МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ВЕТЕРИНАРНОЇ МЕДИЦИНИ ТА БІОТЕХНОЛОГІЙ ІМ. С.З.ГЖИЦЬКОГО

ФАКУЛЬТЕТ МЕХАНІКИ, ЕНЕРГЕТИКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КВАЛІФІКАЦІЙНА РОБОТА

першого (бакалаврського) рівня вищої освіти

на тему: **«Розробка Desktop додатку для визначення фонду часу виконання робіт у рослинництві»**

Виконав: студент 4 курсу групи Кн–41

Спеціальності 122 «Комп'ютерні науки»
(шифр і назва)

Підгорецький Андрій Романович
(Прізвище та ініціали)

Керівник: к.т.н., доцент Татомир А.В.
(Прізвище та ініціали)

Рецензент: к.т.н., доцент Шарибура А.О.
(Прізвище та ініціали)

ДУБЛЯНИ–2025

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ПРИРОДОКОРИСТУВАННЯ
ФАКУЛЬТЕТ МЕХАНІКИ, ЕНЕРГЕТИКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Перший (бакалаврський) рівень вищої освіти
Спеціальність 122 «Комп'ютерні науки»

«ЗАТВЕРДЖУЮ»

Завідувач кафедри_____

д.т.н., проф. А. М. Тригуба
«____» _____2025 р.

ЗАВДАННЯ

на кваліфікаційну роботу студенту

Підгорецькому Андрію Романовичу

1. Тема роботи: «Розробка Desktop додатку для визначення фонду часу виконання робіт у рослинництві»

Керівник роботи Татомир Андрій Володимирович, доцент
затверджені наказом по університету від 25.02.2025 року № 123/к–с.

2. Строк подання студентом роботи 10.06.2025 р.
3. Вихідні дані до роботи: *статистичні дані кліматичних умов; вимоги до фонду часу виконання робіт у рослинництві.*

4. Зміст розрахунково-пояснювальної записки (перелік питань, які необхідно розробити)
Вступ.
1. Аналіз стану питання щодо визначення фонду часу виконання робіт у рослинництві.
2. Вибір інструментарію для створення Desktop додатку визначення фонду часу виконання робіт у рослинництві.
3. Створення програмних модулів Desktop додатку для визначення фонду часу виконання робіт у рослинництві.
4. Розробка інтерфейсу користувача та практичне використання Desktop додатку для визначення фонду часу виконання робіт у рослинництві.
5. Охорона праці.
Висновки та пропозиції.
Список використаної літератури.

5. Перелік ілюстраційного матеріалу (з точним зазначенням обов'язкових креслень): *аналіз стану питання щодо визначення фонду часу виконання робіт у рослинництві; вибір інструментарію для створення Desktop додатку визначення фонду часу виконання робіт у рослинництві; створення програмних модулів Desktop додатку для визначення фонду часу виконання робіт у рослинництві; розробка інтерфейсу користувача та практичне використання Desktop додатку для визначення фонду часу виконання робіт у рослинництві.*

6. Консультанти з розділів:

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1, 2, 3, 4	*Татомир А.В., доцент кафедри ІТ*		
5	*Городецький І.М., доцент кафедри інженерної механіки*		

7. Дата видачі завдання 25 лютого 2025 р.

Календарний план

№ з/п	Назва етапів кваліфікаційної роботи	Терміни виконання етапів роботи	При–мітка
1	*Написання першого розділу*	*25.02–11.03.25*	
2	*Виконання другого розділу та аркушів ілюстраційного матеріалу до нього*	*12.03–15.04.25*	
3.	*Виконання третього розділу та аркушів ілюстраційного матеріалу до нього*	*16.04–20.05.25*	
4.	*Написання розділу «Охорона праці»*	*21.05–21.05.25*	
5.	*Завершення оформлення розрахунково–пояснювальної записки та аркушів ілюстраційного матеріалу*	*22–31.05.25*	
6.	*Завершення роботи вцілому*	*01–10.06.25*	

Студент _____ Підгорецький А.Р.
(підпис)

Керівник роботи_____ Татомир А.В.
(підпис)

УДК 004.4’2:631.5

Розробка Desktop додатку для визначення фонду часу виконання робіт у рослинництві.

Підгорецький А.Р. Кафедра ІТ – Дубляни, Львівський НУВМ, 2025.

Кваліфікаційна робота: 63 с. текст. част., 18 рис., 9 табл., 14 арк. ілюстраційного матеріалу, 39 джерел.

У роботі представлено результати розробки Desktop додатку, призначеного для визначення фонду часу виконання агротехнічних операцій у галузі рослинництва. Актуальність теми зумовлена потребою в автоматизації планування сільськогосподарських робіт із врахуванням виробничих параметрів, погодних умов та ефективності використання техніки. Розроблений додаток реалізовано за допомогою мови програмування Python із використанням бібліотек Tkinter, matplotlib та requests.

Програмний інтерфейс забезпечує зручне введення вхідних даних (площа, тип операції, продуктивність, коефіцієнти використання часу та ризику), а також дозволяє отримати поточні погодні дані через API OpenWeatherMap. Вбудований модуль обчислень враховує погодний вплив на тривалість робіт, що підвищує точність результатів. Крім текстового виводу, додаток формує графік залежності фонду часу та ризику від площі поля, що візуалізується у вигляді нелінійної залежності з подвійною шкалою.

У ході дослідження виконано аналіз небезпечних і шкідливих чинників, пов’язаних із роботою за комп’ютером, та сформовано інструкцію з охорони праці, що враховує ергономічні, електротехнічні та психофізіологічні аспекти. Результати тестування свідчать про ефективність використання створеного додатку для обґрунтованого прийняття рішень в аграрному секторі.

Ключові слова: Desktop додаток, рослинництво, агротехнічні роботи, фонд часу, Python, OpenWeatherMap.

ЗМІСТ

ВСТУП

У галузі рослинництва одним із головних завдань є раціональне планування часу, необхідного для виконання основних технологічних процесів [7]. З огляду на сезонність робіт, вплив погодних умов, а також обмеженість трудових і технічних ресурсів, особливого значення набуває точне визначення фонду часу. Це дозволяє не лише уникнути простоїв і втрати врожаю, а й підвищити загальну ефективність агровиробництва.

Незважаючи на наявність різноманітних аграрних інформаційних систем, на практиці багато фермерських господарств досі користуються ручними методами планування або ж універсальними табличними редакторами, які не враховують специфіки польових робіт [16]. Водночас, із розвитком комп'ютерних технологій відкриваються нові можливості для створення простих і зручних програмних рішень, що можуть стати надійним інструментом у щоденній роботі аграріїв.

Виконана кваліфікаційна робота присвячена розробці desktop-додатку, основне завдання якого – полегшити процес розрахунку фонду часу виконання робіт у рослинництві. Цей додаток має враховувати низку параметрів – площу оброблюваних земель, види культур, обсяги необхідних робіт, продуктивність техніки та інші дані, які користувач може ввести самостійно. Такий підхід дозволяє адаптувати додаток під різні умови господарювання, як для дрібних фермерів, так і для великих агропідприємств.

Об'єктом дослідження виступає процес організації агротехнічних робіт у полі, а предметом – засоби автоматизації обчислень, пов'язаних із плануванням часу. У процесі роботи були проаналізовані існуючі підходи до обліку трудових ресурсів у сільському господарстві, обрано відповідне середовище розробки, а також реалізовано прототип додатку з основним функціоналом.

Вибір теми обумовлений як практичною потребою сільськогосподарських підприємств у сучасних засобах планування, так і

недостатньою кількістю адаптованих програмних рішень саме для галузі рослинництва.

Таким чином, практична цінність розробки полягає в можливості її використання для щоденного планування в умовах реального агровиробництва, а сам додаток є основою для подальших вдосконалень або впровадження в навчальний процес аграрних закладів освіти.

РОЗДІЛ 1.
АНАЛІЗ СТАНУ ПИТАННЯ ЩОДО ВИЗНАЧЕННЯ ФОНДУ ЧАСУ ВИКОНАННЯ РОБІТ У РОСЛИННИЦТВІ

1.1. Сучасний стан інформатизації рослинництва

Розвиток інформаційних технологій у XXI столітті значною мірою змінив підходи до ведення сільськогосподарського виробництва. Зокрема, в галузі рослинництва відбувається активне впровадження цифрових рішень, спрямованих на автоматизацію агротехнологічних процесів, моніторинг полів, управління ресурсами та планування робіт. Інформатизація аграрного сектору дозволяє суттєво підвищити ефективність виробництва, зменшити витрати й поліпшити якість управлінських рішень.

Застосування комп’ютерних систем у рослинництві охоплює різні рівні – від персональних desktop-додатків до повноцінних геоінформаційних систем (ГІС), що інтегруються з супутниковими даними, сенсорними мережами та технологіями точного землеробства. Важливою перевагою таких рішень є можливість вести облік робіт, контролювати витрати пального, добрив, насіння, а також визначати трудовий фонд, необхідний для виконання сільськогосподарських операцій у визначені терміни.

На рис. 1.1 представлено загальну структуру інформаційних технологій, що використовуються у сучасному рослинництві. Вона включає модулі збору даних (сенсори, дрони, супутники), засоби зберігання (бази даних, хмари), аналітичні системи (моделі, обчислення), а також користувацькі інтерфейси для прийняття рішень.

Особливо динамічно розвивається сегмент desktop- і web-додатків, які забезпечують зручний доступ до функцій агропланування. У табл. 1.1 подано порівняння основних функцій сучасних аграрних програмних продуктів.

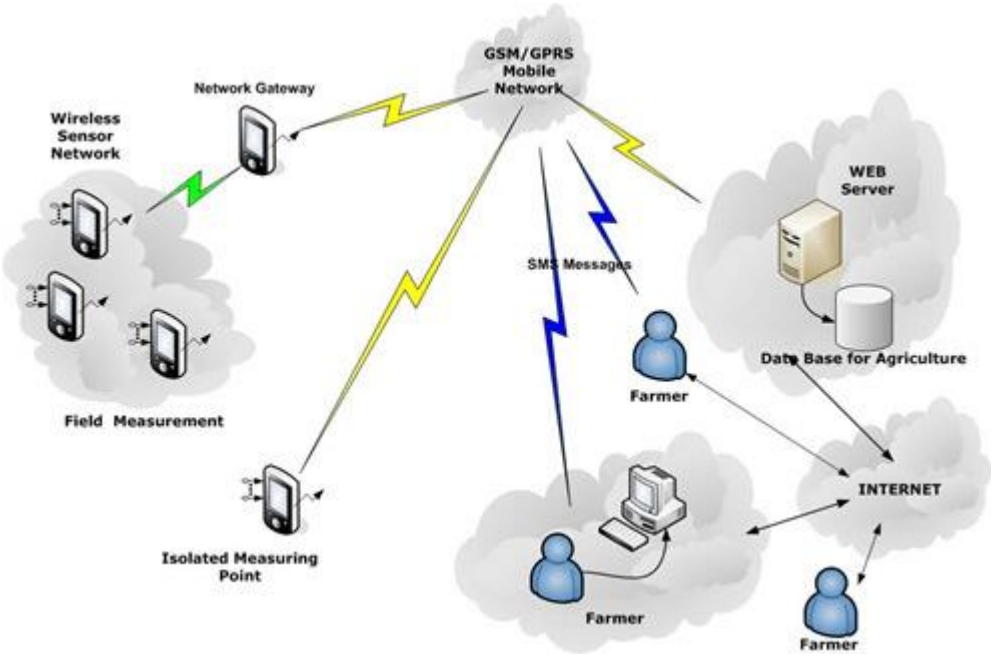

Рисунок 1.1 – Структура інформаційної системи підтримки рішень у рослинництві [30]

Таблиця 1.1 – Порівняння функціоналу сучасних IT-рішень для рослинництва

Назва ПЗ	Тип	Облік робіт	ГІС-модуль	Розрахунок фонду часу	Вартість
AgroOffice	Desktop	+	-	частково	Умовно безкоштовне
Cropio	Web	+	+	немає	Комерційне
FieldBee	Mobile	+	+	немає	Платне
Саморобні Excel-системи	Таблиці	частково	-	вручну	Безкоштовне

Як видно з даних таблиці 1.1, більшість популярних рішень або не містять модулів для автоматизованого розрахунку фонду часу виконання робіт, або реалізують їх частково, через обмеження щодо локальної специфіки агропроцесів. Це створює передумови для розробки індивідуальних програмних засобів, орієнтованих на конкретні умови господарювання та з урахуванням національних агротехнологічних нормативів.

Сучасні агропромислові холдинги поступово переходять до ефективної побудови операційного менеджменту за допомогою інноваційних продуктів автоматизації сільського господарства. Таким продуктом є система дистанційного моніторингу та оперативного обліку Agro Office [1].

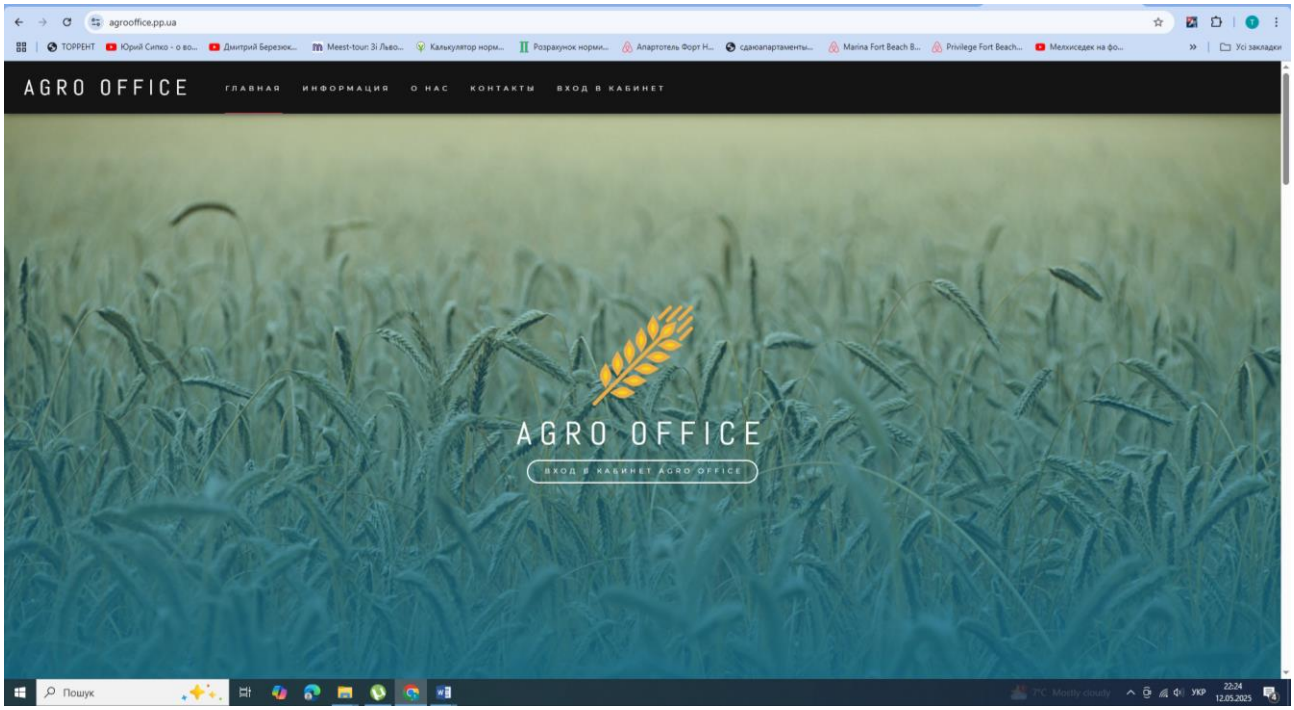

Рисунок 1.2 – Система дистанційного моніторингу та оперативного обліку Agro Office [1]

Заслуговує на увагу платформа Cropwise, що забезпечує підвищення ефективності агровиробництва. Це єдина інтегрована платформа сервісів і рішень, які взаємодіють між собою через обмін даними. Вона надає фермерам і агроконсультантам інструменти для оптимізації як повсякденних операцій, так і загальної діяльності господарства (рис. 1.3).

