилає запит із зображенням у режимі `Agroponic`. Сервер перевіряє, що є рівно одне валідоване зображення, інструменти вимкнені, а `fallback` до `plain` або `grounded` заборонений. Потім викликається сегментаційний модуль, результат проходить поріг калібрування і зіставлення з агрономічним класом.

Якщо сегментація невизначена, клас не готовий, мітка не зіставлена або розділ RAG-джерела відсутній, система завершує сценарій без діагнозу. Якщо всі перевірки пройдено, сервер готує RAG-контекст за агрономічним класом, передає його `Gemma` разом із зображенням і зберігає фінальну відповідь із цитатами.

Повторне формування агрономічної відповіді не повинно повторно вгадувати результат сегментації. Збережений `agronomy_result` і `hash` зображення, моделі та конфігурації використовуються для стабільного повторного запуску, щоб відповідь залишалась прив'язаною до того самого перевіреного стану.

Алгоритмічні стани агрономічного режиму

Стан	Умова	Дія системи
ok	Клас enabled, calibration passed, RAG section знайдено	Формується цитована відповідь
segmentation_uncertain	Confidence нижче порогу	Безпечне завершення без діагнозу
class_not_enabled	Клас структурно є, але не пройшов потрібну перевірку	Діагноз не заявляється
class_unmapped	Raw label не має stable class_key	Сценарій завершується помилкою мапінгу
rag_section_missing	Немає class-key джерела	Відповідь не формується
vision_conflict	Gemma cross-check суперечить gated class	Висновок блокується

Блок-схема збереження фотодіагностичного turn у БД AgroChat

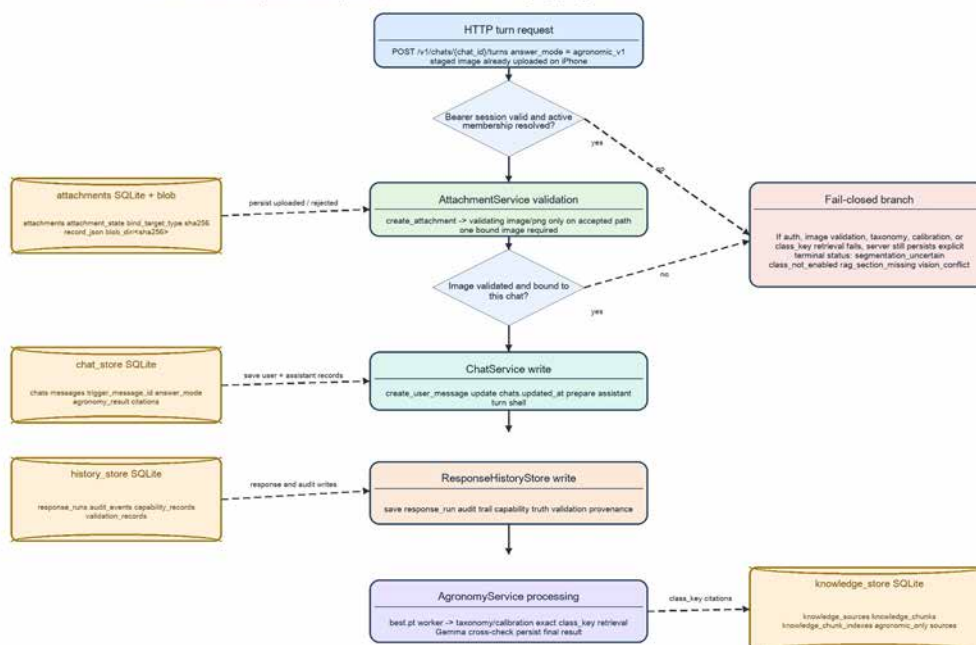


Рисунок 15 - Класична блок-схема алгоритму фотодіагностичного запиту

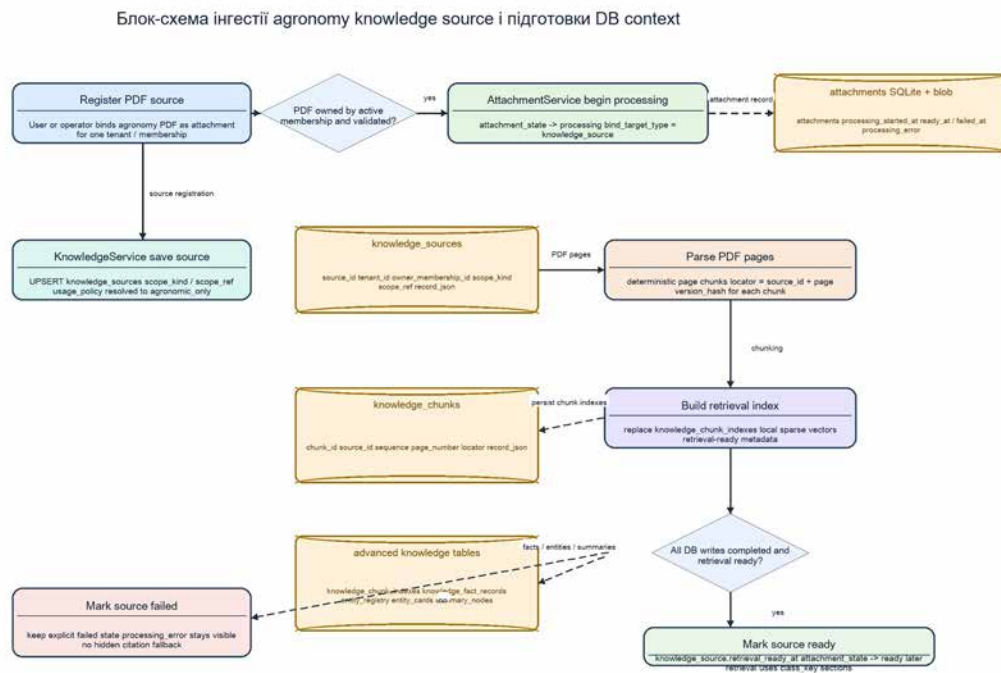


Рисунок 16 - Класична блок-схема формування RAG-контексту з цитуванням

Нижче наведено короткі фрагменти програмної реалізації. Вони скорочені до навчального обсягу, а повніші лістинги винесено в Додаток В.

3.3.1 Фрагмент FastAPI endpoint

```

@app.post("/v1/chats/{chat_id}/turns", response_model=CreateTurnAccepted)
async def create_chat_turn(chat_id: str, request: CreateTurnRequest,
                           authorization: str | None = Header(default=None)):
    context = _require_auth_context(authorization)
    _require_accepted_model(request.model)
    return chat_service.create_turn(context=context, chat_id=chat_id, request=request)
  
```

3.3.2 Фрагмент Pydantic DTO

```

class CreateTurnRequest(BaseModel):
    model: str
    input_text: str = Field(min_length=1)
    answer_mode: AnswerMode | None = None
    vision_attachment_id: str | None = None
  
```

3.3.3 Фрагмент виклику сегментаційного модуля

```

async with httpx.AsyncClient(timeout=timeout) as client:
    response = await client.post(
        f"{worker_url.rstrip('/')}/segment",
        json={"image_sha256": image_sha256, "image_base64": payload},
    )
  
```

3.3.4 Фрагмент збереження citation

```
def attach_citations(self, response_id: str, citations: list[CitationRecord]):
    stored = self._require(response_id)
    stored.citations = list(citations)
    stored.last_event_at = datetime.now(timezone.utc)
    self._persist(stored)
```

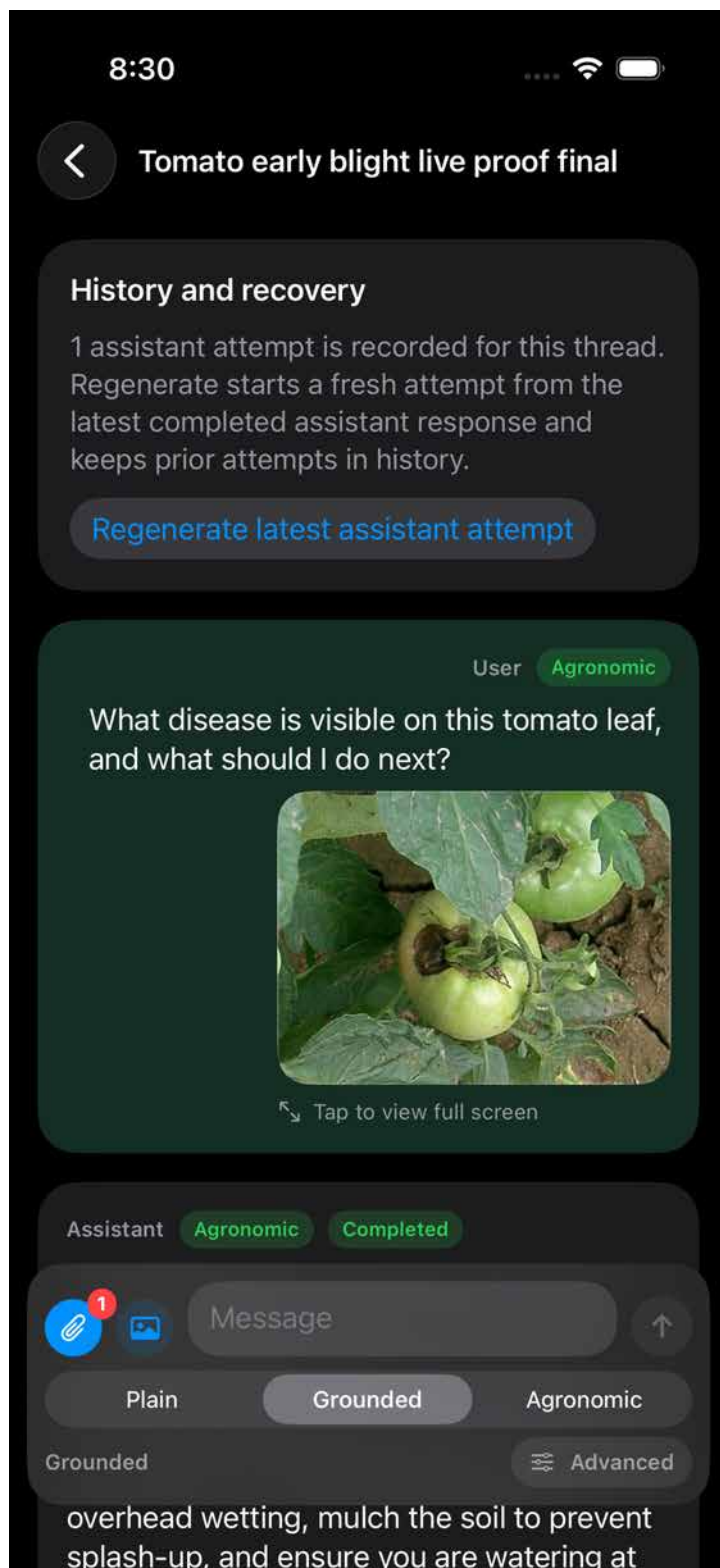


Рисунок 17 - Агрономічний turn із видимим фото листка та початком відповіді

Джерело: знімок екрана симулятора; показано фото рослини, цитату джерела та початок агрономічної відповіді.

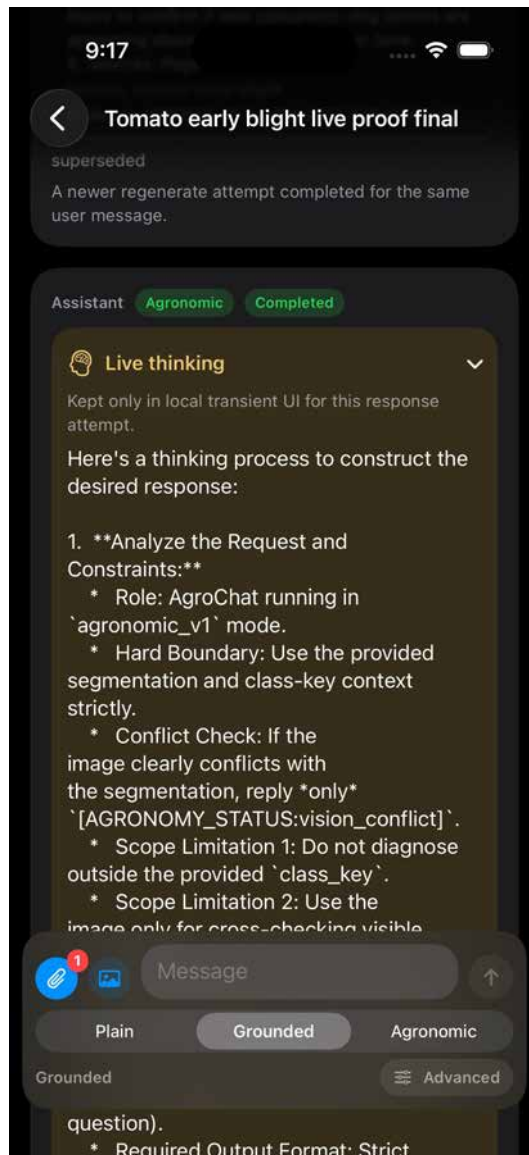


Рисунок 18 - Початок реального Live thinking process після regenerate

Джерело: знімок екрана симулятора після повторного формування відповіді; thinking показаний як тимчасовий UI-блок, а не як збережений текст відповіді.

