Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

Кафедра Програмного забезпечення комп'ютерних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *Заіки Максима Ігоровича*
(ПІБ)

академічної групи *121-20-2*
(шифр)

спеціальності *121 Інженерія програмного забезпечення*
(код і назва спеціальності)

освітньої програми *Інженерія програмного забезпечення*
(назва освітньої програми)

на тему: *Розробка програмного забезпечення веб застосунку обробки графічних зображень сільськогосподарських культур*

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	*проф. Лактіонов І.С.*			
розділів:				
спеціальний	*проф. Лактіонов І.С.*			
економічний	*доц. Касьяненко Л.В.*			
Рецензент				
Нормоконтролер	*доц. Мартиненко А.А.*			

Дніпро
2024

Міністерство освіти і науки України
НТУ «Дніпровська політехніка»

ЗАТВЕРДЖЕНО:
завідувач кафедри
програмного забезпечення комп'ютерних систем
(повна назва)

_____ М.О. Алексєєв
(підпис) (прізвище, ініціали)

« » _____ 2024 року

ЗАВДАННЯ
на кваліфікаційну роботу

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *121-20-2* *Заіки Максима Ігоровича*
(група) (прізвище та ініціали)

тема кваліфікаційної роботи *Розробка програмного забезпечення веб застосунку обробки графічних зображень сільськогосподарських культур*

затверджена наказом ректора НТУ «ДП» від *23.05.2024* № *469-с*

Розділ	Зміст виконання	Термін виконання
Спеціальний	***На основі матеріалів проєктно-технологічної практики та інших науково-технічних джерел провести аналіз стану рішення проблеми та постановку задачі. Обґрунтувати вибір та здійснити реалізацію методів вирішення проблеми***	*10.06.2024 р.*
Економічний	***Провести розрахунок трудомісткості розробки програмного забезпечення, витрат на створення ПЗ й тривалості його розробки***	*14.06.2024 р.*

Завдання видав _____ *проф. Лактіонов І.С.*
(підпис) (посада, прізвище, ініціали)

Завдання прийняв до виконання _____ *Заіка М.І.*
(підпис) (прізвище, ініціали)

Дата видачі завдання: *14.01.2024 р.*

Термін подання кваліфікаційної роботи до ЕК: *17.06.2024 р.*

РЕФЕРАТ

Пояснювальна записка: 121 с., 54 рис., 3 дод., 24 джерела., 6 таблиць.

Об'єкт розробки: програмний веб застосунок обробки графічний зображень сільськогосподарських культур.

Мета кваліфікаційної роботи: підвищення показників ефективності діяльності сільського господарста шляхом впровадження у процеси агромоніторингу веб застосунку обробки зображень сільгоспкультур.

Вступ: постановка задачі, опис об’єкту дослідження та оброблюваних даних, висвітлення актуальності обраної теми, аргументація щодо обраних методів вирішення, галузь розробки та галузь застосування.

Перший розділ: дослідження предметної області, визначення актуальності завдання, функціональне призначення розробки, постановка завдання, основні вимоги.

Другий розділ: вибір платформи для розробки, виконання проектування програми і її розробка, наведення опису алгоритму і структури функціонування системи, визначення вхідних і вихідних даних, наведення характеристики складу параметрів технічних засобів, опис роботи програми.

Економічний розділ: визначення трудомісткості розробленого продукту, проведення підрахунку вартості роботи по створенню застосунку та розрахування необхідного обсягу часу на його створення.

Практичне значення: створення веб застосунку для обробки графічних зображень з метою оптимізації процесу отримання робітниками вихідного зображення сільськогосподарських культур з правильною кольоропередачею.

Актуальність програмного продукту: веб застосунок дозволить зменшити мінімально необхідний рівень теоретичної та технічної складової моніторингових заходів задля отримання візуальних даних. Ефективно аналізована інформація на їх основі дозволяє впроваджувати рішення, що є не тільки теоретично обґрунтованими, а й економічно ефективними.

Ключові слова: ВЕБ ЗАСТОСУНОК, HTML, CSS, JS, PHP, БРАУЗЕР, ЗОБРАЖЕННЯ.

ABSTRACT

Explanatory note: 121 pp., 54 fig., 3 appendix, 24 sources, 6 tables.

Object of development: software web application for editing graphic omages of agricultural crops.

The purpose of the qualification work: increasing the efficiency indicators of agricultural activity by introducing a web application for image processing of agricultural crops into the processes of agromonitoring.

Introduction: statement of the problem, description of the research object and processed data, highlighting the relevance of the chosen topic, justification of the chosen solution methods, the field of development and the field of application.

The first section: research of the subject area, determination of the relevance of the task, functional purpose of the development, formulation of the task, basic requirements.

The second section: choosing a platform for development, performing program design and its development, providing a description of the algorithm and the structure of the system's functioning, determining input and output data, providing characteristics of the composition of technical means parameters, a description of the program's operation.

Economic section: determining the labor intensity of the developed product, calculating the cost of work on creating the application and calculating the required amount of time for its creation.

Practical meaning: creation of a web application for processing graphic images in order to optimize the process of obtaining by workers the original image of agricultural crops with the correct color rendering.

Relevance of the software product: the web application will allow to reduce the minimally necessary level of the theoretical and technical component of monitoring measures to obtain visual data. Effectively analyzed information based on them allows to implement solutions that are not only theoretically justified, but also economically effective..

Keywords: WEB APPLICATION, HTML, CSS, JS, PHP, BROWSER, IMAGE.

ЗМІСТ

СПИСОК УМОВНИХ ПОЗНАЧЕНЬ

ОС – операційна система;

ПК – персональний комп’ютер;

ПЗ – програмне забезпечення;

ІТ – інформаційні технології;

HTML – мова розмітки гіпертексту;

CSS – каскадні таблиці стилів;

JS, Python, Java, Ruby, PHP, Self, Scheme – мови програмування

БД – бза даних;

CMS – система для керування вмістом, програма для адміністрування та організації вебсайтів;

DOM – об’єктна модель документу;

Тег – ключове слово для фрагменту інформації в HTML;

Кросбраузерність – властивість веб застосунку, що визначає можливість коректного відображення елементів інтерфейсу відповідно до початкового дизайну;

БпЛА – безпілотний літальний апарат;

GPU – графічний процесор;

CPU – центральний процесор;

TPU – тензорний блок обробки;

API – прикладний програмний інтерфейс;

IDE – інтегроване середовище розробки.

ВСТУП

Темою даної кваліфікаційної роботи є розробка веб застосунку обробки графічних зображень. Інтернет та його елементи вже давно стали невід'ємною частиною існування не тільки окремо взятої людини, а й бізнесу, громад, об'єднань, професійних спілок, установ тощо. Всі вони для своїх потреб використовують сучасні технології, які спрощують, а іноді й повністю автоматизують ті чи інші процеси та алгоритми. Це може бути моніторинг показників, формування звітності, передача великих об'ємів даних, збереження, видалення та редагування інформації тощо. У всі сфери життя так чи інакше інтегрується розробник та його програмний продукт для вирішення різноманітних задач. Однією з них і є робота з візуальним контентом.

Переважний обсяг інформації людина отримує саме у візуальному контексті, тож невід'ємною частиною Інтернет-простору є графічні дані. Саме візуальне сприйняття стало формувати загальне враження від продукту, послуги та іншої інформації. Навіть ті сфери діяльності, які раніше рідко або взагалі не використовували графічні дані у своїй роботі, тепер активно інтегрують їх у свої рішення. Відповідно все більше зростає потреба у роботі із зображеннями, їх редагування, обробка, розпізнавання.

Для дослідження було обрано сільськогосподарський сектор, для якого створено веб застосунок для швидкої обробки зображень. Передбачається, що вхідними даними для нього будуть зображення, отримані пристроями працівників (смартфони, фотоапарати тощо) в процесі візуального моніторингу захворюваності рослин. Враховуючи специфіку основної аудиторії користувачів, застосунок повинен мати ряд обов'язкових характеристик, серед який можна виділити інтуїтивно зрозумілі елементи керування, доступність з усіх пристроїв. Як основний сценарій використання було виділено можливість змінювати ряд основних фізичних властивостей зображень, такий як яскравість, контраст, прозорість, ступінь розмиття, орієнтацію і так далі. Це має на меті допомогти працівникам оптимізувати процес отримання, передачі, та редагування графічної

інформації. Наприклад, обробка в одній кольоровій гамі фото зернових, технічних та кормових культур з правильними налаштуваннями для чіткої передачі даних про ту чи іншу хворобу рослини, її видозміненість, мутацію, результат використання певного виду добрив тощо.

Враховуючи те, що сільське господарство в Україні є однією з провідних галузей, продукти, створені для неї, матимуть, потенційно, великий дохід, внаслідок ліквідності підприємств та інших форм господарювання. Це великий ринок, який щодня вирішує цілий спектр задач, які потенційно можна автоматизувати та спростити.

Для вирішення поставленої задачі було обрано веб застосунок. Це рішення було обумовлено рядом факторів, один з який це кросплатформеність, тобто властивість програмного забезпечення працювати більш, ніж на одній апаратній платформі і (або) операційній системі. Це важливо, оскільки працівники повинні мати доступ до продукту на будь-якому етапі виробництва. До того ж це дозволить оптимізувати роботу програми на пристроях, що були випущені у різні роки, мають різну будову, способи відображення інформації, її вводу, периферії, архітектуру, кількість вбудованої пам’яті та швидкість доступу до неї, можливість підключення додаткових носіїв і так далі.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Загальні відомості з предметної галузі

Робота із візуальними даними та використання сучасних методів їх обробки є невід'ємними частинами діяльності сучасного сільського господарства. Потреба виникає на етапі контролю перебігу процесів вирощування. Прикладами сценаріїв використання можуть бути: моніторинг з фотофіксацією, аерофотозйомка системами БпЛА для проведення аерофотозйомки, сільгосптехніка з навігаційними системами та відеокамерами для отримання даних із місцезнаходженням об'єктів, супутникові знімки для роботи з ділянками землі великої площі.

Прикладом подібного використання графічних даних є моніторинг захворюваності сільськогосподарських культур. Існує велика кількість його різновидів, серед них найбільш перспективним з точки зору впровадження новітніх технологічних рішень є візуальний, який передбачає візуальний аналіз рослин. Групою кваліфікованих працівників проводяться лабораторні дослідження, дані для яких збираються безпосередньо в польових умовах. Їх метою є своєчасне виявлення хвороби або її ознак у аналізованого виду, а також фіксація цих даних для їх подальшої обробки. Ці дослідження повинні бути комплексними та регулярними задля збереження врожайності, себто основної цілі, а їх дані - чіткими та якісно представленими для формування правильних висновків.

Внаслідок дії на рослину несприятливих умов навколишнього середовища та/або певних збудників (гриби, бактерії, віруси, паразитичні рослини тощо) на її поверхні з'являються візуальні ознаки (зміна кольору, розміру, форми рослини, її елементів тощо), правильне розпізнавання та аналіз яких дозволяє провести діагностування та вжити необхідних заходів знезараження.

Діагностування візуальним видом моніторингу може бути оптимізований та пришвидшений шляхом впровадження у нього інструменту для отримання візуальних даних, інформація з яких дозволить зробити обґрунтований аналіз з точки зорі відповідності існуючій проблемі та впровадити ефективне рішення для знезаражування культур. Цим інструментом може слугувати веб застосунок, функціоналом якого є обробка графічних зображень, що дозволить отримати якісні дані швидше, при цьому не задіюючи додаткових працівників та обладнання.

Використання запропонованої технології веб застосунку має наступні переваги:

- відсутність потреби у встановлення додатку, для його запуску потрібен тільки запит у браузері;
- оновлення на стороні серверу і, як результат, завжди актуальна версія додатку у користувача без потреби в актуалізації версій на кожному окремому пристрої завдяки особливостям клієнт-серверної архітектури (Рис. 1.1);
- можливість запуску на багатьох видах пристроїв, які підтримуються взаємодію з браузером;
- збереження даних на сервері з можливістю повторного звернення до них;
- порівняно низькі технічні вимоги;
- швидкість роботи застосунку;
- інтуїтивно зрозумілий інтерфейс без необхідності у комплексному опануванні.

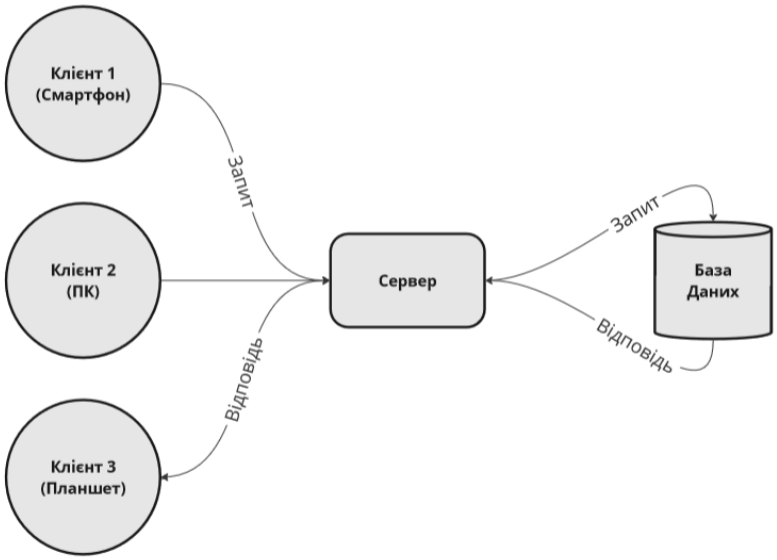

Рис. 1.1. Спрощена ілюстрація принципу роботи клієнт-серверної архітектури

Поточний стан розвитку сектору сільського господарства в Україні та технологічні рішення, які використовує у своїй діяльності ця галузь, свідчить про актуальність впровадження ІТ-продуктів з метою оптимізації багатоетапний процесів аналізу захворюваності сільськогосподарських культур.

Серед існуючих рішень на ринку застосунків обробки зображень у цій галузі виділяються комплексні додатки аналізу врожайності, у яких робота з графічними даними є однією з функціональних можливостей. У своїй роботі вони можуть використовувати дані оптичних датчиків, що можуть бути інтегровані у сільгосптехніку. Одним із джерелом вхідних даних для подальшої обробки можуть слугувати знімки, зроблені з використанням БпЛА або штучних супутників, а також смартфонів та фотоапаратів. Для чіткого розуміння існуючих потреб та запропонованих рішень доцільним є проведення порівняльного аналізу переваг та недоліків існуючих ІТ-продуктів для ринку сільського господарства, що також допоможе сформувати загальні вимогу по програмного забезпечення, його функціональнихї частин та дизайну.

Agrobase – веб застосунок (Рис. 1.2), розроблений у вигляді посібнику, розділеного на категорії в залежності від типу запиту. Користувачу надається детальний опис та фото рослин, їх хвороб, а також бур’янів, комах та добрив з можливістю їх порівняти (Рис. 1.3).

Рис. 1.2. Головна сторінка застосунку Agrobase

Рис. 1.3. Інформаційна сторінка Agrobase з даними про хворобу рослин

Таблиця 1.1

Аналіз застосунку Agrobase

Переваги	Недоліки
– Доступний з браузеру, себто не вимагає встановлення на пристрій користувача; – Зручний розподіл на категорії основних запитів; – Велика кількість зображень, доступних для перегляду та завантаження; – Не потребує реєстрації для перегляду вмісту; – Безкоштовний перегляд.	– Відсутність можливості завантаження власного матеріалів; – Потреба у володінні основною термінологією для пошуку потрібної інформації; – Відсутність взаємодії з іншими користувачами.

Plantix – мобільний застосунок (Рис.1.4), основною функцією якого є визначення уражених хворобою ділянок сільськогосподарських культур за допомогою технологій штучного інтелекту. Plantix веде активну співпрацю з кваліфікованими представниками агрономії з метою покращення алгоритмів машинного навчання та підвищення рівню врожайності (Рис. 1.5).

Застосунок має сучасний дизайн типу Material. Сам інтерфейс розділений на три основні категорії: «Your crops», «Community» та «You». У першому і основному розділі відображається актуальна дата та погодні умови населеного пункту, інформативні блоки щодо популярних добрив, шкідників та калькулятор їх розрахунку в залежності від кількості рослин. Також тут розміщений блок створення фото та діагностики отриманих даних за допомогою нейронної мережі. Другий розділ відображає запити та коментарі користувачів. Останній розділ відображає обліковий запис користувача та його дані, включно з ім’ям та фото.

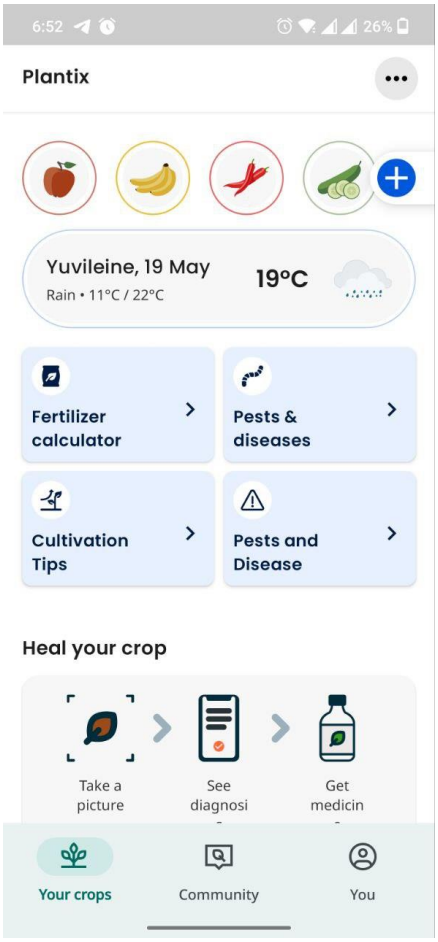

Рис. 1.4. Головна сторінка застосунку Plantix

Рис. 1.5. Результат аналізу зображення у застосунку Plantix

Таблиця 1.2

Аналіз застосунку Plantix

Переваги	Недоліки
– Простота викростання, що не потребує додаткових навичок та знань. – Швидкість отримання та формування готового висновку на основі штучного інтелекту. – Співпраця із кваліфікованими спеціалістами та організаціями. – Велика користувацька спільнота. – Можливість завантаження фото, зробленого на смартфон. – Низькі технічні вимоги. – Не потребує реєстрації для користування функціями. – Безкоштовне ПЗ.	– Відсутність можливості обробки зображення перед його аналізом. – Потреба у встановленні окремого ПЗ на пристрій – Відсутність версії для персональних комп'ютерів – Точність визначення хвороби може бути низькою. – Обмежена база даних культур. – Залежність від якості вхідного зображення – Приватність. Зображення, завантажені у застосунок можуть бути використані для машинного навчання.

Taranis – застосунок, націлений на використання даних аерофотозйомки (Рис. 1.6) для аналізу стану полів. В роботі використовуються дані БпЛА, датчиків та аналітичних систем. Для аналізу використовуються алгоритми штучного інтелекту, дані формуються у щоденні звіти з актуальним станом полів в реальному часі. В Україні вже понад 100 тис. гектарів полів обслуговуються системою Taranis (Рис. 1.7).