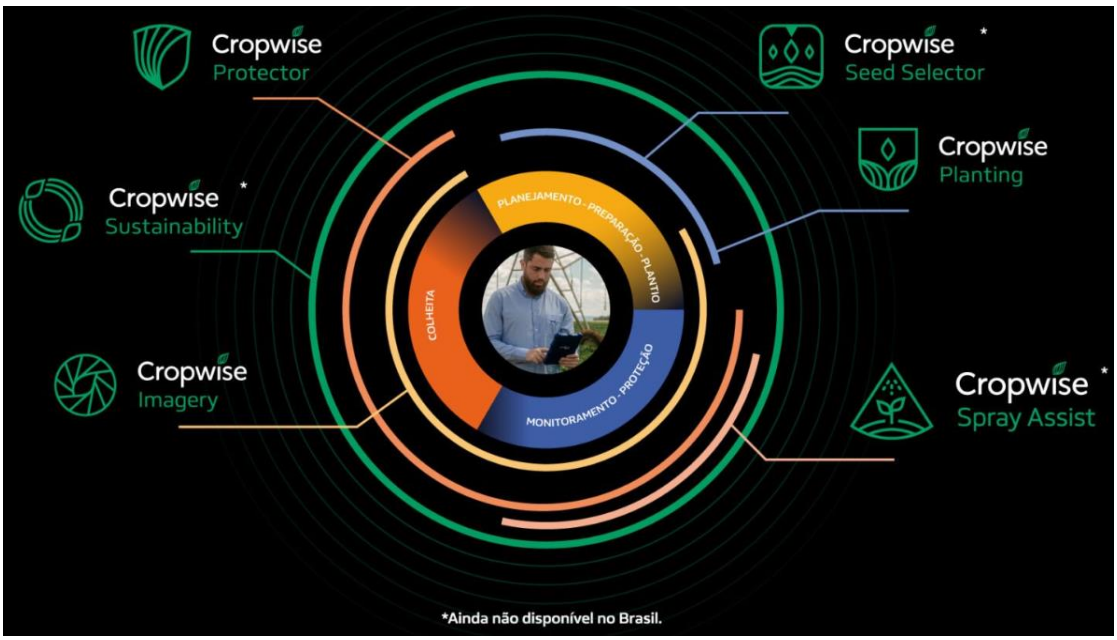

Рисунок 1.3 – Платформа Cropwise [1]

Cropwise – це цифрова платформа від компанії Syngenta, яка об’єднує низку інструментів для підтримки агровиробників на всіх етапах сільськогосподарського циклу – від планування посівів до збору врожаю. Вона забезпечує інтеграцію даних, аналітику та автоматизацію, спрямовані на підвищення продуктивності, ефективності та стійкості агробізнесу.

Основні модулі платформи Cropwise [1]:

Cropwise Operations – система моніторингу полів у реальному часі, яка надає інформацію про стан рослин, рівень вегетації, прогноз погоди, історію полів та інші агрономічні показники. Цей модуль дозволяє оперативно виявляти проблемні ділянки та приймати обґрунтовані рішення для оптимізації виробничих процесів.

Cropwise Protector – інструмент для моніторингу шкідників, хвороб і бур’янів, а також для ведення польового журналу. Він підтримує мобільні та веб-версії, дозволяє додавати нотатки, фотографії та голосові коментарі, а також інтегрується з супутниковими знімками для точного визначення зон обстеження.

Cropwise Imagery – модуль для аналізу супутникових зображень, який допомагає відстежувати стан посівів, виявляти аномалії та приймати рішення щодо зонального управління.

Cropwise Seed Selector – інструмент для підбору оптимальних сортів насіння відповідно до специфіки полів та агрокліматичних умов, що сприяє максимізації врожайності.

Cropwise Financials – модуль для фінансового планування та аналізу, який дозволяє відстежувати витрати, доходи та рентабельність господарства.

Cropwise Sustainability – інструмент для оцінки впливу господарської діяльності на навколишнє середовище, включаючи викиди парникових газів, використання водних ресурсів та біорізноманіття.

Отже, попри активне впровадження IT у рослинництво, наявна потреба в адаптованих локальних рішеннях, зокрема desktop-додатках, які можуть обробляти польові дані, враховувати специфіку регіону та оперативно надавати результат користувачеві без потреби у підключенні до мережі Інтернет. Це визначає актуальність розробки інструментів для визначення фонду часу виконання робіт у вигляді персонального програмного засобу.

1.2. Методологічні основи розрахунку фонду часу агротехнічних робіт

Фонд часу виконання робіт у рослинництві є одним із ключових показників, що визначає ефективність виробничого процесу. Його обчислення ґрунтується на поєднанні агротехнічних нормативів, характеристик техніки, організації праці та особливостей культур, які вирощуються. Методологія визначення фонду часу базується на узагальнених формулах, які враховують площу оброблюваних полів, продуктивність механізмів і коефіцієнти, що відображають реальні умови роботи.

Для розрахунку фонду часу часто використовуються базові формули типу:

$$T = \frac{S}{P \cdot K}, \qquad (1.1)$$

де T – потрібний фонд часу, год.; S – площа оброблюваної ділянки, га; P – продуктивність техніки, га/год; K – коефіцієнт змінності або використання техніки.

Ця формула (1.1) є базовою для розрахунків при плануванні навантаження на машини або працівників. Автоматизація введення вхідних параметрів та обчислень у вигляді desktop-додатку дозволяє зробити розрахунки оперативнішими, а дані більш структурованими.

Під час планування робіт важливо враховувати змінність (кількість змін за день), погодні умови, кількість задіяної техніки та кваліфікацію працівників. У табл. 1.2 наведено приклад розрахунку фонду часу для різних видів польових робіт.

Таблиця 1.2 – Приклад розрахунку фонду часу для окремих агротехнічних операцій

Назва операції	Площа, га	Продуктивність, га/год	Коефіцієнт K	Розрахований фонд часу, год
Оранка	25	1.8	0.85	16.34
Дискування	25	2.5	0.90	11.11
Сівба зернових	25	1.5	0.80	20.83
Внесення добрив	25	3.0	0.95	8.77

У реальних умовах також варто враховувати багатофакторні впливи. Наприклад, затримки через дощ або поломку техніки можуть скоригувати фактичний час виконання робіт. Для таких випадків у плануванні застосовують поправкові коефіцієнти ризику R, що вводяться в основну формулу:

$$T' = T \cdot R, \qquad (1.2)$$

де T' – уточнений фонд часу; $R > 1$ – поправковий коефіцієнт (зазвичай 1.05–1.2).

Таким чином, методологія розрахунку фонду часу є багатоаспектною і вимагає врахування як нормативно-технологічних, так і організаційних параметрів. Автоматизація цих обчислень у вигляді desktop-додатку дозволяє пришвидшити процес планування, зменшити ймовірність помилок та забезпечити адаптивність до змінних умов виробництва.

1.3. Огляд алгоритмів і моделей для визначення фонду часу виконання робіт

Розрахунок фонду часу виконання робіт у рослинництві є не лише прикладною задачею, а й важливою частиною моделювання виробничих процесів у рослинництві. У різних господарствах для планування робочого навантаження використовують набір алгоритмічних та емпіричних моделей, які враховують площу земель, типи культур, технологічні карти, характеристики техніки та людських ресурсів. Залежно від поставленої мети, можна використовувати прості аналітичні формули, табличні нормативи або імітаційні алгоритми, що враховують змінність умов.

Сучасні алгоритми можна класифікувати за кількома ознаками:

- за принципом дії (аналітичні, табличні, імітаційні);
- за рівнем автоматизації (ручні, напівавтоматизовані, повністю автоматизовані);
- за гнучкістю до змін вхідних параметрів.

Найбільш традиційним підходом є застосування аналітичних формул прямого розрахунку часу, таких як наведені у п. 1.2. Їхньою головною перевагою є простота реалізації та висока швидкість обчислення. Водночас, ці

моделі мають обмежену здатність адаптуватися до динамічних зовнішніх умов, таких як затримки, погодні зміни чи технічні простої.

На практиці значного поширення набули також нормативно-довідкові методики, які ґрунтуються на використанні галузевих таблиць продуктивності техніки та норм витрат часу на одиницю площі. Ці таблиці, як правило, розробляються на основі тривалих спостережень, експертних оцінок та експериментальних даних. У них подається перелік робіт (оранка, культивація, сівба тощо), їх технологічні параметри (глибина, швидкість, ширина захвату), а також середній час на виконання. В обчислювальному контексті такі таблиці перетворюються на lookup-масиви, доступ до яких реалізується через інтерфейс програми. Наприклад, у desktop-додатку користувач може вибрати тип операції зі списку, а система автоматично підставить нормативний час.

Більш складними, але й більш гнучкими, є імітаційні моделі. Їх основою є побудова послідовності дій, які імітують процес виконання робіт у часовому вимірі. У таких моделях враховуються не лише продуктивність техніки та площа поля, але й логістика, затримки через зовнішні чинники, обслуговування машин, людський фактор. Наприклад, у випадку обробітку поля кількома агрегатами одночасно, може виникати конфлікт доступу до ділянок або черговість операцій, що впливають на загальну тривалість. Такі моделі реалізуються на основі подійно-орієнтованих симуляторів або агентно-орієнтованих середовищ, які дозволяють створювати повноцінні сценарії з урахуванням варіативності.

На рисунку 1.4 наведено схема типової реалізації алгоритму визначення фонду часу у вигляді програмного модуля. У ній чітко простежується логіка етапів: введення вихідних даних, вибір типу обробки, підрахунок базового часу, застосування поправкових коефіцієнтів та формування результату. Така структура є основою для побудови програмного ядра desktop-додатку.

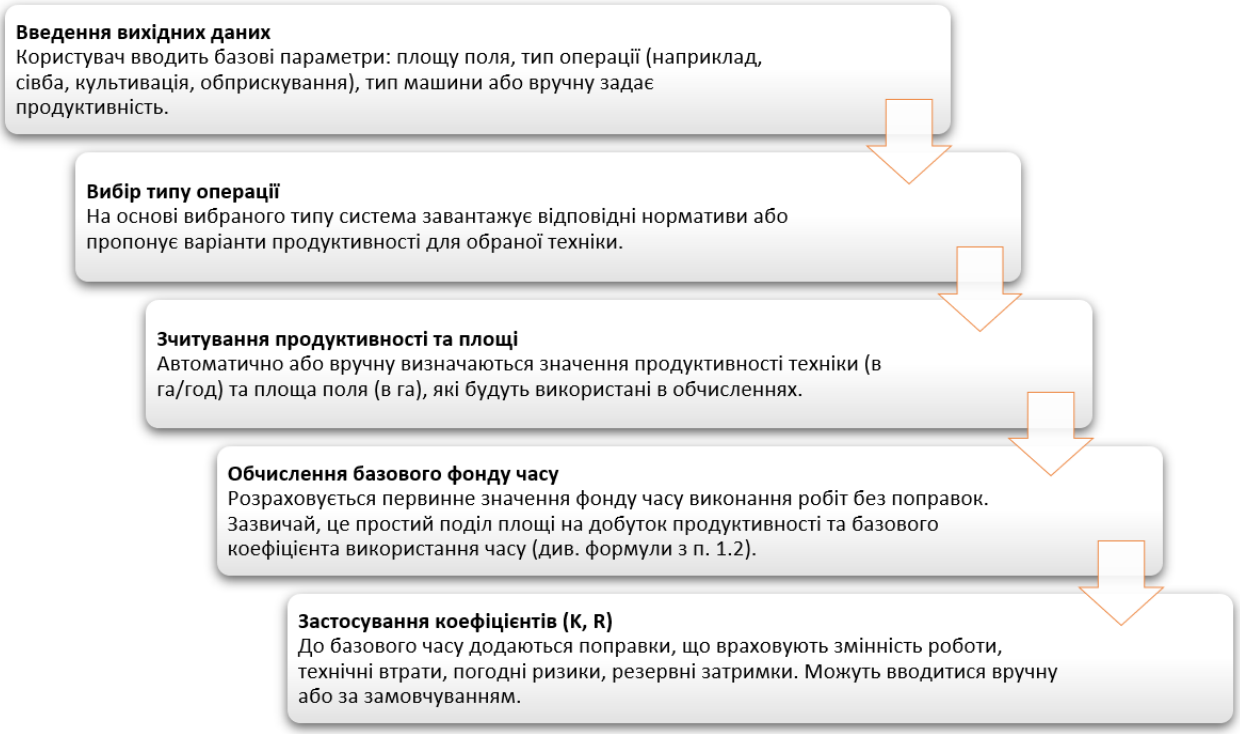

Рисунок 1.4 – Схема визначення фонду часу у програмному модулі

Ще одним підходом, що активно розвивається в останнє десятиліття, є використання методів штучного інтелекту, зокрема регресійного моделювання, дерев рішень та нейромереж. У цих підходах система навчається на історичних даних про фактичні тривалості виконання робіт і потім здатна робити прогнози для нових полів або культур. Наприклад, модель на основі множинної регресії може враховувати такі незалежні змінні, як тип ґрунту, кількість змін, технічний стан машин, погодні умови, а також специфіку культури. Модель формує функцію у вигляді:

$$T = f\left(x_1, x_2, ..., x_n\right), \qquad (1.3)$$

де $x_1, x_2, ..., x_n$ – вхідні характеристики; f – функція, отримана на основі навчання.