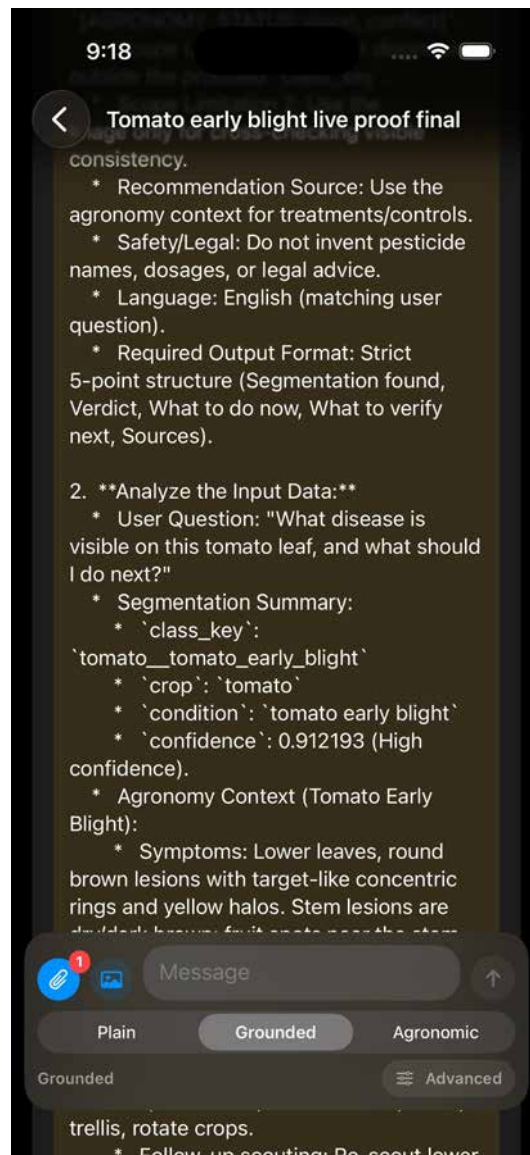


Рисунок 19 - Live thinking із Segmentation Summary та confidence 0.912193

Джерело: знімок екрана симулятора; видимі `class_key tomato__tomato_early_blight`, `crop tomato`, `condition tomato early blight`, `confidence 0.912193`.

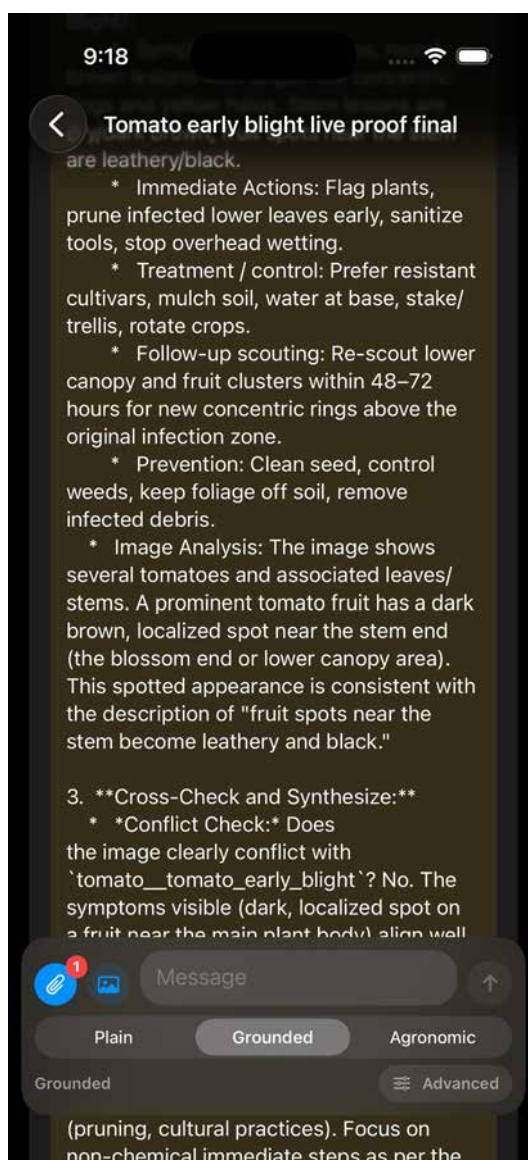


Рисунок 20 - Live thinking з agronomy context і image analysis

Джерело: знімок екрана симулятора; видимі agronomy context, symptoms, immediate actions, follow-up scouting і image analysis.

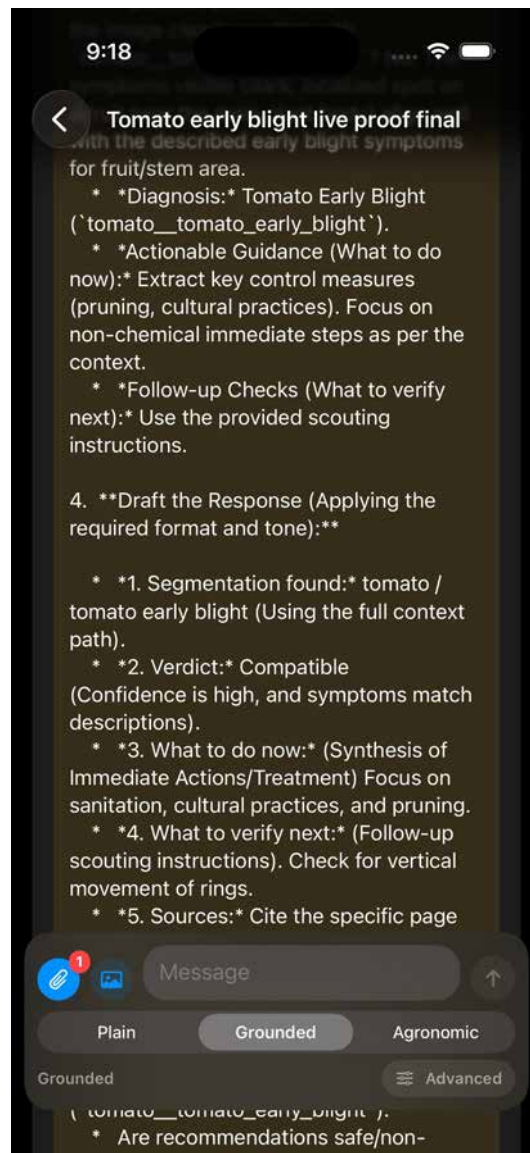


Рисунок 21 - Live thinking з cross-check і draft response

Джерело: знімок екрана симулятора; видимі conflict check, diagnosis, actionable guidance і draft response.

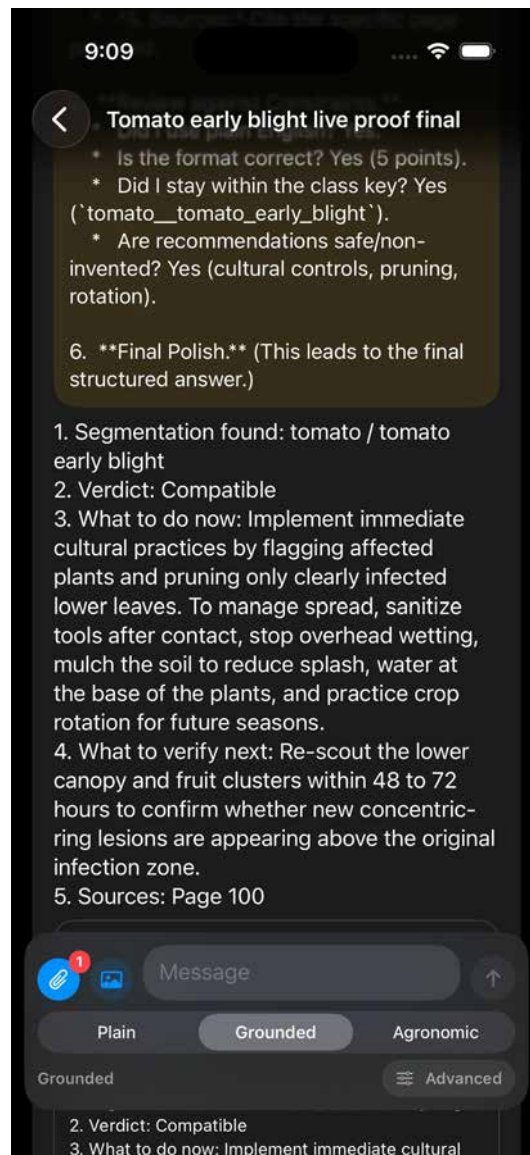


Рисунок 22 - Повна агрономічна відповідь після Live thinking

Джерело: знімок екрана симулятора; видимі пункти 1-5 canonical answer і Sources: Page 100.

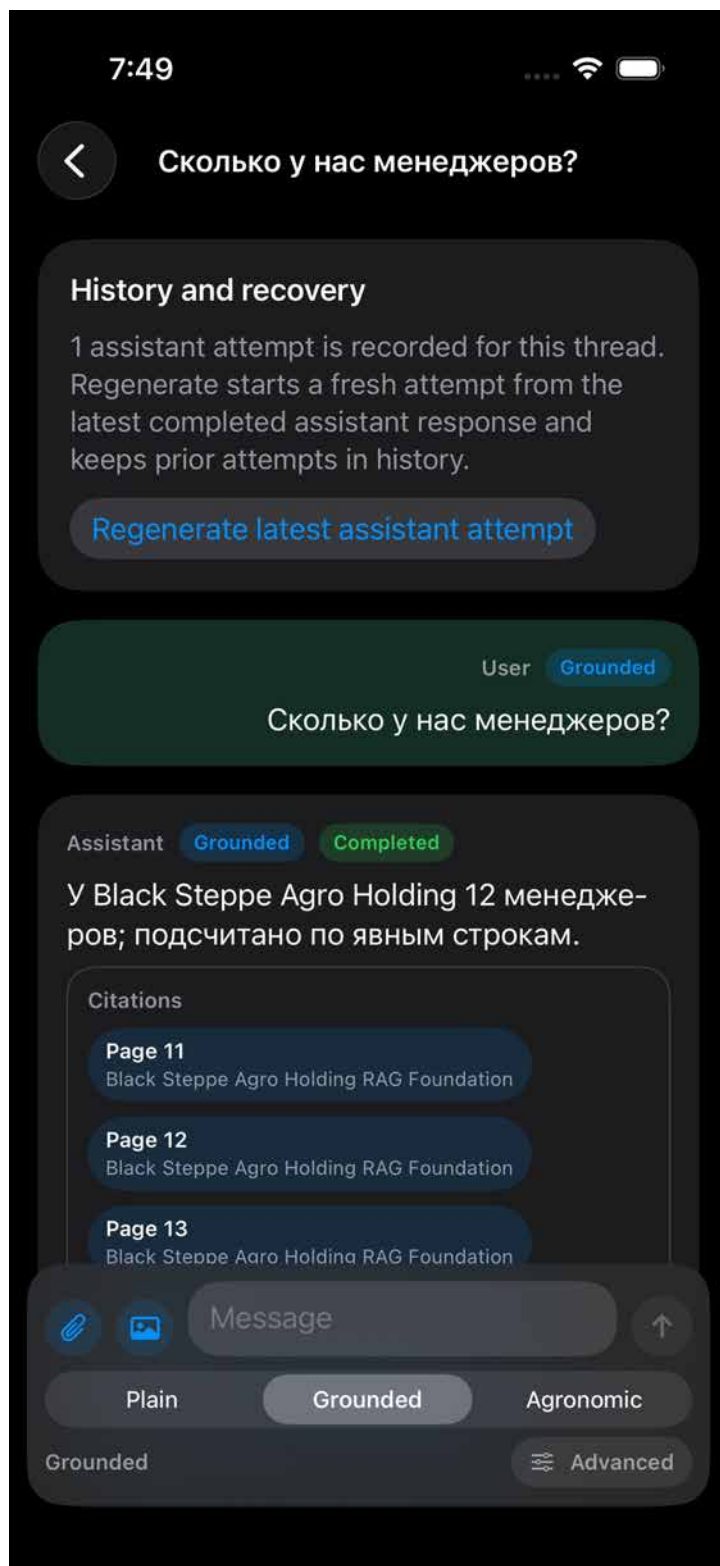


Рисунок 23 - Grounded-відповідь про менеджерів із видимими цитатами сторінок 11-13

Джерело: знімок екрана симулятора; показано відповідь із джерелами та видимими посиланнями на сторінки.

3.4 Програмна реалізація iPhone-клієнта

iPhone-клієнт AgroChat реалізує не просто форму надсилання запиту, а повну робочу оболонку. Він має вкладки Chat, Workspace, Runtime і Settings. Така інформаційна архітектура дає користувачеві доступ до основного сценарію, організаційного контексту, стану системи і локальних налаштувань без перенесення операторського керування на iPhone.