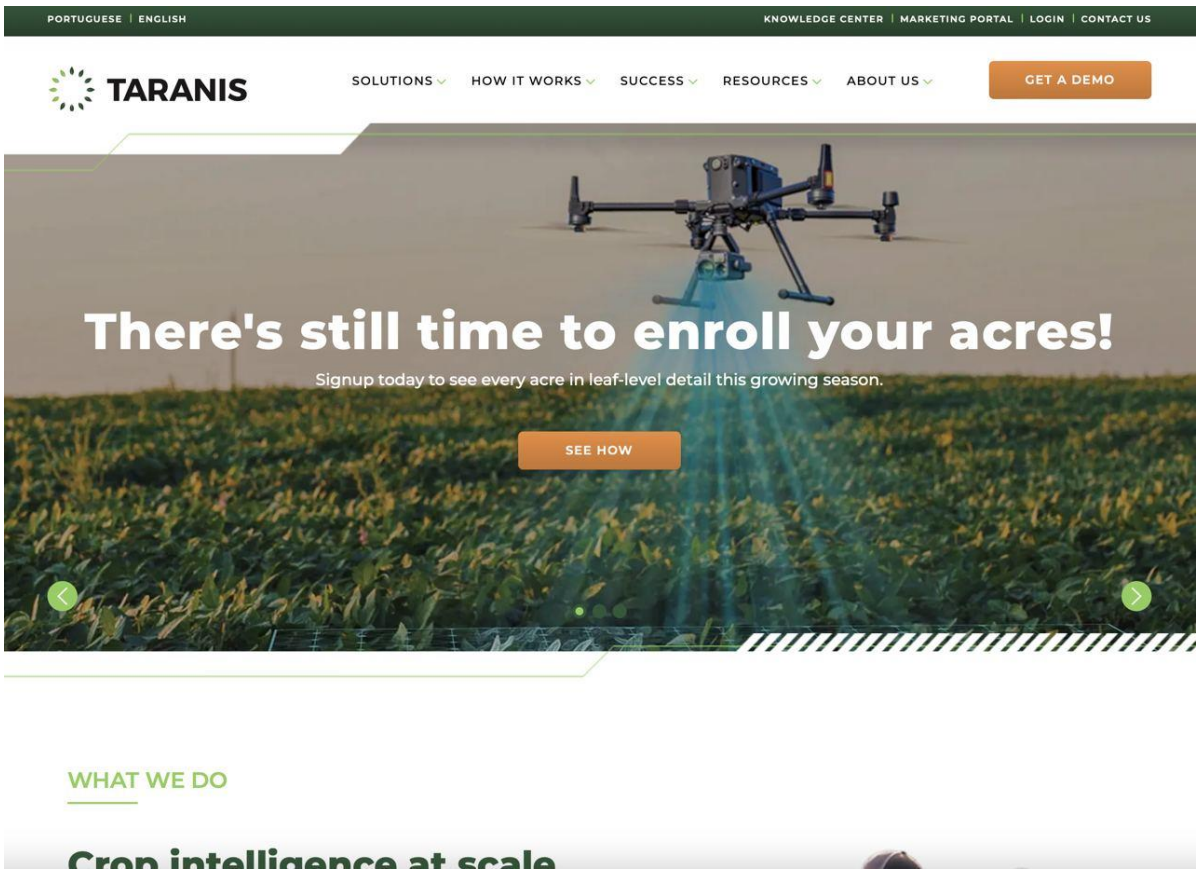

Рис. 1.6. Головна сторінка Taranis

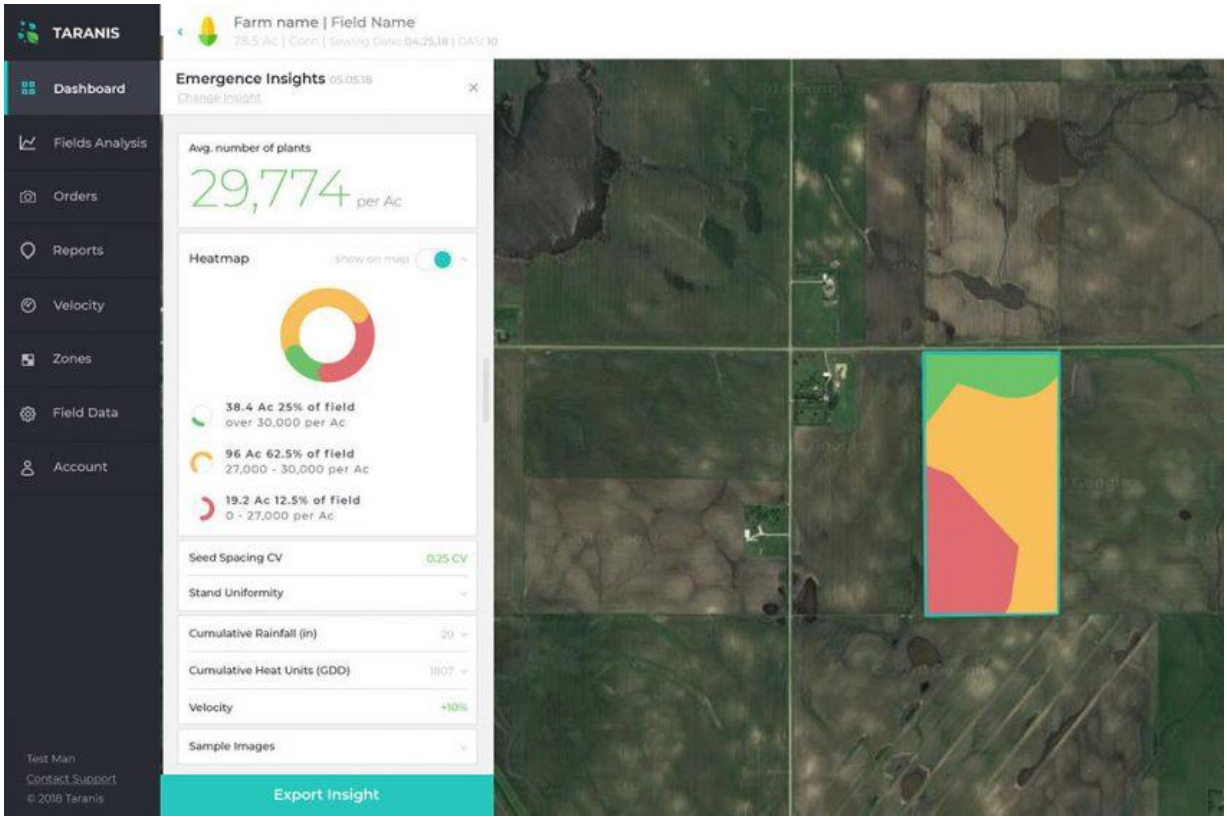

Рис. 1.7. Аналітика на основі супутникового знімку

Таблиця 1.3

Аналіз застосунку Taranis

Переваги	Недоліки
– Робота з високоточними зображенням, отриманих системами БЛПА – Висока швидкість обробки даних. – Моніторинг показників у реальному часі. – Оптимізація процесів моніторингу, часткова їх заміна. – Велика автоматизована система.	– Висока вартість через потребу у використанні БПЛА. – Потреба у навчанні для ефективного користування та аналізу отриманих даних. – Залежність від погодних умов.

Обрані для аналізу застосунки демонструють різні підходи до вирішення задач сільського господарства. Їх користувачі мають різний рівень кваліфікаційної підготовки та різні методи отримання інформації про стан оброблюваних культур. Не зважаючи на тенденцію інтегрування методів розпізнавання технологіями штучного інтелекту, польові дослідження за безпосередньої участі кваліфікаційного спеціаліста залишаються затребуваними.

1.2 Призначення розробки та галузь застосування

Потреба у можливості швидкої обробки зображень виникає на етапі отримання фінального зображення. Оскільки прилади для виконання цих задач часто не мають стандартизації вихідного кольору, виникають похибки і неточна передача тонів, а також експозиції зображення, його чіткості.

Варто додати і те, що процес оволодіння навичками та особливостями використання фотоапарату або інших приладів для фіксації зображень потребує

певного об’єму часу від спеціаліста. Подібне не завжди можливо, оскільки це не є профільною діяльністю працівника агросфери і не завжди є можливість додаткового навчання спеціаліста через брак кадрів відповідної кваліфікації та мотивації персоналу, нестачу базових навичок володіння необхідною технікою, ряд економічних особливостей тощо.

Створений продукт вирішує цю проблему, дозволяючи відкорегувати недоліки зображень, до того ж, якщо задачу з надання зображень виконує група незалежних працівників з різною технікою, то веб застосунок надає можливість створити базу однакових за кольором знімків.

Важливо мати можливість точно налаштовувати потрібний колір на готовому знімку, адже однією із галузей сільського господарства є моніторинг загальних показників захворюваності різних культур. Тож, для виконання відповідних досліджень компанії використовують зображення і виникає задача правильно відобразити візуальні ознаки захворювання, що не завжди вдається зробити одразу, наприклад, через неправильно налаштовані параметри фототехніки (наприклад, баланс білого, який через неправильне налаштування може хибно передати білий колір, тим самим змінивши загальний тон зображення). Подібні дефекти не є припустимими при роботі із зображенням.

1.3 Підстава для розробки

Відповідно до освітньої програми, згідно навчального плану та графіків навчального процесу, в кінці навчання студент виконує кваліфікаційну роботу.

Тема роботи узгоджується з керівником проєкту, випускаючою кафедрою, та затверджується з наказом ректора.

Таким чином підставами для розробки (виконання кваліфікаційної роботи) є:

– освітня програма спеціальності 121 «Інженерія програмного забезпечення»;

– навчальний план та графік навчального процесу;

– наказ ректора Національного технічного університету «Розробка програмного забезпечення веб застосунку обробки графічних зображень сільськогосподарських культур» № 469-с від 23.05.2024 р;

– завдання на кваліфікаційну роботу на тему «Розробка програмного забезпечення для приймальних комісій навчальних закладів».

1.4 Поставновка завдання

Темою даної кваліфікаційної роботи є розробка веб застосунку обробки графічних зображень, який повинен мати наступний функціонал та характеристики:

– завантаження власних графічних даних в якості вхідних для подальшої постобробки;

– графічний інтерфейс із списком базових налаштувань зображення та елементи введення даних, як бажаних параметрів;

– скидання обраних налаштувань;

– відображення внесених у зображення змін шляхом візуалізації обраних параметрів на вхідному зображенні задля можливості аналізу та порівняння (в тому числі після їх скидання);

– можливість створення поверх зображення графічних елементів, форму та колір яких визначається значенням відповідно обраних елементів та полів;

– Відображення результату аналізу зображення нейронною мережею на визначення виду сільськогосподарської культури та (за наявності) виду хвороби на основі наявних візуальних ознак.

– завантаження вихідних даних у форматі зображення із розширенням JPEG або PNG.

Метою кваліфікаційної роботи є підвищення показників ефективності діяльності сільського господарста шляхом впровадження у процеси агромоніторингу веб застосунку обробки зображень сільгоспкультур.

Об'єктом розробки є програмний веб застосунок обробки графічний зображень сільськогосподарських культур.

1.5 Вимоги до програми або програмного виробу

1.5.1. Вимоги до функціональних характеристик

Веб застосунок має бути виконаний у вигляді сайту з клієнтом, роль якого відіграє браузер, та сервером, який представляє собою веб-сервер, задачею якого є обробка запитів з різних пристроїв та виведення інформації.

Кінцевий продукт має відповідати наступним вимогам:

– швидке та безперебійне завантаження сторінки на всіх видах пристроїв, що підтримують системні вимоги;

– адаптивний дизайн, тобто коректне відображення графічних елементів, незалежно від параметрів дисплею, його фізичних характеристик;

– можливість завантажувати на сторінку зображення із власного сховища даних;

– введення користувачем даних з клавіатури або подібної периферії, а також з екранної клавіатур, якщо застосунок відкрито зі смартфону;

– перевірка введених користувачем даних на відповідність типу та формату, а також відсутність сценаріїв некоректної роботи застосунку та окремих його частин через недотримання наведеної вище умови;

– інтуїтивно зрозумілий дизайн інтерфейсу з урахування сучасних тенденцій та розробок.

1.5.2 Вимоги до інформаційної безпеки

Задля дотримання вимог до інформаційної безпеки користувача мають бути дотримані наступні умови користування застосунком:

– запобігання використанню файлів сумнівного походження;

– використання надійних паролів для реєстрації з метою запобігання викрадення даних;

– захищене та стабільне з'єднання із сервером;

– запобігання використанню незахищених мереж Wi-Fi у громадських місцях або інших методів підключення, надійність яких під сумнівом і не може бути перевірена;

– своєчасні оновлення клієнту, за допомогою якого відбувається взаємодія із застосунком, контроль актуальних версій, оновлень безпеки клієнту та пристрою.

1.5.3 Вимоги до складу та параметрів технічних засобів

Коректна робота веб застосунку передбачає технічні вимоги та своєчасну актуалізацію версій не тільки операційної системи, а й клієнту, з якого виконується вхід. Нижче наведено мінімальні вимоги для використання браузеру в залежності від системи (табл. 1.4):

Таблиця 1.4

Мінімальні технічні вимоги

Операційна система	Версія	
	Операційна система	Процесор
Windows	Windows 10 або пізнішої версії або Windows Server 2016 або пізнішої версії[1)]	Intel Pentium 4 або пізнішої версії з підтримкою SSE3
MacOS	macOS Catalina 10.15 або пізнішої версії[1)]	Не зазначено
Linux	Ubuntu 18.04 (64-розрядна версія) або пізнішої версії, Debian 10 або пізнішої версії, openSUSE 15.5 або пізнішої версії, Fedora Linux 38 або пізнішої версії[1)]	Процесор Intel Pentium 4 або пізнішої версії з підтримкою SSE3

Продовження табл. 1.4

Операційна система	Версія	
	Операційна система	Процесор
Android	Android 8.0 (Oreo)[1)]	Не зазначено

Примітки:

1. Для інших версій припинено випуск оновлень клієнту, припинено підтримку операційної системи. Це означає, що веб застосунок може відкритися, але є ймовірність некоректного відображення, в тому числі елементів дизайну.

1.5.4 Вимоги до інформаційної та програмної сумісності

Пристрій повинен мати встановлений клієнт, тобто браузер (останньої версії на момент встановлення), з якого відбувається перегляд та взаємодія із застосунком (рис. 1.8).

Також пристрій користувача у своїй будові повинен передбачати периферію для взаємодії з інтерфейсом, візуалізацією, введенням даних і т. д.

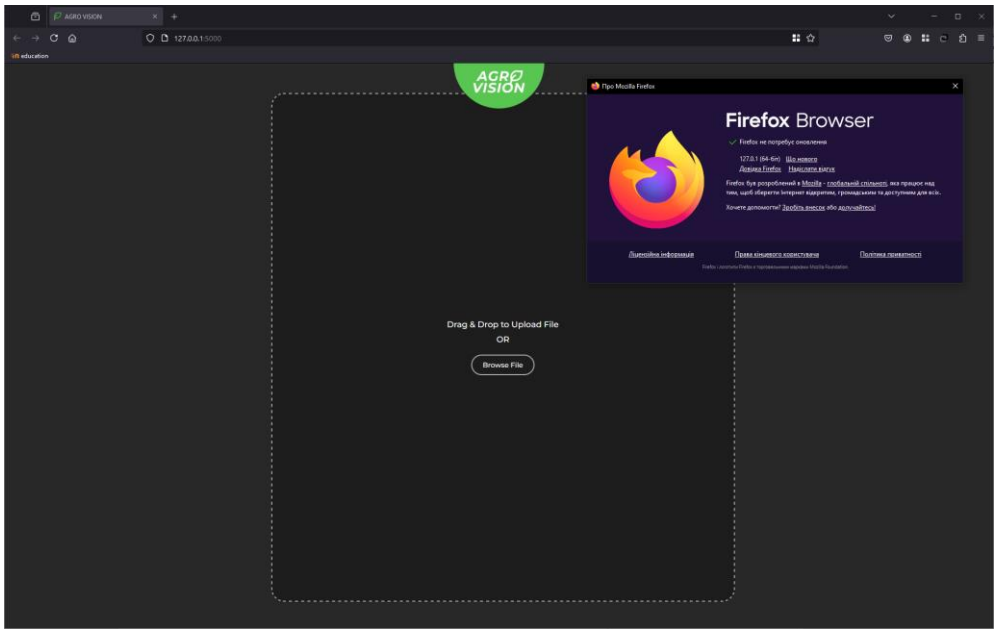

Рис. 1.8. Перегляд веб застосунку через останню актуальну версію браузеру Firefox 127.0.1 для персональних комп’ютерів

РОЗДІЛ 2
ПРОЕКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ

2.1 Функціональне призначення програми

В рамках даної кваліфікаційної роботи створено веб застосунок, функціональним призначенням якого є обробка зображень сільськогосподарських культур. Розроблене програмне забезпечення покликане оптимізувати процес отримання якісних графічних даних з токи зори їх візуальної інтерпретації та аналізу іншими кваліфікованими спеціалістами вищезгаданої галузі.

Користувач матиме можливість завантажити у застосунок файли з власного пристрою або підключеного зовнішнього носія для їх подальшої обробки. Для користувачів, які виконують вхід у застосунок з ПК, реалізовано функцію Drag & Drop, за допомогою якої потрібний файл можна перемістити у відповідне поле для його завантаження.

Після завантаження зображення відбувається його обробка нейронною мережею, яка визначатиме стан рослини, відповідна інформація відображатиметься у застосунку з можливістю змінити назву вхідного файлу з метою класифікації початкових даних.

Внесені зміни мають чітке візуальне представлення та змінюються в реальному часі в залежності від використаних ефектів (фільтрів), що дає можливість точно налаштувати необхідні параметри.

Окрім застосування фільтрів наявне створення графічних елементів поверх початкового зображення. Даних функціонал передбачає можливість вибору товщини пензля та його кольору.

Отримане в результаті виконаної обробки зображення, що включає у себе застосовані фільтри, можна завантажити на пристрій, з якого було виконаний вхід у застосунок.

Також серед представленого функціоналу є можливість повороту та дзеркального відображення вхідного зображення, що може допомогти відкоригувати початкові геометричні властивості для формування стандартизованого розміщення об’єктів.

У будь-який момент внесені зміни можна відмінити та повернутися до початкового представлення. Відбувається очищення намальованих елементів, а також застосованих фільтрів та трансформацій.

В якості прикладу стилізації програми зі схожим функціоналом було обрано плагін Camera Raw графічного редактору Adobe Photoshop. Використано подібні елементи інтерфейсу та загальний дизайн (Рис 2.1).

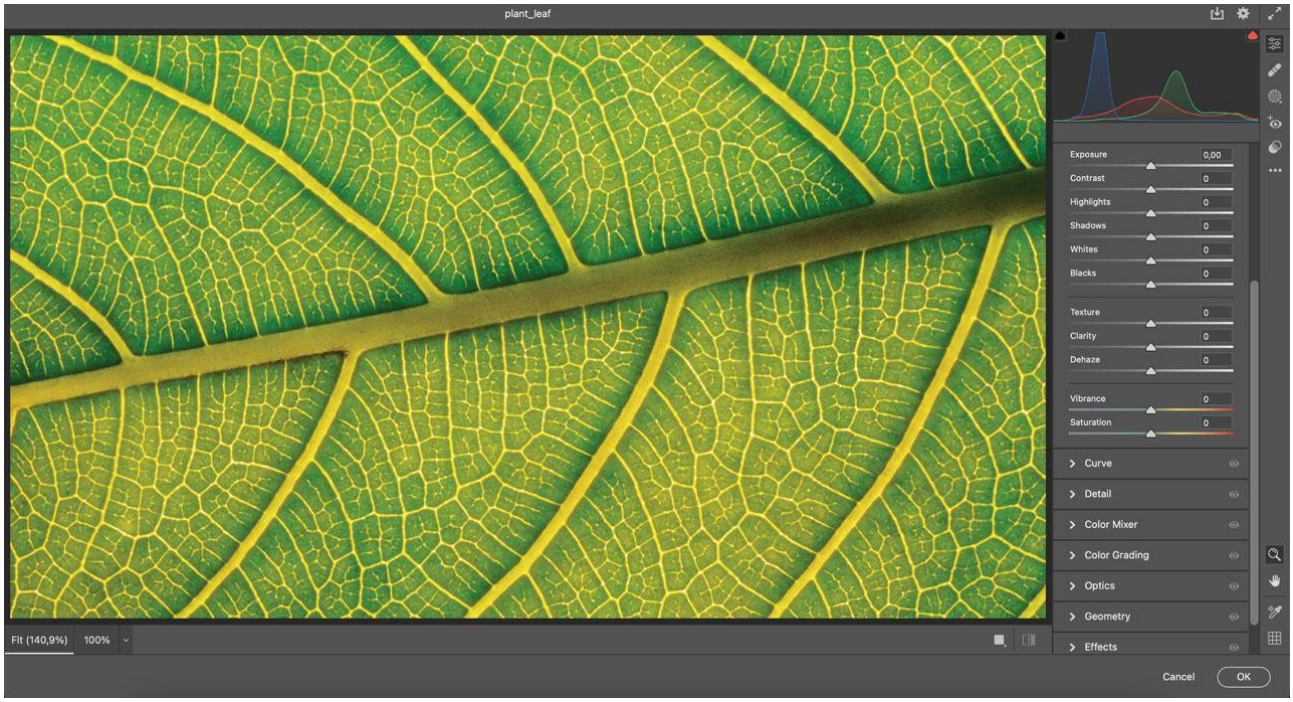

Рис. 2.1. Інтерфейс плагіну Camera Raw

Інтерфейс веб застосунку повинен відповідати стандартам, що передбачають коректне відображення графічних елементів без суттєвих візуальних та функціональних змін, що можуть виникати та пристроях з різними браузерами, операційними системами та їх версіями.

2.2 Опис застосованих математичних методів

В ході розробки веб застосунку використовувалися стандартні арифметичні операції, такі як додавання, віднімання, ділення з остачею тощо. Обробка певних подій, що виникають в результаті взаємодії користувача зі сторінкою, були виконана з використанням операторів порівняння. Вищеназвані операції не потребують використання сторонніх бібліотек.

2.3 Опис використаної архітектури та шаблонів проектування

Процес створення веб застосунку можна умовно розділити на дві частини: створення та навчання нейронної мережі, створення веб сторінки та користувацького інтерфейсу.

Підготовчим етапом до створення нейронної мережі є вибір масиву графічних даних (dataset), які будуть використовуватися для навчання та тренування. У застосунку використано масив даних з веб платформи Kaggle (Рис. 2.2). У цьому сервісі аналітики розміщують свої масиви даних для подальшого машинного навчання на їх основі. Датасет передбачає наявність директорій тренувальних та валідаційних файлів, розділених між собою (Рис. 2.3). Правильні відповіді, тобто правильні результати розпізнавання для нейронної мережі, зберігаються у назвах директорій, що мають дані одного типу. Вони представлені різними зображеннями одного об’єкту з метою збільшення кількості варіацій образу, що можна розпізнати.

З метою збільшення об’єму графічних даних масиву існує підхід використання одного зображення з різними варіаціями представлення, це може бути фрагмент початкових даних, змінена яскравість, контраст, змішення по шкалі HUE тощо (Рис. 2.4).