Такий підхід дозволяє гнучко адаптувати розрахунки до змін середовища, однак вимагає наявності репрезентативної бази даних і значних обчислювальних ресурсів.

Слід також згадати про експертні системи, в яких знання про тривалість робіт кодуються у вигляді правил типу "якщо – то". Наприклад: якщо вологість

ґрунту > 25%, то збільшити час виконання культивації на 15%. Такі правила можуть формуватися агрономами або експертами і вноситися в систему для подальшого використання. Подібні алгоритми добре себе зарекомендували в умовах відсутності кількісних даних, однак поступаються в точності регресійним моделям при достатньому обсязі спостережень.

У таблиці 1.3 наведено коротке порівняння основних груп моделей за критеріями адаптивності, точності, обчислювальної складності та можливості реалізації в desktop-додатку.

Таблиця 1.3 – Порівняння моделей для розрахунку фонду часу

Модель	Адаптивність	Точність	Складність реалізації	Придатність для desktop
Аналітична	Низька	Середня	Низька	Висока
Таблична	Середня	Середня	Низька	Висока
Імітаційна	Висока	Висока	Висока	Середня
Регресійна / ML	Висока	Висока	Висока	Залежить від реалізації
Експертна система	Середня	Середня	Середня	Висока

Отже, сучасні алгоритми для визначення фонду часу в агротехнологіях охоплюють широкий спектр підходів – від класичних формул до складних моделей на основі машинного навчання. Вибір конкретної моделі залежить від цілей застосування, наявності вихідних даних, рівня автоматизації та технічних можливостей користувача. Для desktop-додатків, що орієнтовані на невеликі фермерські господарства або агрономічні служби, оптимальним є використання аналітико-нормативного гібриду, який дозволяє швидко й наочно розраховувати необхідний час за базовими параметрами та адаптувати його з використанням поправкових коефіцієнтів або правил.

РОЗДІЛ 2.
ВИБІР ІНСТРУМЕНТАРІЮ ДЛЯ СТВОРЕННЯ DESKTOP ДОДАТКУ ВИЗНАЧЕННЯ ФОНДУ ЧАСУ ВИКОНАННЯ РОБІТ У РОСЛИННИЦТВІ

2.1. Мета та завдання роботи

Метою цієї кваліфікаційної роботи є розробка програмного засобу у вигляді desktop-додатку, який забезпечує можливість точного та оперативного визначення фонду часу виконання агротехнічних робіт у рослинництві. Створення подібного інструменту покликане спростити процес планування трудових та технічних ресурсів, зробити його доступним для широкого кола користувачів – від агрономів-практиків до менеджерів агропідприємств. Реалізація функціонального, інтуїтивно зрозумілого та адаптивного додатку дозволить забезпечити автоматизацію типових розрахунків, які сьогодні часто виконуються вручну або із використанням неспеціалізованих засобів.

У контексті поставленої мети сформульовано низку завдань, які потребують вирішення у межах даного дослідження. Насамперед необхідно проаналізувати існуючі підходи та програмні продукти, що стосуються планування фонду часу у сільському господарстві, зокрема в рослинництві. Важливо визначити методологічну основу майбутньої реалізації, включаючи математичні моделі, алгоритми розрахунку, систему вхідних параметрів і поправкових коефіцієнтів. Паралельно з цим необхідно обґрунтувати вибір мови програмування, бібліотек, середовища розробки, а також засобів побудови користувацького інтерфейсу.

Наступним кроком є розробка структури додатку та його функціональних модулів, які повинні охоплювати введення даних, обчислення фонду часу, застосування коефіцієнтів, формування результатів і збереження звітів. У процесі реалізації важливо також врахувати можливість масштабування додатку, наприклад, через подальше доповнення модулями експертної оцінки

або геоінформаційної інтеграції. Крім цього, потрібно здійснити тестування програми на прикладах реальних або умовно змодельованих сценаріїв, що дозволить виявити помилки, оптимізувати обчислення і скоригувати логіку роботи окремих блоків.

Завершальним завданням виступає оформлення результатів роботи у вигляді програмного продукту з відповідною документацією, що містить опис структури, принципів функціонування та приклади застосування. Таким чином, реалізація поставленої мети передбачає не лише створення технічного інструменту, але й забезпечення його практичної цінності для галузі рослинництва.

2.2. Вибір технологій та засобів для розробки Desktop додатку визначення фонду часу виконання робіт у рослинництві

Для реалізації комп’ютерної програми, яка допомагає аграріям визначати фонд часу польових робіт, потрібно обрати такі технології, що дадуть змогу зручно вводити дані, проводити розрахунки та показувати результати у зрозумілому вигляді. Оскільки передбачається використання програми на локальному комп’ютері, перевага була надана розробці саме desktop-додатку.

Після порівняння різних варіантів мов програмування, найбільш доцільним виявився вибір Python. Це мова, яка поєднує простоту синтаксису з великими можливостями для математичних обчислень, графічної візуалізації й побудови інтерфейсів. Вона активно використовується як у навчанні, так і в промисловій розробці програмного забезпечення, має велику кількість бібліотек, а також підтримується на різних операційних системах.

Для створення вікон, кнопок, полів введення та інших елементів інтерфейсу обрано бібліотеку Tkinter, яка йде в комплекті з Python і не потребує додаткових інсталяцій (рис. 2.1). Вона дозволяє побудувати просте й логічне вікно програми, де користувач вводить площу, обирає тип роботи, натискає

кнопку – і відразу бачить результат. При потребі інтерфейс можна вдосконалити, наприклад, стилізувати через додаткові пакети або перевести додаток у формат .exe.

Рисунок 2.1 – Логотип вибраної бібліотеки Tkinter на мові програмування Python

Оскільки у розрахунках потрібно працювати з числовими масивами, таблицями й формулами, для цього залучається NumPy – потужна бібліотека для виконання математичних операцій. Саме з її допомогою реалізуються обчислення, про які йшлося в попередньому розділі, включаючи врахування коефіцієнтів і побудову залежностей.

Рисунок 2.2 – Логотип вибраних бібліотек NumPy та Matplotlib на мові програмування Python

Щоб зробити результати зрозумілими та наочними, додаток підтримує побудову графіків – наприклад, як змінюється час залежно від площі або продуктивності. Для цього використовується бібліотека Matplotlib, яка дозволяє створювати діаграми прямо у вікні програми або зберігати їх у файл.

Для загального уявлення нижче наведено таблицю, яка показує, чому саме ці засоби були обрані, які представлені на рис. 2.1.

Таблиця 2.1 – Результати вибору технологій для розробки Desktop додатку визначення фонду часу виконання робіт у рослинництві

Компонент	Інструмент	Переваги
Мова програмування	Python	Зручна, популярна, проста у вивченні
Інтерфейс	Tkinter	Вбудований, легкий, без зайвих залежностей
Обчислення	NumPy	Швидкі операції з масивами
Побудова графіків	Matplotlib	Гнучкі можливості для візуалізації

Також варто згадати, що обрані засоби дозволяють розширити додаток у майбутньому. Зокрема, додати збереження у формат Excel або PDF, формувати звіти, підключити базу даних для зберігання полів, робіт та результатів.

Загальна структура додатку передбачає поділ на кілька логічних блоків: введення даних, обробка, обчислення, вивід результату та побудова графіка. На схемі нижче умовно зображено, як ці блоки взаємодіють між собою (рис. 2.3).

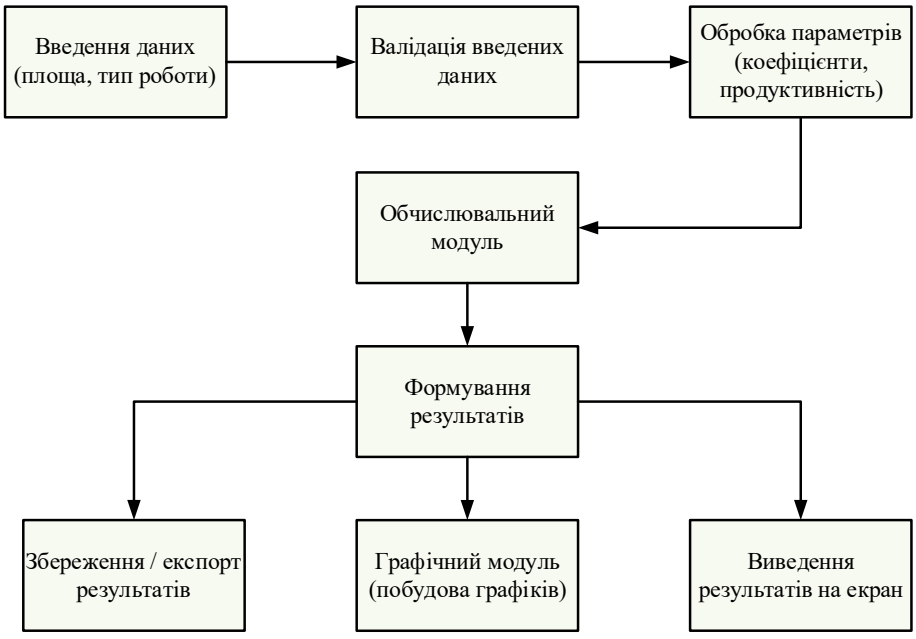

Рисунок 2.3 – Структура взаємодії модулів програми

Таким чином, вибрані інструменти дозволяють реалізувати всю потрібну функціональність: від простого розрахунку до формування повного звіту. При цьому програма залишиться доступною для користувача без спеціальних технічних знань, а її робота не залежатиме від наявності інтернету чи складних систем.

2.3. Математичний опис використовуваних моделей Desktop додатку для визначення фонду часу виконання робіт у рослинництві

Базові моделі, що використовуються для визначення фонду часу, розширюються у desktop-додатку до форми, яка дозволяє врахувати не лише продуктивність техніки та площу, а й додаткові змінні: типи технологічних операцій, коефіцієнти змінності, перерви, час на підготовку та обслуговування, логістичні втрати, ризики і навіть щільність техніки на одиницю площі. Це дозволяє адаптувати розрахунки до реальних умов господарств, де рідко все відбувається за еталонними показниками.

Загальна формула для розрахунку повного фонду часу $T_{повн}$ з урахуванням додаткових коефіцієнтів і фаз виглядає так:

$$T_{повн} = \left(\frac{S}{P \cdot K \cdot Z} \right) \cdot (1 + \alpha + \beta + \gamma), \qquad (2.1)$$

де S – площа ділянки, га; P – номінальна продуктивність техніки, га/год; K – коефіцієнт ефективності використання змінного часу; K – кількість змін на добу становить $K = 1...3$ од.; α – частка часу, втрачена через технічні простої (поломки, обслуговування); β – поправка на втрати через логістику (очікування техніки, віддаленість ділянки); γ – поправка на погодні умови або непередбачувані затримки.

Значення α, β, γ можуть вводитися вручну або братись із довідників, ґрунтуючись на досвіді господарства. У середньому кожен із цих коефіцієнтів варіюється в межах 0.05…0.25 (тобто 5…25% втрат на кожному етапі).

Для багатофазних робіт (наприклад, сівба з одночасним внесенням добрив) використовується сумарний розрахунок часу для кожної операції окремо:

$$T_{заг} = \sum_{i=1}^{n} \left(\frac{S_i}{P_i \cdot K_i \cdot Z_i} \right) \cdot \left(1 + \alpha_i + \beta_i + \gamma_i\right), \qquad (2.2)$$

де i – номер технологічної операції (оранка, сівба, культивація тощо); S_i, P_i, K_i, Z_i – параметри, специфічні для кожної фази; $\alpha_i, \beta_i, \gamma_i$ – поправки на втрати під час виконання конкретної операції.

У разі планування робіт протягом кількох днів, модель може бути адаптована до динамічного виду з урахуванням календарного графіка:

$$T_{план}(d) = \sum_{d=1}^{D} \left(\frac{S_d}{P_d \cdot K_d \cdot Z_d} \cdot (1 + \omega_d) \right), \qquad (2.3)$$

де d – день планування; ω_d – коефіцієнт втрат у конкретний день, визначений за погодними даними або прогнозами ризику; D – загальна кількість днів.

Окремий блок моделі дозволяє врахувати щільність техніки, якщо на ділянці працює кілька агрегатів одночасно. Тоді застосовується коригування продуктивності на співвідношення агрегатів до площі:

$$P_{еф} = P \cdot N \cdot \delta, \qquad (2.4)$$

де N – кількість однакових машин; $\delta \in \lceil 0;1 \rceil$ – коефіцієнт зниження ефективності при роботі в групі (враховує затори, маневрування тощо).

Підсумкова формула повного часу набуває вигляду:

$$T = \frac{S \cdot (1 + \alpha + \beta + \gamma)}{P \cdot K \cdot Z \cdot \delta \cdot N}. \qquad (2.5)$$

Ця модель дозволяє надзвичайно гнучко адаптувати додаток під конкретне господарство. Наприклад, у фермерському господарстві з однією машиною та одним оператором зазвичай приймається $Z = 1,\ N = 1$, а коефіцієнти втрат задаються вручну на основі власного досвіду.

У розробленому додатку користувач може задати більшість параметрів вручну, що дозволяє програмі бути не просто калькулятором, а моделлю підтримки прийняття рішень, яка працює в умовах невизначеності. Такий підхід відповідає сучасним принципам адаптивного планування в агросфері.

РОЗДІЛ 3.
СТВОРЕННЯ ПРОГРАМНИХ МОДУЛІВ DESKTOP ДОДАТКУ ДЛЯ ВИЗНАЧЕННЯ ФОНДУ ЧАСУ ВИКОНАННЯ РОБІТ У РОСЛИННИЦТВІ

3.1. Опис основних об’єктів desktop-додатку

Побудова ефективного desktop-додатку для визначення фонду часу виконання агротехнічних робіт у рослинництві вимагає чіткої внутрішньої організації – як у вигляді логіки взаємодії модулів, так і на рівні програмних об’єктів. Кожен із модулів, що реалізує ключову функцію програми – введення даних, обробку параметрів, обчислення, формування та виведення результатів – має відповідати за роботу конкретного класу або групи функцій. Така структурна декомпозиція забезпечує зручність у підтримці коду, гнучкість розширення функціоналу, а також зрозумілий поділ відповідальностей між частинами додатку.