Екран Auth підтримує збережену session і login token flow. Сирий login token не повинен залишатися в довготривалому сховищі iPhone; після exchange клієнт зберігає bearer session. Це відрізняє login token від постійного пароля і відповідає операторськи виданому доступу.

Екран Chat є центральним екраном. У ньому користувач бачить історію повідомлень, live assistant block, mode chips, citations, tool summaries і attachment preview. Для дипломної системи важливо, що UI не створює fake thinking або fake success: live state прив'язаний до подій сервера.

Composer підтримує Plain, Grounded і Agronomic режими. Plain використовується для звичайної відповіді, Grounded - для пошуку з цитуванням, Agronomic - для фотодіагностики. Наявність сегментованого вибору зменшує ризик, що користувач випадково очікує RAG або diagnosis у неправильному режимі.

Image staging має окрему логіку. Коли користувач вибирає фото, воно стає чернеткою вкладення, але не перетворює автоматично весь чат на vision mode без явного надсилання. Це важливо для передбачуваності: вкладення має бути пов'язане з конкретним запитом.

Розділ Workspace показує роль користувача, організаційну видимість, notes і knowledge sources. Це підтримує ідею, що користувач працює не в абстрактному чаті, а в межах організації. Private notes можуть бути частиною знань, але залишаються owner-only.

Розділ Runtime показує автентифікований стан роботи системи та можливостей. Звичайний користувач може бачити базову готовність системи, а роль з операторськими правами може бачити ширший read-only admin drill-down. При цьому iPhone не має кнопок load/unload/restart моделі.

Екран Settings містить локальну конфігурацію, зокрема server base URL. Це практично потрібно для локального Mac-сервера, але не повинно перетворюватися на вибір різних продуктних серверів для різних режимів відповіді. Plain, Grounded і Agronomic використовують один app backend.

Локальне збереження в iPhone клієнті використовується допоміжно. Токен сесії зберігається в захищеному сховищі, байти попереднього перегляду - у локальній файловій області, а налаштування UI - у UserDefaults. Узгоджений стан чатів, відповідей, цитат і стану системи залишається на сервері.

У дизайні iPhone-клієнта важливо не змішувати UI language policy і answer language. UI labels залишаються English-only, але answer text може відповідати мові питання. Це дозволяє користувачу ставити питання українською або російською без порушення English-only UI constraint.

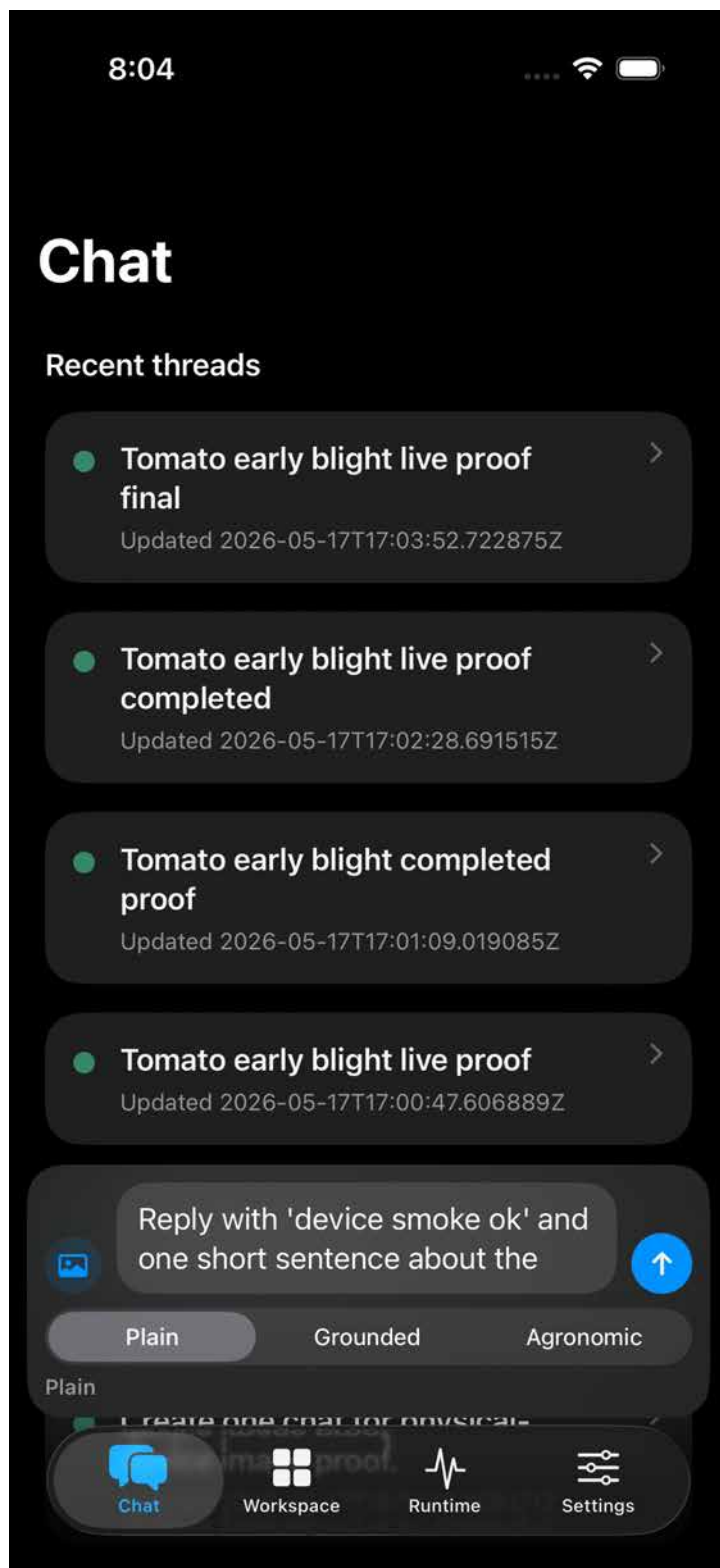


Рисунок 24 - Основна чатова поверхня після згенерованих відповідей

Джерело: знімок екрана симулятора після відповіді з джерелами та агрономічної відповіді.

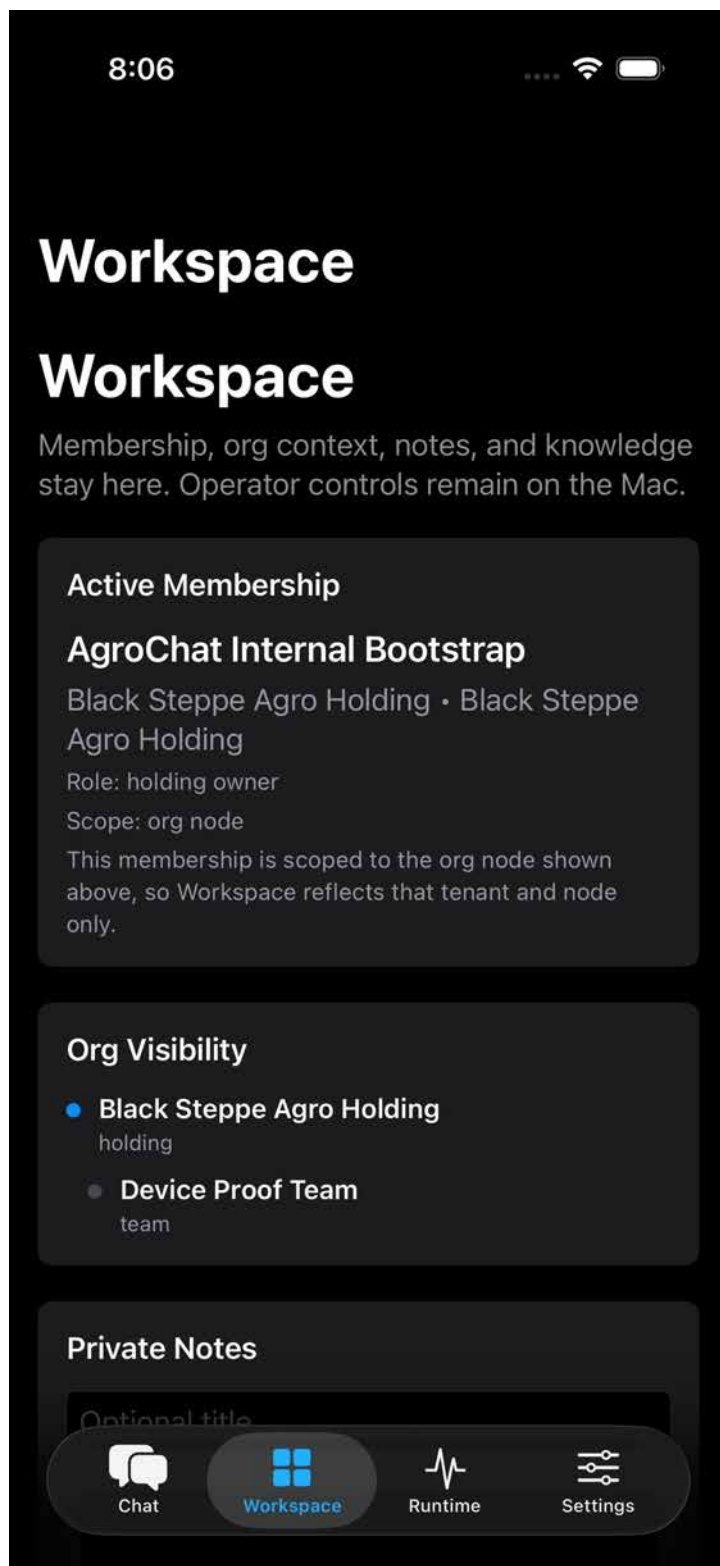


Рисунок 25 - Розділ Workspace з роллю користувача, організаційною видимістю і knowledge sources

Джерело: знімок екрана симулятора; показано організаційний контекст і джерела знань.

3.5 Програмна реалізація серверної частини

Серверна частина AgroChat є канонічним власником бізнес-логіки. Вона приймає запити iPhone, перевіряє bearer session, визначає активну роль користувача, застосовує контроль доступу і тільки після цього переходить до чату, вкладень, знань або операцій стану системи.

FastAPI routes організовані за ресурсами. Auth endpoints відповідають за session і login-token exchange; chat endpoints створюють chats, turns і streams; attachment endpoints завантажують і прив'язують файли; knowledge endpoints ведуть sources, context і citations; runtime endpoints показують автентифікований стан системи.

Contract layer на Pydantic описує payloads і responses. Це важливо для iPhone-клієнта: він не повинен здогадуватися, чи поле означає ready, failed або unsupported. Типізовані DTO зменшують кількість неявних UI-станів.

Сервісний шар реалізує предметну логіку. Сервіс авторизації веде користувачів, рольові участі, сесії і токени входу. Сховище чатів зберігає повідомлення. Сховище вкладень працює з файловими об'єктами. Сервіси інгестії знань і пошуку готують джерела для відповіді на основі документів. Сервіс стану системи збирає модельний і системний стан.

Інтеграційний шар мовної моделі відокремлює сервер від конкретного serving interface. Сервер знає, що використовує локальне середовище LM Studio, але не розсипає деталі upstream API по всій кодовій базі. Це дозволяє контролювати вибір моделі і не допустити silent fallback на іншу модель.

Admission policy керує чергою і busy state. Локальна модель може обробляти обмежену кількість важких генерацій. Якщо система зайнята, користувач повинен бачити busy або queued state, а не нескінченне очікування без пояснення.

Response history store дає змогу відновити completed attempts, citations і terminal states. Це важливо для chat history: завершена відповідь повинна

зберігатися як `canonical assistant message`, а `transient thinking` не має ставати частиною канонічного тексту.

`Admin routes` розділені від `user routes`. Операторські дії можуть завантажувати, вивантажувати або перезапускати модель на Mac, але iPhone має лише `read-only visibility` за наявності `operator-read grant`. Це зберігає межу між користувацьким застосунком і операторським керуванням.

`Knowledge usage policy` розділяє `general_grounded` і `agronomic_only sources`. Завдяки цьому агрономічний PDF pack не забруднює `ordinary grounded retrieval`, а загальний RAG-корпус не використовується для `diagnosis` за агрономічним класом без спеціального `routing`.

Серверна реалізація є основою для чесності дипломного проєкту: саме сервер вирішує, чи можливість доступна, недоступна, завершена помилкою або працює з обмеженнями, і саме ці стани повинні відображатися в UI та документації.

3.6 Потокова взаємодія і формування відповіді

Потокова взаємодія реалізується через SSE. Це дозволяє iPhone отримувати події поступово, а не чекати повного завершення генерації. Для користувача це створює відчуття живого діалогу, але з технічного погляду важливіше те, що `stream` має типізований конверт і коректне завершення.

Кожна подія потоку відповіді повинна мати зрозумілу семантику: запит прийнято, додано фрагмент тексту, додано цитату, завершено, завершено помилкою або скасовано. UI може реагувати на ці події без парсингу довільного тексту.