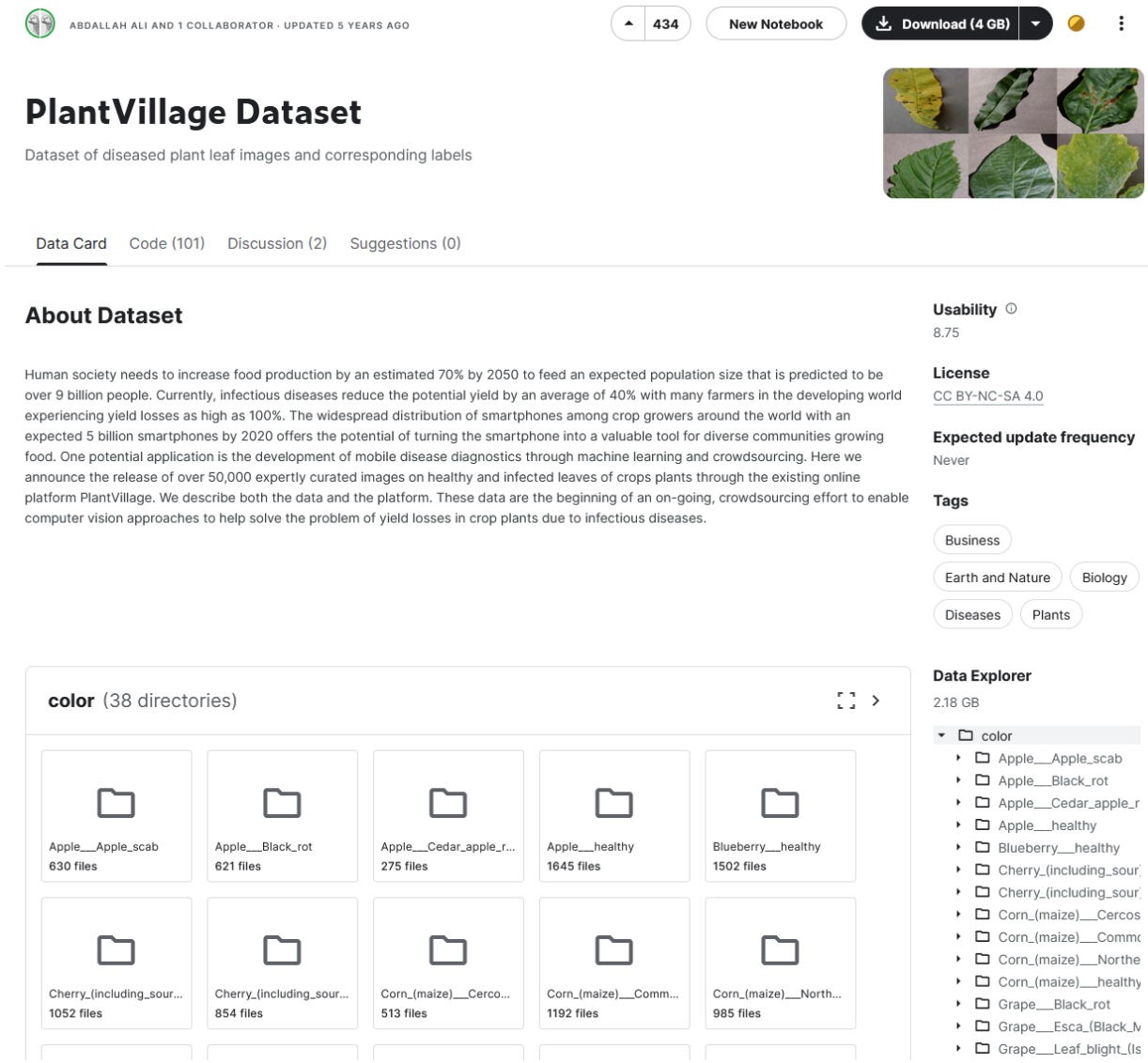

Рис. 2.2. Використаний у веб застосунку масив даних, розміщений на платформі Kaggle

Рис. 2.3. Приклад зображення в одній із директорій

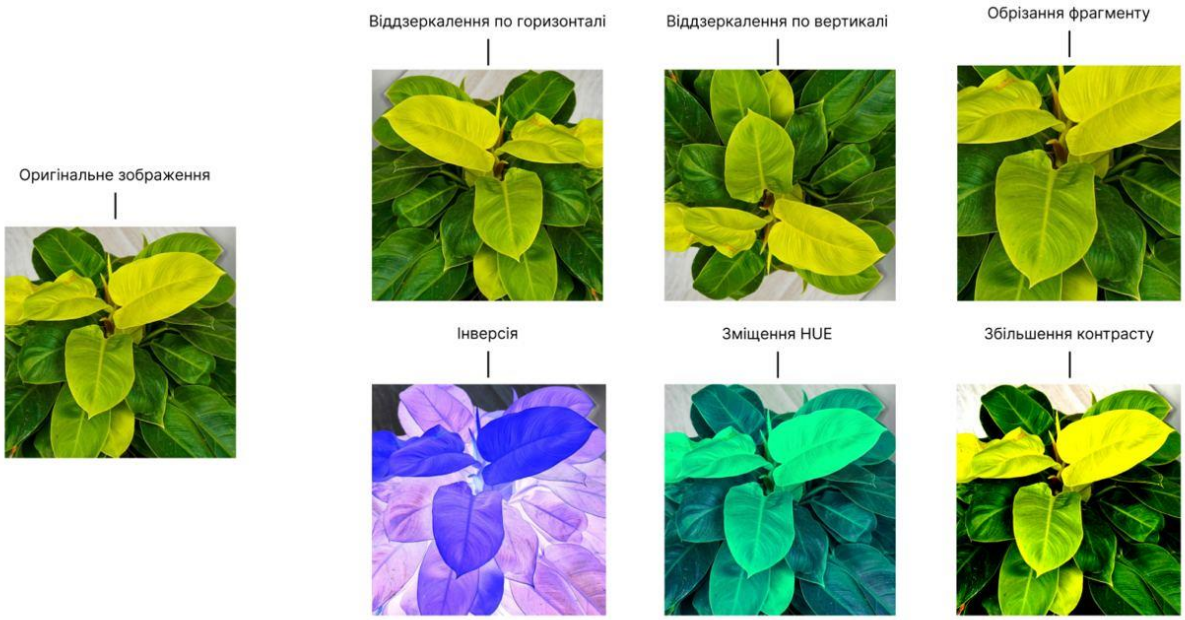

Рис. 2.4. Використання інтерпретацій до початкового зображення

Підхід до написання алгоритмів нейронної мережі передбачає використання сумісних версій бібліотек та фреймворків, що може викликати проблеми при локальному завантаженні. Задля оптимізації цього процесу популярною є практика використання веб середовищ програмування, які вже мають встановлені бібліотеки, які потрібно тільки імпортувати у проєкт. Одним із таких рішень є хмарна платформа Google Colaboratory. Це IDE, що працює з браузеру та надає безкоштовний доступ до графічних процесорів, які використовуються для обчислень багатовимірних масивів даних моделей машинного навчання. Дане середовище підтримує написання коду мовою програмування Python та не потребує встановлення встановлення бібліотек, для використання їх треба тільки імпортувати у самому файлі, який у Google Colaboratory реалізований у форматі Jupyter Notebook. Ця технологія дозволяє впроваждувати у код графічні елементи (зображення, графіки, коментарі), ділити код на блоки за конвертувати у інші формати (Рис. 2.5).

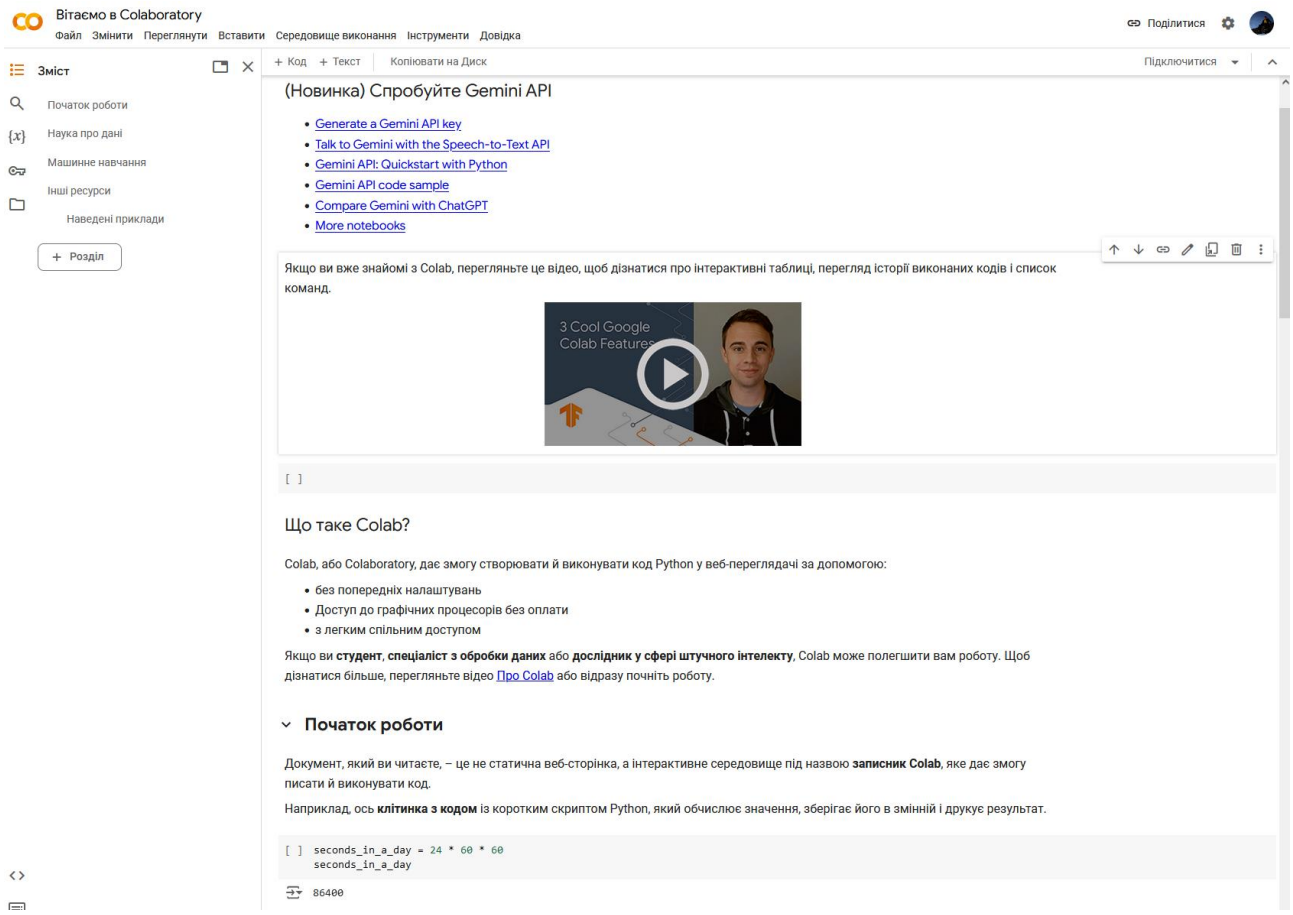

Рис. 2.5. Структура файлу Jupyter Notebook у Google Colaboratory

Додатковою перевагою Google Colaboratory є підключення хмарного диску даних (облікового запису, з якого було виконано вхід) з можливістю безпосередньої взаємодії з файловою структурою, що дозволяє не тільки переглядати та отримувати, а й редагувати та доповнювати вміст. Таким чином можна завантажити масив даних на локальний диск та звернутися до нього з браузеру, або ж підключитися до вже існуючої бази даних без потреби у завантаженні матеріалу.

Функціонал Kaggle (API) можна використовувати у Python, це дозволяє завантажувати дані безпосередньо з сервісу. Алгоритм встановлення:

1) Створення облікового запису на платформі Kaggle (Рис. 2.6).

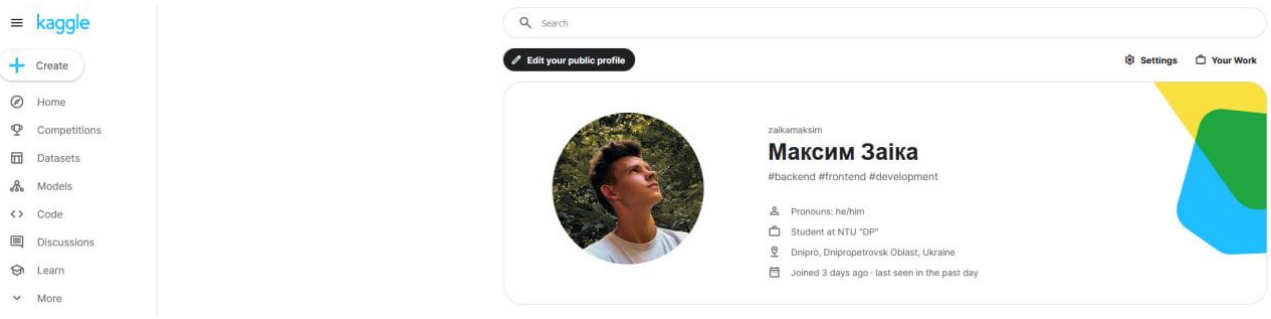

Рис. 2.6. Створений обліковий запис у Kaggle

2) Генерація API-токену у форматі JSON-файлу (Рис. 2.7).

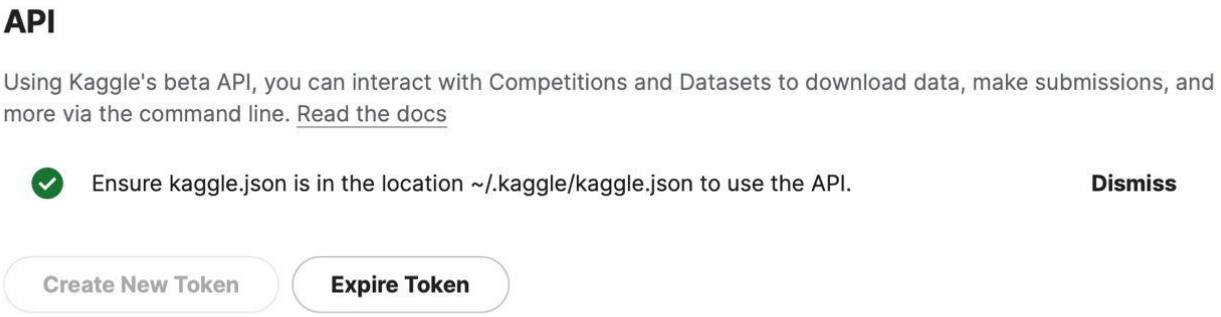

Рис. 2.7. Створення нового токену

3) Встановлення пакету kaggle у файлі Google Colaboratory (Рис. 2.8).

```
[ ] !pip install kaggle

Requirement already satisfied: kaggle in /usr/local/lib/python3.10/dist-packages (1.6.14)
Requirement already satisfied: six>=1.10 in /usr/local/lib/python3.10/dist-packages (from kaggle) (1.16.0)
Requirement already satisfied: certifi>=2023.7.22 in /usr/local/lib/python3.10/dist-packages (from kaggle) (2024.2.2)
Requirement already satisfied: python-dateutil in /usr/local/lib/python3.10/dist-packages (from kaggle) (2.8.2)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from kaggle) (2.31.0)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from kaggle) (4.66.4)
Requirement already satisfied: python-slugify in /usr/local/lib/python3.10/dist-packages (from kaggle) (8.0.4)
Requirement already satisfied: urllib3 in /usr/local/lib/python3.10/dist-packages (from kaggle) (2.0.7)
Requirement already satisfied: bleach in /usr/local/lib/python3.10/dist-packages (from kaggle) (6.1.0)
Requirement already satisfied: webencodings in /usr/local/lib/python3.10/dist-packages (from bleach->kaggle) (0.5.1)
Requirement already satisfied: text-unidecode>=1.3 in /usr/local/lib/python3.10/dist-packages (from python-slugify->kaggle) (1.3)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests->kaggle) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests->kaggle) (3.7)
```

Рис. 2.8. Процес інсталяції пакету kaggle

4) Завантаження вмісту API-токену (Рис. 2.9).

```
kaggle_credentails = json.load(open("kaggle.json"))
```

Рис. 2.9. Завантаження токену

5) Перенесення значень у змінні (Рис. 2.10).

```
os.environ['KAGGLE_USERNAME'] = kaggle_credentails["username"]
os.environ['KAGGLE_KEY'] = kaggle_credentails["key"]
```

Рис. 2.10. Ініціалізація змінних

6) Завантаження контенту (Рис. 2.11).

```
!kaggle datasets download -d abdallahalidev/plantvillage-dataset

Dataset URL: https://www.kaggle.com/datasets/abdallahalidev/plantvillage-dataset
License(s): CC-BY-NC-SA-4.0
Downloading plantvillage-dataset.zip to /content
100% 2.04G/2.04G [00:18<00:00, 181MB/s]
100% 2.04G/2.04G [00:18<00:00, 118MB/s]
```

Рис. 2.11. Завантаження масиву даних

7) Розпакування архіву (Рис. 2.12).

```
with ZipFile("plantvillage-dataset.zip", 'r') as zip_ref:
    zip_ref.extractall()
```

Рис. 2.12. Розпакування завантаженого архіву

Другим етапом розробки веб застосунку є створення інтерфейсу користувача та обробки подій на сторінці з вмістом, що буде завантажений на сторінку з додатковим контентом на основі обробки зображення нейронною мережею.

Для перегляду та редагування вмісту веб сторінки було використано вбудовані у систему браузерів інструменти розробника (Рис. 2.13). Серед основних переваг можна виділити:

– інтерактивна взаємодія з елементами DOM, відображення їх розмірів;

– відлагодження Java Script коду;

– аналіз мережевої активності;

– перегляд наявних помилок у коді та попереджень;

– редагування HTML без впливу на основний каталог;

– застосування CSS-стилів з підказками синтаксису та їх значень, відображення змін у реальному часі та місцем розташування правила у коді;

– симуляції перегляду з мобільних пристроїв (Рис. 2.14).

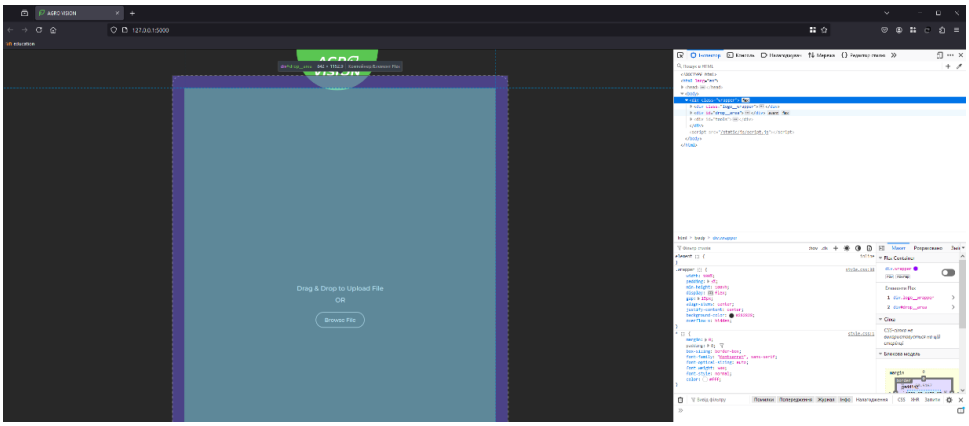

Рис. 2.13. Інструменти розробника на прикладі браузеру Mozilla Firefox

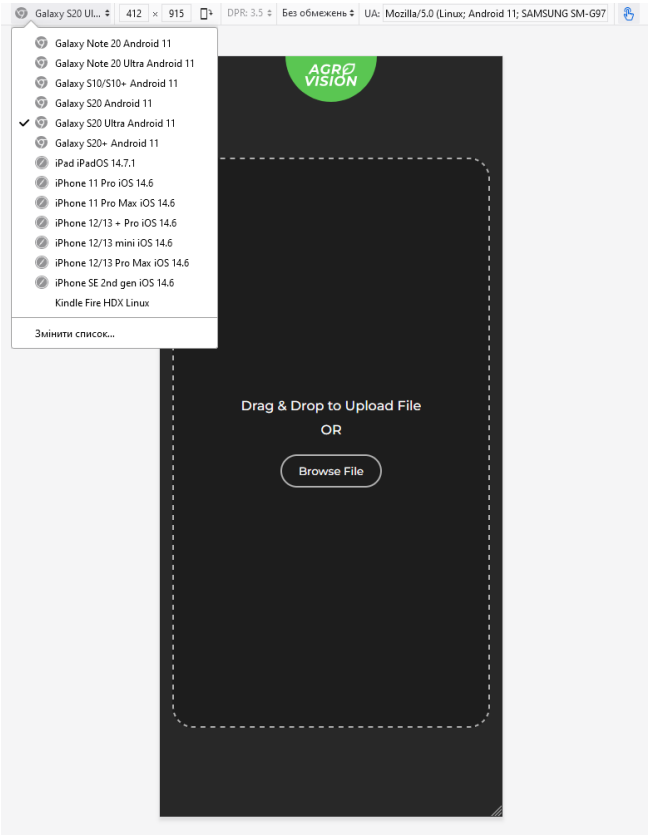

Рис. 2.14. Візуалізація відображення веб сторінки на смартфоні

2.4 Опис використаних технологій та мов програмування

Технологічний стек розроблюваного продукту складається з інструментів, функціональним призначенням яких є відображення, обробка та структуризація інтерфейсу застосунків веб середовища, а також даних, які вони обробляють.

HTML – мова розмітки гіпертексту. Являє собою структурований набір елементів веб застосунків, що оформлені у вигляді тегів, які оточують контент сторінки. Це дескриптори, в основу який закладено певні властивості відображення.

За синтаксисом тег повинен бути прописаний у кутових дужках та, якщо тег парного типу, мати другу частину, доповнену слешом (косою лінією) перед назвою елементу. Такий тег називають закриваючим (Рис. 2.15).

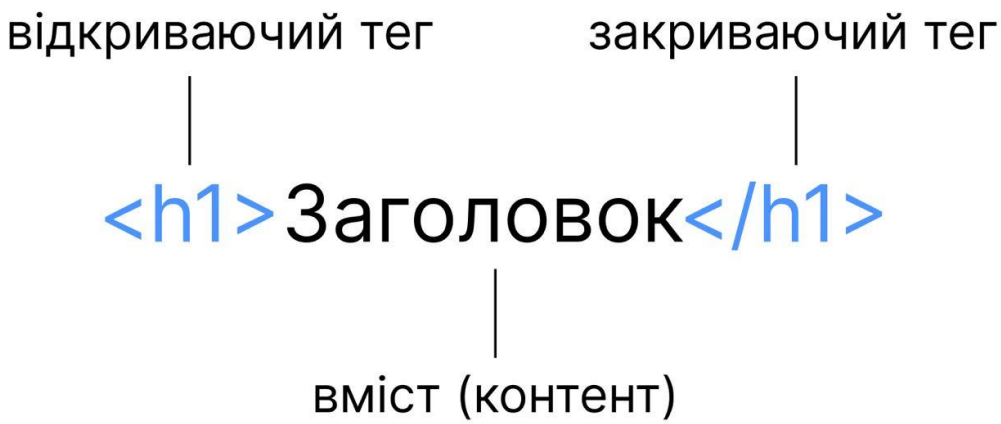

Рис. 2.15. Приклад структури дескриптора HTML

Деякі властивості HTML-елементів можна видозмінити або доповнити за допомогою атрибутів, що прописуються у тілі тегу. З метою класифікації та розділення елементів та групи за логікою існують атрибути class та id. Один елемент може мати декілька класів (їх поєднання формує ClassList, структура якого можна змінювати), а також їх можна визначити для інших елементів. Атрибут id має логіку унікального ідентифікатору, який можна визначити лише для одного елементу, при цьому їх не може бути декілька в рамках одного елементу (Рис. 2.16).

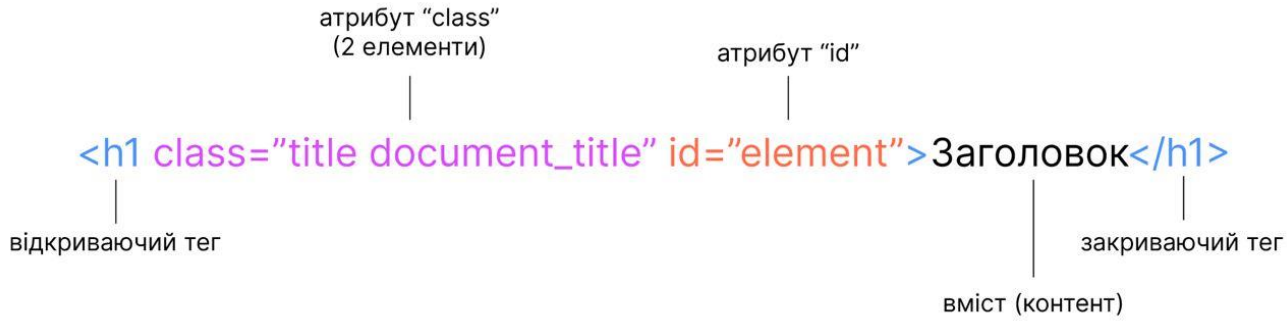

Рис. 2.16. Приклад присвоєння атрибутів для елементу

Така структура передбачає наявність вмісту всередині тегу, це може бути як текстовий контент, так і інші теги, їх групи, що у поєднанні створюють вкладену структуру. Вона формує загальний вигляд сторінки, логіку розміщення елементів, що необхідне не тільки для відображення, а й стилізації (Рис. 2.17).