Створення додатку для визначення фонду часу виконання робіт у рослинництві передбачає організацію даних у вигляді об’єктів та класів, що дозволяє структурувати логіку, спростити обчислення та забезпечити гнучкість у подальшому доопрацюванні додатку. Застосування об’єктно-орієнтованого підходу дає змогу чітко відокремити вхідні дані, алгоритмічну частину та результативні структури, а також впорядкувати процес взаємодії між модулями програми.

Центральною одиницею є клас WorkSession, який виступає як об’єкт одного сеансу розрахунку. Саме через нього проходять усі ключові параметри, необхідні для визначення фонду часу, зокрема площа поля, тип агротехнічної операції, продуктивність, а також коефіцієнти ефективності виконання та ризиків. Цей клас містить не лише змінні, але й методи для валідації введених даних, запуску розрахунків, форматування результату та передачі його в інші модулі.

Допоміжні класи – це окремі об’єкти, які відповідають за специфічні задачі. Наприклад, клас Validator перевіряє правильність введених числових і текстових даних, забезпечуючи, щоб розрахунок виконувався лише у випадку наявності коректних значень. Клас ParameterProcessor може бути відповідальним за уточнення або нормалізацію коефіцієнтів – тобто виконання операцій над вихідними параметрами перед передачею їх у математичний модуль.

Розрахунок здійснюється у межах класу CalculationModule, який реалізує метод, що використовує формули, описані у розділі 1. Важливо, що цей клас ізольовано від інтерфейсу – він приймає готові значення параметрів, виконує обчислення, а результат передає назад у вигляді структурованої змінної. Це дозволяє в разі потреби замінити або вдосконалити алгоритм, не змінюючи логіку програми загалом.

Результати обчислень передаються у клас ResultBuilder, який формує текстовий і табличний звіт. Окремий клас GraphModule відповідає за візуалізацію: побудову графіків залежностей, зокрема впливу площі або продуктивності на фонд часу. Завдяки винесенню в окрему структуру цей модуль можна легко замінити або доповнити іншими типами діаграм.

Останній блок – OutputManager, який обробляє збереження результатів у файл (TXT, CSV, PDF) або їх вивід на екран. Він взаємодіє з усіма іншими модулями, отримуючи готові дані для відображення або експорту.

У таблиці 3.1 подано аналіз основних класів і їх функціонального призначення у межах структури додатку.

Класи взаємодіють між собою згідно з логікою, представленою у структурі програми. Зв’язки реалізовані за допомогою передачі об’єктів як параметрів до функцій або методів.

Таким чином, запропонована модель забезпечує чітке логічне розділення функцій програми, що позитивно впливає на її підтримку, масштабованість та зручність використання.

Таблиця 3.1 – Основні класи desktop-додатку та їх призначення

Клас	Основне призначення
WorkSession	Централізований об’єкт сесії – зберігання параметрів, запуск розрахунку
Validator	Перевірка правильності вхідних даних
ParameterProcessor	Обробка і трансформація коефіцієнтів, переведення одиниць
CalculationModule	Реалізація математичних обчислень за формулою
ResultBuilder	Формування текстового або табличного звіту
GraphModule	Візуалізація результатів – побудова графіків
OutputManager	Виведення даних на екран, збереження у файл

Структура дозволяє ефективно виконувати всі етапи – від введення до збереження результатів – у межах єдиного об’єктно-орієнтованого середовища.

На рисунку 3.1 наведено UML-діаграму класів, яка ілюструє взаємозв’язки між основними об’єктами desktop-додатку.

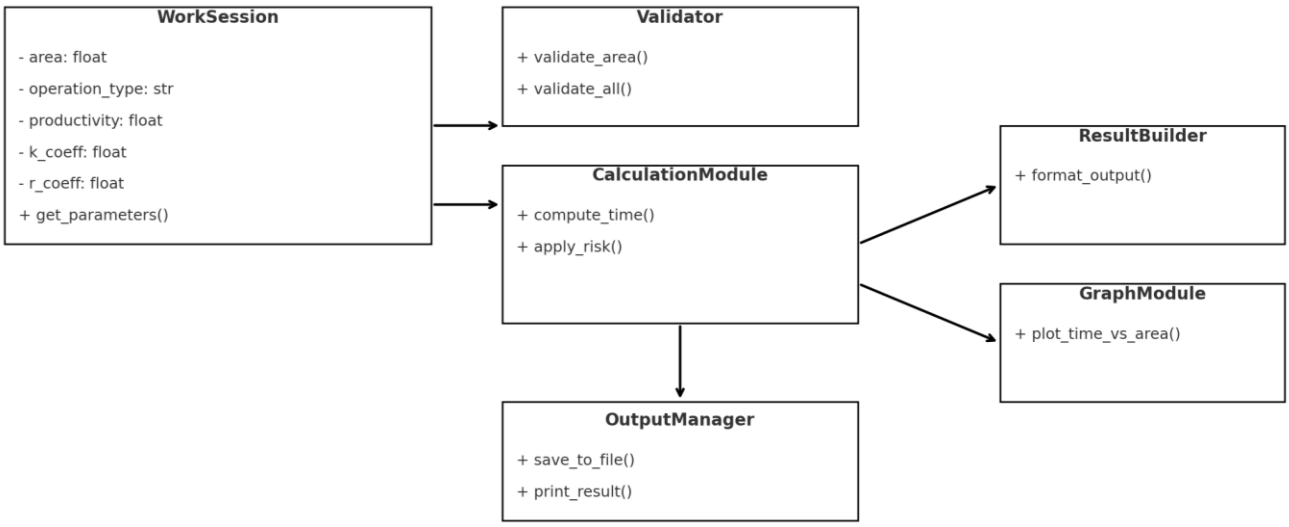

Рисунок 3.1 – UML-діаграма основних класів desktop-додатку

Таким чином, структура об’єктів desktop-додатку орієнтована на ефективне обслуговування користувацького запиту, гнучкість до змін

параметрів, повторне використання коду й простоту підтримки. Обраний підхід поєднує класичну об’єктно-орієнтовану архітектуру з практичними структурами даних, що забезпечують продуктивну й масштабовану реалізацію функціоналу desktop-додатку.

3.2. Структура класу WorkSession

Основним класом у системі є WorkSession, який виступає контейнером для всієї інформації, необхідної для розрахунку. До цього класу входять властивості, що описують площу обробки, тип агротехнічної операції, продуктивність, коефіцієнти поправок, а також методи, які реалізують обчислення та форматування результатів.

У таблиці 3.2 представлено загальну структуру класу WorkSession та його основні атрибути і функції.

Таблиця 3.2 – Структура основного класу WorkSession

Елемент	Тип	Призначення
area	float	Площа поля, га
operation_type	str	Назва агротехнічної операції (оранка, сівба, обприскування тощо)
productivity	float	Продуктивність техніки, га/год
k_coeff	float	Коефіцієнт використання змінного часу
r_coeff	float	Коефіцієнт ризику
calculate_time()	метод	Обчислює фонд часу на основі вхідних даних
export_result()	метод	Формує звіт або текстовий висновок

Цей клас використовується як об’єкт-сесія, який ініціалізується при кожному новому розрахунку. Користувач вводить вхідні дані через інтерфейс, і програма створює новий екземпляр об’єкта, який виконує всі обчислення та

генерує результат. Важливою перевагою такого підходу є можливість одночасно створювати декілька сесій розрахунку без взаємного впливу.

```
operations = {
    "оранка": 1.8,
    "культивація": 2.2,
    "сівба": 1.5,
    "обприскування": 3.0
}
```

Рисунок 3.2 – Фрагмент формування списку доступних агротехнічних операцій

Для зберігання списку доступних агротехнічних операцій використовується структура типу словник (dict), у якому ключем виступає назва операції, а значенням є – середня продуктивність або типовий коефіцієнт (рис. 3.2).

3.3. Структура класу Validator

Клас Validator відповідає за перевірку правильності введених даних, які надходять від користувача на початковому етапі функціонування програми. Його основна мета – забезпечити надійність і коректність вхідної інформації, попередити критичні помилки при виконанні обчислень, а також зменшити ймовірність збоїв під час роботи додатку. Робота цього класу є обов'язковою на етапі після введення, але до запуску основного обчислювального модуля.

Основними параметрами, які проходять перевірку, є площа поля (в гектарах), значення продуктивності техніки (в га/год), значення коефіцієнтів k та r, а також тип обраної агротехнічної операції. Клас здійснює перевірку на числовий формат, недопущення від'ємних або нульових значень, а також логічну узгодженість – наприклад, виключення ситуацій, у яких коефіцієнт ефективності перевищує 1.0 або продуктивність дорівнює нулю.

Методи класу реалізовані у вигляді окремих функцій, які можна використовувати автономно або в сукупності в загальному методі validate_all(). У разі виявлення некоректного значення система формує повідомлення про помилку, яке виводиться користувачеві через інтерфейс. Це попереджує перехід до обчислювального етапу до моменту повного виправлення даних.

Фрагмент коду реалізації цього класу у Python представлено на рис. 3.3.

```python
class Validator:
    def validate_area(self, area):
        return isinstance(area, float) and area > 0

    def validate_productivity(self, productivity):
        return isinstance(productivity, float) and productivity > 0

    def validate_coefficients(self, k, r):
        return 0 < k <= 1.0 and 0 < r <= 1.5

    def validate_operation(self, operation, operations_dict):
        return operation in operations_dict

    def validate_all(self, area, productivity, k, r, operation, operations_dict):
        return (
            self.validate_area(area) and
            self.validate_productivity(productivity) and
            self.validate_coefficients(k, r) and
            self.validate_operation(operation, operations_dict)
        )
```

Рисунок 3.3 – Фрагмент коду реалізації класу Validator

У таблиці 3.3 наведено характеристики основних функцій класу Validator та їх призначення.

Таблиця 3.3 – Основні функції класу Validator

Назва методу	Призначення
validate_area()	Перевіряє, чи площа є додатним числом
validate_productivity()	Перевіряє, чи продуктивність задана правильно
validate_coefficients()	Визначає допустимість значень коефіцієнтів
validate_operation()	Перевіряє, чи вибраний тип операції присутній у базі
validate_all()	Підсумкова перевірка всіх параметрів перед передачею до обчислень

Клас Validator є незалежним і може використовуватись у будь-якому програмному середовищі, яке потребує перевірки вхідних аграрних або технічних параметрів. Його ізольованість дозволяє з легкістю підключати модуль до інших частин системи або використовувати в майбутніх розширеннях – зокрема, при введенні імпорту даних з файлів, де перевірка також необхідна.

Таким чином, використання Validator у структурі desktop-додатку дозволяє реалізувати принцип «захисту від помилок на вході», що є невід'ємною частиною надійного програмного забезпечення для агропланування.

3.4. Структура класу CalculationModule

Клас CalculationModule є ядром логіки обчислень у desktop-додатку для визначення фонду часу виконання робіт у рослинництві. Його основне призначення полягає в реалізації алгоритмів розрахунку на основі параметрів, які були введені користувачем або оброблені попередніми модулями. Саме цей модуль виконує ключову математичну операцію – визначення часу, необхідного для виконання польових робіт на заданій площі з урахуванням продуктивності, коефіцієнтів ефективності використання техніки та факторів ризику.

Розрахунок використовується у вигляді виклику методу всередині класу. Вхідними даними для методу є площа поля, продуктивність техніки, коефіцієнт використання змінного часу, а також коефіцієнт ризику. Результатом роботи методу є конкретне числове значення часу в годинах, яке потім передається у наступні модулі – для побудови графіків, формування звітів або виводу на екран.

Клас взаємодіє з об'єктом WorkSession, отримуючи з нього необхідні параметри. Така реалізація дозволяє зберегти цілісність даних протягом усього

циклу взаємодії програми. Приклад структури класу в Python наведено на рис. 3.4.

```python
class CalculationModule:
    def __init__(self, session):
        self.session = session

    def compute_time(self):
        base_time = self.session.area / (self.session.productivity * self.session.k_coeff)
        total_time = base_time * self.session.r_coeff
        return round(total_time, 2)
```

Рисунок 3.4 – Фрагмент коду реалізації класу CalculationModule

Важливо, що метод compute_time() не лише виконує обчислення, але й округлює результат до зручної форми, що полегшує подальше використання значення у текстових звітах чи таблицях.

У таблиці 3.4 подано основні методи класу CalculationModule та їх функціональні призначення.

Таблиця 3.4 – Основні методи класу CalculationModule

Назва методу	Призначення
compute_time()	Основний метод обчислення фонду часу
round_result()	Округлення результату до певного знаку після коми
get_raw_data()	Повернення проміжних розрахунків для детального аналізу

Клас CalculationModule не залежить від інтерфейсу або структури збереження даних, що дозволяє масштабувати його в майбутньому. Наприклад, у разі потреби можна реалізувати підтримку складніших формул, врахування сезонності, погодних коефіцієнтів або поетапного виконання робіт з розбиттям на змінні.

Таким чином, CalculationModule є ключовим компонентом функціональної частини desktop-додатку. Його ефективна реалізація забезпечує точність результатів, швидкість обчислень та можливість інтеграції з іншими модулями програми – від графіки до збереження звітів.

3.5. Структура класу GraphModule

Клас GraphModule реалізує функції графічного представлення результатів розрахунку фонду часу, що дозволяє користувачеві не лише бачити числове значення, але й візуально оцінити, як змінюється час виконання робіт залежно від площі, продуктивності або інших параметрів. Графіки є зручним інструментом для прийняття управлінських рішень, зокрема під час вибору оптимального сценарію навантаження на техніку чи планування обсягів робіт у господарстві.

У структурі програми клас GraphModule приймає об’єкт WorkSession або результат із CalculationModule, зчитує основні параметри і формує масиви даних для побудови діаграм. Стандартним є створення лінійного графіка залежності фонду часу від площі при фіксованій продуктивності, або навпаки – при змінній продуктивності для заданої площі. Візуалізація виконується за допомогою бібліотеки Matplotlib, яка дозволяє будувати графіки із сіткою, підписами осей, легендами та збереженням у форматах PNG, SVG або PDF.

Основні методи класу реалізують такі функції, як побудова графіка, відображення на екрані користувача, збереження в файл, а також додавання аналітичних елементів (ліній допустимого ризику, вертикальних маркерів, підписів максимумів тощо).