`Thinking visibility` дозволяється лише тоді, коли модельне середовище реально віддає окремий `reasoning channel`. Якщо такого каналу немає, система не повинна вигадувати `reasoning`. Це відрізняє AgroChat від демонстраційних продуктів, де `thinking` часто імітується для ефекту.

Під час grounded answer citations можуть додаватися в stream або в metadata відповіді. Користувач бачить не лише текст відповіді, а й джерела. Для дипломної системи це особливо важливо, бо аграрні поради мають бути прив'язані до матеріалів, а не лише до генеративної моделі.

Завершальна подія є обов'язковою. Якщо stream обірвався, клієнт не повинен робити вигляд, що відповідь завершена. Система не заявляє повний replay/resume там, де його немає.

Cancellation має межі. Якщо upstream не підтримує running cancel повноцінно, система повинна чесно показувати розрив або запит на скасування, а не гарантувати те, чого не може виконати. Це ще один приклад чесного опису можливостей.

Для потік подій агрономічного контуру може завершитися завершальним статусом (terminal status) без генерації довгого текст відповіді. Наприклад, segmentation_uncertain - це коректна відповідь системи, хоча вона не містить діагнозу. UI повинен показати це як зрозумілий результат, а не як помилку мережі.

Після completed server виконує canonical persistence. Це означає, що transient live block у UI має бути замінений або узгоджений із збереженим assistant message. Інакше користувач може побачити дубльовані або суперечливі відповіді.

Потокова модель також впливає на тестування. Потрібно перевіряти не лише текст, а порядок подій, завершальну подію, збереження після завершення, видимість цитат і контрольовану поведінку при помилках модельного середовища.

Таким чином, SSE у AgroChat є не просто UX-покращенням, а частиною contract-first design. Воно дозволяє описати весь процес відповіді як послідовність контрольованих подій.

4 РЕКОМЕНДАЦІЇ ЩОДО ВПРОВАДЖЕННЯ ТА ЕКСПЛУАТАЦІЇ СИСТЕМИ

4.1 Тестування системи

Тестування AgroChat має будуватися не лише навколо компіляції, а навколо спостережуваної поведінки системи. Для цієї системи важливо перевіряти, що API повертає правильні стани, механізм контролю доступу блокує заборонені сценарії, цитати не виходять за межі дозволеного організаційного контуру, стан роботи системи відповідає фактичній інтеграції з мовною моделлю, а iPhone показує узгоджений стан серверної частини.

У проєкті підготовлено перевірки для авторизації, організаційної ієрархії, чатів, вкладень, RAG-цитування, стану системи, PDF-джерел і агрономічного режиму. Вони підтверджують, що ключові контракти реалізовані не лише в документації.

Матеріали перевірки виконують роль журналу підтвердження результатів. Вони містять структуровані звіти, знімки екрана, підсумки приймальних перевірок, збережені стани системи, відомості про джерела та результати перевірки на пристрої або симуляторі. Для диплома важливо використовувати їх як підтвердження конкретних сценаріїв, а не як заяву про необмежену промислову готовність.

Окремо перевіряється відповідність заяв фактичним результатам. Якщо можливість не підтверджена матеріалами перевірки, вона не описується як реалізована. Саме тому аудіо, універсальне агрономічне покриття, хмарний інференс і загальне розуміння будь-яких документів не заявляються як функції поточної роботи.

Таблиця 4.1

Основні напрями тестування

Напря́м	Приклади перевірок	Очікуваний результат
Авторизація і ролі	Вхід за виданим token, відкликання доступу, запит чужого ресурсу	Доступ дозволяється лише активній ролі користувача
Робота з вкладеннями	Завантаження фото, перевірка типу, прив'язка до чату, обробка помилки	Файл використовується тільки після валідованого стану
RAG і цитування	Питання з готовим PDF-джерелом, перегляд цитати, спроба чужого джерела	Відповідь має дозволене джерело або завершується відмовою
Стан системи	Модель готова, модель недоступна, сегментаційний модуль недоступний	Користувач бачить фактичний стан, а не прихований збій
Агрономічний режим	Успішне фото, невизначена сегментація, відсутній агрономічний клас	Діагноз формується лише для перевіреного сценарію
iPhone-клієнт	Вхід, вибір режиму, відображення статусу, перегляд цитат	Інтерфейс не показує хибний успіх

Таблиця 4.2

Протокол тестування програмного продукту

№	Сценарій	Вхідні дані	Очікуваний результат	Фактичний результат	Статус
1	Вхід за виданим token	Дійсний операторський token	Сесію створено	Сесію створено, активна роль визначена	Passed
2	Відкликаний доступ	Відкликаний token або session	Доступ заборонено	API повертає відмову авторизації	Passed
3	Відповідь із джерелами	Питання і готове PDF-джерело	Відповідь містить цитату	Збережено посилання на сторінку	Passed

				джерела	
4	Успішний агрономічний сценарій	Фото перевіреного агрономічного класу	Відповідь із цитованим агрономічним контекстом	Створено результат фотодіагностик и і цитату	Passed
5	Невизначена сегментація	Фото нижче порогу впевненості	Безпечне завершення	Повернуто завершальний стан без діагнозу	Passed
6	Доступ до чужих даних	Чужий source/chat id	Доступ заборонено	Контроль доступу блокує читання	Passed

Таблиця 4.3

Негативні сценарії тестування

Сценарій	Причина ризику	Очікувана поведінка
Невалідне зображення	Модель не повинна отримати пошкоджений файл	Вкладення не переходить у готовий стан
Сегментаційний модуль недоступний	Сегментація неможлива	Агрономічний режим показує обмежену працездатність або недоступність
Клас не пройшов перевірку	Для класу бракує калібрування або джерела	Діагноз не формується
Відсутній розділ джерела	Немає агрономічного джерела для визначеного класу	Завершальний стан без вигаданої рекомендації
Недозволена цитата	Ризик витоку даних іншого господарства	Перегляд цитати завершується безпечною відмовою

Таблиця 4.4

Матеріали підтвердження приймання

Матеріал	Що підтверджує	Обмеження
Структурований звіт агрономічного сценарію	Успішний і безпечний шлях фотодіагностики	Не означає універсальне покриття всіх класів
Знімки екрана iPhone-клієнта	Відображення станів, режимів і результатів	Не замінюють усі перевірки на реальному пристрої
Серверний протокол тестування	Оркестрацію відповіді та завершальні стани	Не перевіряє візуальне компонування UI
Журнал приймальних рішень	Поточні межі реалізованих можливостей	Не розширює архітектуру без окремого рішення

Таблиця 4.5

Протокол виправлення виявлених помилок

Дефект	Прояв	Причина	Виправлення	Доказ
Некоректна відмова доступу	Користувач бачив загальну помилку	Не було окремого повідомлення для забороненого ресурсу	Додано явну відповідь про відмову доступу	Перевірка сценарію чужого ресурсу
Порожня відповідь із джерелами	Модель повернула службовий текст без відповіді	Не було збереженого тексту відповіді	Додано контрольований фінальний текст	Перевірка відповіді з джерелами
Невизначена сегментація	Модель нижче порогу впевненості	Фото або клас не проходить поріг	Повертається завершальний стан без діагнозу	Приймальна перевірка агрономічного сценарію
Застаріле агрономічне	Пошук міг брати старе	Вибір першого готового	Вибір останнього	Перевірка оновлення

джерело	джерело	джерела	ГОТОВОГО агрономічного джерела	джерел
---------	---------	---------	--------------------------------------	--------

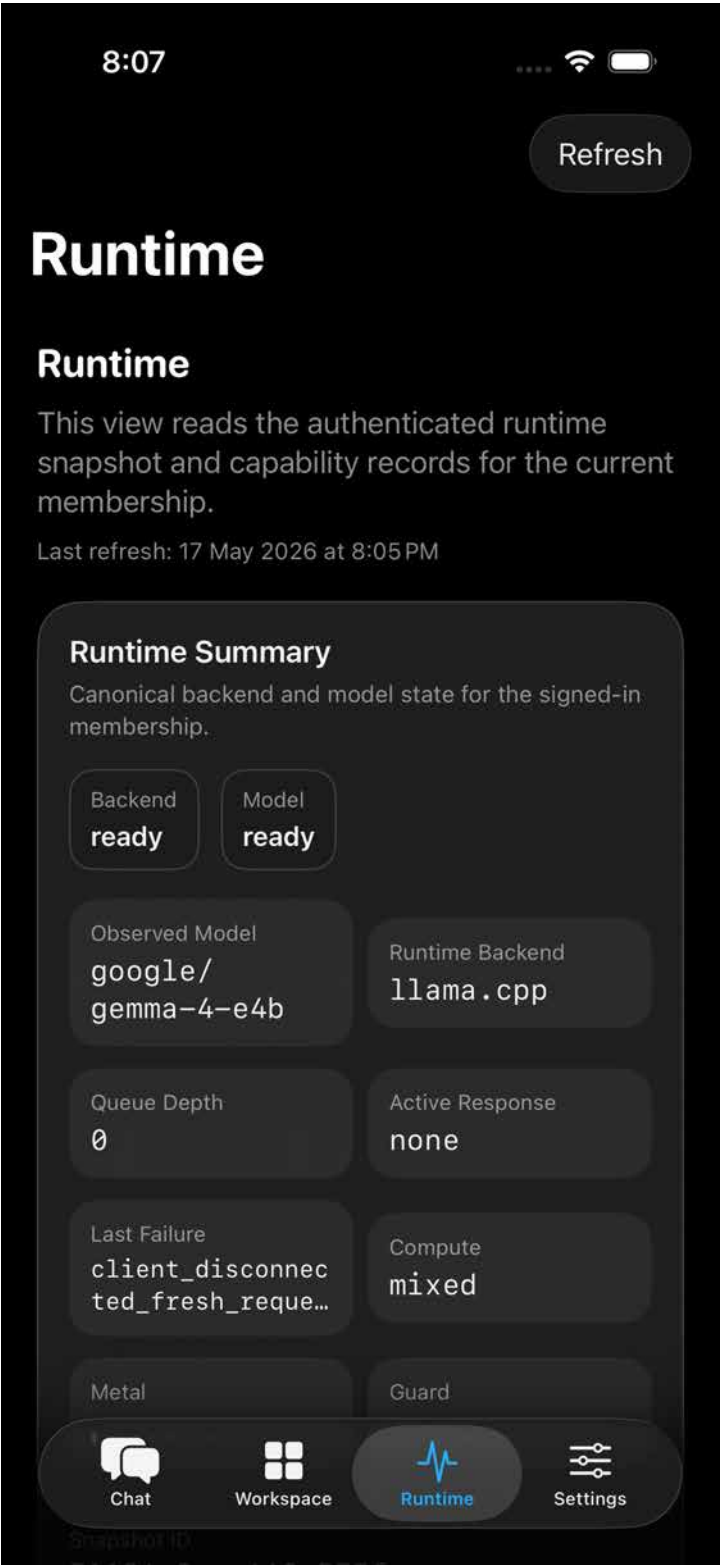


Рисунок 26 - Екран стану системи на iPhone без дій адміністративного керування

Джерело: знімок екрана симулятора; iPhone показує стан системи лише для перегляду, без дій load/unload/restart.

4.2 Вимоги до апаратного та програмного забезпечення

Для локального впровадження потрібні iPhone з установленим AgroChat iOS app, MacBook як сервер та вузол інференсу, Python-середовище для FastAPI, LM Studio для локального Gemma 4 E4B, локальні файли моделей і ресурсів, SQLite-backed каталог даних та мережеве з'єднання між iPhone і Mac.

Mac виконує роль локального адміністративного та обчислювального вузла. Він запускає серверний API, інтеграційний шар мовної моделі, довговічні сховища, content-addressed file storage, сегментаційний модуль і операторську адміністративну консоль. iPhone не повинен напряму керувати моделлю або базою даних.

Програмна конфігурація повинна містити ідентифікатор локальної моделі, цільове контекстне вікно, каталог стану, корінь файлового сховища, дозволений host/port сервера, URL сегментаційного модуля і налаштування операторськи виданого login token. Неправильна конфігурація повинна проявлятися як явний збій або обмежена працездатність, а не як прихована підміна можливостей.

У production-подібному майбутньому серверна частина може бути перенесена на іншу інфраструктуру, але клієнтські контракти не повинні змінюватися. Саме тому в дипломі наголошено на REST/SSE, типізованих DTO і серверному володінні станами ресурсів.

Таблиця 4.6

Мінімальні вимоги для локального запуску

Елемент	Вимога	Примітка
iPhone	Сумісна версія iOS і	Клієнтська робоча поверхня

	встановлений native app	
MacBook	Локальний сервер з доступом до LM Studio	Сервер, інференс і операторське керування
Python	Середовище для FastAPI server і сегментаційного модуля	Локальні залежності проєкту
LM Studio	Завантажена локальна модель Gemma 4 E4B	Без хмарного інференсу
Storage	Локальний каталог стану і файлове сховище	SQLite-backed stores та blobs
Network	Доступ iPhone до Mac-сервера	Лише дозволений local/LAN контур

4.3 Склад інсталяційного пакету

Інсталяційний пакет для AgroChat повинен включати iOS project або зібраний застосунок, серверний пакет, конфігураційні файли, інструкції запуску локальної моделі, агрономічні ресурси, сегментаційний модуль, короткі SQL/store-фрагменти, checklist перевірки і посібник оператора. Для дипломного подання частина цих елементів наведена в додатках як структура, а не як комерційний installer.