Рис. 2.17. Приклад вкладеності тегів

У будові HTML-файлу присутні обов’язкові теги, які визначають ключові елементи сторінки та присутні у структурі всіх веб застосунків (табл. 2.1).

Таблиця 2.1

Структура файлу HTML

Елемент	Опис
<!DOCTYPE HTML>	Визначення версії HTML, у даному випадку позначення п’ятої версії (HTML 5)
<html></html>	Контейнер для вмісту веб сторінки

Продовження табл. 2.1

Елемент	Опис
<head></head>	Блок метаданих, містить в собі елементи, що не відображаються безпосередньо на сторінці. В цьому блоці відбувається підключення скриптів, шрифтів, стилів та інших файлів
<body></body>	Блок побудови HTML-розмітки, в якому розміщені всі елементи сторінки

Задання візуальних властивостей для елементів відбувається в окремому файлі – CSS. Це каскадні таблиці стилів, що описують правила відображення сторінки та її елементів. Структура файлу містить в собі набір окремих правил для кожного елементу або їх груп (класів). Синтаксис окремого правила передбачає ідентифікатор елементу або групи елементів та тіла правила, розміщеного у фігурних дужках. Кожна властивість відокремлена крапкою з комою, нове правило пишеться з нової строки. Саме правило складається з назви властивості та значення, які відокремлені двокрапкою. Описати правило для стилізації елементів можна описавши їх ідентифікатор у вигляді класу, id або системної назви. Перші два випадки передбачають написання додаткового символу для визначення ідентифікатору: символ крапки для класу та символ решітки для id (Рис. 2.18).

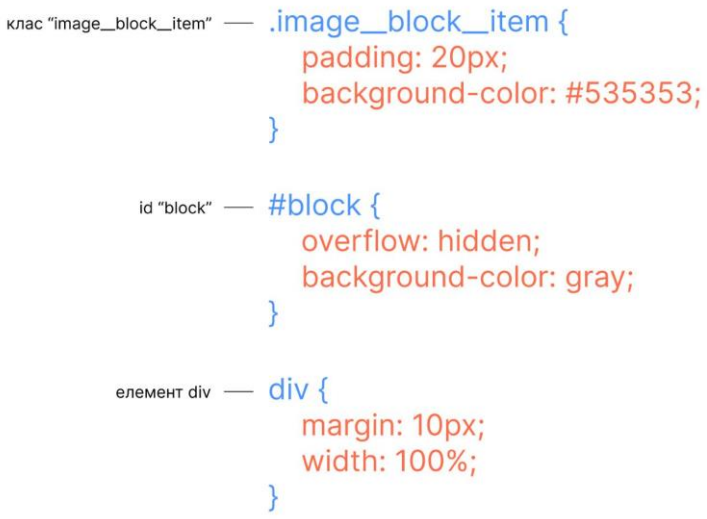

Рис. 2.18. Синтаксис CSS-правила в залежності від обраного методу ідентифікації елементу

Підключення CSS-файлу до потрібної розмітки здійснюється шляхом його опису у HTML-файлі у вигляді тегу link, що є дочірнім елементом тегу head, з атрибутами: rel=stylesheet та href=path, де path - це шлях до директорії розміщення CSS-файлу.

Сформована структура HTML-файлу являє собою набір DOM-елементів, які прийнято називати DOM-деревом. Інтегрування у розробку мов програмування дозволяє динамічно змінювати структуру цього DOM-дерева, формуючи потрібну структуру, а також створювати свої елементи, формувати їх контент та атрибути.

Для вирішення цієї задачі було обрано скриптову мову програмування Java Script. Вона була спроектована для опису сценаріїв, що виникають на сторінці. У створеному веб застосунку цими подіями є заповнення текстових полей, натискання на кнопки, перебування курсову миші на одному з елементів тощо. Дана технологія відповідає за динаміку та інтерактивність сторінки, має ряд визначних переваг, в порівнянні з іншими мовами програмування:

- відсутність попередньої компіляції, що дозволяє в реальному часі відображати зміни у коді;
- інструменти інтеграції з HTML та CSS тісно пов’язують ці технології, дозволяючи створювати повнофункціональне ПЗ;
- динамічна типізація оптимізує процес написання коду без необхідності попереднього визначення типу змінної;
- популярність серед розробників даної мови програмування і як наслідок наявність великої кількості бібліотек, фреймворків та надбудов, що пришвидшують роботу над програмою;
- кросплатформеність Java Script, себто підтримка цієї мови програмування на різних типах пристроїв, що є важливою властивістю для веб розробки, оскільки вона націлена саме на використання програмного продукту через різні пристрої;
- вбудована підтримка DOM. Для роботи з елементом HTML-файлу достатньо отримати його (або масив елементів) у змінну шляхом опису одного з

його ідентифікаторів. Отримані елементи можна повністю видозмінювати, переписуючи або додаючи CSS-властивості, атрибути, їх контент, обробляти і використовувати їх у функціях та повертати результат виконання у назад до HTML-файлу (Рис. 2.19);

– відкритий код, що надає розробникам широкий спектр можливостей модифікацій існуючих технологій та сценаріїв. Дана властивість також характерна мові програмування Python.

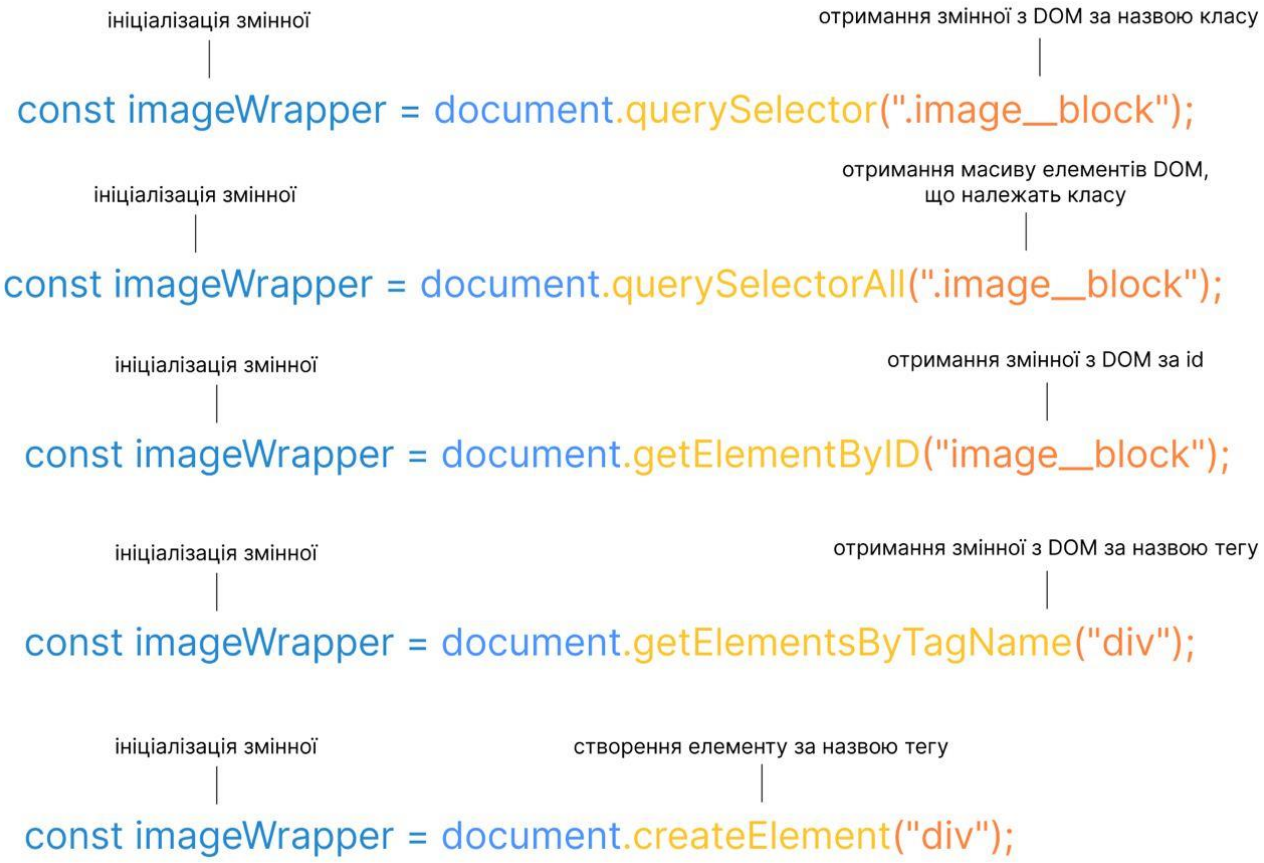

Рис. 2.19. Принцип роботу взаємодії з елементами HTML у Java Script

Вищезгадана мова програмування використана у веб додатку з метою інтеграції технологій штучного інтелекту. Це багатоплатформна інтерпретована об’єктно-орієнтовна мова програмування високого рівня з динамічною типізацією. Вона була обрана для створення, тренування та інтеграції нейронної мережі в проєкт з використанням технологій TensorFlow та Flask.

Бібліотека Tensorflow являє собою бібліотеку з відкритим кодом, створену для машинного навчання та штучного інтелекту. Її верша версія була опублікована компанією Google Brain 2015 року та продовжує оновлюватися, окрім Python наразі підтримується такими мовами програмування, як JavaScript, C++, Java. Бібліотека є багатоплатформною, підтримується на 64-бітних версіях операційних систем Windows, macOS та Linux, а також мобільних платформах Android та iOS. Алгоритми нейронних мереж передбачають обробку багатовимірних масивів даних, для цього у Tensorflow реалізовану можливість розгортки обчислень і таких платформах, як GPU, TPU та CPU. Це дозволяє пришвидшити процес обчислень та використовувати технічні можливості пристроїв максимально ефективно.

Фреймворк Flask відноситься до класу мікрофреймворків для створення веб застосунків мовою програмування Python. Розроблена Арміном Ронакеном у 2010 році технологія надає базовий інструментарій для веб розробки. Даний підхід дозволяє швидко опанувати загальні принципи використання фреймворку та впровадити його у програму, що особливо підходить для розробок невеликого масштабу, для яких існує потреба організації веб середовища.

Серед основних етапів розробки можна виділити наступні:

1) Створення нейронної мережі. У проєкт інтегруються фреймворки та бібліотеки, обраний dataset. Для мережі описуються параметри, надаються зображення, правильні відповіді та визначається кількість епох, себто кількість ітерацій циклу навчання. Під час цього циклу є можливість відображення основних показників, таких як коефіцієнт втрат, точності визначення (Рис. 2.20). Вищеописані параметри використовуються для запобігання “перенавчанню”. Цей процес описує таку кількість епох (Рис. 2.21), за яких нейронна мережа здатна розпізнавати лише тренувальні дані, при цьому зменшуючи коефіцієнт точності визначення валідаційного масиву значень (Рис. 2.22). У зв’язку з цим важливим є вибір кількості епох, що забезпечують коефіцієнт кількості втрат, максимально близький до одиниці.

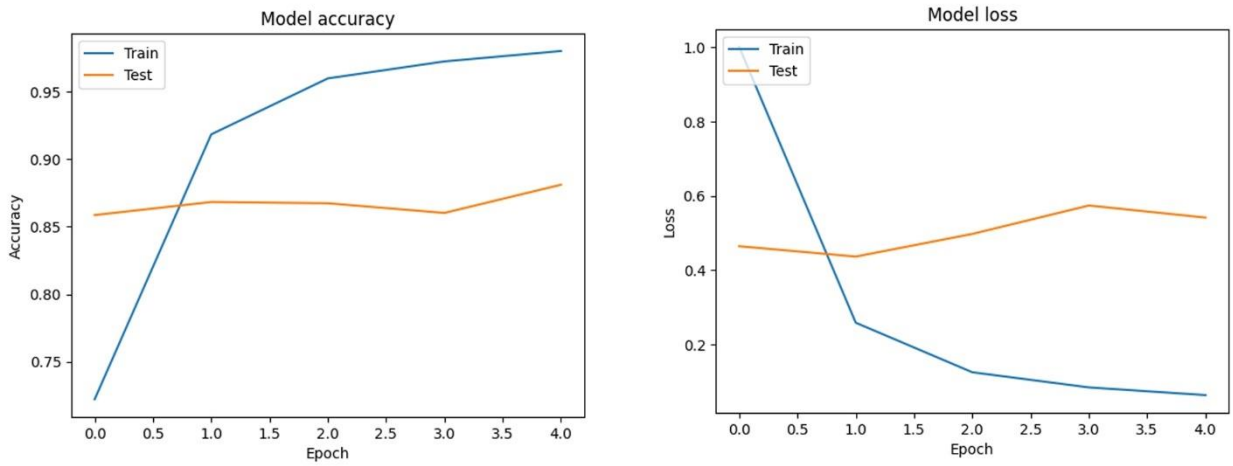

Рис. 2.20. Графіки ефективності навчання нейронної мережі

```
Epoch 1/5
1358/1358 [==============================] - 113s 79ms/step - loss: 1.0004 - accuracy: 0.7220 - val_loss: 0.4642 - val_accuracy: 0.8587
Epoch 2/5
1358/1358 [==============================] - 106s 78ms/step - loss: 0.2585 - accuracy: 0.9186 - val_loss: 0.4363 - val_accuracy: 0.8684
Epoch 3/5
1358/1358 [==============================] - 102s 75ms/step - loss: 0.1252 - accuracy: 0.9601 - val_loss: 0.4972 - val_accuracy: 0.8674
Epoch 4/5
1358/1358 [==============================] - 109s 80ms/step - loss: 0.0844 - accuracy: 0.9727 - val_loss: 0.5737 - val_accuracy: 0.8603
Epoch 5/5
1358/1358 [==============================] - 107s 79ms/step - loss: 0.0637 - accuracy: 0.9804 - val_loss: 0.5412 - val_accuracy: 0.8812
```

Рис. 2.21. Процес тренування нейронної мережі та змінні ефективності

```
Evaluating model...
339/339 [==============================] - 20s 59ms/step - loss: 0.5412 - accuracy: 0.8812
Validation Accuracy: 88.12%
```

Рис. 2.22. Розпізнавання валідаційного масиву даних

2) Експорт файлів. Результатом тренування нейронної мережі є модель розпізнавання, найефективніша з них зберігається у файл, який можна використовувати у інших локальних проєктах. Окремо зберігається текстовий масив значень, що відповідають образам;

3) Створення віртуального середовища Python. Для окремих проєктів прийнято встановлювати локальну версію Python у віртуальному середовищі. Робиться це з метою відокремлення встановлених бібліотек задля запобігання конфлікту версій як самого Python, так і встановлених файлів. Таке середовище працює незалежно та має свій набір встановлених файлів (Рис. 2.23).

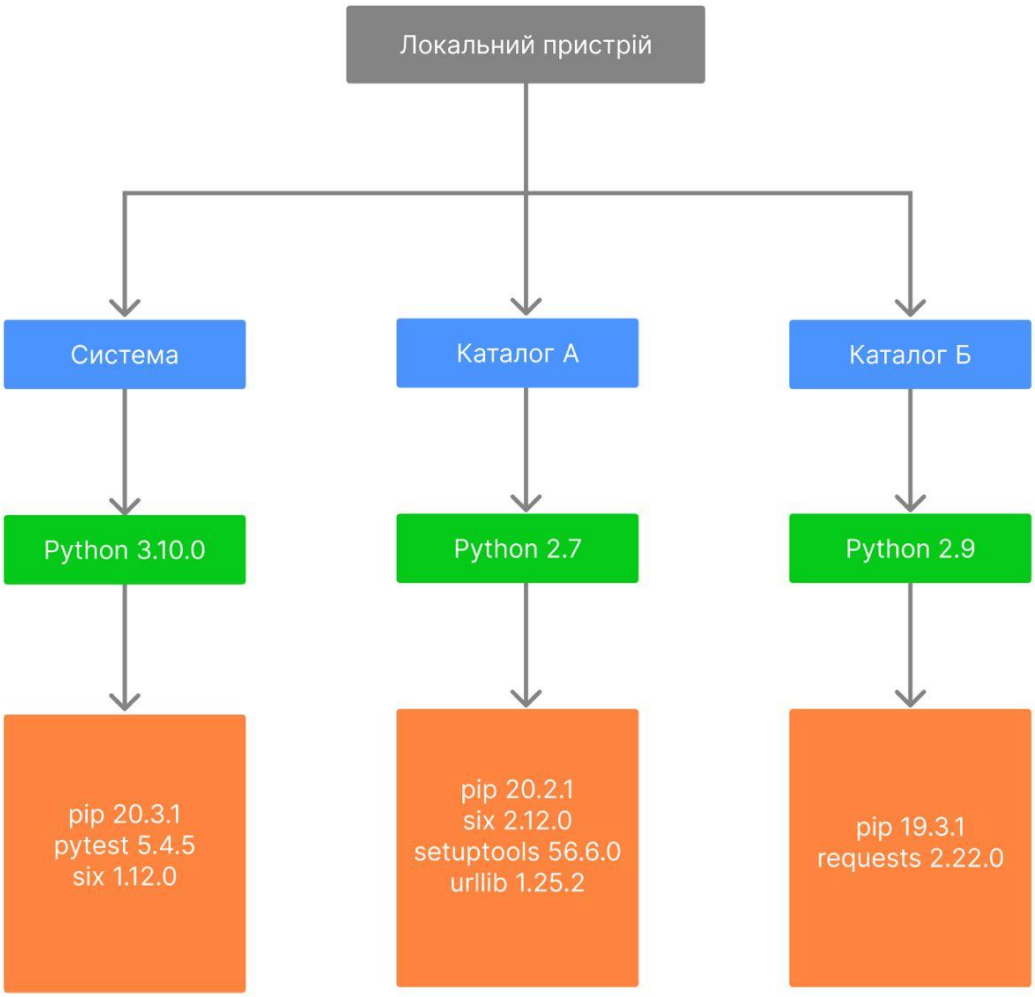

Рис. 2.23. Відокремлені віртуальні оточення для Python

Ініціалізація віртуального середовища відбувається у директорії програми шляхом створення директорії venv, що містить в собі файли Python (Рис. 2.24). Для виконання дій з локальною версією відбувається активація віртуального середовища (Рис. 2.25).

```
PS C:\Users\User\Desktop\AVGO VISION +ph> python -m venv venv
```

Рис. 2.24. Створення директорії віртуального середовища у каталозі застосунку

```
PS C:\Users\User\Desktop\AVGO VISION +ph> .\venv\Scripts\Activate
(venv) PS C:\Users\User\Desktop\AVGO VISION +ph>
```

Рис. 2.25. Активація віртуального середовища

4) Імпорт бібліотек та фреймворків. Для окремо створеного віртуального середовища встановлюються необхідні версії бібліотек, версії який мають бути сумісними. Прийнято створювати окремий текстовий файл requirements.txt з використаними версіями з метою їх швидкого імпорту на іншому пристрої (табл. 2.2).

Таблиця 2.2

Бібліотеки та фреймворки

Назва	Версія
TensorFlow	2.16.1
Flask	3.0.3
Numpy	1.26.4
Pillow	10.3.0

5) Імпорт файлів у локальний проєкт веб застосунку. Завантажена модель та список класів зберігаються у директорії та використовується Python-файлом. На цьому етапі формується передбачення нейронної мережі за правилами розпізнавання імпортованої моделі;

6) Відображення результату обробки нейронною мережею на сторінці. Функція повертає результат у вигляді класу-відповідника та передає його у підключену HTML-розмітку в елемент DOM.

7) Відображення елементів керування та завантаженого користувачем зображення. Після завантаження відправленого зображення формується інтерфейс користувача з елементами, що дозволяють використовувати методи обробки, а також малювати графічні елементи на зображенні з можливістю завантажити оброблене зображення. До кінцевої назви завантаженого зображення додається назва веб застосунку – Agro Vision.

2.5 Опис структури програми та алгоритми її функціонування

Стани інтерфейсу веб застосунку розділені на два стани: до завантаження зображення та після. На сторінці розміщений логотип сайту та поле завантаження, що розташоване по центру екранного середовища. Цей елемент доповнений текстовим супроводом про можливість завантаження файлу шляхом його переміщення до активної зони елементу та окремою кнопкою для стандартного вибору зображення з файлової системи.

Таку технологію прийнято називати Drag & Drop, а самі поле виділяти пунктирним обведенням з метою демонстрації того, що елемент можна доповнити файлом. При цьому в момент перебування файлу в зоні дії події завантаження обведення змінюється на суцільне, а текстовий блок змінює інформацію (Рис. 2.26).