Перевагою використання GraphModule є те, що візуалізація не впливає на обчислювальний модуль і може бути легко замінена або вдосконалена. Така ізоляція дозволяє, наприклад, адаптувати графіки для мобільної версії додатку або автоматично формувати графічні звіти для друку.

У випадку розширення програми графічний модуль може бути доповнений функціями порівняльного аналізу (наприклад, накладення кількох графіків), 3D-візуалізації або інтерактивного масштабування, що особливо корисно у великих господарствах із комплексним плануванням.

У таблиці 3.5 представлено ключові методи класу GraphModule.

Таблиця 3.5 – Основні методи класу GraphModule

Назва методу	Призначення
plot_time_vs_area()	Побудова графіка залежності фонду часу від площі
plot_time_vs_productivity()	Побудова графіка залежності фонду часу від продуктивності
add_optimal_marker()	Додавання вертикальної лінії або підпису «Оптимум»
save_plot(filename)	Збереження побудованого графіка у вказаний файл
show_plot()	Відображення графіка в інтерактивному режимі

```python
class GraphModule:
    def __init__(self, session):
        self.session = session

    def plot_time_vs_area(self):
        import matplotlib.pyplot as plt
        areas = list(range(5, 55, 5))
        times = [a / (self.session.productivity * self.session.k_coeff) * self.session.r_coeff for a in areas]
        plt.plot(areas, times, marker='o')
        plt.title("Залежність фонду часу від площі")
        plt.xlabel("Площа, га")
        plt.ylabel("Фонд часу, год")
        plt.grid(True)
        plt.show()
```

Рисунок 3.5 – Фрагмент коду реалізації класу CalculationModule

Таким чином, клас GraphModule є візуально-аналітичним компонентом системи, який підвищує інформативність результатів розрахунків і забезпечує зручність для прийняття аграрних рішень на основі даних.

РОЗДІЛ 4.
РОЗРОБКА ІНТЕРФЕЙСУ КОРИСТУВАЧА ТА ПРАКТИЧНЕ ВИКОРИСТАННЯ DESKTOP ДОДАТКУ ДЛЯ ВИЗНАЧЕННЯ ФОНДУ ЧАСУ ВИКОНАННЯ РОБІТ У РОСЛИННИЦТВІ

4.1. Розробка інтерфейсу користувача Desktop додатку для визначення фонду часу виконання робіт у рослинництві

Інтерфейс користувача є важливим елементом будь-якого програмного забезпечення, адже саме він забезпечує взаємодію між кінцевим користувачем і логікою додатку. У розробленому desktop-додатку для визначення фонду часу виконання робіт у рослинництві інтерфейс реалізовано з урахуванням принципів простоти, інтуїтивної зрозумілості та мінімалізму. Його основне завдання – забезпечити зручне введення даних, контроль за їх коректністю, запуск обчислень і виведення результатів у текстовій та графічній формах.

Для побудови інтерфейсу використано бібліотеку Tkinter, що є стандартною для мови програмування Python. Вона забезпечує достатній рівень функціональності для створення інтерактивних вікон, кнопок, полів введення, списків вибору та елементів управління подіями. Такий вибір обумовлений як простотою реалізації, так і сумісністю з іншими модулями програми, включно з графікою (Matplotlib) та обробкою даних (NumPy).

Головне вікно додатку поділено на кілька логічних зон (рис. 4.1):

– Область введення даних, де користувач зазначає площу ділянки (в гектарах), вибирає тип агротехнічної операції зі списку, а також задає продуктивність техніки, коефіцієнти ефективності використання часу та ризику;

– Область управління обчисленнями, що включає кнопку запуску розрахунку, скидання полів і формування звіту;

– Область результатів, у якій виводиться отримане значення фонду часу в годинах;

– Графічна панель, де будуються відповідні графіки залежностей (наприклад, площа → час, продуктивність → час).

На рисунку 4.1 представлено приклад головного вікна додатку, сформованого на основі зазначених елементів.

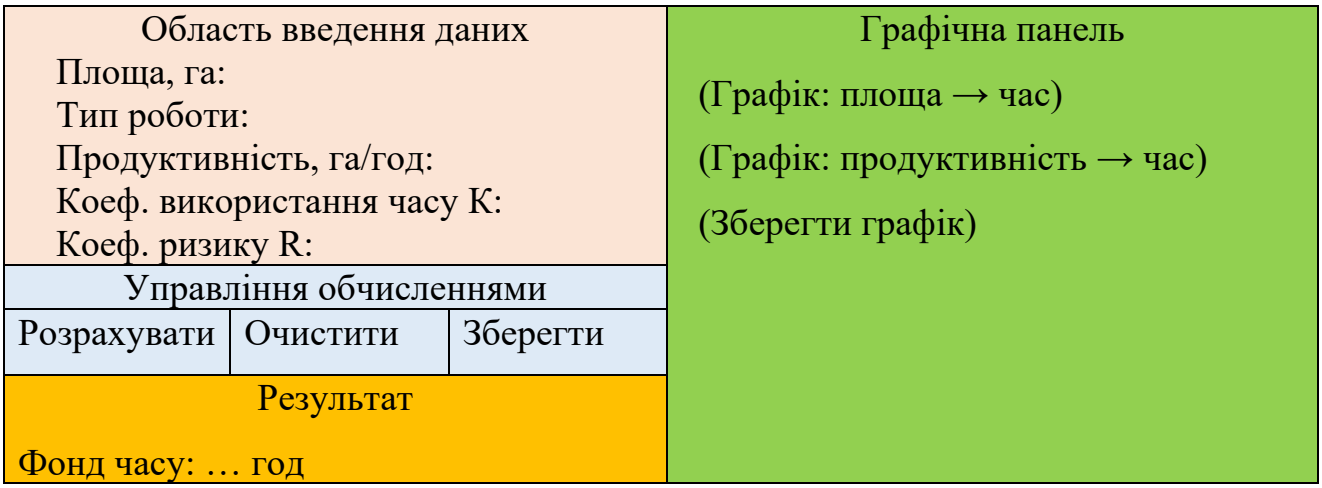

Рисунок 4.1 – Головне вікно інтерфейсу desktop-додатку

Однією з ключових особливостей інтерфейсу є використання перевірки даних у реальному часі. Наприклад, при введенні площі або продуктивності система автоматично перевіряє, чи є значення числовим і чи не виходить воно за межі допустимих діапазонів. Якщо виявлено помилку, виводиться попередження, а поле підсвічується. Це дозволяє зменшити кількість обчислювальних помилок і підвищити достовірність результатів.

Додатково передбачено функції збереження результатів до текстового файлу, а також формування графіка в окремому вікні або його експорт у файл зображення. Такі можливості реалізовані через модуль OutputManager, що взаємодіє з частиною інтерфейсу, відповідальною за обробку подій натискання кнопок.

Інтерфейс адаптовано до використання як на ноутбуках із середньою роздільною здатністю, так і на стаціонарних ПК з великим екраном. Завдяки цьому програма може бути застосована як в офісних умовах агрофірми, так і безпосередньо в польових лабораторіях.

Отже, створення зручного, функціонального й адаптивного інтерфейсу є невід’ємною частиною програмного рішення. Реалізований дизайн не лише відповідає сучасним вимогам до desktop-програм, а й підвищує практичну цінність додатку для фахівців у сфері рослинництва.

4.2. Результати розробки інтерфейсу користувача Desktop додатку для визначення фонду часу виконання робіт у рослинництві

У межах розробки програмного забезпечення особливу увагу було приділено створенню зручного, інтуїтивно зрозумілого та функціонально повного інтерфейсу користувача. Завдяки використанню бібліотеки Tkinter було реалізовано багатовіконну структуру додатку, яка забезпечує як введення даних, так і візуалізацію результатів розрахунків. Основною метою стало забезпечення аграрних фахівців засобом, який дозволяє не лише проводити точні розрахунки фонду часу, але й враховувати реальні погодні умови у розрахунках.

Головне вікно додатку містить логічно структуровані блоки введення вхідних параметрів (рис. 4.2). Серед них: площа ділянки, тип агротехнічної операції, продуктивність техніки, коефіцієнт ефективності використання часу (K), коефіцієнт ризику (R), а також нововведене поле – вибір міста, що використовується для визначення актуального стану погоди. Завдяки інтеграції з відкритим API OpenWeatherMap, додаток здатен у реальному часі отримувати метеодані для обраного регіону та визначати погодний коефіцієнт впливу.

Після натискання кнопки «Показати погоду» відкривається окреме вікно з інформацією про поточний стан атмосфери в обраному місті. Виводяться дані про основну характеристику погоди (наприклад, "хмарно"), температуру повітря, рівень вологості та коефіцієнт впливу, який автоматично інтегрується в обчислення. У разі помилки (наприклад, відсутність інтернет-з’єднання або

неправильне написання міста), інтерфейс виводить повідомлення з точним кодом помилки та її описом, наприклад: HTTP 404: city not found.

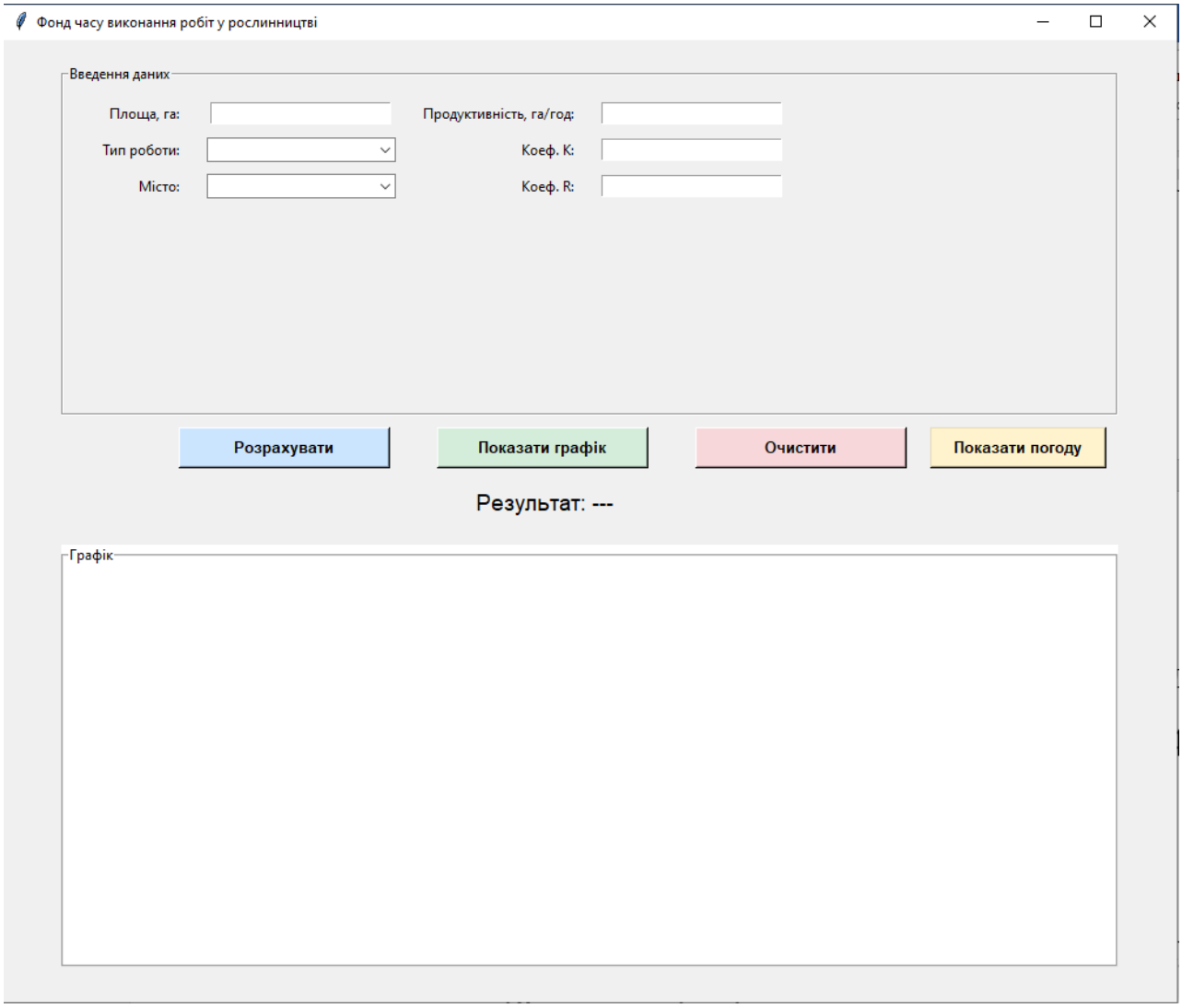

Рисунок 4.2 – Вікно інтерфейсу desktop-додатку для визначення фонду часу виконання робіт у рослинництві

Форма для розрахунку містить кнопку запуску обчислень, після чого у центральній частині вікна виводиться результат у годинах. Крім того, у нижньому блоці вікна відображається графік залежності фонду часу та ризику від площі поля. Графік реалізований за допомогою бібліотеки matplotlib та містить дві осі: перша відображає змінений (квадратний корінь) фонд часу, а друга – ймовірність ризику у вигляді нелінійної функції. Це дозволяє оцінити, як зміна площі при фіксованих параметрах впливає на загальний час виконання робіт та пов’язані ризики.

На рисунку 4.3 представлено зовнішній вигляд головного вікна додатку з усіма реалізованими функціональними компонентами.

```
def get_weather_info(city_name):
    url = f"http://api.openweathermap.org/data/2.5/weather?q={city_name}&appid={API_KEY}&units=metric"
    response = requests.get(url)
    data = response.json()
    weather_main = data['weather'][0]['main']
    return WEATHER_MAP.get(weather_main, 1.2)
```

Рисунок 4.3 – Фрагмент коду запиту до API сервера погоди OpenWeatherMap

Цей блок відповідає за отримання погодного коефіцієнта на основі поточних умов в обраному місті. Далі цей коефіцієнт інтегрується у формулу обчислення фонду часу, що дозволяє відображати більш реалістичні результати з урахуванням зовнішніх чинників.

У підсумку, розроблений інтерфейс повністю відповідає вимогам сучасних інформаційних систем аграрного призначення. Він є зручним для використання на різних етапах прийняття рішень щодо планування польових робіт, забезпечує достатню гнучкість та точність, а також легко адаптується до нових умов завдяки модульності архітектури та відкритості використаних технологій.