До серверної частини входять Pydantic-контракти, service layer, інтеграційний шар мовної моделі, тести і конфігурація Python-пакету. До клієнтської частини входять SwiftUI views, API clients, session store і збереження попереднього перегляду вкладень. До агрономічної частини входять сегментаційний модуль і агрономічні ресурси.

Окремим елементом пакету є матеріали перевірки. Вони потрібні не для запуску, а для доказовості: у них зберігаються записи, які підтверджують або обмежують заяви про можливості. Для системи з LLM і CV це принципово,

оскільки без таких матеріалів легко помилково заявити ширші межі роботи, ніж реально перевірено.

Під час експлуатації оператор повинен контролювати стан серверної частини, завантаження моделі, готовність сегментаційного модуля, чергу, знання, аудити й сесії. Користувач на iPhone бачить лише ті стани та адміністративні відомості, які дозволені його роллю і операторськими правами, без дій керування Mac-середовищем.

Таблиця 4.7

Склад інсталяційного пакету

Частина	Вміст	Призначення
iPhone app	AgroChatiOS project/build	Користувацький інтерфейс
Backend	server package, contracts, services	API, контроль доступу, chats, RAG, стан системи
Сегментаційний модуль	scripts and code	Сегментація зображень
Resources	taxonomy, calibration, model registry, agronomy PDF	Перевірений агрономічний режим
Storage setup	State dir, file storage, SQLite stores	Канонічні локальні дані
Матеріали перевірки	протоколи тестування і звіти приймання	Докази прийнятих можливостей

4.4 Порядок експлуатації системи

Експлуатація AgroChat у локальному середовищі починається з підготовки Mac-сервера. Оператор повинен переконатися, що каталог стану доступний, SQLite-backed stores можуть бути створені або відкриті, файлове сховище має права запису, LM Studio запущено, а локальне середовище Gemma 4 E4B доступне.

Наступним кроком є запуск серверного API. Сервер повинен опублікувати `/healthz` як відкриту перевірку доступності, але стан системи, перелік можливостей і адміністративні відомості мають залишатися за автентифікованою межею. Це дозволяє iPhone перевірити з'єднання без витoku модельної інформації до неавторизованого користувача.

Після серверної частини запускається або перевіряється агрономічний сегментаційний модуль. Він повинен пройти початкову перевірку ваг моделі, таксономії, калібрування та працездатності. Якщо модуль не готовий, агрономічний режим має показувати обмежену працездатність або недоступність, а не давати користувачу фальшиву готовність.

Оператор створює або знаходить користувача і його рольову участь, після чого видає login token. Сирий token показується один раз і зберігається на сервері лише в non-recoverable формі. Користувач вводить token на iPhone, отримує bearer session і надалі працює через збережену сесію.

Користувач відкриває Chat і вибирає режим. Для звичайного питання достатньо Plain; для відповіді з джерелами використовується Grounded; для фото рослини - Agnomic. Режим повинен бути явним, бо кожний режим має різні гарантії та обмеження.

Під час фотодіагностики користувач вибирає зображення, перевіряє preview і надсилає запит. Якщо фото невіддале або клас не готовий для використання, система повідомляє про причину. Якщо діагностика успішна, відповідь повинна включати результат визначення класу і цитований агрономічний контекст.

Workspace використовується для перегляду організаційного контексту, нотаток і джерел знань. Це допомагає користувачу розуміти, з якими даними він працює. Однак Workspace не повинен ставати операторською панеллю керування; він лише показує дозволenu інформацію.

Екран стану системи використовується для діагностики проблем. Якщо модель не завантажена, сегментаційний модуль не готовий або джерело знань

відсутнє, користувач і оператор можуть бачити відповідний стан. Це зменшує кількість ситуацій, коли помилка сприймається як погана відповідь моделі.

Періодично потрібно переглядати журнал дій і матеріали перевірки. Для аграрної системи важливо знати, хто видав token, хто виконав запит, яке джерело було використано і чи не було подій безпечного завершення, які вимагають покращення даних або моделі.

Оновлення агрономічних ресурсів повинно виконуватися контрольовано. Якщо додати нові класи без калібрування або підготовлених джерел, система може почати заявляти непідтверджені можливості. Тому нові класи повинні входити лише після окремої перевірки, а не через автоматичне ввімкнення всього списку таксономії.

4.5 Критерії приймання та межі подальшого розвитку

Критерієм приймання системи не може бути лише успішна компіляція або запуск екрана. Для AgroChat приймання означає, що сценарій має спостережуваний результат і матеріали перевірки. Жодна можливість не повинна описуватися як реалізована без перевірки.

Для авторизації критерієм є успішний login-token exchange, безпечне завершення відкликаноного або простроченого доступу і відсутність публічної самореєстрації.

Для чату критерієм є створення повідомлення, цикл формування відповіді, потокові події, завершення обробки і збереження історії відповіді. Якщо мовна модель не повертає змістовного тексту, сервер повинен або сформувавати дозволений фінальний текст на основі джерел, або чесно завершити сценарій без фальшивої відповіді.

Для вкладень критерієм є окреме завантаження і прив'язка, правильна перевірка типу, володіння в межах дозволеної області доступу і відмова від непідтриманих медіа. Audio STT у прийнятому локальному середовищі LM

Studio не підтримується, тому audio не входить у приймання дипломної реалізації.

Для RAG критерієм є перевірка доступу до добору джерел, відповідь із цитатами і відсутність витоку недозволених джерел. Важливо, що підтримано PDF-джерела, але DOCX, CSV, URL і OCR не заявляються.

Для агрономічного режиму критерієм є сегментація, визначення агрономічного класу, поріг калібрування, наявність відповідного розділу джерела, відповідь із цитатами або завершальний стан безпечної відмови. Універсальна підтримка всіх 116 класів не є критерієм поточної роботи, бо вона не має відповідного підтвердження.

Для стану системи критерієм є збіг того, що показують серверна частина, адміністративна поверхня і read-only екран iPhone. Якщо контекстне вікно, ідентичність моделі або готовність сегментаційного модуля невідомі, система має повертати null або явну недоступність, а не вигадувати значення.

Подальший розвиток може йти в кількох напрямках. Перший - розширення груп агрономічних класів із підготовленими джерелами і перевіркою на пристрої. Другий - покращення якості пошуку й відповіді після створення відповідних тестових наборів. Третій - підготовка production infrastructure без зміни клієнтських контрактів.

Ще одним напрямом є покращення UX для польового процесу: кращі підказки щодо якості фото, локальні чернетки, offline capture queue або детальніший перегляд цитат. Проте такі функції повинні додаватися лише після оновлення плану і не повинні порушувати серверно-орієнтовану (server-first) межу відповідальності.

Найважливішою межею розвитку залишається чесність. Якщо майбутня версія додасть dense reranking, audio або ширшу агрономічну підтримку, кожна з цих можливостей повинна мати власне підтвердження. Інакше система втратить головну інженерну перевагу - відповідність між заявленим і реально перевіреним.

Таблиця 4.8

Критерії приймання програмного продукту

Критерій	Умова приймання	Доказ
Авторизація	Login-token exchange створює сесію, відкликаний або прострочений token завершується без доступу	Протокол тестування auth-сценаріїв і негативних перевірок
Рольовий доступ	Кожний запит виконується в межах активної рольової участі та організаційного контуру	Тести ACL і матеріали evidence artifacts
Чат	Повідомлення користувача створює цикл відповіді, потік подій і збережений фінальний результат	Таблиця тест-кейсів і скріншоти iPhone-сценарію
Вкладення	Файл проходить upload, validate, bind і не використовується поза дозволеною областю доступу	Негативні сценарії для malformed/foreign attachment
RAG	Відповідь на основі джерел містить дозволені цитати і не використовує недоступні матеріали	Evidence artifacts із citations і provenance
Агрономічний режим	Фото проходить сегментацію, калібрування, добір джерела і дає цитовану відповідь або безпечне завершення	Протокол агрономічного сценарію та terminal-state перевірки
Стан системи	Сервер, admin-контур і iPhone truth surface показують узгоджені можливості або явну недоступність	Скріншоти runtime surface і capability records
Обмеження	Аудіо, хмарний інференс, публічна реєстрація і універсальне покриття класів не заявляються як підтримані	Список claims_not_made у manifest і текстові обмеження роботи

4.6 Узагальнення проєктних рішень основної частини

Узагальнення проєктних рішень показує, що AgroChat побудовано як цілісну інформаційну систему для фотодіагностики рослин, а не як окремий класифікатор зображень. Основний сценарій починається в мобільному чаті: користувач фіксує фото, додає запит і отримує відповідь із поясненням, станом виконання та джерелами.

Ключове рішення полягає в серверно-орієнтованій архітектурі. iPhone-застосунок відповідає за зручну взаємодію, вибір режиму, вкладення і відображення результату, а локальний Mac-сервер виконує автентифікацію, контроль доступу, збереження історії, роботу з RAG-контекстом, виклик моделі та аудит. Такий розподіл зменшує ризик розходження між інтерфейсом і фактичним станом системи.

Інформаційна модель підтримує доказовість відповіді: користувачі працюють у межах ролей і організацій, зображення зберігаються як контрольовані вкладення, джерела знань мають стан готовності, а цитати пов'язують відповідь із конкретними матеріалами. Для агрономічного режиму додатково використовується перевірка сегментації, класу рослини та наявності відповідного джерела знань.

Окрему увагу приділено чесному відображенню можливостей. Якщо модель, джерело або агрономічний контур не дають достатніх підстав для висновку, система повинна завершити сценарій зрозумілим завершальним статусом, а не формувати неперевірену пораду. Це важливо для предметної області, де помилкова діагностика може впливати на господарські рішення.

Отже, обрані рішення забезпечують керованість, приватність локального прототипу, відтворюваність перевірок і можливість подальшого розвитку без зміни базової логіки взаємодії. У першому розділі обґрунтовано потребу в мобільній фотодіагностиці, у другому описано інформаційну модель, у третьому розкрито програмні компоненти, а в четвертому визначено впровадження, тестування та межі використання системи.

ВИСНОВКИ

У бакалаврській кваліфікаційній роботі розроблено інформаційну систему AgroChat для фотодіагностики рослин за допомогою нейронних мереж. Система реалізує iPhone-first підхід, у якому мобільний застосунок є головною користувацькою поверхнею, а локальний Mac-сервер відповідає за доступ, збереження, інференс, RAG, аудит і стан роботи системи.

Проведено системний аналіз предметної області та показано, що для практичної фотодіагностики недостатньо лише класифікації зображення. Потрібний повний інформаційний контур, який поєднує фото, запит користувача, знання, рольовий доступ, пояснення і доказові джерела. AgroChat реалізує такий контур через чат-орієнтований (chat-first) UX і серверно-орієнтовану (server-first) архітектуру.

Розроблено інформаційне забезпечення системи: модель організаційних контурів, ролей користувачів, чатів, повідомлень, вкладень, джерел знань, одиниць пошуку, цитат, станів системи та результатів фотодіагностики. Для поточного локального контуру обґрунтовано використання durable SQLite-backed stores і локального content-addressed file storage.

Розроблено прикладне програмне забезпечення: SwiftUI iPhone app, FastAPI backend, інтеграційний шар мовної моделі Gemma 4 E4B, PDF/RAG контур, екрани стану системи та агрономічний сегментаційний модуль. Алгоритм агрономічної відповіді побудовано як послідовність із безпечним завершенням: зображення -> сегментація -> перевірка таксономії та калібрування -> RAG за агрономічним класом -> відповідь із цитатами.

Проведено аналіз впровадження й тестування. Система має набір тестів і матеріалів перевірки, які підтверджують окремі сценарії, але не розширюють межі роботи без доказів. У роботі свідомо не заявлено аудіопідтримку, хмарний

інференс, публічну самореєстрацію, GPU-only execution, універсальне агрономічне покриття або загальне розуміння будь-яких документів.

Перевагою розробки є поєднання мобільності, локальної приватності, доказового RAG, ролей, аудиту й чесного відображення можливостей. Подальший розвиток може включати розширення переліку перевірених агрономічних класів, покращення якості пошуку після відповідних перевірок і підготовку production deployment без зміни клієнтських контрактів.

Порівняння із завданням показує виконання основних пунктів. Завдання 1 виконано: проведено аналіз предметної області та існуючих рішень через огляд наукових джерел і продуктів Plantix, PictureThis, Pl@ntNet та Agrio. Завдання 2 виконано: спроектовано інформаційне забезпечення через логічну модель, життєві цикли даних і аналіз ЗНФ. Завдання 3 виконано: розроблено програмне забезпечення через native iPhone app, FastAPI backend, RAG-цитування і сегментаційний модуль. Завдання 4 виконано: проведено тестування та опис впровадження через протокол тестування, негативні сценарії, матеріали підтвердження приймання і опис експлуатації.