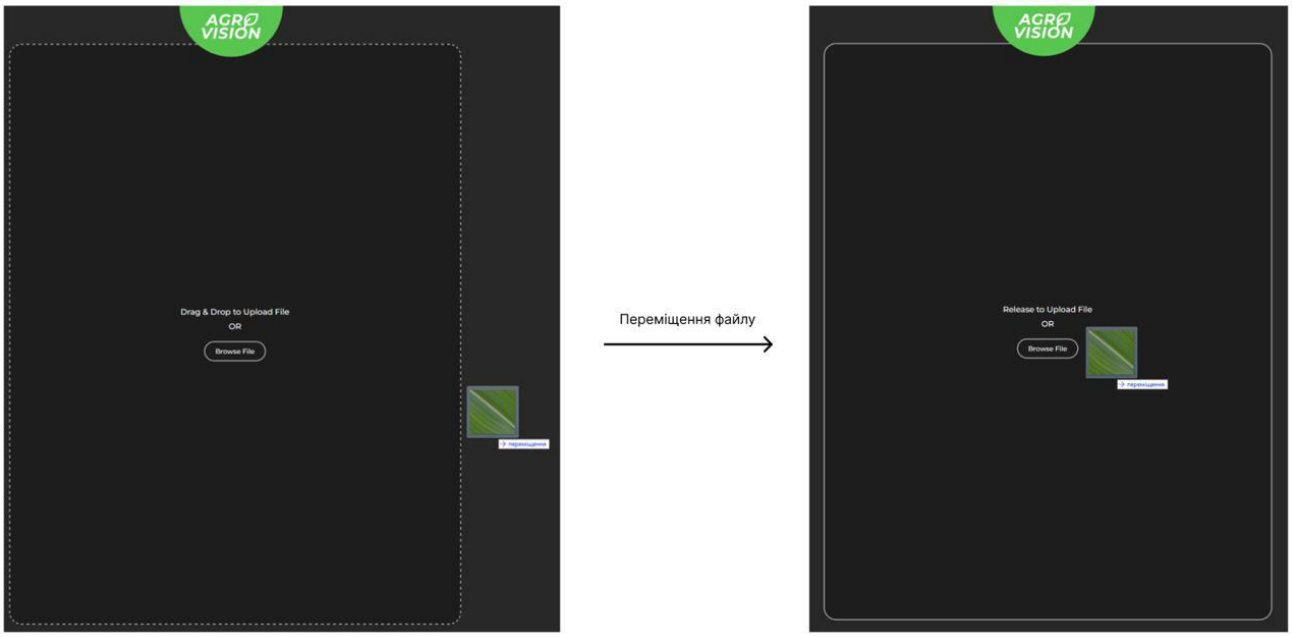

Рис. 2.26. Відображення станів блоку до та після переміщення

Після переміщення та завантаження зображення відбувається зміна структури DOM-дерева. Блок завантаження змінює свій вміст: поле завантаження файлу стає неактивним, текст з кнопкою та вибором файлу не

відображається, додається суцільне обведення до блоку. Сам блок при цьому доповнюється новими елементами:

1) Текстове поле, в якому відображається назва завантаженого на сторінку файлу з можливістю вводу нової назви, яка відображатиметься на заголовку сторінки та автоматично змінюватиметься під час введення (Рис. 2.27).

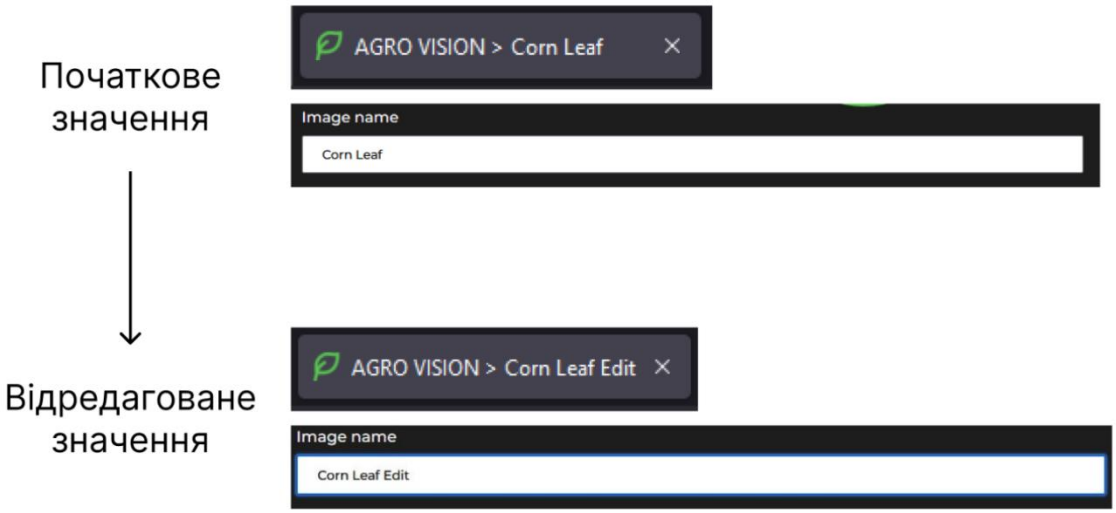

Рис. 2.27. Зміна назви заголовку сторінки

2) Зображення, на якому в реальному часті відображаються внесені зміни.

3) Блок з результатами розпізнавання нейронної мережі, в якому розміщена назва рослини, її стан та прапорець вибору, активувавши який зміниться значення назви файлу і заголовку сторінки. Деактивація прапорця поверне минуле значення назви файлу.

4) Кнопка закриття зображення. Після активації відбувається повернення до початкового вигляду сторінки, поле завантаження файлу знову стає активним.

5) Кнопка завантаження зображення. Після внесення змін відредаговане зображення разом із внесеними зміна завантажується на пристрій, зберігаючи при цьому початковим розмір без втрати якості. До імені файлу на початку додається назва «AGRO VISION».

Після завантаження на сторінку зображення також стає активним блок інструментів редагування та його елементи:

1) Фільтри. Це блоки, у складі яких є назва фільтру, повзунок налаштування та текстове поле з числовим показником. Змінити інтенсивність фільтру можна не тільки повзунком, а я вписавши бажане значення у текстове поле, при цьому зміниться положення маркеру на відповідну величину.

2) Блок трансформацій. В ньому розміщені кнопки, що відповідають за віддзеркалення зображення за віссю абсцис та ординат, а також повороту на 90° за або проти годинникової стрілки.

3) Блок малювання. В ньому розміщені перемикачі кольору та текстове поле введення розміру пензля в діапазоні 1-30 пікселів (за замовчуванням – 5 пікселів). Режим малювання активується тільки тоді, коли обрано колір. При цьому вибір можна відмінити, зробивши всі перемикачі неактивними, що є модифікацією стандартного функціоналу, в якому вибір можна тільки змінити.

4) Кнопка повернення до значень за замовчуванням. Після натискання всі значення фільтрів та трансформації повертаються до початкових, також стираються всі намальовані елементи.

2.6 Обґрунтування та організація вхідних та вихідних даних

До вхідних даних створеного веб застосунку входять:

– зображення. Після вибору користувачем та завантаження воно конвертується у Data URL. Синтаксис Data URL складається з приставки data:, MIME-типу (Multipurpose Internet Mail Extensions), що описує формат файлу у стандартизованомуц форматі, приставки base64 та даних зображення, конвертованих у тектовий формат. Такий підхід дозволяє зменшити кількість HTTP-запитів та інтегрувати зображення безпосередньо у DOM-дерево сторінки;

– текстові поля. Інформація з них використовується для опису внесених у зображення змін, їх текстове представлення. Досутп до них

виконується з клавіатури (екранної у випадку, якщо пристрій перегляду це смартфон або фізичної периферії у випадку з ПК);

- поля вибору числового значення. Ці елементи мають визначений проміжок, в межах якого обирається значення інтенсивності обраного фільтру;
- кнопки внесення трансформацій. Взаємодія з ними впливає на значення змінних, що будуть застосовані до зображення;
- кнопки вибору кольору. Дані поля відповідають за кінцеві візуальні властивості графічних елементів, що будуть відображені поверх зображення.

До вихідних даних можна віднести:

- завантажене зображення. Обрані у застосунку візуальні властивості інтерпретуються у новий файл;
- реакції інтерфейсу на події сторінки. Це натискання кнопок, зміна розмірів вікна перегляду, зміна розміщення елементів;
- відображення дії фільтрів. Застосовані зміни у реальному часі відображаються на сторінці шляхом видозмінення вхідного зображення, використовуючи для цього обрані числові дані;
- відображення графічних елементів. Обрані значення розміру лінії та її кольору визначають кінцевий вигляд графічних елементів, розміщених на зображенні.

2.7 Опис розробленого програмного продукту

2.7.1. Використані технічні засоби

Розробку веб застосунку було виконано з використанням персонального комп’ютеру (ПК) з наступними характеристиками комплектуючих:

- процесор AMD Ryzen 5 5600G, 4400MHz;
- інтегрована графіка Radeon Vega 7 Graphics;
- SSD-диск Kingston SSDNow A400 TLC 480GB;
- жорсткий диск Western Digital Blue 1TB;

- материнская плата ASRock B550M-HDV;
- оперативна пам’ять (ОЗП) HyperX DDR4 32GB (2x16GB) 3200Mhz Predator;
- блок живлення Be Quiet! System Power 9 600W CM.

Периферійні пристрої:

- монітор BenQ PD2700Q 27" QHD (2560 x 1440);
- комп’ютерна мина Logitech G502 Hero;
- клавіатура HATOR Rockfall EVO TKL;

2.7.2. Використані програмні засоби

Для написання коду веб застосунку з можливістю підключення віртуального середовища було обрано Visual Studio Code. Це безкоштовне програмне забезпечення від компанії Microsoft з широким спектром функціоналу та підтримуваних технологій, перша версія якого була випущена 2015 року. Редактор надає доступ до каталогів проєктів, має систему контролю версій, вбудований термінал, додаткові розширення (VS Code Marketplace), можливість персоналізації, підсвічування синтаксису більшості мов програмування тощо.

Для створення дизайну інтерфейсу було використано застосунок Adobe Illusrtator (Рис. 2.28) від компанії Adobe Systems, перша версія якого була випущена 1987 року. Це графічний редактор векторної графіки, націлений на професійне використання графічними дизайнерами. Серед переваг застосунку можна виділити:

- робота з шарами та масками для свторення елементів, що довзоляє на будь-якому етапі відредагувати минулі операції;
- малювання пензлем та його автоматична конвертація у векторну графіку;
- підтрика багатоьх векторних форматів;
- форматування тексту та ефекти;
- трансформація растрових зображень;

– можливість повного налаштування робочого середовища.

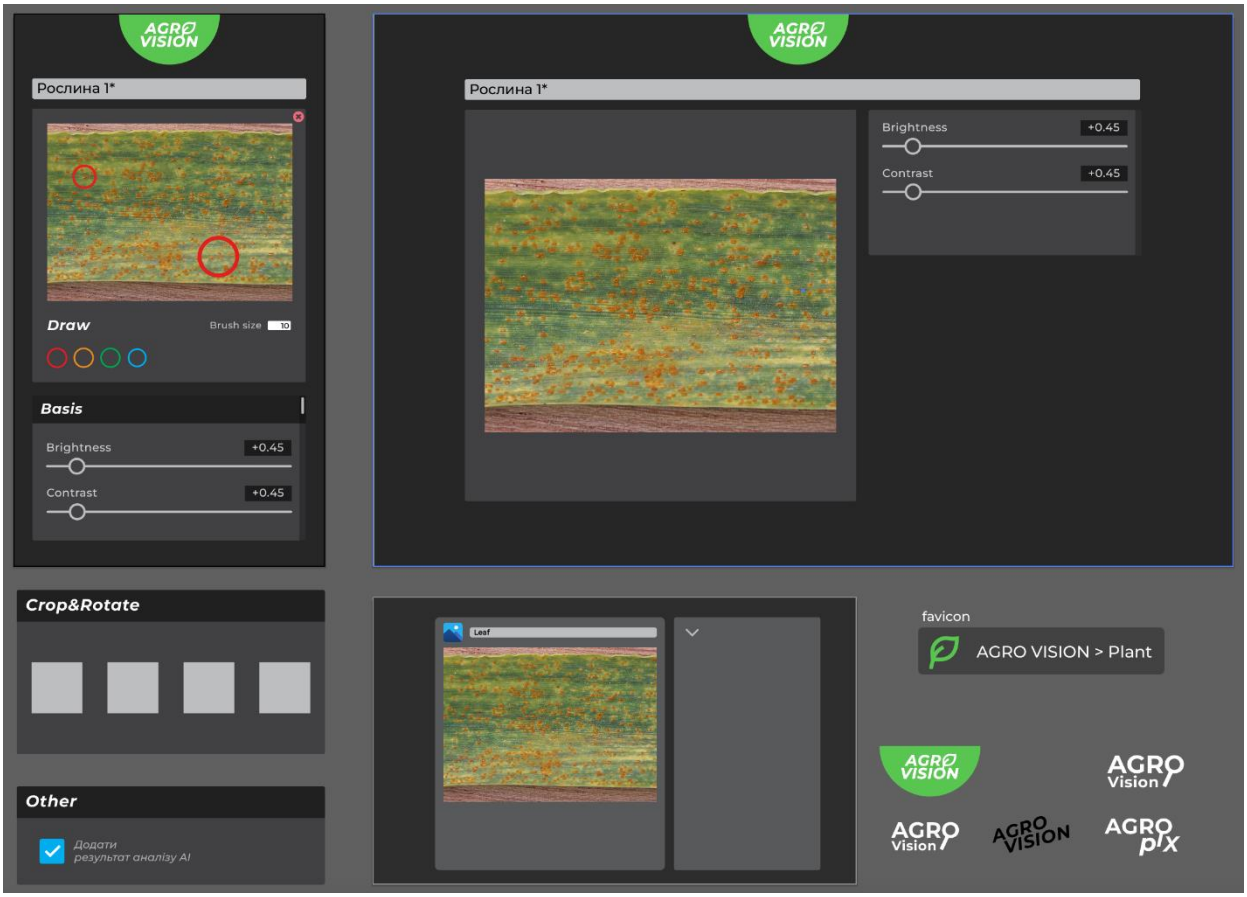

Рис. 2.28. Дизайн елементів інтерфейсу, створений у програмі Adobe Illustrator

2.7.3. Виклик та завантаження програми

За умови встановлення каталогу проєкту на локальний пристрій виклик та завнтаження програми відбувається через командний термінал з активованим venv-віртуальним середовищем. Після активації потрбіно запустити python-файл відповідною командою командою (Рис. 2.29).

```
PS C:\Users\User\Desktop\AVGO VISION +ph> .\venv\Scripts\Activate
(venv) PS C:\Users\User\Desktop\AVGO VISION +ph> python app.py
```

Рис. 2.29. Запуск веб застосунку

Після активації команди відбувається запуск локального серверу Flask та формується локальне посилання (Рис. 2.30) на сформовану HTML-сторінку (Рис. 2.31).

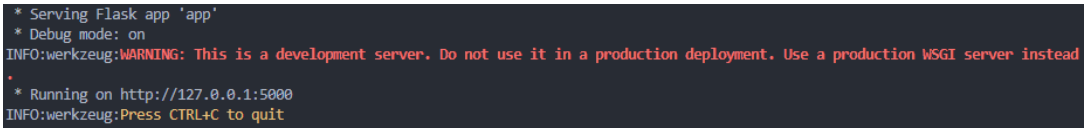

Рис. 2.30. Візуалізація запуску локального серверу

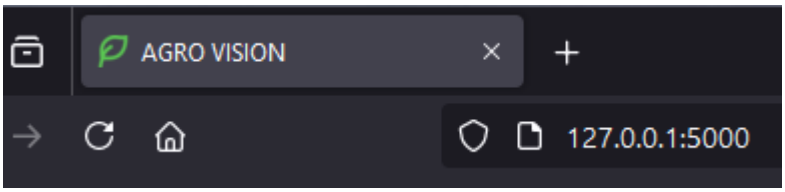

Рис. 2.31. Відкритий у браузері застосунок

Зупинка серверу і зупинка роботи програми відбуається ініціалізацію команди у терміналі поєднанням клавіш Ctrl + C.

За умови публікації проєкту на домені досуп до застосунку можна отримати без завантаження директорії. Для цього у браузері у відповідному поллі вводиться URL-посилання. Після цього відбувається стандартна процедура завантаження.

2.7.4. Опис інтерфейсу користувача

У веб застосунку реалізовано стандартний інтерфейс графічного редактору із демонстрацією зображення та елементів керування. До завантаження файлу зайві блоки приховуються, а після закриття зображення повертаються до початкового вигляду.

На початку запуску застосунку у браузері користовачу відображається головне меню (Рис. 2.32) із можливістю вибору зображення. Після завантаження обраного файлу на сторінці з’являється два блоки: дані зображення та інструменти керування (Рис. 2.33).

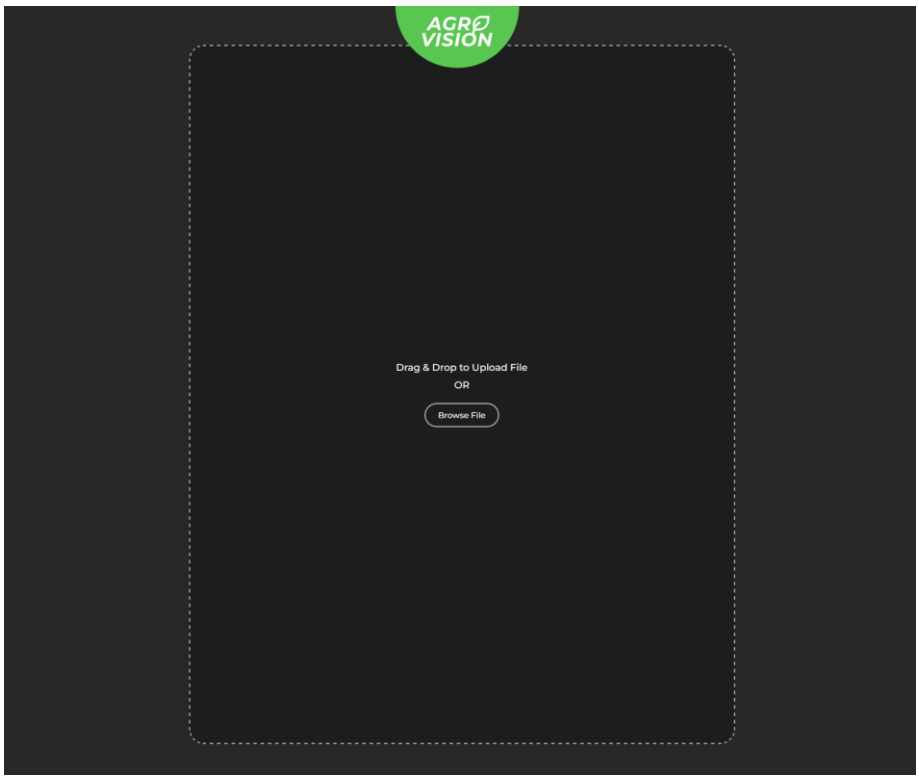

Рис 2.32. Головне меню застосунку до завантаження зображення

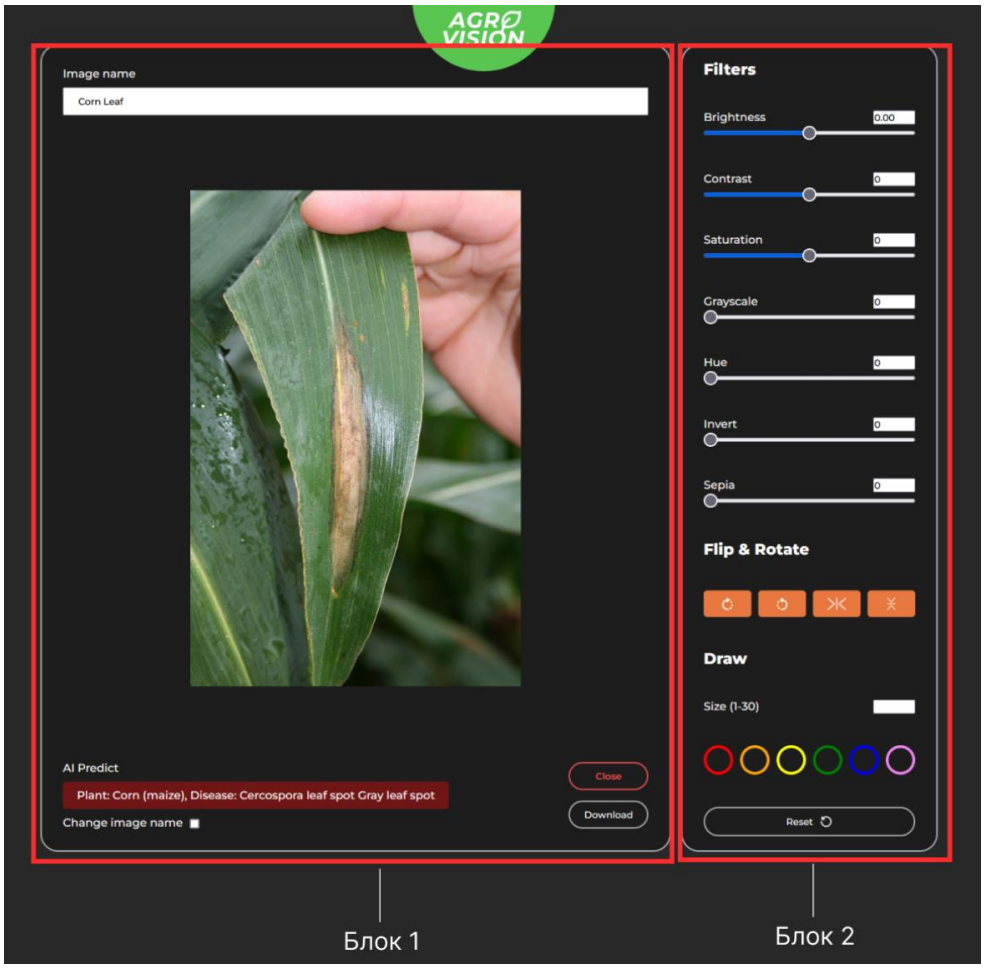

Рис. 2.33. Структура веб застосунку, розділена на два елементи

У лівому блоці розміщене зображення та інструменти взаємодії:

– назва файлу, релізована у форматі текстового поля, яке займає всю ширину блоку для можливості відображення великої назви. Такий підхід дозволяє змінювати назву файлу (Рис. 2.34) з клавіатури без необхідності додаткового налаштування даних у файловому провіднику пристрою (Рис. 2.35);

Рис. 2.34. Властивості зображення до та після зміни текстового поля

Рис. 2.35. Блок назви зображення

– результат аналізу нейронною мережею. Оформлений у вигляді прямокутнику з текстом, колір якого залежить від результатів передбачення (зелений колір для здорової рослини (Рис. 2.36) і червоний для хворої) (Рис. 2.37). У блоці наявний прапорець, активувавши який початкова назва файлу зміниться на новий з висновком нейронної мережі (Рис. 2.38). Ці елементи супроводжуються пояснювальним текстом;