4.3. Результати використання створеного Desktop додатку для визначення фонду часу виконання робіт у рослинництві

Після завершення етапу розробки Desktop додатку було проведено серію тестувань із метою перевірки працездатності програмного забезпечення в реальних сценаріях аграрної діяльності. Тестові приклади охоплювали виконання польових робіт на різних ділянках, із застосуванням різних типів агротехнічних операцій, продуктивності техніки та погодних умов.

У першому прикладі користувач задав площу 150 гектарів для операції «Сівба» із продуктивністю 4.5 га/год, коефіцієнтом використання часу K = 0.85

та ризиком R = 1.1. Місто для отримання погодних умов було обрано «Lviv», а погодні дані в момент запиту відповідали стану «Clouds». У результаті автоматично було визначено коефіцієнт впливу погоди рівний 1.1, що інтегрувався у фінальне обчислення фонду часу. Отриманий результат становив 43.37 год, що підтверджує адекватність застосованої математичної моделі.

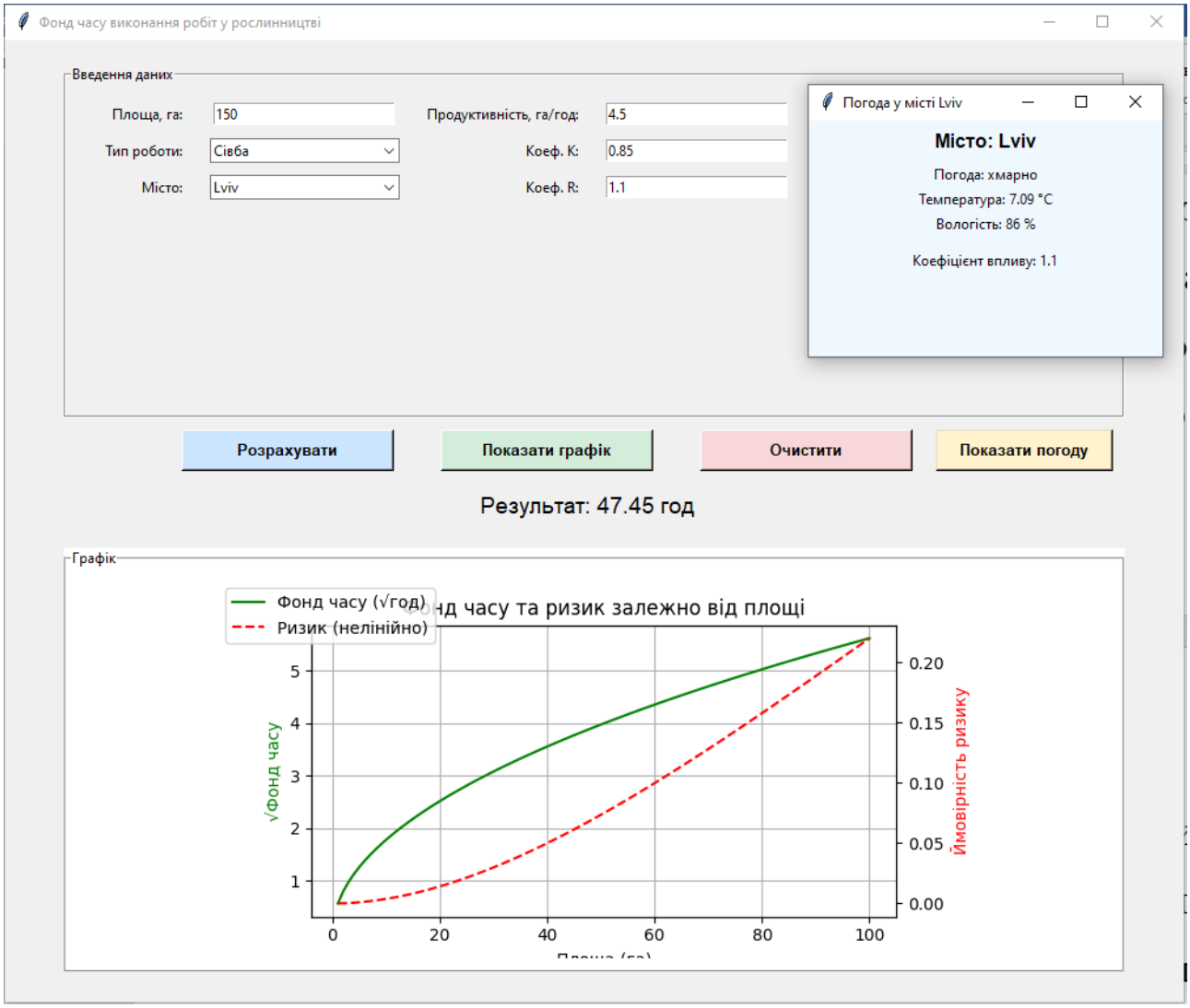

Рисунок 4.4 – Вікно desktop-додатку із результатами визначення фонду часу виконання робіт у рослинництві

Після розрахунку користувач має можливість переглянути динамічний графік, який відображає залежність фонду часу та рівня ризику від площі поля. Графік візуалізується у нижній частині програми й дозволяє оцінити зміну навантаження на техніку в разі збільшення площі або зміни коефіцієнтів. Наприклад, при площі понад 80 гектарів різке зростання ризику стає помітним

через нелінійне зростання часу виконання робіт і відповідно ймовірність виходу за межі агротехнічних термінів.

У другому прикладі модель була використана для розрахунку обробітку поля в місті «Odesa» за умов дощової погоди. У цьому випадку коефіцієнт погодного впливу автоматично визначився як 1.3. Це спричинило помітне збільшення загального фонду часу – з базових 33,4 год до понад 50,2 год, що змусило користувача переглянути терміни виконання та логістику виконання робіт. Саме така гнучкість у коригуванні параметрів дає змогу приймати обґрунтовані управлінські рішення.

Слід зазначити, що додаток демонструє високу стійкість до помилок користувача. У разі неправильного введення числових даних (наприклад, літери замість чисел) або відсутності інтернет-з’єднання, інтерфейс виводить чітке повідомлення про помилку (рис. 4.5).

```
HTTP 404: city not found

requests.exceptions.ConnectionError: [Errno ...] Failed to establish a new connection
```

Рисунок 4.5 – Фрагмент повідомлення у разі неправильного введення числових даних або відсутності інтернет-з’єднання

Ці повідомлення (рис. 4.5) дозволяють користувачу швидко виявити і виправити проблему, не зупиняючи роботу з програмою.

Загалом результати використання показали, що розроблений додаток дозволяє не лише автоматизувати розрахунки, а й адаптувати їх до зовнішніх чинників, таких як погода, що особливо важливо для сучасного точного землеробства. Візуалізація, модульність, захист від помилок і можливість розширення функціоналу свідчать про доцільність практичного впровадження цього рішення у фермерських господарствах, агрофірмах та дорадчих службах.

РОЗДІЛ 5.
ОХОРОНА ПРАЦІ

5.1. Аналіз стану умов праці на робочому місці

У процесі розробки Desktop додатку для аграрного сектора програміст безпосередньо працює у середовищі з переважно розумовими навантаженнями, тривалим перебуванням перед монітором комп’ютера та взаємодією з електронною технікою. Робоче місце користувача програмного забезпечення, а також розробника, як правило, розташоване у приміщенні офісного типу або лабораторії з відповідним технічним оснащенням. Тому надзвичайно важливою складовою є створення безпечних і комфортних умов праці, що відповідають чинному законодавству та нормативам охорони праці.

Умови праці за характером виконуваної діяльності класифікуються як роботи з переважним застосуванням зорово-рухового аналізатора та нервово-емоційного напруження. За класифікацією, це належить до категорії праці легкої інтенсивності, однак вона може супроводжуватись статичною втомою та зоровим перенапруженням, якщо не дотримані ергономічні вимоги.

Однією з ключових характеристик є рівень освітлення робочої зони. Згідно з ДБН В.2.5-28:2018 «Природне і штучне освітлення», мінімальна освітленість робочого столу при роботі з комп’ютером має становити не менше 300 лк. У разі недостатнього природного освітлення повинне застосовуватись комбіноване (загальне та локальне) штучне освітлення з використанням ламп денного світла або LED-технологій. Нижче наведено приклад порівняльної таблиці фактичних і нормативних параметрів освітлення:

Ще одним важливим аспектом є мікроклімат. Згідно з ДСН 3.3.6.042-99 «Санітарні норми мікроклімату», температура повітря в робочих приміщеннях повинна підтримуватись у межах 20–24°C при відносній вологості 40–60%. Перевищення або зниження цих показників може призвести до зниження працездатності, підвищення втомлюваності та зростання ризику захворювань.

Таблиця 5.1 – Порівняння фактичних параметрів освітлення з нормативними

Параметр	Нормативне значення	Фактичне значення	Висновок
Освітленість, лк	≥ 300	320	Відповідає нормі
Тип ламп	LED/ДСП	LED	Відповідає
Контрастність зображення	Висока	Висока	Норма

Умови праці на робочому місці розробника повинні враховувати й рівень шуму. Джерелами шуму є системний блок комп’ютера, вентилятори охолодження, принтери або інші периферійні пристрої. Згідно з ГОСТ 12.1.003-83, допустимий рівень шуму у приміщенні не повинен перевищувати 50 дБ. У разі використання кількох ПК в одному офісі рекомендовано застосування шумопоглинальних панелей, акустичних екранів або програмного розкладу роботи пристроїв для зменшення акустичного навантаження.

На рисунку 5.1 наведено приклад правильно організованого робочого місця користувача програмного забезпечення. Видно, що екран монітора розташовано на рівні очей, клавіатура розміщена з урахуванням зручного кута нахилу зап’ясть, а спинка крісла забезпечує підтримку поперекового відділу хребта.

Важливим чинником є також режим праці та відпочинку. При тривалій роботі за комп’ютером рекомендовано робити перерви кожні 60 хвилин по 10–15 хвилин із виконанням вправ для зняття зорового й статичного напруження. Доцільним є впровадження програм нагадування про перерви або використання режиму затемнення екрана у фоновому режимі.

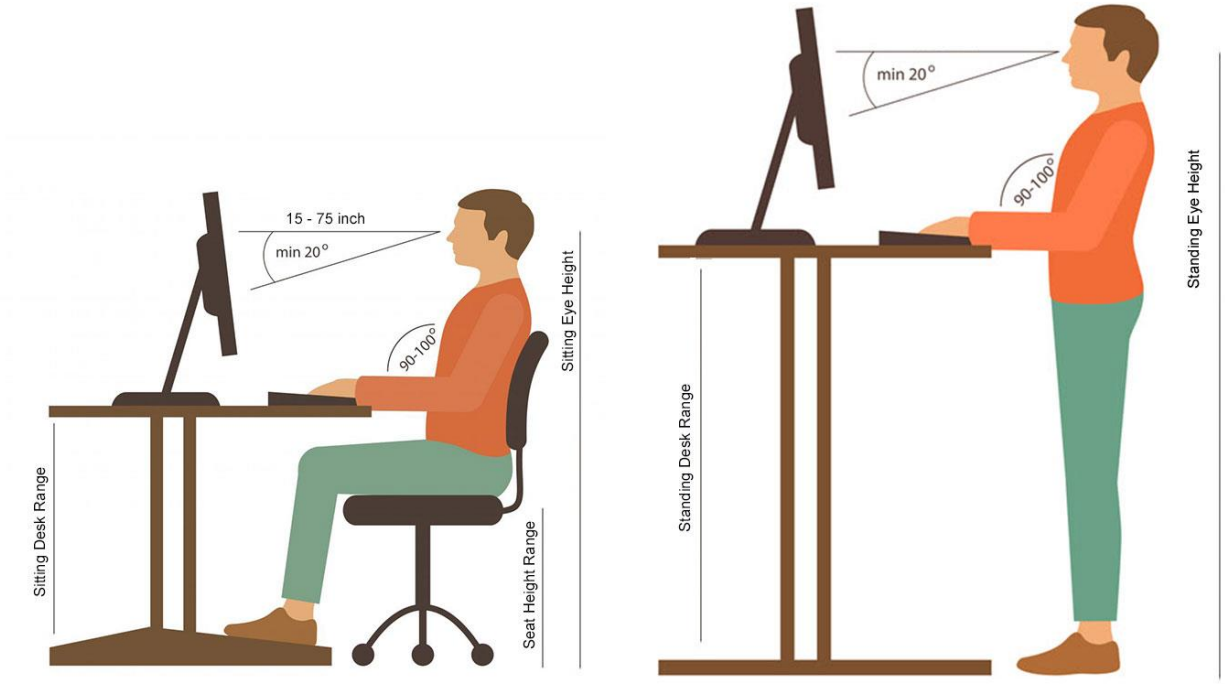

Рисунок 5.1 – Схема ергономічного робочого місця користувача Desktop додатку для визначення фонду часу виконання робіт у рослинництві

Загалом умови праці при використанні Desktop додатку вважаються сприятливими, якщо дотримано норм мікроклімату, освітлення, ергономіки й психоемоційного навантаження. У разі потреби застосування програми в умовах агропідприємства (наприклад, польові лабораторії, віддалені офіси) слід додатково подбати про джерела резервного живлення, належну вентиляцію приміщення та дотримання електробезпеки.

5.2. Небезпечні та шкідливі чинники

Під час виконання робіт, пов’язаних із розробкою та експлуатацією програмного забезпечення, у тому числі Desktop додатку для аграрного сектору, існує низка небезпечних і шкідливих виробничих чинників, що впливають на здоров’я працівника. Незважаючи на те, що програміст чи користувач не взаємодіє безпосередньо з важким обладнанням або

небезпечними хімічними речовинами, його робоче середовище все ж не позбавлене ризиків.

Серед основних небезпечних чинників, які супроводжують роботу за комп’ютером, слід зазначити можливість ураження електричним струмом у разі несправності електромережі або порушення правил експлуатації електрообладнання. Також потенційну небезпеку можуть становити перегрів блоків живлення, коротке замикання, або падіння техніки (монітора, системного блоку) з поверхні столу через неправильне встановлення.