Порівняно з Plantix і Agrio, система AgroChat акцентує локальний інференс, операторськи виданий доступ, рольовий контроль доступу і аудит. Порівняно з PictureThis вона орієнтована не на споживчий догляд за рослинами, а на контрольований аграрний чат із джерелами. Порівняно з Pl@ntNet вона вирішує не лише задачу ідентифікації виду, а й задачу доказової фотодіагностики з RAG за агрономічним класом.

Особистий внесок автора полягає в тому, що особисто розроблено архітектуру AgroChat, серверну частину, native iPhone-клієнт, логічну та SQL-модель інформаційної бази, інтеграцію сегментаційного модуля, RAG-цитування, тестовий протокол, матеріали перевірки і пояснювальну документацію. Автор також виконав узгодження тексту диплома з реальними межами системи, щоб не заявляти непідтверджені можливості.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Методичні вказівки до розробки бакалаврської кваліфікаційної роботи для студентів спеціальності F3 «Комп'ютерні науки». Київ: НУБіП України, 2025.
2. ДСТУ 8302:2015. Бібліографічне посилання. Загальні положення та правила складання. Київ: ДП «УкрНДНЦ», 2016.
3. Plant Disease Detection and Classification: A Systematic Literature Review. *Sensors*. 2023. Vol. 23, No. 10. Article 4769. DOI: 10.3390/s23104769.
4. Role of Internet of Things and Deep Learning Techniques in Plant Disease Detection and Classification: A Focused Review. *Sensors*. 2023. Vol. 23, No. 18. Article 7877. DOI: 10.3390/s23187877.
5. Shoaib M., Shah B., El-Sappagh S. et al. An advanced deep learning models-based plant disease detection: a review of recent research. *Frontiers in Plant Science*. 2023. URL: <https://www.frontiersin.org/articles/10.3389/fpls.2023.1158933/full> (дата звернення: 19.05.2026).
6. A systematic review of deep learning techniques for plant diseases. *Artificial Intelligence Review*. 2024. Vol. 57. Article 304. DOI: 10.1007/s10462-024-10944-7.
7. Hughes D. P., Salathé M. An open access repository of images on plant health to enable the development of mobile disease diagnostics. arXiv:1511.08060. URL: <https://arxiv.org/abs/1511.08060> (дата звернення: 19.05.2026).
8. Mohanty S. P., Hughes D. P., Salathé M. Using Deep Learning for Image-Based Plant Disease Detection. *Frontiers in Plant Science*. 2016. Vol. 7. Article 1419. DOI: 10.3389/fpls.2016.01419. URL: <https://www.frontiersin.org/articles/10.3389/fpls.2016.01419/full> (дата звернення: 19.05.2026).

9. Redmon J., Divvala S., Girshick R., Farhadi A. You Only Look Once: Unified, Real-Time Object Detection. arXiv:1506.02640. URL: <https://arxiv.org/abs/1506.02640> (дата звернення: 19.05.2026).
10. Selvaraju R. R. et al. Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization. arXiv:1610.02391. URL: <https://arxiv.org/abs/1610.02391> (дата звернення: 19.05.2026).
11. Lewis P. et al. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. Advances in Neural Information Processing Systems. 2020. URL: <https://papers.neurips.cc/paper/2020/hash/6b493230205f780e1bc26945df7481e5-Abstract.html> (дата звернення: 19.05.2026).
12. Gao Y. et al. Retrieval-Augmented Generation for Large Language Models: A Survey. arXiv:2312.10997. URL: <https://huggingface.co/papers/2312.10997> (дата звернення: 19.05.2026).
13. A Survey on Hallucination in Large Language Models: Principles, Taxonomy, Challenges, and Open Questions. arXiv:2311.05232. URL: <https://arxiv.org/abs/2311.05232> (дата звернення: 19.05.2026).
14. Plantix. Plantix - free app for crop diagnosis and treatments. URL: <https://plantix.net/en/> (дата звернення: 19.05.2026).
15. PictureThis. Online plant encyclopedia and plant identifier. URL: <https://www.picturethisai.com/app> (дата звернення: 19.05.2026).
16. Pl@ntNet. About Pl@ntNet. URL: <https://plantnet.org/en/about/> (дата звернення: 19.05.2026).
17. Agrio. Plant health and crop monitoring application materials. URL: <https://agrio.app/> (дата звернення: 19.05.2026).
18. Національний університет біоресурсів і природокористування України. Конференція студентів та аспірантів «Теоретичні та прикладні аспекти розробки комп'ютерних систем '2026». URL:

- <https://nubip.edu.ua/news/konferentsiya-studentiv-ta-aspirantiv-teoretychni-ta-prykladni-aspekty-rozrobky-kompyuternykh> (дата звернення: 19.05.2026).
19. Howard A. G. et al. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. arXiv:1704.04861. URL: <https://arxiv.org/abs/1704.04861> (дата звернення: 19.05.2026).
 20. Tan M., Le Q. V. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. arXiv:1905.11946. URL: <https://arxiv.org/abs/1905.11946> (дата звернення: 19.05.2026).
 21. Apple Inc. SwiftUI Documentation. URL: <https://developer.apple.com/documentation/swiftui> (дата звернення: 19.05.2026).
 22. Apple Inc. URLSession Documentation. URL: <https://developer.apple.com/documentation/foundation/urlsession> (дата звернення: 19.05.2026).
 23. FastAPI Documentation. URL: <https://fastapi.tiangolo.com/> (дата звернення: 19.05.2026).
 24. Pydantic Documentation. URL: <https://docs.pydantic.dev/> (дата звернення: 19.05.2026).
 25. SQLite Documentation. URL: <https://www.sqlite.org/docs.html> (дата звернення: 19.05.2026).
 26. LM Studio Documentation. URL: <https://lmstudio.ai/docs> (дата звернення: 19.05.2026).
 27. Google. Gemma model family documentation. URL: <https://ai.google.dev/gemma> (дата звернення: 19.05.2026).
 28. Ultralytics YOLO Documentation. URL: <https://docs.ultralytics.com/> (дата звернення: 19.05.2026).
 29. Лесь М.О. Інформаційна система фотодіагностики рослин на основі нейронних мереж: тези доповіді. VII Всеукраїнська науково-практична

конференція студентів та аспірантів «Теоретичні та прикладні аспекти розробки комп'ютерних систем '2026», 29 квітня 2026 року, НУБіП України, м. Київ. Науковий керівник: PhD, старший викладач Золотуха Р.А.