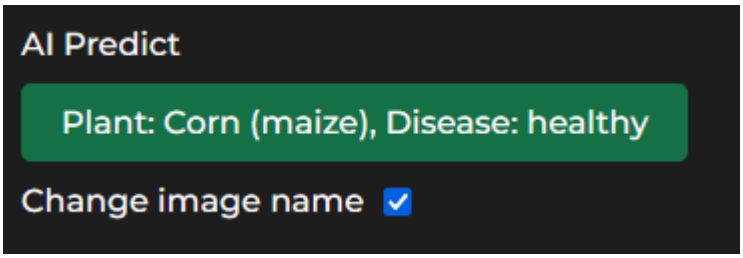

Рис. 2.36. Блок із висновками нейронної мережі для здорового зразку

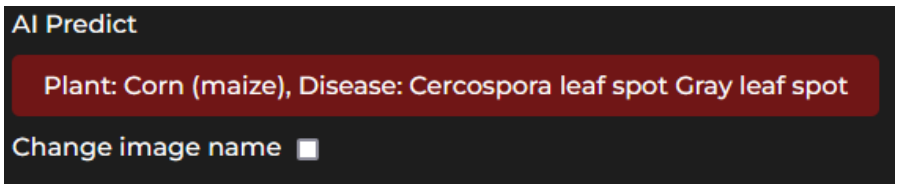

Рис. 2.37. Блок із висновками нейронної мережі для здорового зразку

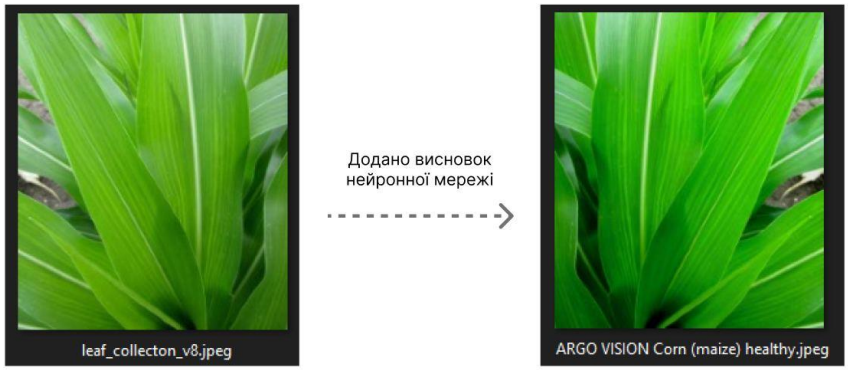

Рис. 2.38. Зміна назви файлу з результатом обробки нейронною мережею

– кнопки завантаження зображення. Інтерфейси графічних редакторів передбачають завантаження відредагованого зображення, у застосунку ця функція реалізована кнопкою завантаження файлу з відповідним текстом. Назвою нового завантажуваного файлу є значення текстовго поля назви файлу з приставкою «AGRO VISION >» на її початку (Рис. 2.39);

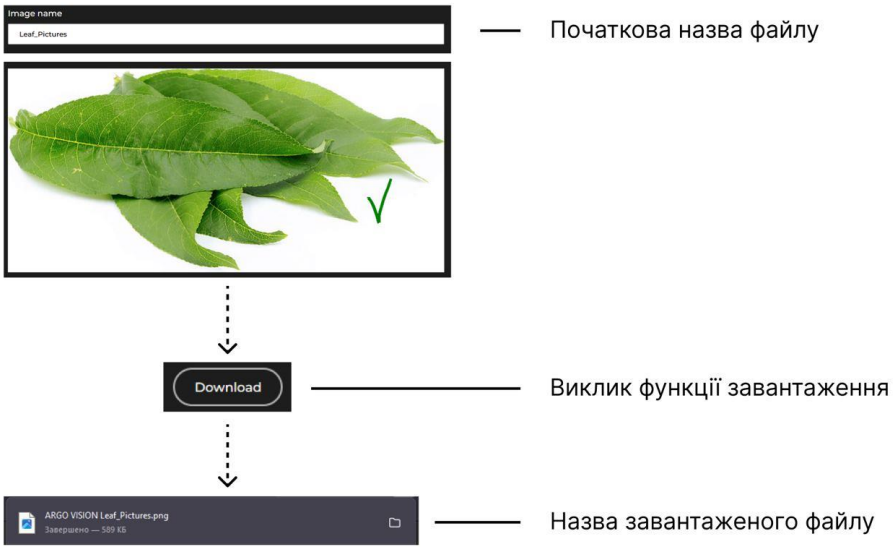

Рис. 2.39. Зміна назви файлу після завантаження

– кнопка закриття зображення. Натискання на елемент повертає сторінку до початкового вигляду (Рис. 2.40).

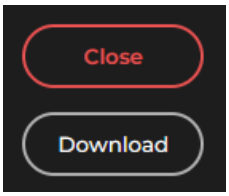

Рис. 2.40. Кнопки лівого блоку

У правому блоці розміщені наступні елементи:

– блок фільтрів. Функціонал цього блоку передбачає наявність текстовго супроводу, елементу вибору значення в межах діапазону та поля виводу обраного значення. Такий тип інтерфейсу реалізований у додатку Adobe Photoshop. Також реалізовано можливість зміни значення з клавіатури у відповідного для фільтру полі, яке змінить положення основного елементу. До зображення застосовуються CSS-фільтри, окрему дію яких імінтовано у застосунку (Рис. 2.41);

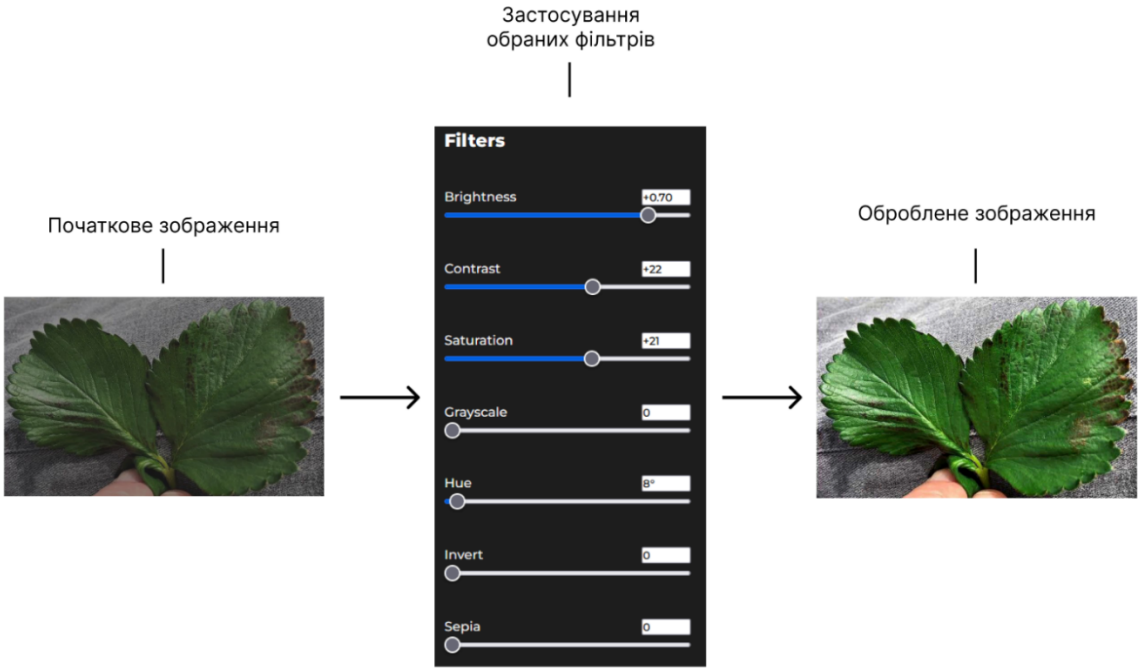

Рис. 2.41. Процес обробки зображення

– блок трансформацій. В ньому розміщені 4 кнопки (віддзеркалення по горизонталі та вертикалі (Рис. 2.42), поворот на 90° за або проти годинникової стрілки), всередині яких знаходяться графічні елементи, демонструючі поведінку зображення після натиснення на них (Рис. 2.43);

Рис. 2.42. Приклад віддзеркаленого по горизонталі зображення

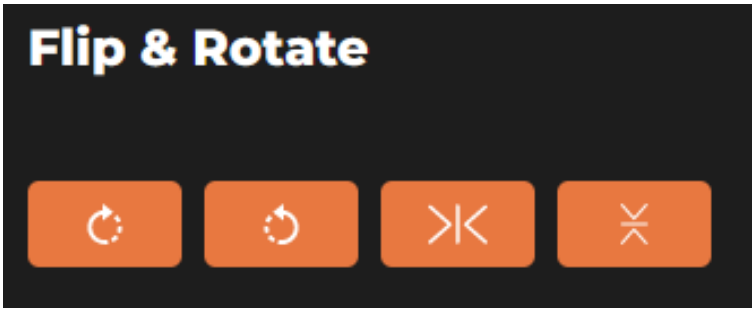

Рис. 2.43. Блок трансформацій зображення

– блок малювання. Цей елемент складається з текстового поля та блоків палітри кольорів. Введене у поле значення змінює товщину пензля для малювання, а вибір кольору змінює візуальні властивості створених графічних елементів (Рис. 2.44). Малювання лінії здійснюється під час одночасного виконання трьох умов: активовано один з кольорів, курсор миші перебуває на зображенні, виконано клік (Рис. 2.45);

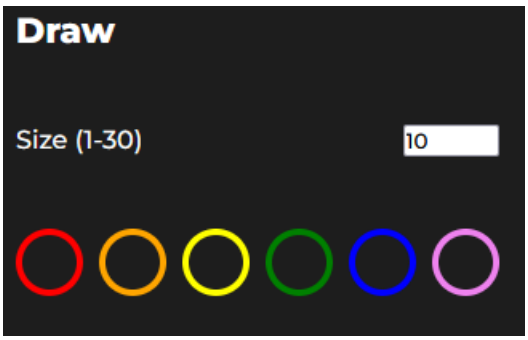

Рис. 2.44. Блок малювання

Рис. 2.45. Приклад намальованого на зображенні елементу

– кнопка відміни внесених змін. Даний елемент повертає відредаговане зображення до початкового вигляду, при цьому до значень за замовчуванням повертаються поля фільтрів (Рис. 2.46).

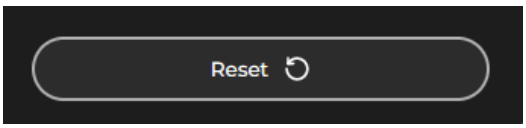

Рис. 2.46. Кнопка відміни внесених змін

РОЗДІЛ 3
ЕКОНОМІЧНИЙ РОЗДІЛ

3.1 Розрахунок трудомісткості та вартості розробки програмного продукту

Початкові дані:

1. q – передбачуване число операторів програми – 1219;
2. C – коефіцієнт складності програми – 1,35;
3. p – коефіцієнт корекції програми в ході її розробки – 0,2;
4. $C_{пр}$ – годинна заробітна плата програміста – 231 грн/год;

Згідно статистики (за даними DOU.ua) середня величина рівню заробітної плати Full-Stack розробника рівня Junior становить 1000$ станом на кінець 2023 року. За даними НБУ на початок червня 2024 року вартість одного долару США становить 40,65 грн, тож в гривнях це значення становитиме 40,650 грн. Беручи значення робочого графіку 176 годин/місяць, отримуємо значення годинної заробітної плати – 231 грн.

5. B – коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі – 1,3;
6. k – коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності – 0,8;
7. $C_{мв}$ – вартість машино-години ЕОМ – 5,68 грн/год.
8. B_k – число виконавців – 1 людина;
9. F_p – місячний фонд робочого часу (при 40-годинному робочому тижні) – 176 годин.

Вихідними даними дял прорахунку вартості машино-годин є витрати на електроенергію, доступ до мережі Інтернет, вартість амортизації пристрою.

Для розрахунку вартості амортизації використовуються показники споживання електроенергії, більшість ресурсу яких використовуються процесором та монітором використовуваної ЕОМ.

Величина показової потужності процесора становить 65 Вт (разом з інтегрованою графікою). Для визначення енергоспоживання монітору використаємо середнє арифметичне показників нормального та максимального споживання – 43 Вт. Тож, середній рівень споживання ЕОМ становить 108 Вт.

Станом на початок червня 2024 року тариф на електроенергію (за даними холдингу Yasno) становить 4,32 грн / кВт · год.

Розрахунок ціни на годину:

$$4{,}32 * 0{,}108 \text{ кВт} = 0{,}47 \text{ грн/год.}$$

Вартість пакету підключення до інтренту на місяць (від компанії Київстар, пакет «Інтернет.Оберіг») становить 150 грн.

Розрахунок ціни на годину:

$$150 / 176 \approx 0{,}85 \text{ грн/год}$$

Погодинна амортизація ЕОМ за умови початкової вартості 36800 грн, (з яких 21800 грн становить вартість комплектуючих, а 15000 грн – монтіору), терміну корисного використання – 3 роки прямолінійним методом розраховується відповідно:

$$36800 \text{ грн} / (12 \text{ місяців} * 4 \text{ роки} * 176 \text{ годин}) \approx 4{,}36 \text{ грн/год.}$$

Тож, сумарна вартість машино-годин ЕОМ становить:

$$0{,}47 + 0{,}85 + 4{,}36 = 5{,}68 \text{ грн/год.}$$

Нормування праці в процесі створення ПЗ істотно ускладнено в силу творчого характеру праці програміста. Тому трудомісткість розробки ПЗ може бути розрахована на основі системи моделей з різною точністю оцінки.

Трудомісткість розробки ПЗ можна розрахувати за формулою:

$$t = t_o + t_u + t_a + t_n + t_{отл} + t_д \text{, людино-годин,} \qquad (3.1)$$

де t_o - витрати праці на підготовку й опис поставленої задачі (приймається 50 людино-годин);

t_u - витрати праці на дослідження алгоритму рішення задачі;

t_a - витрати праці на розробку блок-схеми алгоритму;

t_n - витрати праці на програмування по готовій блок-схемі;

$t_{отл}$ - витрати праці на налагодження програми на ЕОМ;

$t_д$ - витрати праці на підготовку документації.

Складові витрати праці визначаються через умовне число операторів у програмному забезпеченні, яке розробляється.

Умовне число операторів:

$$Q = q \cdot C \cdot (1 + p) \qquad (3.2)$$

де q - передбачуване число операторів (1219);

C - коефіцієнт складності програми (1,35);

p - коефіцієнт корекції програми в ході її розробки (0,2).

Отже, умовне число операторів в програмі має значення:

$$Q = 1219 * 1,35 * (1 + 0,2) = 1975$$

Витрати праці на вивчення опису задачі t_u визначається з урахуванням уточнення опису і кваліфікації програміста:

$$t_u = \frac{Q \cdot B}{(75..85) \cdot k} \text{ , людино-годин} \qquad (3.3)$$

де B - коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі;

k - коефіцієнт кваліфікації програміста, обумовлений стажем роботи у даній сфері. Для періоду до 2 років він складає 0,8.

З урахуванням коефіцієнта кваліфікації k = 0,8, отримуємо витрати праці на вивчення опису завдання:

$$t_u = (1975 * 1,3) / (75 * 0,8) = 42,79 \text{ людино-годин.}$$

Витрати праці на розробку алгоритму рішення задачі визначаються за формулою:

$$t_a = \frac{Q}{(20\ldots25) \cdot k} \text{ , людино-годин} \qquad (3.4)$$

де Q – умовне число операторів програми;

k – коефіцієнт кваліфікації програміста.

Підставивши відповідні значення в формулу (3.4), отримуємо:

$$t_a = 1975 / (25 * 0,8) = 123,44.$$

Витрати на складання програми по готовій блок-схемі:

$$t_n = \frac{Q}{(20\ldots25) \cdot k} \text{ , людино-годин} \qquad (3.5)$$

Підставимо значення у формулу (3.5) та отримаємо витрати на складання програми по готовій блок-схемі:

$$t_n = 1975 / (20 * 0{,}8) = 123{,}44 \text{ людино-годин.}$$

Витрати праці на налагодження програми на ЕОМ:

- за умови автономного налагодження одного завдання:

$$t_{отл} = \frac{Q}{(4..5) \cdot k} \text{ , людино-годин} \qquad (3.6)$$

$$t_{отл} = 1975 / (4 * 0{,}8) = 617{,}19 \text{ людино-годин.}$$

- за умови комплексного налагодження завдання:

$$t_{отл}^{k} = 1{,}5 \cdot t_{отл} \text{ , людино-годин} \qquad (3.7)$$

$$t_{отл}^{k} = 1{,}5 * 617{,}19 = 925{,}78 \text{ людино-годин.}$$

Витрати праці на підготовку документації визначаються за формулою:

$$t_д = t_{др} + t_{до} \text{ , людино-годин} \qquad (3.8)$$

де $t_{др}$-трудомісткість підготовки матеріалів і рукопису:

$$t_{др} = \frac{Q}{(15..20) \cdot k} \text{ , людино-годин} \qquad (3.9)$$

$t_{до}$ - трудомісткість редагування, печатки й оформлення документації:

$$t_{до} = 0{,}75 \cdot t_{др} \text{ , людино-годин.} \quad (3.9)$$

Підставляючи відповідні значення, отримаємо:

$$t_{др} = 1975 / (15 * 0{,}8) = 164{,}58 \text{ людино-годин.}$$

$$t_{до} = 0{,}75 * 164{,}58 = 123{,}43 \text{ людино-годин.}$$

$$t_{д} = 164{,}58 + 123{,}43 = 288{,}01 \text{ людино-годин.}$$

Повертаючись до формули (3.1), отримаємо повну оцінку трудомісткості розробки програмного забезпечення:

$$t = 50 + 42{,}79 + 123{,}44 + 123{,}44 + 617{,}19 + 288{,}01 = 1244{,}87 \text{ людино-годин.}$$

3.2. Розрахунок витрат на створення програми

Витрати на створення ПЗ $К_{ПО}$ включають витрати на заробітну плату виконавця програми $З_{ЗП}$ і витрат машинного часу, необхідного на налагодження програми на ЕОМ:

$$К_{ПО} = З_{ЗП} + З_{МВ}, \text{ грн} \quad (3.10)$$

Заробітна плата виконавців визначається за формулою:

$$З_{ЗП} = t \cdot C_{ПР}, \text{ грн} \qquad (3.11)$$

де: t - загальна трудомісткість, людино-годин;

$C_{ПР}$ - середня годинна заробітна плата програміста, грн/година

З урахуванням того, що середня годинна зарплата програміста становить 231 грн / год, отримуємо:

$$З_{ЗП} = 1244,87 * 231 = 287564,97 \text{ грн.}$$

Вартість машинного часу, необхідного для налагодження програми на ЕОМ, визначається за формулою:

$$З_{мв} = t_{отл} \cdot C_{мч}, \text{ грн,} \qquad (3.12)$$

де $t_{отл}$ - трудомісткість налагодження програми на ЕОМ, год;

$C_{мч}$ - вартість машино-години ЕОМ, грн/год (5,68 грн/год).

Підставивши в формулу (3.3) відповідні значення, визначимо вартість необхідного для налагодження машинного часу:

$$З_{мв} = 617,19 * 5,68 = 3505,64 \text{ грн.}$$

Звідси витрати на створення програмного продукту:

$$К_{ПО} = 287564,97 + 3505,64 = 291070,61 \text{ грн.}$$

Очікуваний період створення ПЗ:

$$T = \frac{t}{B_k \cdot F_p}, \text{ міс.} \qquad (3.13)$$

де B_k - число виконавців (дорівнює 1);

F_p - місячний фонд робочого часу (при 40 годинному робочому тижні F_p=176 годин).

Очікуваний період створення ПЗ:

$$T = 1244{,}87 / (1 * 176) \approx 7 \text{ міс.}$$

Отже, було визначено вартість розробки веб застосунку, що складає 291070,61 грн. Очікуваний час розробки становить 7 місяців за умови 40-годинного робочого тижня. Трудомісткість розробленого програмного забезпечення становить 1244,87 людино-годин.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Мазур О.В., Кравець Р.А. Сільське господарство та лісівництво: журнал науково-виробничого та навчального спрямування. – Вінниця: Вінницький національний аграрний університет, 2023 р. – 84 с. URL: http://socrates.vsau.org/repository/getfile.php/35171.pdf

2. Черкаський С.С. Система виявлення хвороб сільськогосподарських культур. – Київ: Київський політехнічний інститут ім. Ігоря Сікорського, 2024 р. – 15 с. URL: https://ela.kpi.ua/server/api/core/bitstreams/55df9a20-9f41-46fd-a019-64507d5f513f/content

3. Ipsita Samal, Deepak kumar Mahanta, Tanmaya Kumar Bhoi. Van Sangyan: Plantix app: A success story of Artificial Intelligence in plant protection, 2023 p. – 24 c. URL: https://www.researchgate.net/profile/Deepak-Kumar-Mahanta/publication/370650382_Plantix_app_A_Success_story_of_artificial_intelligence_in_plant_protection/links/645b9e4e2edb8e5f094db226/Plantix-app-A-Success-story-of-artificial-intelligence-in-plant-protection.pdf#page=29

4. Житкевич В.В. Метод автоматизованого діагностування хвороб рослинних культур. – Хмельницький: Хмельницький національний університет, 2021 р. – 18 с. URL: https://elar.khmnu.edu.ua/server/api/core/bitstreams/4c813642-75bc-450c-a969-b1ffb4629e0b/content

5. Возняк О.М., Штуць А.А., Булига А.І., Харченко Р.Є. Дослідження та удосконалення процесу обробки зображень навігаційної системи сільськогосподарських машин з урахуванням умов обмеженої видимості. – Вінниця: Вінницький національний аграрний університет, 2023 р. – 126 с. URL: http://socrates.vsau.org/repository/getfile.php/35271.pdf

6. Констанченко О.Е. Огляд використання систем комп’ютерного зору в сільському господарстві. – Харків: Харківський національний університет радіоелекторніки, 2023 р. – 193 с. URL: https://files.znu.edu.ua/files/Bibliobooks/Inshi72/0053083.pdf#page=194

7. Д. М. Квашук, О. М. Густера, І. О. Юнашов. Ідентифікація хвороб рослин в сільському господарстві з використанням машинного зору. – Київ: Національний авіаційний університет, 2020 р. – 62 с. URL: http://www.investplan.com.ua/pdf/4_2020/12.pdf

8. Петренко А. І. Система розпізнавання овочів та фруктів для сільського господарства. Київ: Київський політехнічний інститут ім. Ігоря Сікорського, 2018 р. – 45 с. URL: https://ela.kpi.ua/server/api/core/bitstreams/52f5312f-90d8-4c1b-8a48-f4c9cde5b8a9/content

9. Несмянович А.В. Дослідження моделей та методів визначення типів сільськогосподарських культур. – Харків: Харківський національний університет радіоелектроніки, 2022 р. – 11 с. URL: https://openarchive.nure.ua/bitstreams/f5fa0c63-1c9d-4b04-9fd6-e9d0c98171d3/download

10. Білинський Й. Й., Книш Б. П., Кулик Я. А. Обробка та використання мультиспектральних зображень в агромоніторингу. Вінниця: Вінницький національний технічний університет, 2020 р. – 4 с. URL: https://praci.vntu.edu.ua/index.php/praci/article/view/619/581