Таблиця 5.2 – Небезпечні та шкідливі виробничі чинники

Категорія чинника	Приклад	Можливі наслідки	Заходи захисту
Електричний	Пошкодження кабелю	Ураження струмом	Заземлення, справна ізоляція, УЗО
Ергономічний	Невірне розташування екрана	Біль у шиї, порушення постави	Регулювання висоти монітора та крісла
Світловий	Недостатнє освітлення	Напруження зору, головний біль	Настільна лампа, LED-освітлення
Психоемоційний	Стрес при дедлайнах	Виснаження, зниження працездатності	Раціональне планування, перерви
Електромагнітний	Випромінювання монітора	Потенційний вплив на нервову систему	Використання сертифікованих дисплеїв
Температурний	Перегрів ПК	Пожежна небезпека	Очищення вентиляції, температура ≤ 25°C

До шкідливих чинників належать тривала дія монотонного навантаження на зір, недостатня рухова активність, психоемоційне перевантаження під час вирішення складних логічних задач, а також вплив електромагнітного випромінювання моніторів і радіочастотних пристроїв. Окрім цього, в умовах поганого мікроклімату – високої температури або сухого повітря – посилюється втомлюваність і зростає ризик професійних захворювань.

У таблиці нижче наведено типові небезпечні та шкідливі чинники, які мають бути враховані під час організації робочого місця розробника або користувача Desktop додатку.

Важливим фактором є також постійна дія статичної напруги на опорно-руховий апарат через тривале сидіння. Якщо працівник не робить регулярних перерв або не змінює положення тіла, зростає ризик розвитку остеохондрозу, варикозного розширення вен та інших захворювань опорно-рухової системи. З цією метою на робочих місцях рекомендується встановлення ергономічних меблів, зокрема крісел з регульованою спинкою, підлокітниками та підтримкою поперекового відділу.

Усі вищенаведені чинники не повинні залишатися поза увагою керівництва підприємства та фахівців з охорони праці. Навіть у галузях, де основна діяльність є інтелектуальною, необхідно впроваджувати профілактичні заходи, спрямовані на запобігання професійним ризикам, підвищення ергономічності робочого простору та дотримання санітарно-гігієнічних норм.

5.3. Інструкція із охорони праці під час розробки Desktop додатку для визначення фонду часу виконання робіт у рослинництві

Розробка програмного забезпечення, зокрема Desktop додатку для агропромислових потреб, передбачає тривале перебування працівника за комп’ютером. У зв’язку з цим виникає потреба у дотриманні інструкцій з охорони праці, що регламентують безпечні умови праці в офісних та

лабораторних приміщеннях. Такі інструкції повинні бути спрямовані на мінімізацію впливу шкідливих факторів на організм розробника, а також на запобігання нещасним випадкам, пов’язаним з порушенням правил електробезпеки, ергономіки чи мікроклімату.

Перед початком виконання службових обов’язків працівник має пройти вступний інструктаж з охорони праці, а також ознайомитись із правилами експлуатації офісної техніки, використанням комп’ютерного обладнання, пожежної безпеки та поведінки у разі надзвичайної ситуації. Робоче місце повинне бути організоване відповідно до державних стандартів із врахуванням нормативів зосередження уваги, фізіологічного комфорту та мінімізації навантаження на опорно-руховий апарат.

Усі електроприлади, що використовуються в процесі програмування, повинні бути справними, сертифікованими та мати заводське заземлення. Категорично забороняється експлуатація пошкоджених кабелів, неізольованих розеток або використання саморобних подовжувачів. Для зменшення ризику ураження електричним струмом рекомендується підключення через мережеві фільтри із захистом від перепадів напруги та наявність джерела безперебійного живлення (UPS).

Особливу увагу необхідно приділяти ергономічним параметрам робочого місця. Монітор повинен розташовуватись на відстані не менше 60 см від очей користувача, а його верхній край має бути приблизно на рівні очей. Крісло має бути оснащене спинкою з можливістю регулювання кута нахилу, що дозволяє підтримувати правильну поставу впродовж усього робочого дня. Клавіатура повинна розміщуватись так, щоб руки знаходились у природному положенні без надмірного напруження.

Робочий простір повинен бути забезпечений належним освітленням, бажано природним у денний час, або комбінованим – із застосуванням загального та місцевого штучного освітлення. Яскравість екрана має бути відрегульована відповідно до умов освітлення, а кольорова гама інтерфейсу програм повинна не викликати зорового перевантаження. Температурно-

вологісний режим також підлягає контролю – температура має утримуватись у межах 20–24°C, відносна вологість – 40–60 %.

Тривалість безперервної роботи за комп’ютером не повинна перевищувати однієї години, після чого слід передбачити короткочасну перерву тривалістю 10–15 хвилин. Під час перерв рекомендовано виконувати вправи для очей, змінювати положення тіла, проводити розминку рук і плечового поясу. У разі перевантаження або симптомів втоми працівник зобов’язаний звернутись до керівника для коригування режиму праці.

У разі виявлення несправностей у роботі обладнання, мерехтіння зображення, сторонніх шумів, запаху горілої проводки або перегріву пристроїв, забороняється продовжувати роботу. У таких випадках працівник має негайно вимкнути пристрої від живлення і повідомити відповідальну особу. Самостійне втручання у конструкцію техніки або її ремонт не допускається без спеціальної підготовки.

Інструкція також повинна передбачати алгоритм дій у разі виникнення надзвичайної ситуації – задимлення, пожежі, аварії електромережі. У приміщенні розробника має бути доступний вогнегасник, інструкція з евакуації, план розміщення засобів пожежогасіння та вільний доступ до аварійного виходу. Працівник повинен бути ознайомлений із сигналами тривоги, місцезнаходженням аптечки та телефонними номерами аварійних служб.

Таким чином, дотримання інструкцій з охорони праці під час розробки програмного забезпечення є невід’ємною частиною організації сучасного виробничого процесу. Воно сприяє не лише забезпеченню безпечного середовища, а й збереженню здоров’я працівників, підвищенню продуктивності праці та формуванню відповідального ставлення до професійної діяльності в сфері інформаційних технологій.

ВИСНОВКИ І ПРОПОЗИЦІЇ

Виконана кваліфікаційна робота присвячена розробці desktop-додатку, основне завдання якого – полегшити процес розрахунку фонду часу виконання робіт у рослинництві. Цей додаток має враховувати низку параметрів – площу оброблюваних земель, види культур, обсяги необхідних робіт, продуктивність техніки та інші дані, які користувач може ввести самостійно. Такий підхід дозволяє адаптувати додаток під різні умови господарювання, як для дрібних фермерів, так і для великих агропідприємств.

Більшість популярних рішень або не містять модулів для автоматизованого розрахунку фонду часу виконання робіт, або реалізують їх частково, через обмеження щодо локальної специфіки агропроцесів. Це створює передумови для розробки індивідуальних програмних засобів, орієнтованих на конкретні умови господарювання та з урахуванням національних агротехнологічних нормативів.

Попри активне впровадження ІТ у рослинництво, наявна потреба в адаптованих локальних рішеннях, зокрема desktop-додатках, які можуть обробляти польові дані, враховувати специфіку регіону та оперативно надавати результат користувачеві без потреби у підключенні до мережі Інтернет. Це визначає актуальність розробки інструментів для визначення фонду часу виконання робіт у вигляді персонального програмного засобу.

Методологія розрахунку фонду часу є багатоаспектною і вимагає врахування як нормативно-технологічних, так і організаційних параметрів. Автоматизація цих обчислень у вигляді desktop-додатку дозволяє пришвидшити процес планування, зменшити ймовірність помилок та забезпечити адаптивність до змінних умов виробництва.

Сучасні алгоритми для визначення фонду часу в агротехнологіях охоплюють широкий спектр підходів – від класичних формул до складних моделей на основі машинного навчання. Вибір конкретної моделі залежить від цілей застосування, наявності вихідних даних, рівня автоматизації та технічних

можливостей користувача. Для desktop-додатків, що орієнтовані на невеликі фермерські господарства або агрономічні служби, оптимальним є використання аналітико-нормативного гібриду, який дозволяє швидко й наочно розраховувати необхідний час за базовими параметрами та адаптувати його з використанням поправкових коефіцієнтів або правил.

Нами здійснено вибір технологій та засобів для розробки Desktop додатку визначення фонду часу виконання робіт у рослинництві. Обрані засоби дозволяють розширити додаток у майбутньому. Зокрема, додати збереження у формат Excel або PDF, формувати звіти, підключити базу даних для зберігання полів, робіт та результатів.

Загальна структура додатку передбачає поділ на кілька логічних блоків: введення даних, обробка, обчислення, вивід результату та побудова графіка. На схемі нижче умовно зображено, як ці блоки взаємодіють між собою (рис. 2.3).

Вибрані інструменти дозволяють реалізувати всю потрібну функціональність: від простого розрахунку до формування повного звіту. При цьому програма залишиться доступною для користувача без спеціальних технічних знань, а її робота не залежатиме від наявності інтернету чи складних систем.

Вибрана математична модель дозволяє надзвичайно гнучко адаптувати додаток під конкретне господарство. Наприклад, у фермерському господарстві з однією машиною та одним оператором зазвичай приймається , а коефіцієнти втрат задаються вручну на основі власного досвіду. У розробленому додатку користувач може задати більшість параметрів вручну, що дозволяє програмі бути не просто калькулятором, а моделлю підтримки прийняття рішень, яка працює в умовах невизначеності. Такий підхід відповідає сучасним принципам адаптивного планування в агросфері.

Створення додатку для визначення фонду часу виконання робіт у рослинництві передбачає організацію даних у вигляді об’єктів та класів, що дозволяє структурувати логіку, спростити обчислення та забезпечити гнучкість у подальшому доопрацюванні додатку. Застосування об’єктно-орієнтованого

підходу дає змогу чітко відокремити вхідні дані, алгоритмічну частину та результативні структури, а також впорядкувати процес взаємодії між модулями програми.

Структура об’єктів desktop-додатку орієнтована на ефективне обслуговування користувацького запиту, гнучкість до змін параметрів, повторне використання коду й простоту підтримки. Обраний підхід поєднує класичну об’єктно-орієнтовану архітектуру з практичними структурами даних, що забезпечують продуктивну й масштабовану реалізацію функціоналу desktop-додатку.

Основним класом у системі є WorkSession, який виступає контейнером для всієї інформації, необхідної для розрахунку. До цього класу входять властивості, що описують площу обробки, тип агротехнічної операції, продуктивність, коефіцієнти поправок, а також методи, які реалізують обчислення та форматування результатів.

Використання Validator у структурі desktop-додатку дозволяє реалізувати принцип «захисту від помилок на вході», що є невід’ємною частиною надійного програмного забезпечення для агропланування.

Клас CalculationModule не залежить від інтерфейсу або структури збереження даних, що дозволяє масштабувати його в майбутньому. Наприклад, у разі потреби можна реалізувати підтримку складніших формул, врахування сезонності, погодних коефіцієнтів або поетапного виконання робіт з розбиттям на змінні.

Клас GraphModule є візуально-аналітичним компонентом системи, який підвищує інформативність результатів розрахунків і забезпечує зручність для прийняття аграрних рішень на основі даних.

Створення зручного, функціонального й адаптивного інтерфейсу є невід’ємною частиною програмного рішення. Реалізований дизайн не лише відповідає сучасним вимогам до desktop-програм, а й підвищує практичну цінність додатку для фахівців у сфері рослинництва.

Розроблений інтерфейс завдяки використанню бібліотеки Tkinter повністю відповідає вимогам сучасних інформаційних систем аграрного призначення. Завдяки інтеграції з відкритим API OpenWeatherMap, додаток здатен у реальному часі отримувати метеодані для обраного регіону та визначати погодний коефіцієнт впливу. Він є зручним для використання на різних етапах прийняття рішень щодо планування польових робіт, забезпечує достатню гнучкість та точність, а також легко адаптується до нових умов завдяки модульності архітектури та відкритості використаних технологій.

Результати використання показали, що розроблений додаток дозволяє не лише автоматизувати розрахунки, а й адаптувати їх до зовнішніх чинників, таких як погода, що особливо важливо для сучасного точного землеробства. Візуалізація, модульність, захист від помилок і можливість розширення функціоналу свідчать про доцільність практичного впровадження цього рішення у фермерських господарствах, агрофірмах та дорадчих службах.

У межах розробки Desktop додатку для визначення фонду часу виконання робіт у рослинництві нами обґрунтовано комплекс заходів з організації безпечних умов праці для розробника. Розроблено інструкцію з охорони праці, яка враховує ергономічні вимоги до робочого місця та особливості експлуатації електротехнічного обладнання.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Горбатюк В.І. Програмування мовою Python з використанням Tkinter. – Київ: КНЕУ, 2023. – 178 с.

2. Гузенко Ю.І., Король В.І. Безпека життєдіяльності та охорона праці: навч. посіб. – Київ: Кондор, 2021. – 256 с. ISBN 978-617-8028-88-1.

3. ДБН В.2.5-28:2018. Природне і штучне освітлення. – К.: Мінрегіонбуд України, 2018. – 62 с.

4. ДСН 3.3.6.042-99. Державні санітарні норми мікроклімату виробничих приміщень. – К.: МОЗ України, 1999. – 36 с.

5. ДСТУ 2293:2010. Охорона праці. Терміни та визначення основних понять. – [Чинний]. – К.: ДП «УкрНДНЦ», 2010. – 22 с.

6. ДСТУ EN ISO 9241-5:2004. Ергономіка взаємодії людини та системи. Частина 5. Розміщення елементів робочого місця з відеодисплейним терміналом. – К.: Держспоживстандарт України, 2004. – 24 с.

7. Єгоров Д.І., Бутенко А.С. Агроінформатика: цифрові технології в сільському господарстві. – Полтава: ПДАА, 2023. – 204 с.

8. Жукова І.М. Системи підтримки прийняття рішень у сільському господарстві: навч. посіб. – Кропивницький: ЦНТУ, 2021. – 188 с.

9. Корж Р.Я., Волошин І.П. Проєктування користувацьких інтерфейсів: теорія і практика. – Львів: Видавництво ЛНУ, 2020. – 164 с.

10. Кравець В.М. Цифрові рішення в аграрному секторі: сучасні виклики. – Вінниця: ВНАУ, 2022. – 148 с.

11. Майстренко О.Ю. Ергономічне проєктування комп’ютеризованих робочих місць: навч. посіб. – Харків: ХНУМГ ім. О.М. Бекетова, 2020. – 147 с.

12. Олійник Т.М., Шевченко Ю.Р. Розробка додатків мовою Python: прикладне програмування. – Харків: Ранок, 2022. – 212 с.

13. Пономарьов В.І., Сидоренко А.М. Основи програмування мовою Python. – Київ: Ліра-К, 2023. – 304 с.

14. Станіславський О.Ю. Моделювання виробничих процесів у рослинництві. – Київ: НАУ, 2021. – 196 с.