ДОДАТОК А. ГРАФІЧНА ЧАСТИНА

Графічні матеріали до проєкту



Рисунок 27 - Діаграма прецедентів фотодіагностичної системи AgroChat



Рисунок 28 - Логічна модель інформаційної бази AgroChat

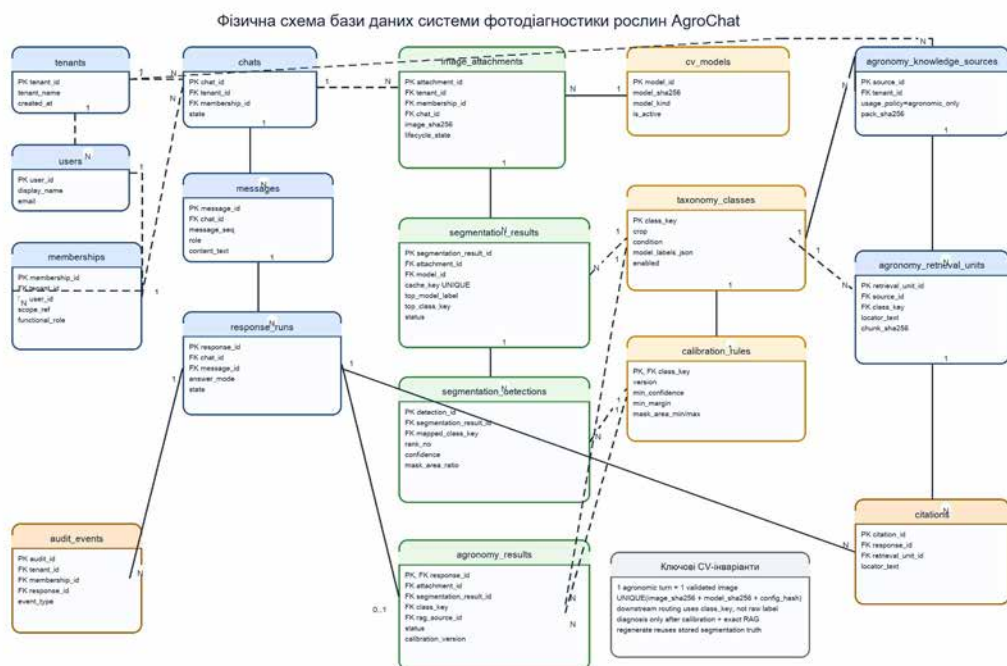


Рисунок 29 - Фізична модель зберігання даних фотодіагностичної системи

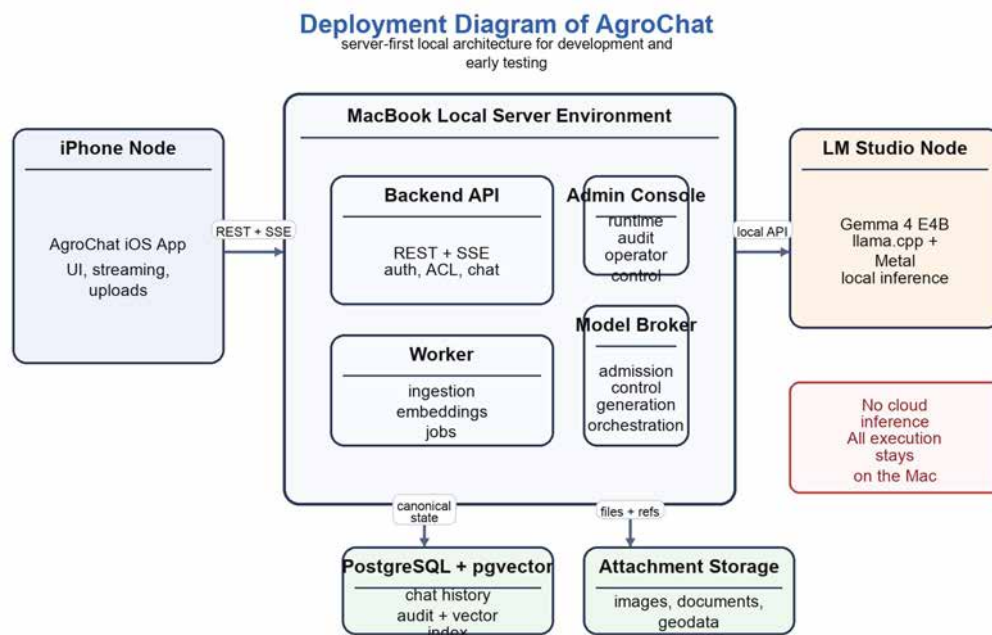


Рисунок 30 - Діаграма розгортання AgroChat у локальному середовищі

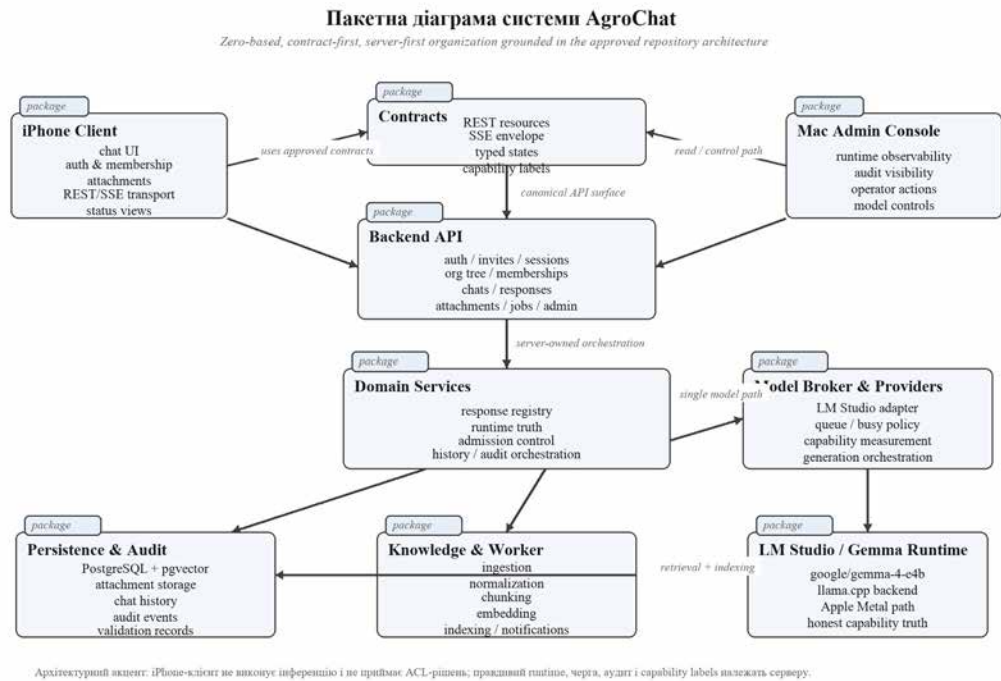


Рисунок 31 - Пакетна структура програмного забезпечення AgroChat

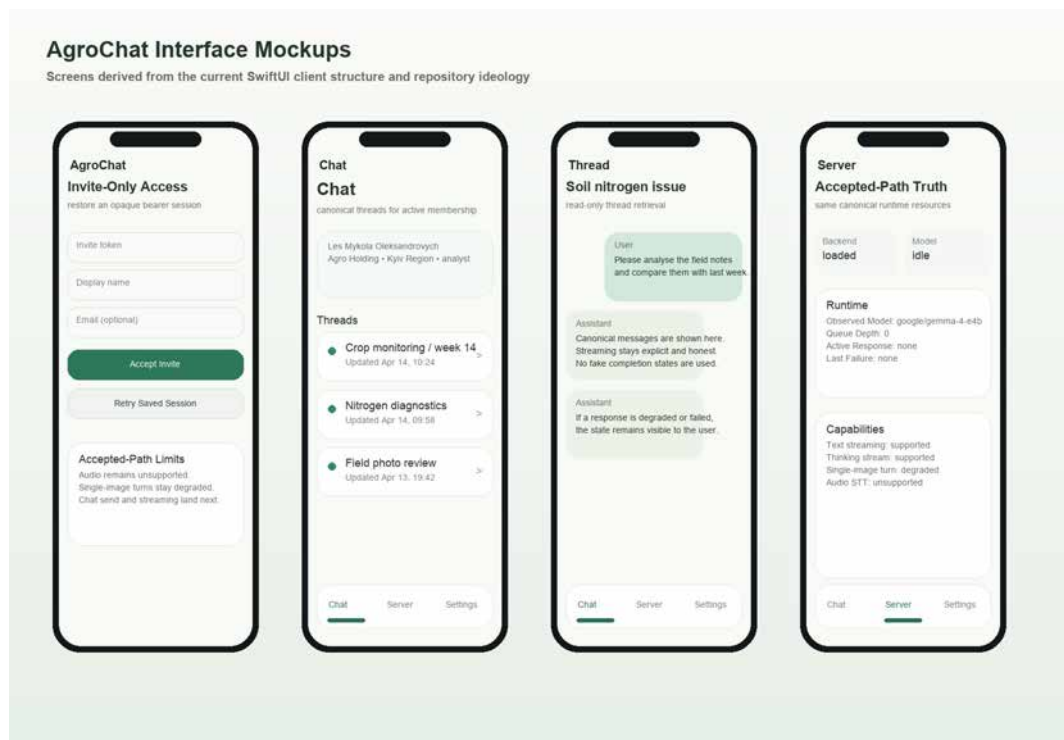


Рисунок 32 - Логіка взаємодії основних екранів iPhone-застосунку

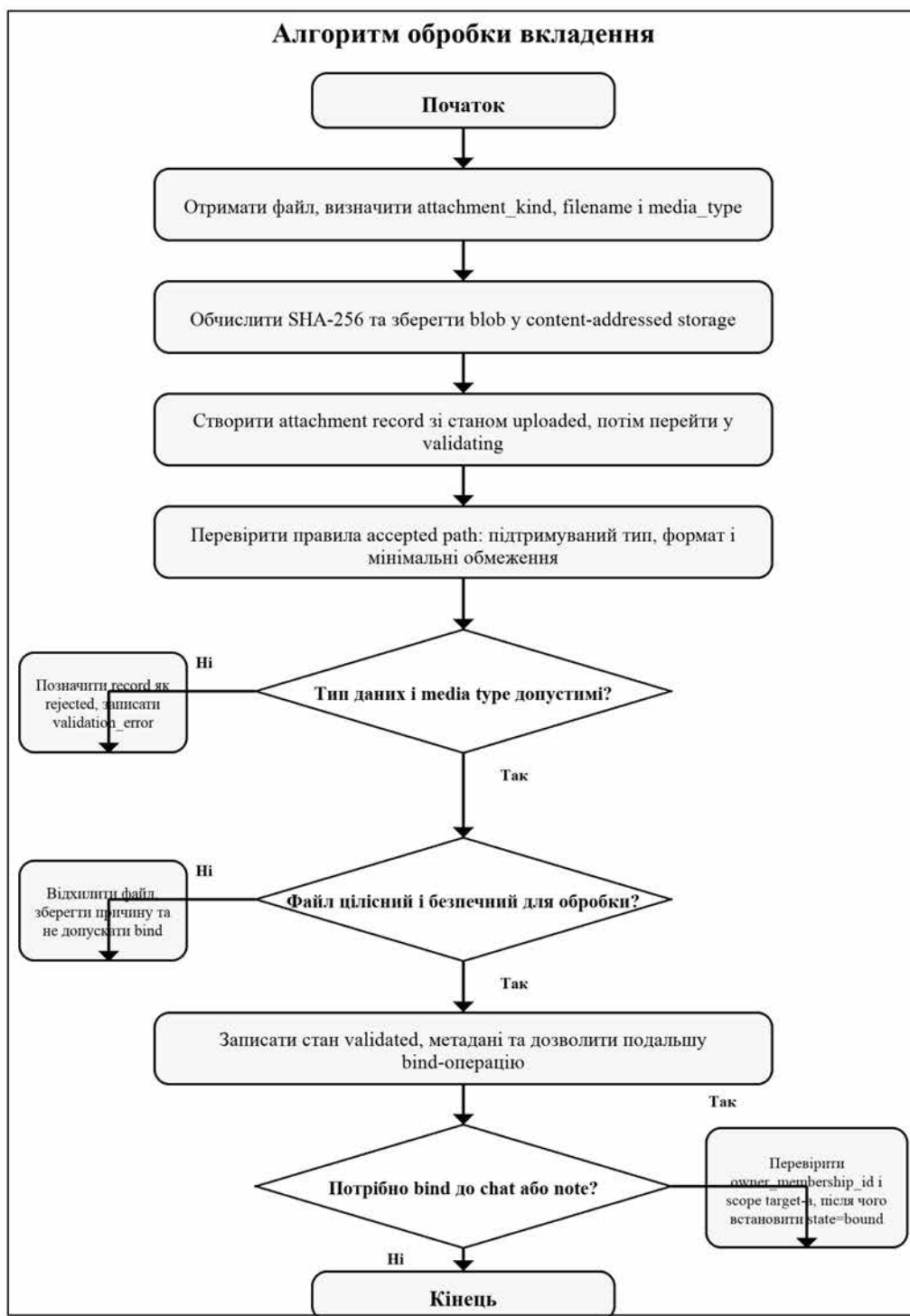


Рисунок 33 - Схема збереження та обробки фотодіагностичного запиту

Алгоритм відбору даних для звітно-статистичної інформації AgroChat

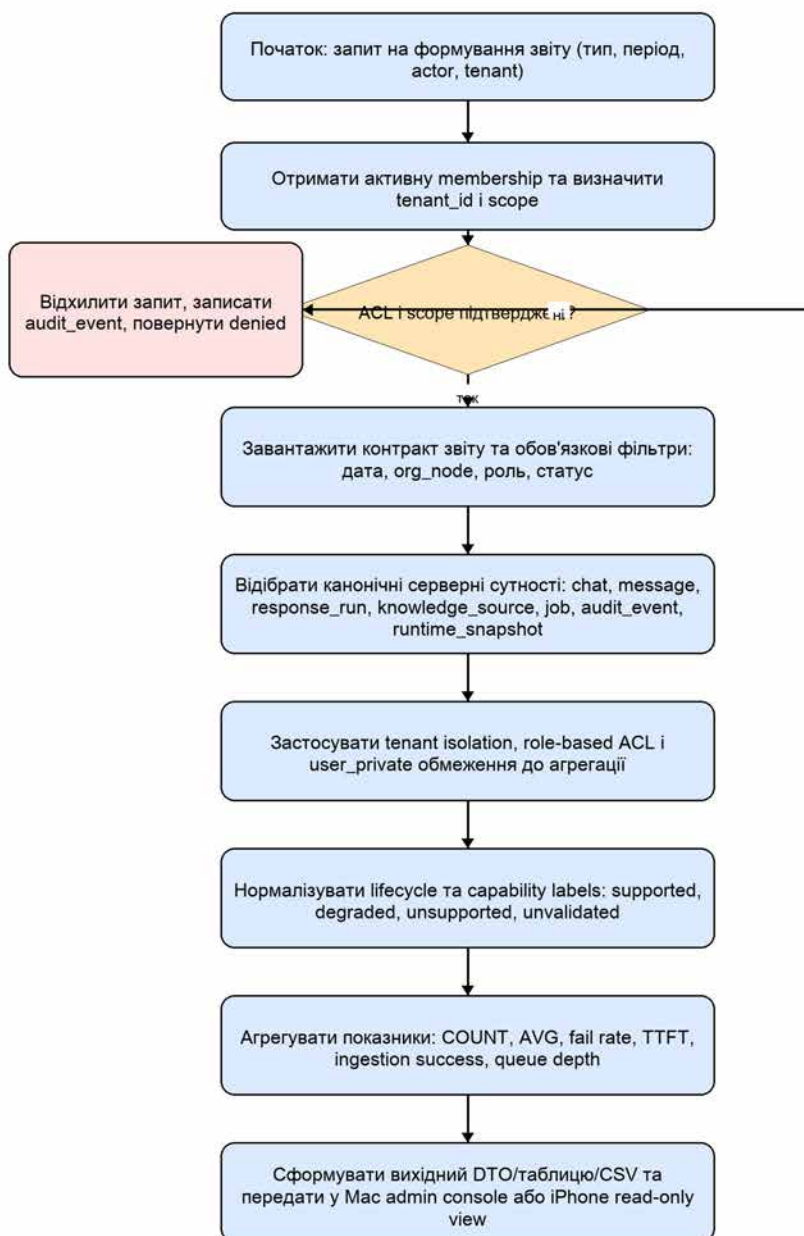


Рисунок 34 - Схема формування звітних даних системи

Наведені графічні матеріали доповнюють опис архітектури, інформаційної моделі, інтерфейсу та процесів AgroChat. Текстові висновки у пояснювальній записці приведені до актуального опису архітектури системи.

ДОДАТОК Б. SQL-ЛІСТИНГ ІНФОРМАЦІЙНОЇ БАЗИ

SQL-лістинг фізичної схеми інформаційної бази

У додатку наведено повний релевантний навчальний SQL-лістинг фізичної схеми інформаційної бази, яка відповідає сутностям, описаним у розділі 2. Лістинг подано у SQLite-сумісному стилі для пояснювальної записки; робоча реалізація може розподіляти ці сутності між кількома локальними сховищами за межами сервісів.