11. Файфура В.В., Співак І.Я., Файфура В.В. Удосконалення методів дистанційного розпізнавання поширення інвазійних видів рослин в контексті оцінювання потенційних збитків навколишньому середовищу та сільському господарству. Тернопіль: Західноукраїнський національний університет, 2023 р. – 119 с. URL: http://inneco.org/index.php/innecoua/article/view/1022/1114

12. Petre Lameski, Eftim Zdravevski, Vladimir Trajkovik, Andrea Kulakov. Weed Detection Dataset with RGB Images Taken Under Variable Light Conditions, 2017 р. – 1 с. URL: https://www.researchgate.net/publication/319570220_Weed_Detection_Dataset_with_RGB_Images_Taken_Under_Variable_Light_Conditions

13. Irma Rochmawati. User Interface Design of Mobile Photo Editors, 2018 р. – 5 с. URL: https://www.atlantis-press.com/article/25906864.pdf

14. Marcus Bevilaqua. Guide to image editing and production of figures for scientific publications with an emphasis on taxonomy Image editing for scientific publications, 2020 p. – 143 c. URL: https://www.researchgate.net/publication/341144996_Guide_to_image_editing_and_production_of_figures_for_scientific_publications_with_an_emphasis_on_taxonomy_Image_editing_for_scientific_publications

15. MDN Web Docs. URL: https://developer.mozilla.org/en-US/

16. Mike McGrath. HTML, CSS & JavaScript in easy steps, 2020 p. – 11 c. URL: https://books.google.com.ua/books?hl=uk&lr=&id=C3L1DwAAQBAJ&oi=fnd&pg=PT11&dq=html+css+javascript&ots=FOgNoa0uA2&sig=uRr6_hJW1Ol-Rt1YCkKt1wXGwlM&redir_esc=y#v=onepage&q=html%20css%20javascript&f=false

17. Ashis Kumar Ratha, Shibani Sahu, Priya Meher. HTML5 in Web Development: A New Approach, 2018 p. – 552 c. URL: https://d1wqtxts1xzle7.cloudfront.net/56779892/IRJET-V5I3123-libre.pdf?1528799028=&response-content-disposition=inline%3B+filename%3DHTML5_in_Web_Development_A_New_Approach.pdf&Expires=1718558545&Signature=cT1PR6ivIO9DOIJMZKUy2Jz5yuGS5wLbQ9toHN8~pfFEwc40TjSeWJS-SgmvUzzYifW3Ayl9B0pUlVyYvIlWnP0lv952c3zFUYUzPCMGlBrGuk-XWS1~TUwUGMAePVUHv8B9k9Hp9BFqAEy7vDVu2bPU0YUzS7UfL~sGug7W1tLwozjli7kLECpjM-IFNH8mjDKLpzMCmk1Y3bVCVHmJbFuV0ez4iqDZUvEwuGRQE4dA9BJH6kpmZej5l7btXqzMEP0TeBuLiJmX7aRlnvQ-Aiuu-9RbbPjC7uGQ~UQG5NINoIqIBI0iejdUPr39NVzdOU8HBj6BCO5iGhO2UWJYxg__&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA

18. Ben Frain. Responsive Web Design with HTML5 and CSS: Build future-proof responsive, 2022 p. – 254 c.

19. Miguel Grinber. Flask Web Development 2nd Edition, 2018 p. – 14 c.

20. W3Schools. URL: https://www.w3schools.com/

21. Marijn Haverbeke. Eloquent JavaScript 4th Edition, 2024 p. – Part 2: Browser. The Document Object Model. URL: https://eloquentjavascript.net/14_dom.html

22. Kyle Simpson. You don't Know JS: Up & Going, 2020 p. – 58 c.

23. David Flanagan. JavaScript. The Definitive Guide. Master the World's Most-Used Programming Language 7th Edition, 2020 p. – 742 c.

24. Eric Matthes. Python Crash Course. A Hands-On, Project-Based Introduction to Programming, 2023 p. – 3 c.

ДОДАТОК А

ЛІСТИНГ ПРОГРАМИ

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>AGRO VISION</title>
            <link rel="stylesheet" href="{{ url_for('static', filename='css/style.css')
}}">
            <link rel="shortcut icon" href="{{ url_for('static',
filename='favicon/favicon_agro_vision.ico') }}" type="image/x-icon">
</head>
<body>
      <div class="wrapper">
            <div class="logo__wrapper">
                  <img id="logo" src="{{ url_for('static',
filename='logo/agro_vision_logo.svg') }}">
            </div>
            <div id="drop__area">
                  <div id="drop__area__info">
                        <span id="drop__area__info__text">Drag & Drop to
Upload File</span>
                        <span>OR</span>
                        <form id="uploadForm" enctype="multipart/form-data">
                              <input type="file" id="file__input" name="file"
accept="image/*" hidden>
```

```
                    </form>
                    <button id="upload__button">Browse File</button>
                </div>
                <div id="drop__area__content" hidden>
                </div>
            </div>
            <div id="tools">
                <label class="tools__title tools__block">Filters</label>

                <div class="tools__brightness tools__block">
                    <div class="brighness__info tools__block__info">
                        <label
for="brightness__input">Brightness</label>
                        <input type="text"
name="brightness__input" class="brightness__input tools__input"
id="brightness__input" value="0.00" maxlength="5">
                    </div>
                    <input type="range" name="brightness__range"
class="brightness__range" id="brightness__range" step="0.05" min="-1" max="1"
value="0">
                </div>
                <div class="tools__contrast tools__block">
                    <div class="contrast__info tools__block__info">
                        <label
for="contrast__input">Contrast</label>
                        <input type="text"
name="contrast__input" class="contrast__input tools__input" id="contrast__input"
value="0" maxlength="4">
                    </div>
```

```
                    <input type="range" name="contrast__range"
class="contrast__range" id="contrast__range" step="1" min="0" max="200"
value="100">
                </div>
                <div class="tools__saturation tools__block">
                    <div class="saturation__info tools__block__info">
                        <label
for="saturation__input">Saturation</label>
                        <input type="text"
name="saturation__input" class="saturation__input tools__input"
id="saturation__input" value="0" maxlength="4">
                    </div>
                    <input type="range" name="saturation__range"
class="saturation__range" id="saturation__range" step="1" min="0" max="200"
value="100">
                </div>
                <div class="tools__grayscale tools__block">
                    <div class="grayscale__info tools__block__info">
                        <label
for="grayscale__input">Grayscale</label>
                        <input type="text"
name="grayscale__input" class="grayscale__input tools__input"
id="grayscale__input" value="0" maxlength="4">
                    </div>
                    <input type="range" name="grayscale__range"
class="grayscale__range" id="grayscale__range" step="1" min="0" max="100"
value="0">
                </div>
                <div class="tools__hue tools__block">
                    <div class="hue__info tools__block__info">
```

```
                                        <label for="hue__input">Hue</label>
                                        <input type="text" name="hue__input"
class="hue__input tools__input" id="hue__input" value="0" maxlength="4">
                                    </div>
                                    <input type="range" name="hue__range"
class="hue__range" id="hue__range" step="1" min="0" max="360" value="0">
                                </div>
                                <div class="tools__invert tools__block">
                                    <div class="invert__info tools__block__info">
                                        <label
for="invert__input">Invert</label>
                                        <input type="text" name="invert__input"
class="invert__input tools__input" id="invert__input" value="0" maxlength="4">
                                    </div>
                                    <input type="range" name="invert__range"
class="invert__range" id="invert__range" step="1" min="0" max="100" value="0">
                                </div>
                                <div class="tools__sepia tools__block">
                                    <div class="sepia__info tools__block__info">
                                        <label for="sepia__input">Sepia</label>
                                        <input type="text" name="sepia__input"
class="sepia__input tools__input" id="sepia__input" value="0" maxlength="4">
                                    </div>
                                    <input type="range" name="sepia__range"
class="sepia__range" id="sepia__range" step="1" min="0" max="100" value="0">
                                </div>
                                <label class="tools__title tools__block">Flip & Rotate</label>

                                <div class="tools__flipRotate">
```

```
                    <button id="rotate__right"
class="tools__flipRotate__item"><img src="{{ url_for('static',
filename='icons/rotate_right_icon.svg') }}"></buttoon>
                    <button id="rotate__left"
class="tools__flipRotate__item"><img src="{{ url_for('static',
filename='icons/rotate_left_icon.svg') }}"></buttoon>
                    <button id="flip__horizontal"
class="tools__flipRotate__item"><img src="{{ url_for('static',
filename='icons/flip_horizontal_icon.svg') }}"></buttoon>
                    <button id="flip__vertical"
class="tools__flipRotate__item"><img src="{{ url_for('static',
filename='icons/flip_vertical_icon.svg') }}"></buttoon>
                </div>

                <label class="tools__title tools__block">Draw</label>

                <label class="brushsize" for="brushSizeInput"><span
class="brushSizeText">Size (1-30)</span><input type="text" id="brushSizeInput"
maxlength="2"></label>

                <div class="tools__pallette">
                    <div class="tools__pallette__item"><input
type="radio" name="pallette" id="red"></div>
                    <div class="tools__pallette__item"><input
type="radio" name="pallette" id="orange"></div>
                    <div class="tools__pallette__item"><input
type="radio" name="pallette" id="yellow"></div>
                    <div class="tools__pallette__item"><input
type="radio" name="pallette" id="green"></div>
```

```
                                <div class="tools__pallette__item"><input type="radio" name="pallette" id="blue"></div>
                                <div class="tools__pallette__item"><input type="radio" name="pallette" id="violet"></div>
                        </div>

                        <div class="tools__buttons">
                                <button class="tools__buttons__reset tools__buttons__item"><span class="tools__buttons__text">Reset</span><img src="{{ url_for('static', filename='icons/reset_icon.svg') }}" alt="" srcset=""></button>
                        </div>
        </div>

        </div>
  <script src="{{ url_for('static', filename='js/script.js') }}"></script>
</body>
</html>
```

style.css

```
* {
        margin: 0;
        padding: 0;
        box-sizing: border-box;
        font-family: "Montserrat", sans-serif;
        font-optical-sizing: auto;
        font-weight: 500;
        font-style: normal;
        color: #fff;
}
```

```
body {
	overflow: hidden;
}

input[type="text"], input[type="number"] {
	color: #000;
}

button {
	cursor: pointer;
  transition: .3s;
  border-radius: 50px;
  overflow: hidden;
  background-color: transparent;
  border: 2px solid #b0b0b0;
}

#logo {
	position: absolute;
  top: 0;
  z-index: 999;
  left: 50%;
  transform: translateX(-50%);
  max-height: 8%;
}

.wrapper {
	width: 100%;
	/* 123 */
```

```
        padding: 3%;
        /* 123 */
        min-height: 100vh;
        display: flex;
        gap: 15px;
        align-items: center;
        justify-content: center;
        background-color: #282828;
        overflow-x: hidden;
}

#tools {
        opacity: 0;
        visibility: hidden;
        display: none;
  flex-direction: column;
  height: 90vh;
  justify-content: space-between;
  background-color: #1d1d1d;
  padding: 20px 30px;
  border-radius: 25px;
  border: 2px solid #b0b0be;
}

.tools__flipRotate, .tools__buttons {
        display: flex;
        justify-content: space-between;
        gap: 10px;
}
```

```
.tools__flipRotate__item {
	display: flex;
	background-color: #e87840;
	flex: 1;
	border: none;
	padding: 10px 5px;
	border-radius: 5px;
	align-items: center;
	justify-content: center;
}

.tools__flipRotate__item img {
	min-width: 12px;
	max-height: 18px;
}

.tools__title {
	font-size: 22px;
	font-weight: 800;
}

.tools__block {
	display: flex;
	flex-direction: column;
	gap: 2px;
}

.tools__block__info {
	display: flex;
	justify-content: space-between;
```

```
}

.tools__input, #brushSizeInput {
	max-width: 20%;
}

.brushsize {
	display: flex;
	justify-content: space-between;
}

.tools__buttons__reset {
	width: 100%;
  display: flex;
  align-items: center;
  justify-content: center;
  gap: 10px;
  padding: 10px 20px;
}

.tools__buttons__reset img {
	width: 7%;
}

.tools__buttons button:hover {
	background-color: #f8f8f834;
}

.tools__pallette {
	display: flex;
```

```
        flex-wrap: wrap;
        justify-content: space-between;
        align-items: center;
        gap: 10px;
}

.tools__pallette__item {
        flex-grow: 1;
        display: flex;
        align-items: center;
        justify-content: center;
        padding: 10px;
        border-radius: 50%;
        transition: .3s;
        cursor: pointer;
}

.tools__pallette__item input[type="radio"]{
        visibility: hidden;
}

.tools__pallette__item input[type="color"]{
        display: none;
}

.tools__pallette__item:has(#red) {
        border: 4px solid red;
}

.tools__pallette__item:has(#red:checked) {
```

```
  background-color: red;
}

.tools__pallette__item:has(#orange) {
        border: 4px solid orange;
}

.tools__pallette__item:has(#orange:checked) {
  background-color: orange;
}

.tools__pallette__item:has(#yellow) {
        border: 4px solid yellow;
}

.tools__pallette__item:has(#yellow:checked) {
  background-color: yellow;
}

.tools__pallette__item:has(#green) {
        border: 4px solid green;
}

.tools__pallette__item:has(#green:checked) {
  background-color: green;
}

.tools__pallette__item:has(#blue) {
        border: 4px solid blue;
}
```

```
.tools__pallette__item:has(#blue:checked) {
  background-color: blue;
}

.tools__pallette__item:has(#violet) {
        border: 4px solid violet;
}

.tools__pallette__item:has(#violet:checked) {
  background-color: violet;
}

.tools__pallette__item:has(#custom) {
        border: 4px solid rgb(121, 121, 121);
}

.tools__pallette__item:has(#custom:checked) {
  background-color: rgb(121, 121, 121);
}

#drop__area {
        display: flex;
        flex-direction: column;
        justify-content: center;
        align-items: center;
        position: relative;
        gap: 10px;
        width: 50vw;
        height: 90vh;
```

```
        padding: 30px;
        background: #1d1d1d;
        border-radius: 25px;
        border: 2px dashed #b0b0b0;
}

#drop__area.active {
        border: 2px solid #b0b0b0;
}

#drop__area__info {
        display: flex;
        flex-direction: column;
        align-items: center;
        gap: 10px;
}

#upload__button {
        padding: 10px 20px;
}

#upload__button:hover {
        background-color: #f8f8f834;
}

#drop__area__content {
        display: none;
        flex-direction: column;
        justify-content: space-between;
        gap: 30px;
```

```
        width: 100%;
        height: 100%;
        max-height: 100%;
}

.fileNameBlock {
        display: flex;
        flex-direction: column;
        gap: 10px;
}

#image__name {
        padding: 10px 20px;
}

#imgElementWrapper {
  display: inline-flex;
  align-items: center;
  justify-content: center;
        height: 65%;
        overflow: hidden;
        transition: .3s;
}

#imgElementWrapper img{
        max-width: 100%;
        max-height: 100%;
        width: auto;
        height: auto;
  object-fit: contain;
```

```
}

.imgActionsBlock {
	display: flex;
	justify-content: space-between;
	align-items: center;
	gap: 10px;
}

.AIWrapper {
	display: flex;
	flex-direction: column;
	gap: 10px;
}

.imgButtonsWrapper {
	display: flex;
	flex-direction: column;
	gap: 15px;
}

#predictionCheckbox {
	margin-left: 10px;
}

#close__button {
  padding: 10px 20px;
	border: 2px solid #e65151;
  color: #e65151;
  font-weight: 600;
```

```
}

#close__button:hover {
	background-color: #aa404075;
}

#download__button {
	padding: 10px 20px;
}

#download__button:hover {
	background-color: #f8f8f834;
}

@media (max-width: 900px) {
	.wrapper {
		flex-direction: column;
		padding: 15px;
	}

	#logo {
		max-height: 6%;
	}

	#drop__area {
		height: 75vh;
		width: 100%;
		padding: 25px;
	}
```

```
#tools {
        width: 100%;
        height: calc(25vh - 45px);
        overflow-y: scroll;
overflow-x: hidden;
        gap: 15px;
}

.tools__pallette {
        justify-content: space-around;
}

.tools__pallette__item {
        padding: 5%;
        flex-grow: 0;
}

.tools__flipRotate__item {
        padding: 3% 1%;
}

.tools__flipRotate__item {
        width: 20%;
}

.AIWrapper {
        width: 50%;
}

.imgButtonsWrapper {
```

```
        width: 50%;
    }
}

@media (min-width: 1200px) {
    .tools {
        max-width: 500px;
    }

    #drop__area {
        max-width: 900px;
    }
}
```

script.js

```
//Get drop area elements
const fileInput = document.getElementById("file__input");
const uploadButton = document.getElementById("upload__button");
const dropArea = document.getElementById("drop__area");
const dropAreaInfo = document.getElementById("drop__area__info");
const dropAreaContent = document.getElementById("drop__area__content");
const dropAreaInfoText = document.getElementById("drop__area__info__text");
const imageTools = document.getElementById("tools");

//Get tools elements
const brightnessInput = document.querySelector(".brightness__input");
const brightnessRange = document.querySelector(".brightness__range");

const contrastInput = document.querySelector(".contrast__input");
const contrastRange = document.querySelector(".contrast__range");
```

```
const saturationInput = document.querySelector(".saturation__input");
const saturationRange = document.querySelector(".saturation__range");

const grayscaleInput = document.querySelector(".grayscale__input");
const grayscaleRange = document.querySelector(".grayscale__range");

const hueInput = document.querySelector(".hue__input");
const hueRange = document.querySelector(".hue__range");

const invertInput = document.querySelector(".invert__input");
const invertRange = document.querySelector(".invert__range");

const sepiaInput = document.querySelector(".sepia__input");
const sepiaRange = document.querySelector(".sepia__range");

const toolsBlock = document.querySelector(".tools");

const rotateFlipOptions = document.querySelectorAll(".tools__flipRotate__item");

const brushSizeInput = document.getElementById("brushSizeInput");

const resetButton = document.querySelector(".tools__buttons__reset");

//Top block
const imgNameBlock = document.createElement("div");
imgNameBlock.classList.add("fileNameBlock");

const fileNameBlockText = document.createElement("span");
fileNameBlockText.textContent = "Image name";
```

```
const imgNameInput = document.createElement("input");
imgNameInput.type = "text";
imgNameInput.id = "image__name";

imgNameBlock.appendChild(fileNameBlockText);
imgNameBlock.appendChild(imgNameInput);

//Image element
const imgElementWrapper = document.createElement("div")
imgElementWrapper.id = "imgElementWrapper";
imgElementWrapper.style.position = "relative";

const imgElement = document.createElement("img");
imgElement.id = "image__edit";
imgElement.classList.add("rotate");
imgElement.style.position = 'relative';

imgElementWrapper.appendChild(imgElement);

//Bottom element
const imgActionsBlock = document.createElement("div");
imgActionsBlock.classList.add("imgActionsBlock");

const AIWrapper = document.createElement("div");
AIWrapper.classList.add("AIWrapper");

const imgPredictInfo = document.createElement("span");
imgPredictInfo.textContent = "AI Predict";
imgPredictInfo.classList.add("AILabel");
```

```
const imgPredictText = document.createElement("span");
imgPredictText.id = "image__predict";
imgPredictText.style.padding = "10px 20px";
imgPredictText.style.borderRadius = "5px"

const imgButtonsWrapper = document.createElement("div");
imgButtonsWrapper.classList.add("imgButtonsWrapper");

const closeImageButton = document.createElement("button");
closeImageButton.id = "close__button";
closeImageButton.textContent = "Close";

const downloadImageButton = document.createElement("button");
downloadImageButton.id = "download__button";
downloadImageButton.textContent = "Download";

imgButtonsWrapper.appendChild(closeImageButton);
imgButtonsWrapper.appendChild(downloadImageButton);

const changeNameCheckbox = document.createElement("input");
changeNameCheckbox.type = "checkbox";
changeNameCheckbox.id = "predictionCheckbox";

const changeNameLabel = document.createElement("label");
changeNameLabel.id = "predictionLabel";
changeNameLabel.textContent = "Change image name";
changeNameLabel.setAttribute("for", `${changeNameCheckbox.id}`);

changeNameLabel.appendChild(changeNameCheckbox);
```

```
AIWrapper.appendChild(imgPredictInfo);
AIWrapper.appendChild(imgPredictText);
AIWrapper.appendChild(changeNameLabel);

imgActionsBlock.appendChild(AIWrapper);
imgActionsBlock.appendChild(imgButtonsWrapper);

let file;
let fileType;

let prediction;

let documentDefaultTitle = document.title;

let oldFileName;

let isDrawing = false;

let brushSize = 5;

let selectedColor = "";

const settingsFilter = { brightness: 1, contrast: 100, saturation: 100, grayscale: 0, hue:
0, invert: 0, sepia: 0};
const settingsFlipRotate = { rotate: 0, scalex: 1, scaley: 1};
let rotateDegree = 0;
let flipHorizontalIndex = 1;
let flipVerticalIndex = 1;
```

```
let insertTransformStyle = `rotate(${settingsFlipRotate["rotate"]}deg)
scaleX(${settingsFlipRotate["scalex"]}) scaleY(${settingsFlipRotate["scaley"]})`;
let insertFilterStyle = `contrast(${settingsFilter["contrast"]}%)
saturate(${settingsFilter["saturation"]}%) grayscale(${settingsFilter["grayscale"]}%)
hue-rotate(${settingsFilter["hue"]}deg) invert(${settingsFilter["invert"]}%)
sepia(${settingsFilter["sepia"]}%) brightness(${settingsFilter["brightness"]})`;

//Make canvas for drawing
const drawingCanvas = document.createElement("canvas");
const ctx = drawingCanvas.getContext("2d");

//Filter Listeners
brightnessInput.addEventListener("input", () => updateInput(brightnessRange,
brightnessInput, "brightness"));
brightnessRange.addEventListener('input', () => updateRange(brightnessRange,
brightnessInput, "brightness"));

contrastInput.addEventListener("input", () => updateInput(contrastRange,
contrastInput, "contrast"));
contrastRange.addEventListener('input', () => updateRange(contrastRange,
contrastInput, "contrast"));

saturationInput.addEventListener("input", () => updateInput(saturationRange,
saturationInput, "saturation"));
saturationRange.addEventListener('input', () => updateRange(saturationRange,
saturationInput, "saturation"));

grayscaleInput.addEventListener("input", () => updateInput(grayscaleRange,
grayscaleInput, "grayscale"));
```