15. Стеблюк П.І. Інформаційні системи в агропромисловому виробництві. – Київ: КНЕУ, 2021. – 278 с.

16. Tryhuba A., Boyarchuk V., Tryhuba I., Ftoma O., Padyuka R., Rudynets M., Forecasting the Risk of the Resource Demand for Dairy Farms Basing on Machine Learning, Proceedings of the 2nd International Workshop on Modern Machine Learning Technologies and Data Science (MoMLeT+DS 2020). Volume I: Main Conference. Lviv–Shatsk, Ukraine, June 2–3, 2020. pp.327–340.

17. Тригуба А. Системна модель цифрової трансформації сільських територіальних громад на основі обчислювального інтелекту. Вісник Львівського національного екологічного університету. Агроінженерні дослідження, 2022, № 26, р. 177–184.

18. Харківський В.Й., Токарєв В.В. Охорона праці в галузі інформаційних технологій: навч. посіб. – Львів: Львівська політехніка, 2022. – 184 с.

19. Харченко С.П. Комп'ютерна ергономіка: методи і технології. – Запоріжжя: ЗНТУ, 2020. – 134 с.

20. Ярова Н.М. Програмне забезпечення для аграрного сектору: метод. рекомендації. – Суми: СНАУ, 2020. – 56 с.

21. AgroOffice. URL: https://agrooffice.pp.ua/

22. International Labour Office. Guidelines on occupational safety and health management systems: ILO-OSH 2001. – 2nd ed. – Geneva: ILO, 2023. – 64 p. URL: https://www.ilo.org/global/topics/safety-and-health-at-work/lang--en/index.htm (дата звернення: 15.03.2025).

23. ISO 9241-5:1998. Ergonomic requirements for office work with visual display terminals (VDTs) – Part 5: Workstation layout and postural requirements. – Geneva: ISO, 2023. – 28 p.

24. Koval N., Tryhuba A., Kondysiuk I., Tryhuba I., Boiarchuk O. Forecasting the Fund of Time for Performance of Works in Hybrid Projects Using Machine Training Technologies. MoMLeT+ DS, 196–206.

25. Malanchuk O., Tryhuba A., Tryhuba I., Sholudko R., Pankiv O., A Neural Network Model–based Decision Support System for Time Management in Pediatric Diabetes Care Projects. International Scientific and Technical Conference on Computer Sciences and Information Technologies, 2023.

26. Nielsen J. Usability Engineering. – San Francisco: Morgan Kaufmann, 2020. – 362 p. ISBN 978-0125184069.

27. OpenWeatherMap. API Documentation. – 2024. – [Електронний ресурс]. – Режим доступу: https://openweathermap.org/api (дата звернення: 10.05.2025).

28. Pandas [Електронний ресурс]. URL: http://pandas.pydata.org/pandas–docs/stable/overview.html#license.

29. Python (programming language). URL: https://bitly.su/paHJe.

30. Stamenkovic Z., Ranđić S., Santamaria I., Pešović U. Advanced Wireless Sensor Nodes and Networks for Agricultural Applications // 24th Telecommunications Forum (TELFOR), 22–23 Nov. 2016, Belgrade, Serbia. – IEEE, 2016. – P. 420–423. – DOI: 10.1109/TELFOR.2016.7818709.

31. Tryhuba A., Boyarchuk V., Tryhuba I., Francik S., Rudynets M., Method and software of planning of the substantial risks in the projects of production of raw material for biofuel, CEUR Workshop Proceedings, 2020, 2565, pp. 116–129.

32. Tryhuba A., Kondysiuk I., Tryhuba I., Boiarchuk O., Tatomyr A., Intellectual information system for formation of portfolio projects of motor transport enterprises, CEUR Workshop Proceedings, 2022, 3109, pp. 44–52.

33. Tryhuba A., Kotenko V. Intelligent information system for resource planning in grain crops delivery projects on the basis of machine learning. IEEE 18th International Conference on Computer Science and Information Technologies (CSIT) 2023. P. 1–4.

34. Tryhuba A., Koval N., Tryhuba I., Boiarchuk O., Application of Sarima Models in Information Systems Forecasting Seasonal Volumes of Food Raw

Materials of Procurement on the Territory of Communities. CEUR Workshop Proceedings, 2022, 3295, pp. 64–75.

35. Tryhuba A., Padyuka R., Tymochko V., Lub P., Mathematical model for forecasting product losses in crop production projects. CEUR Workshop Proceedings, 2022, 3109, pp. 25–31.

36. Tryhuba A., Tryhuba I., Ftoma O., Boyarchuk O., Method of quantitative evaluation of the risk of benefits for investors of fodder–producing cooperatives. International Scientific and Technical Conference on Computer Sciences and Information Technologies, 2019, 3, pp. 55–58, 8929788.

37. Tryhuba I., Hutsol T., Tryhuba A., Cieszewska A., Kovalenko N., Mudryk K., Glowacki S., Bryś A., Tulej W., Sojak M. Analysis and Quantification of the Generation of Organic Waste by Households for Energy Production. Sustainability 2023 , 15 (22), 15922; https://doi.org/10.3390/su152215922

38. Tryhuba I., Tryhuba A., Hutsol T., Cieszewska A., Andrushkiv O., Glowacki S., Bryś A., Slobodian S., Tulej W., Sojak M. Prediction of Biogas Production Volumes from Household Organic Waste Based on Machine Learning. Energies 2024, 17(7), 1786; https://doi.org/10.3390/en17071786

39. Tryhuba I., Tryhuba A., Hutsol T., Lopushniak V., Cieszewska A., Andrushkiv O., Barabasz W., Pikulicka A., Kowalczyk Z., Vasyuk V. European Green Deal: Justification of the Relationships between the Functional Indicators of Bioenergy Production Systems Using Organic Residential Waste Based on the Analysis of the State of Theory and Practice. Energies. Енергія 2024 , 17 (6), 1461; https://doi.org/10.3390/en17061461

Додатки

Додаток А

Фрагмент програмного коду функціонального модуля (створення класів, обчислення, запити до API, побудову графіка)

```
import requests
import numpy as np
import matplotlib.pyplot as plt

API_KEY = "cce3670275c325cfc5e4bac6353a5d17"

OPERATIONS = {
    "Оранка": 1.8,
    "Культивація": 2.2,
    "Сівба": 1.5,
    "Обприскування": 3.0
}

WEATHER_MAP = {
    "Clear": 1.0,
    "Clouds": 1.1,
    "Rain": 1.3,
    "Drizzle": 1.3,
    "Thunderstorm": 1.3,
    "Snow": 1.4
}

class WorkSession:
    def __init__(self, area, operation, productivity, k_coeff, r_coeff,
weather_factor):
        self.area = area
        self.operation = operation
        self.productivity = productivity
        self.k_coeff = k_coeff
        self.r_coeff = r_coeff
        self.weather_factor = weather_factor

class Validator:
    def validate_all(self, area, productivity, k, r, operation):
        try:
            area = float(area)
            productivity = float(productivity)
            k = float(k)
            r = float(r)
        except ValueError:
            return False
        return area > 0 and productivity > 0 and 0 < k <= 1 and 1 <= r <= 1.5
and operation in OPERATIONS

class CalculationModule:
    def __init__(self, session):
        self.session = session

    def compute_time(self):
        base_time = self.session.area / (self.session.productivity *
self.session.k_coeff)
        total_time = base_time * self.session.r_coeff *
self.session.weather_factor
        return round(total_time, 2)

class GraphModule:
```

```
    def __init__(self, session):
        self.session = session

    def plot_complex_graph(self):
        areas = np.linspace(1, 100, 100)
        base_time = areas / (self.session.productivity * self.session.k_coeff)
        adjusted_time = base_time * self.session.r_coeff *
self.session.weather_factor
        risk = np.clip(1 - np.exp(-adjusted_time / 50), 0, 1)

        fig, ax1 = plt.subplots()
        ax2 = ax1.twinx()

        ax1.plot(areas, np.sqrt(adjusted_time), 'g-', label='Фонд часу (√год)')
        ax2.plot(areas, risk**2, 'r--', label='Ризик (нелінійно)')

        ax1.set_xlabel('Площа (га)')
        ax1.set_ylabel('√Фонд часу', color='g')
        ax2.set_ylabel('Ймовірність ризику', color='r')
        ax1.set_title('Фонд часу та ризик залежно від площі')
        ax1.grid(True)

        fig.legend(loc="upper left")
        return fig

def get_weather_info(city_name):
    try:
        url =
f"http://api.openweathermap.org/data/2.5/weather?q={city_name}&appid={API_KEY}&u
nits=metric&lang=ua"
        response = requests.get(url, timeout=5)
        if response.status_code != 200:
            return {"error": f"HTTP {response.status_code}:
{response.json().get('message', 'Unknown error')}"}
        data = response.json()
        weather_main = data['weather'][0]['main']
        weather_desc = data['weather'][0]['description']
        temp = data['main']['temp']
        humidity = data['main']['humidity']
        return {
            "main": weather_main,
            "description": weather_desc,
            "temp": temp,
            "humidity": humidity,
            "factor": WEATHER_MAP.get(weather_main, 1.2)
        }
    except requests.exceptions.RequestException as e:
        return {"error": str(e)}
```

Фрагмент програмного коду створення графічного інтерфейсу користувача з використанням Tkinte

```
import tkinter as tk
from tkinter import ttk, messagebox, Toplevel
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
from functionality_module import *

CITIES = ["Kyiv", "Lviv", "Odesa", "Kharkiv", "Dnipro", "Zaporizhzhia", "Ivano-
Frankivsk", "Ternopil"]

class TimeCalculatorApp:
    def __init__(self, root):
        self.root = root
        self.root.title("Фонд часу у рослинництві (з погодою)")
        self.root.geometry("1000x820")
        self.root.configure(bg="#f0f0f0")
        self.create_widgets()

    def create_widgets(self):
        frame_inputs = tk.LabelFrame(self.root, text="Введення даних", padx=20,
pady=10, bg="#f0f0f0")
        frame_inputs.place(x=50, y=20, width=900, height=300)

        labels = ["Площа, га:", "Тип роботи:", "Продуктивність, га/год:", "Коеф.
K:", "Коеф. R:", "Місто:"]
        entries = []
        for i, label in enumerate(labels):
            row = i // 2
            col = (i % 2) * 2
            tk.Label(frame_inputs, text=label, bg="#f0f0f0").grid(row=row,
column=col, padx=10, pady=5, sticky='e')
            entry = ttk.Combobox(frame_inputs, width=23) if "Тип роботи" in
label or "Місто" in label else tk.Entry(frame_inputs, width=25)
            entry.grid(row=row, column=col+1, padx=10)
            entries.append(entry)

        self.area_entry, self.operation_cb, self.prod_entry, self.k_entry,
self.r_entry, self.city_cb = entries
        self.operation_cb['values'] = list(OPERATIONS.keys())
        self.city_cb['values'] = CITIES

        self.weather_button = tk.Button(self.root, text="Показати погоду",
command=self.show_weather_window, bg="#fff3cd", font=('Arial', 10, 'bold'))
        self.weather_button.place(x=790, y=330, width=150, height=35)

        self.calc_button = tk.Button(self.root, text="Розрахувати",
command=self.calculate, bg="#cce5ff", font=('Arial', 10, 'bold'))
        self.calc_button.place(x=150, y=330, width=180, height=35)

        self.graph_button = tk.Button(self.root, text="Показати графік",
command=self.show_graph, bg="#d4edda", font=('Arial', 10, 'bold'))
        self.graph_button.place(x=370, y=330, width=180, height=35)

        self.clear_button = tk.Button(self.root, text="Очистити",
command=self.clear_fields, bg="#f8d7da", font=('Arial', 10, 'bold'))
        self.clear_button.place(x=590, y=330, width=180, height=35)

        self.result_label = tk.Label(self.root, text="Результат: ---",
font=('Arial', 14), bg="#f0f0f0")
```

```
        self.result_label.place(x=400, y=380)

        self.graph_frame = tk.LabelFrame(self.root, text="Графік", padx=10,
pady=10, bg="#ffffff")
        self.graph_frame.place(x=50, y=430, width=900, height=360)

    def calculate(self):
        area = self.area_entry.get()
        operation = self.operation_cb.get()
        productivity = self.prod_entry.get()
        k = self.k_entry.get()
        r = self.r_entry.get()
        city = self.city_cb.get()

        validator = Validator()
        if not validator.validate_all(area, productivity, k, r, operation):
            messagebox.showerror("Помилка", "Будь ласка, перевірте правильність
введених даних.")
            return

        weather_data = get_weather_info(city)
        if not weather_data or "error" in weather_data:
            messagebox.showerror("Помилка", f"Не вдалося отримати дані погоди:
{weather_data.get('error', 'Невідома причина')}")
            return

        session = WorkSession(float(area), operation, float(productivity),
float(k), float(r), weather_data["factor"])
        calc = CalculationModule(session)
        result = calc.compute_time()

        self.session = session
        self.result_label.config(text=f"Результат: {result} год")

    def show_graph(self):
        for widget in self.graph_frame.winfo_children():
            widget.destroy()
        if not hasattr(self, 'session'):
            messagebox.showwarning("Увага", "Спочатку виконайте розрахунок.")
            return
        graph = GraphModule(self.session)
        fig = graph.plot_complex_graph()
        canvas = FigureCanvasTkAgg(fig, master=self.graph_frame)
        canvas.draw()
        canvas.get_tk_widget().pack()

    def show_weather_window(self):
        city = self.city_cb.get()
        data = get_weather_info(city)
        if not data or "error" in data:
            messagebox.showerror("Помилка", f"Не вдалося отримати дані погоди:
{data.get('error', 'Невідома причина')}")
            return
        win = Toplevel(self.root)
        win.title(f"Погода у місті {city}")
        win.geometry("300x200")
        win.configure(bg="#f0f8ff")
        for key in ["description", "temp", "humidity", "factor"]:
            label = f"{key.capitalize()}: {data.get(key, '---')}"
            tk.Label(win, text=label, bg="#f0f8ff").pack(pady=2)

    def clear_fields(self):
        self.area_entry.delete(0, tk.END)
        self.operation_cb.set('')
```

```
        self.prod_entry.delete(0, tk.END)
        self.k_entry.delete(0, tk.END)
        self.r_entry.delete(0, tk.END)
        self.city_cb.set('')
        self.result_label.config(text="Результат: ---")
        for widget in self.graph_frame.winfo_children():
            widget.destroy()

if __name__ == "__main__":
    root = tk.Tk()
    app = TimeCalculatorApp(root)
    root.mainloop()
```