Схема охоплює довідники, організаційні контури, користувачів і рольові участі, сесії, чати, повідомлення, вкладення, джерела знань, одиниці пошуку, цитати, агрономічні результати, журнал дій і записи стану виконання.

```
PRAGMA foreign_keys = ON;

CREATE TABLE ref_membership_roles (
    role_code TEXT PRIMARY KEY,
    role_label TEXT NOT NULL
);

CREATE TABLE ref_lifecycle_states (
    state_code TEXT PRIMARY KEY,
    state_group TEXT NOT NULL,
    is_terminal INTEGER NOT NULL CHECK (is_terminal IN (0, 1))
);

CREATE TABLE ref_answer_modes (
    mode_code TEXT PRIMARY KEY,
    mode_label TEXT NOT NULL
);

CREATE TABLE ref_audit_event_types (
    event_type TEXT PRIMARY KEY,
    description TEXT NOT NULL
);

CREATE TABLE tenants (
    tenant_id TEXT PRIMARY KEY,
    name TEXT NOT NULL,
    lifecycle_state TEXT NOT NULL REFERENCES ref_lifecycle_states(state_code),
    created_at TEXT NOT NULL
);

CREATE TABLE org_nodes (
    org_node_id TEXT PRIMARY KEY,
    tenant_id TEXT NOT NULL REFERENCES tenants(tenant_id),
    parent_org_node_id TEXT REFERENCES org_nodes(org_node_id),
    node_type TEXT NOT NULL,
    name TEXT NOT NULL,
    UNIQUE (tenant_id, parent_org_node_id, name)
);

CREATE TABLE users (
    user_id TEXT PRIMARY KEY,
```

```

        display_name TEXT NOT NULL,
        email TEXT UNIQUE,
        created_at TEXT NOT NULL
    );

CREATE TABLE memberships (
    membership_id TEXT PRIMARY KEY,
    tenant_id TEXT NOT NULL REFERENCES tenants(tenant_id),
    org_node_id TEXT NOT NULL REFERENCES org_nodes(org_node_id),
    user_id TEXT NOT NULL REFERENCES users(user_id),
    role_code TEXT NOT NULL REFERENCES ref_membership_roles(role_code),
    lifecycle_state TEXT NOT NULL REFERENCES ref_lifecycle_states(state_code),
    UNIQUE (tenant_id, org_node_id, user_id, role_code)
);

CREATE TABLE sessions (
    session_id TEXT PRIMARY KEY,
    user_id TEXT NOT NULL REFERENCES users(user_id),
    active_membership_id TEXT REFERENCES memberships(membership_id),
    issued_at TEXT NOT NULL,
    expires_at TEXT NOT NULL,
    revoked_at TEXT
);

CREATE TABLE login_tokens (
    token_id TEXT PRIMARY KEY,
    user_id TEXT NOT NULL REFERENCES users(user_id),
    token_hash TEXT NOT NULL UNIQUE,
    issued_by_membership_id TEXT NOT NULL REFERENCES memberships(membership_id),
    issued_at TEXT NOT NULL,
    expires_at TEXT NOT NULL,
    consumed_at TEXT,
    revoked_at TEXT
);

CREATE TABLE chats (
    chat_id TEXT PRIMARY KEY,
    tenant_id TEXT NOT NULL REFERENCES tenants(tenant_id),
    owner_membership_id TEXT NOT NULL REFERENCES memberships(membership_id),
    title TEXT,
    lifecycle_state TEXT NOT NULL REFERENCES ref_lifecycle_states(state_code),
    created_at TEXT NOT NULL,
    updated_at TEXT NOT NULL
);

CREATE TABLE messages (
    message_id TEXT PRIMARY KEY,
    chat_id TEXT NOT NULL REFERENCES chats(chat_id) ON DELETE CASCADE,
    membership_id TEXT REFERENCES memberships(membership_id),
    role TEXT NOT NULL CHECK (role IN ('user', 'assistant', 'system')),
    body TEXT NOT NULL,
    created_at TEXT NOT NULL
);

CREATE TABLE response_runs (
    response_id TEXT PRIMARY KEY,
    chat_id TEXT NOT NULL REFERENCES chats(chat_id) ON DELETE CASCADE,
    trigger_message_id TEXT NOT NULL REFERENCES messages(message_id),
    answer_mode TEXT NOT NULL REFERENCES ref_answer_modes(mode_code),
    lifecycle_state TEXT NOT NULL REFERENCES ref_lifecycle_states(state_code),
    attempt_no INTEGER NOT NULL CHECK (attempt_no > 0),
    started_at TEXT,
    completed_at TEXT,
    terminal_reason TEXT
);

CREATE TABLE attachments (
    attachment_id TEXT PRIMARY KEY,
    tenant_id TEXT NOT NULL REFERENCES tenants(tenant_id),
    owner_membership_id TEXT NOT NULL REFERENCES memberships(membership_id),
    media_type TEXT NOT NULL,
    content_sha256 TEXT NOT NULL,
    byte_size INTEGER NOT NULL CHECK (byte_size >= 0),

```

```

        lifecycle_state TEXT NOT NULL REFERENCES ref_lifecycle_states(state_code),
        created_at TEXT NOT NULL
    );

CREATE TABLE message_attachments (
    message_id TEXT NOT NULL REFERENCES messages(message_id) ON DELETE CASCADE,
    attachment_id TEXT NOT NULL REFERENCES attachments(attachment_id),
    PRIMARY KEY (message_id, attachment_id)
);

```

Окремий блок таблиць знань забезпечує зв'язок між PDF-джерелом, нормалізованими фрагментами, одиницями пошуку та цитатами у відповіді.

```

CREATE TABLE knowledge_sources (
    source_id TEXT PRIMARY KEY,
    tenant_id TEXT REFERENCES tenants(tenant_id),
    owner_membership_id TEXT REFERENCES memberships(membership_id),
    source_kind TEXT NOT NULL CHECK (source_kind IN ('pdf','agronomy_pack')),
    title TEXT NOT NULL,
    content_sha256 TEXT NOT NULL,
    lifecycle_state TEXT NOT NULL REFERENCES ref_lifecycle_states(state_code),
    created_at TEXT NOT NULL
);

CREATE TABLE knowledge_chunks (
    chunk_id TEXT PRIMARY KEY,
    source_id TEXT NOT NULL REFERENCES knowledge_sources(source_id) ON DELETE CASCADE,
    chunk_index INTEGER NOT NULL CHECK (chunk_index >= 0),
    locator TEXT NOT NULL,
    normalized_text TEXT NOT NULL,
    UNIQUE (source_id, chunk_index)
);

CREATE TABLE retrieval_units (
    retrieval_unit_id TEXT PRIMARY KEY,
    source_id TEXT NOT NULL REFERENCES knowledge_sources(source_id),
    chunk_id TEXT NOT NULL REFERENCES knowledge_chunks(chunk_id),
    class_key TEXT,
    usage_policy TEXT NOT NULL,
    version_hash TEXT NOT NULL
);

CREATE TABLE citations (
    citation_id TEXT PRIMARY KEY,
    response_id TEXT NOT NULL REFERENCES response_runs(response_id) ON DELETE CASCADE,
    retrieval_unit_id TEXT NOT NULL REFERENCES retrieval_units(retrieval_unit_id),
    source_id TEXT NOT NULL REFERENCES knowledge_sources(source_id),
    locator TEXT NOT NULL,
    version_hash TEXT NOT NULL,
    created_at TEXT NOT NULL,
    UNIQUE (response_id, retrieval_unit_id, locator)
);

CREATE TABLE taxonomy_classes (
    class_key TEXT PRIMARY KEY,
    crop_name TEXT NOT NULL,
    condition_name TEXT NOT NULL,
    is_enabled INTEGER NOT NULL CHECK (is_enabled IN (0, 1))
);

CREATE TABLE segmentation_results (
    segmentation_id TEXT PRIMARY KEY,
    attachment_id TEXT NOT NULL REFERENCES attachments(attachment_id),
    model_version TEXT NOT NULL,
    raw_label TEXT NOT NULL,
    class_key TEXT REFERENCES taxonomy_classes(class_key),
    confidence REAL NOT NULL CHECK (confidence >= 0 AND confidence <= 1),
    image_sha256 TEXT NOT NULL,

```

```

        created_at TEXT NOT NULL
    );

CREATE TABLE agronomy_results (
    result_id TEXT PRIMARY KEY,
    response_id TEXT NOT NULL REFERENCES response_runs(response_id),
    attachment_id TEXT NOT NULL REFERENCES attachments(attachment_id),
    segmentation_id TEXT REFERENCES segmentation_results(segmentation_id),
    class_key TEXT REFERENCES taxonomy_classes(class_key),
    calibration_version TEXT NOT NULL,
    confidence REAL CHECK (confidence >= 0 AND confidence <= 1),
    terminal_status TEXT NOT NULL,
    created_at TEXT NOT NULL
);

```

Журнал дій і записи стану виконання фіксують службову поведінку системи, щоб можливості, обмеження та дії оператора були перевірюваними після завершення сценарію.

```

CREATE TABLE audit_events (
    audit_event_id TEXT PRIMARY KEY,
    tenant_id TEXT REFERENCES tenants(tenant_id),
    actor_membership_id TEXT REFERENCES memberships(membership_id),
    event_type TEXT NOT NULL REFERENCES ref_audit_event_types(event_type),
    target_kind TEXT NOT NULL,
    target_id TEXT,
    event_payload_json TEXT NOT NULL DEFAULT '{}',
    created_at TEXT NOT NULL
);

CREATE TABLE runtime_snapshots (
    snapshot_id TEXT PRIMARY KEY,
    backend_name TEXT NOT NULL,
    model_identity TEXT,
    context_window INTEGER,
    queue_state TEXT NOT NULL,
    segmentation_state TEXT NOT NULL,
    captured_at TEXT NOT NULL
);

CREATE TABLE capability_records (
    capability_id TEXT PRIMARY KEY,
    capability_name TEXT NOT NULL UNIQUE,
    support_state TEXT NOT NULL CHECK (support_state IN
('supported','unsupported','unvalidated','deferred')),
    evidence_ref TEXT,
    updated_at TEXT NOT NULL
);

CREATE INDEX idx_memberships_user ON memberships(user_id);
CREATE INDEX idx_chats_owner ON chats(owner_membership_id);
CREATE INDEX idx_messages_chat ON messages(chat_id);
CREATE INDEX idx_attachments_owner ON attachments(owner_membership_id);
CREATE INDEX idx_citations_response ON citations(response_id);
CREATE INDEX idx_audit_actor_time ON audit_events(actor_membership_id, created_at);

```

ДОДАТОК В. ФРАГМЕНТИ ПРОГРАМНИХ МОДУЛІВ

Фрагменти програмних модулів

FastAPI endpoint створення chat turn

```

1048:     @app.post("/v1/chats/{chat_id}/turns", response_model=CreateTurnAccepted)
1049:     async def create_chat_turn(
1050:         chat_id: str,
1051:         request: CreateTurnRequest,
1052:         authorization: str | None = Header(default=None),
1053:     ) -> CreateTurnAccepted:
1054:         context = _require_auth_context(authorization)
1055:         _require_accepted_model(request.model)
1056:         try:
1057:             chat = app.state.chat.require_chat_access(context, chat_id)
1058:         except AuthError as exc:

```

Pydantic DTO для chat turn

```

0075: class CreateTurnRequest(BaseModel):
0076:     model: str
0077:     input_text: str = Field(min_length=1)
0078:     stream: bool = True
0079:     answer_mode: AnswerMode | None = None
0080:     grounding_mode: GroundingMode = "disabled"
0081:     tool_mode: ToolMode = "disabled"
0082:     vision_mode: VisionMode = "disabled"
0083:     vision_attachment_id: str | None = None
0084:

```

SwiftUI view головної chat-поверхні

```

0060: struct ChatPlaceholderView: View {
0061:     @Environment(AppEnvironment.self) private var appEnvironment
0062:     @Environment(\.colorScheme) private var colorScheme
0063:     @State private var pendingNavigationChat: ChatSummaryRecord?
0064:
0065:     var body: some View {
0066:         ZStack {
0067:             palette.screenBackground
0068:                 .ignoresSafeArea()
0069:

```

Swift model/state для createTurn

```

2965:     private func createTurn(
2966:         chatID: String?,
2967:         inputText: String,
2968:         answerMode: ChatAnswerMode? = nil,
2969:         groundingMode: String = AppEnvironment.groundedGroundingMode,
2970:         requestedToolMode: String,
2971:         visionMode: String = AppEnvironment.disabledVisionMode,
2972:         visionAttachmentID: String? = nil,
2973:         allowComposerSubmissionInFlight: Bool = false
2974:     ) async -> ChatSummaryRecord? {
2975:         guard let sessionToken else {

```

Backend-виклик segmentation worker

```

0481:         async def _segment_image(self, *, image_sha256: str, content: bytes) -> dict[str,
Any]:
0482:             if not self._settings.agronomy_worker_base_url:
0483:                 return {}
0484:             timeout = httpx.Timeout(10.0, read=self._settings.read_timeout_s)
0485:             async with httpx.AsyncClient(timeout=timeout) as client:
0486:                 response = await client.post(
0487:                     f"{self._settings.agronomy_worker_base_url.rstrip('/')}/segment",
0488:                     json={
0489:                         "image_sha256": image_sha256,
0490:                         "image_base64": base64.b64encode(content).decode("ascii"),

```

Збереження citation records

```

0258:         def attach_citations(self, response_id: str, citations: list[CitationRecord]) ->
ResponseRunRecord:
0259:             stored = self._require(response_id)
0260:             stored.citations = list(citations)
0261:             stored.last_event_at = datetime.now(timezone.utc)
0262:             self._persist(stored)
0263:             return self._to_record(stored)
0264:

```

**ДОДАТОК Г. ДЕКЛАРАЦІЯ ПРО АКАДЕМІЧНУ
ДОБРОЧЕСНІСТЬ ТА ВИКОРИСТАННЯ ІІІ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І
ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ**

Факультет інформаційних технологій

Декларація про академічну доброчесність та використання ІІІ

Я, Лесь Микола Олександрович, здобувач НУБіП України, усвідомлюючи відповідальність за порушення академічної доброчесності, цією заявою підтверджую, що:

Моя бакалаврська кваліфікаційна робота на тему «Розробка інформаційної системи для фотодіагностики рослин за допомогою нейронних мереж» є результатом моєї особистої праці.

Усі запозичення з текстів інших авторів мають відповідні посилання згідно з чинними стандартами.

Щодо використання інструментів ІІІ зазначаю, що інструменти ІІІ ChatGPT, Gemini були використані для:

перекладу іншомовних джерел;

стилістичного редагування та перевірки граматики;

допомоги у написанні/відлагодженні програмного коду;

інше: пошуку додаткових інструментів у розробці програмного продукту.

Я розумію, що надання недостовірної інформації може призвести до скасування результатів захисту та відрахування з числа здобувачів НУБіП України.

«22» травня 2026 р. _____

Микола ЛЕСЬ