```
grayscaleRange.addEventListener('input', () => updateRange(grayscaleRange,
grayscaleInput, "grayscale"));

hueInput.addEventListener("input", () => updateInput(hueRange, hueInput, "hue"));
hueRange.addEventListener('input', () => updateRange(hueRange, hueInput, "hue"));

invertInput.addEventListener("input", () => updateInput(invertRange, invertInput,
"invert"));
invertRange.addEventListener('input', () => updateRange(invertRange, invertInput,
"invert"));

sepiaInput.addEventListener("input", () => updateInput(sepiaRange, sepiaInput,
"sepia"));
sepiaRange.addEventListener('input', () => updateRange(sepiaRange, sepiaInput,
"sepia"));

brushSizeInput.addEventListener("input", () =>
validBrushInput(brushSizeInput.value));

resetButton.onclick = resetSettings;
downloadImageButton.onclick= downloadImage;

uploadButton.onclick = ()=>{
  fileInput.click();
}

dropArea.addEventListener("dragover", (event)=>{
  event.preventDefault();
  dropArea.classList.add("active");
  dropAreaInfoText.textContent = "Release to Upload File";
```

```
});

dropArea.addEventListener("dragleave", ()=>{
  dropArea.classList.remove("active");
  dropAreaInfoText.textContent = "Drag & Drop to Upload File";
});

function showFile() {
  if (!file) {
    return; // If file is not defined, do nothing
  }
  fileType = file.type;
  let validExtensions = ["image/png", "image/jpeg", "image/jpg"];
  if (validExtensions.includes(fileType)) {
    let fileReader = new FileReader();
    fileReader.onload = ()=>{
      imgNameInput.value = (file.name).slice(0, (file.name).lastIndexOf("."));
                          oldFileName = imgNameInput.value;
      document.title = `${documentDefaultTitle} > ${imgNameInput.value}`;
      imgElement.src = fileReader.result;

      dropAreaContent.style.display = "flex";
      dropAreaContent.style.opacity = "1";
      dropAreaContent.style.visibility = "visible";

                          imageTools.style.display = "flex";
      imageTools.style.opacity = "1";
      imageTools.style.visibility = "visible";

      dropAreaInfo.style.display = "none";
```

```
      dropAreaInfo.style.opacity = "0";
      dropAreaInfo.style.visibility = "hidden";
      dropAreaContent.removeAttribute('hidden');

      let ContenthChildList = [imgNameBlock, imgElementWrapper,
imgActionsBlock];
      ContenthChildList.forEach(child => {
        dropAreaContent.appendChild(child);
      });

                        imgElement.onload = () => {
                              const imgElementRect =
imgElement.getBoundingClientRect();
                              drawingCanvas.id = "canvas__wrapper";
                              drawingCanvas.style.position = "absolute";
                              drawingCanvas.width = imgElementRect.width;
                              drawingCanvas.height = imgElementRect.height;
                              drawingCanvas.style.top = "50%";
                              drawingCanvas.style.left = "50%";
                              drawingCanvas.style.transform = "translateX(-50%)
translateY(-50%)";
                              imgElementWrapper.appendChild(drawingCanvas);
                        };

      closeImageButton.onclick = closeImage;
    }
    fileReader.readAsDataURL(file);
  } else {
    alert("Chose the image");
    dropArea.classList.remove("active");
```

```
    dropAreaInfoText.textContent = "Drag & Drop to Upload File";
  }
}

dropArea.addEventListener("drop", (event)=>{
  event.preventDefault();
  file = event.dataTransfer.files[0];
  showFile();
  const changeEvent = new Event("change");
  fileInput.dispatchEvent(changeEvent);
});

fileInput.addEventListener('change', function(event) {
  dropArea.classList.add("active");
  file = event.target.files[0];
  showFile();
  if (file) {
    var formData = new FormData();
    formData.append('file', file);

    fetch('/', {
      method: 'POST',
      body: formData
    })
    .then(response => response.json())
    .then(data => {
                        function parsePrediction(prediction) {
                                // Розділяємо текст передбачення за допомогою знака
підкреслення
                                const parts = prediction.split("___");
```

```
                        // Перевіряємо, чи було знайдено дві частини
                        if (parts.length === 2) {
                                // Визначаємо вид рослини та хворобу
                                const plant = parts[0].replace(/[_|,]/g, " ");
                const disease = parts[1].replace(/[_|,]/g, " ");
                                // Повертаємо об'єкт з видом рослини та
хворобою
                                return {
                                        plant: plant,
                                        disease: disease
                                };
                        } else {
                                // Якщо не було знайдено дві частини,
повертаємо порожні значення
                                return {
                                        plant: "",
                                        disease: ""
                                };
                        }
                }
                prediction = parsePrediction(data.prediction);
                imgPredictText.textContent = `Plant: ${prediction.plant}, Disease:
${prediction.disease}`;
                if (prediction.disease === "healthy") {
                        imgPredictText.style.backgroundColor = "#167046";
                } else {
                        imgPredictText.style.backgroundColor = "#701616";
                }
    })
    .catch(error => {
```

```
      console.error('Error:', error);
    });
  }
});

imgNameInput.addEventListener('input', ()=>{
        document.title = `${documentDefaultTitle} > ${imgNameInput.value}`;
        oldFileName = imgNameInput.value;
});

changeNameCheckbox.addEventListener("change", function() {
        if (this.checked) {
                imgNameInput.value = `${prediction.plant} ${prediction.disease}`;
                document.title = `${documentDefaultTitle} > ${imgNameInput.value}`;
        } else {
                imgNameInput.value = oldFileName;
                document.title = `${documentDefaultTitle} > ${imgNameInput.value}`;
        }
});

function updateSettings() {
        insertFilterStyle = `contrast(${settingsFilter["contrast"]}%)
saturate(${settingsFilter["saturation"]}%) grayscale(${settingsFilter["grayscale"]}%)
hue-rotate(${settingsFilter["hue"]}deg) invert(${settingsFilter["invert"]}%)
sepia(${settingsFilter["sepia"]}%) brightness(${settingsFilter["brightness"]})`;
        insertTransformStyle = `rotate(${settingsFlipRotate["rotate"]}deg)
scaleX(${settingsFlipRotate["scalex"]}) scaleY(${settingsFlipRotate["scaley"]})`;
        imgElement.style.filter = insertFilterStyle;
        imgElementWrapper.style.transform = insertTransformStyle;
}
```

```
function updateRange(range, input, keyword) {
	if (keyword == "brightness") {
		settingsFilter[keyword] = (parseFloat(range.value) +
1).toFixed(2);
	} else {
		settingsFilter[keyword] = range.value;
	}
	updateSettings();
	switch (keyword) {
		case "brightness":
			if(input.value > 0){
				input.value =
`+${Number(range.value).toFixed(2)}`;
			} else {
				input.value =
Number(range.value).toFixed(2);
			}
			break;

		case "contrast":
			if (input.value > 0) {
				input.value = `+${range.value - 100}`;
			} else {
				input.value = range.value - 100;
			}
			break;

		case "saturation":
			if (input.value > 0) {
```

```
                    input.value = `+${range.value - 100}`;
                } else {
                    input.value = range.value - 100;
                }
                break;

            case "grayscale":
                input.value = `+${range.value}`;
                break;

            case "hue":
                input.value = `${range.value}°`;
                break;

            case "invert":
                input.value = `${range.value}%`;
                break;

            case "sepia":
                input.value = range.value;
                break;

            default:
                break;
        }
}

function updateInput(range, input, keyword) {
        switch (keyword) {
            case "brightness":
```

```
                let brightnessRegex = /[^\d.-]|(?<=\d)-/g;
                // Використання методу replace() для вилучення символів, що не є числом та крапкою, зберігаючи мінус
                range.value = input.value.replace(brightnessRegex, '');
                if (Number(input.value.replace(brightnessRegex, '')) >= -1 && Number(input.value.replace(brightnessRegex, '')) <= 1) {
                    settingsFilter[keyword] = (parseFloat(input.value) + 1).toFixed(2);
                    updateSettings();
                }
                break;

            case "contrast":
                let contrastRegex = /\B\+/g;
                input.value = input.value.replace(contrastRegex, '');
                range.value = Number(input.value.replace(contrastRegex, '')) + 100;
                if (Number(input.value) >= -100 && Number(input.value) <= 100) {
                    settingsFilter[keyword] = Number(input.value) + 100;
                    updateSettings();
                }
                break;

            case "saturation":
                let saturationRegex = /\B\+/g;
                input.value = input.value.replace(saturationRegex, '');
```

```
                range.value =
Number(input.value.replace(saturationRegex, '')) + 100;
                if (Number(input.value) >= -100 &&
Number(input.value) <= 100) {
                    settingsFilter[keyword] =
Number(input.value) + 100;
                    updateSettings();
                }
                break;

            case "grayscale":
                let grayscaleRegex = /\B\+/g;
                input.value = input.value.replace(grayscaleRegex, '');
                range.value = input.value;
                if (Number(input.value) >= 0 &&
Number(input.value) <= 100) {
                    settingsFilter[keyword] = input.value;
                    updateSettings();
                }
                break;

            case "hue":
                let hueRegex = /\B\+/g;
                input.value = input.value.replace(hueRegex, '');
                range.value = input.value;
                if (Number(input.value) >= 0 &&
Number(input.value) <= 360) {
                    settingsFilter[keyword] = input.value;
                    updateSettings();
                }
```

```
                    break;

                case "invert":
                    let invertRegex = /\B\+/g;
                    input.value = input.value.replace(invertRegex, '');
                    range.value = input.value;
                    if (Number(input.value) >= 0 &&
Number(input.value) <= 100) {
                        settingsFilter[keyword] = input.value;
                        updateSettings();
                    }
                    break;

                case "sepia":
                    let sepiaRegex = /\B\+/g;
                    input.value = input.value.replace(sepiaRegex, '');
                    range.value = input.value;
                    if (Number(input.value) >= 0 &&
Number(input.value) <= 100) {
                        settingsFilter[keyword] = input.value;
                        updateSettings();
                    }
                    break;
                default:
                    break;
        }
}

rotateFlipOptions.forEach(option => {
        option.addEventListener("click", () => {
```

```
                switch (option.id) {
                        case "rotate__right":
                                settingsFlipRotate["rotate"] =
(settingsFlipRotate["rotate"] += 90) % 360;
                                updateSettings();
                                break;

                        case "rotate__left":
                                settingsFlipRotate["rotate"] =
(settingsFlipRotate["rotate"] -= 90) % -360;;
                                updateSettings();
                                break;

                        case "flip__horizontal":
                                if (settingsFlipRotate["rotate"] === 90 ||
settingsFlipRotate["rotate"] === 270 || settingsFlipRotate["rotate"] === -90 ||
settingsFlipRotate["rotate"] === -270) {
                                        settingsFlipRotate["scaley"]
= (settingsFlipRotate["scaley"] === 1) ? -1 : 1;
                                        updateSettings();
                                } else {
                                        settingsFlipRotate["scalex"]
= (settingsFlipRotate["scalex"] === 1) ? -1 : 1;
                                        updateSettings();
                                }
                                break;

                        case "flip__vertical":
```

```
                                        if (settingsFlipRotate["rotate"] === 90 ||
settingsFlipRotate["rotate"] === 270 || settingsFlipRotate["rotate"] === -90 ||
settingsFlipRotate["rotate"] === -270) {
                                            settingsFlipRotate["scalex"]
= (settingsFlipRotate["scalex"] === 1) ? -1 : 1;
                                            updateSettings();
                                        } else {
                                            settingsFlipRotate["scaley"]
= (settingsFlipRotate["scaley"] === 1) ? -1 : 1;
                                            updateSettings();
                                        }
                                        break;
                    }
        });
});

function resetSettings() {
    brightnessRange.value = "0";
    brightnessInput.value = "0.00";

    contrastRange.value = "100";
    contrastInput.value = "0";

    saturationRange.value = "100";
    saturationInput.value = "0";

    grayscaleRange.value = "0";
    grayscaleInput.value = "0";

    hueRange.value = "0";
```

```
    hueInput.value = "0";

    invertRange.value = "0";
    invertInput.value = "0";

    sepiaRange.value = "0";
    sepiaInput.value = "0";

    settingsFlipRotate["rotate"] = 0;
    settingsFlipRotate["scalex"] = 1;
    settingsFlipRotate["scaley"] = 1;

    settingsFilter["brightness"] = 1;
    settingsFilter["contrast"] = 100;
    settingsFilter["saturation"] = 100;
    settingsFilter["grayscale"] = 0;
    settingsFilter["hue"] = 0;
    settingsFilter["invert"] = 0;
    settingsFilter["sepia"] = 0;

                ctx.clearRect(0, 0, drawingCanvas.width, drawingCanvas.height);

    updateSettings();
}

const startDrawing = (e) => {
        // Перевірка, чи обрано колір перед початком малювання
  const anyChecked = document.querySelector('input[name="pallette"]:checked');
  if (!anyChecked) return; // При відсутності обраного кольору завершуємо
функцію
```

```
        isDrawing = true;
        ctx.beginPath();
        ctx.lineWidth = brushSize;
        ctx.strokeStyle = selectedColor;
        ctx.moveTo(e.offsetX, e.offsetY);
}

const drawing = (e) => {
        if (!isDrawing) return;
        ctx.lineTo(e.offsetX, e.offsetY);
        ctx.stroke();
        ctx.moveTo(e.offsetX, e.offsetY);
}

const stopDrawing = () => {
        isDrawing = false;
        ctx.closePath();
}

drawingCanvas.addEventListener("mousedown", startDrawing);
drawingCanvas.addEventListener("mousemove", drawing);
drawingCanvas.addEventListener("mouseup", stopDrawing);
drawingCanvas.addEventListener("mouseout", stopDrawing);

function validBrushInput(value) {
        let brushSizeValue = value;
        brushSizeValue = brushSizeValue.replace(/[^0-9]/g, '');
        if (brushSizeValue.length !== 0) {
                brushSizeValue = parseInt(brushSizeValue);
                if (brushSizeValue >= 1 && brushSizeValue <= 30) {
```

```
                    brushSize = brushSizeValue;
                }
        }
}

function closeImage() {
  dropArea.classList.remove("active");

  dropAreaContent.style.display = "none";
  dropAreaContent.style.opacity = "0";
  dropAreaContent.style.visibility = "hidden";

        imageTools.style.display = "none";
  imageTools.style.opacity = "0";
  imageTools.style.visibility = "hidden";

  dropAreaInfo.style.display = "flex";
  dropAreaInfo.style.opacity = "1";
  dropAreaInfo.style.visibility = "visible";
  document.title = documentDefaultTitle;
};

//Pallette items

//Click on div make click on input
document.querySelectorAll('.tools__pallette__item').forEach(item => {
  item.addEventListener('click', function() {
    const input = this.querySelector('input[type="radio"]');
    input.click();
  });
```

```
});

// Логіка для перевірки вибору та зняття вибору радіо-кнопок
document.querySelectorAll('input[name="pallette"]').forEach(input => {
  input.addEventListener('click', function() {
    if (this.previousChecked) {
      this.checked = false;
      this.previousChecked = false;
      this.parentElement.classList.remove('checked');
    } else {
      document.querySelectorAll('input[name="pallette"]').forEach(i => {
        i.previousChecked = false;
        i.parentElement.classList.remove('checked');
      });
      this.previousChecked = true;
      this.parentElement.classList.add('checked');
                        selectedColor = this.id;
    }
  });
});

function downloadImage() {
        // Створюємо копію drawingCanvas
        const drawingCanvasCopy = document.createElement('canvas');
        const drawingCtxCopy = drawingCanvasCopy.getContext('2d');

        // Встановлюємо розміри копії drawingCanvas такими ж, як у imgElement
        drawingCanvasCopy.width = imgElement.naturalWidth;
        drawingCanvasCopy.height = imgElement.naturalHeight;
```

```
        drawingCtxCopy.translate(
  settingsFlipRotate["scalex"] === -1 ? drawingCanvasCopy.width : 0,
  settingsFlipRotate["scaley"] === -1 ? drawingCanvasCopy.height : 0
 );
 drawingCtxCopy.scale(settingsFlipRotate["scalex"], settingsFlipRotate["scaley"]);

        // Малюємо малюнок з drawingCanvas на копію, масштабуючи його
        drawingCtxCopy.drawImage(drawingCanvas, 0, 0, imgElement.naturalWidth,
imgElement.naturalHeight);

        // Створюємо копію image
        const imageCanvasElement = document.createElement('canvas');
        const imageElementCtx = imageCanvasElement.getContext('2d');

        // Встановлюємо розміри копії image такими ж, як у imgElement
        imageCanvasElement.width = imgElement.naturalWidth;
        imageCanvasElement.height = imgElement.naturalHeight;

        //Застосовуємо фільтри до малювання
        imageElementCtx.filter = insertFilterStyle;

        imageElementCtx.translate(
  settingsFlipRotate["scalex"] === -1 ? imageCanvasElement.width : 0,
  settingsFlipRotate["scaley"] === -1 ? imageCanvasElement.height : 0
 );
 imageElementCtx.scale(settingsFlipRotate["scalex"], settingsFlipRotate["scaley"]);

        // Малюємо малюнок з imgElement
        imageElementCtx.drawImage(imgElement, 0, 0, imgElement.naturalWidth,
imgElement.naturalHeight);
```

```
  // Create new canvas
  const combinedCanvas = document.createElement('canvas');
  const combinedCtx = combinedCanvas.getContext('2d');

  // Встановлюємо розміри об'єднаного канваса такими ж, як у вихідного зображення
        if (settingsFlipRotate["rotate"] === 90 || settingsFlipRotate["rotate"] === 270 || settingsFlipRotate["rotate"] === -90 || settingsFlipRotate["rotate"] === -270) {
                combinedCanvas.width = imgElement.naturalHeight;
        combinedCanvas.height = imgElement.naturalWidth;
        } else {
                combinedCanvas.width = imgElement.naturalWidth;
        combinedCanvas.height = imgElement.naturalHeight;
        }

        // Повертаємо зображення на вказаний градус
  const radians = (settingsFlipRotate["rotate"] * Math.PI) / 180;
  combinedCtx.rotate(radians);

        switch (settingsFlipRotate["rotate"]) {
                case -90:
                        combinedCtx.translate(-combinedCanvas.height, 0);
                        break;

                case 90:
                        combinedCtx.translate(0, -combinedCanvas.width);
                        break;

                case -180:
```

```
                combinedCtx.translate(-combinedCanvas.width, -
combinedCanvas.height);
                break;

            case 180:
                combinedCtx.translate(-combinedCanvas.width, -
combinedCanvas.height);
                break;

            case -270:
                combinedCtx.translate(0, -combinedCanvas.width);
                break;

            case 270:
                combinedCtx.translate(-combinedCanvas.height, 0);
                break;

                default:
                break;
        }

  combinedCtx.drawImage(imageCanvasElement, 0, 0, imgElement.naturalWidth,
imgElement.naturalHeight);
  combinedCtx.drawImage(drawingCanvasCopy, 0, 0, imgElement.naturalWidth,
imgElement.naturalHeight);

  // Визначаємо тип файлу та видаляємо префікс "image/"
  const fileType = file.type.split('/')[1];

  // Конвертуємо об'єднане зображення в Data URL з відповідним форматом
```

```
  const dataURL = combinedCanvas.toDataURL(`image/${fileType}`);

  // Створюємо посилання для завантаження
  const downloadLink = document.createElement('a');
  downloadLink.href = dataURL;

  // Додаємо ім'я файлу з відповідним розширенням
  downloadLink.download = `ARGO VISION ${imgNameInput.value ||
"image"}.${fileType}`;

  // Імітуємо клік на посиланні для завантаження файлу
  downloadLink.click();
}
```

app.py

```
import os
import json
from PIL import Image
import numpy as np
import tensorflow as tf
from flask import Flask, request, jsonify, url_for, render_template

app = Flask(__name__)

working_dir = os.path.dirname(os.path.abspath(__file__))
model_path = f"{working_dir}/trained_model/plant_model.h5"
model = tf.keras.models.load_model(model_path)

class_indices = json.load(open(f"{working_dir}/class_indices.json"))
```

```
UPLOAD_FOLDER = 'static/uploads/'
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER

def load_and_preprocess_image(image_path, target_size=(224, 224)):
    img = Image.open(image_path)
    img = img.resize(target_size)
    img_array = np.array(img)
    img_array = np.expand_dims(img_array, axis=0)
    img_array = img_array.astype('float32') / 255.
    return img_array

def predict_image_class(model, image_path, class_indices):
    preprocessed_img = load_and_preprocess_image(image_path)
    predictions = model.predict(preprocessed_img)
    predicted_class_index = np.argmax(predictions, axis=1)[0]
    predicted_class_name = class_indices[str(predicted_class_index)]
    return predicted_class_name

@app.route('/', methods=['GET', 'POST'])
def index():
    if request.method == 'POST':
        if 'file' not in request.files:
            return jsonify({'error': 'No file part'})

        file = request.files['file']
        if file.filename == '':
            return jsonify({'error': 'No selected file'})

        file_path = os.path.join(app.config['UPLOAD_FOLDER'], file.filename)
        file.save(file_path)
```

```
        prediction = predict_image_class(model, file_path, class_indices)

        image_url = url_for('static', filename='uploads/' + file.filename)
        response_data = {
            'prediction': prediction,
            'image_url': image_url
        }
        return jsonify(response_data)

    return render_template('index.html')

if __name__ == '__main__':
    app.run(debug=True)
```

tarining.py

```
import random
random.seed(0)

import numpy as np
np.random.seed(0)

import tensorflow as tf
tf.random.set_seed(0)

import os
import json
from zipfile import ZipFile
from PIL import Image

import numpy as np
```

```
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras import layers, models

!pip install kaggle

kaggle_credentails = json.load(open("kaggle.json"))

# setup Kaggle API key as environment variables
os.environ['KAGGLE_USERNAME'] = kaggle_credentails["username"]
os.environ['KAGGLE_KEY'] = kaggle_credentails["key"]

!kaggle datasets download -d abdallahalidev/plantvillage-dataset

!ls

# Unzip the downloaded dataset
with ZipFile("plantvillage-dataset.zip", 'r') as zip_ref:
    zip_ref.extractall()

print(os.listdir("plantvillage dataset"))



print(len(os.listdir("plantvillage dataset/segmented")))
print(os.listdir("plantvillage dataset/segmented")[:5])

print(len(os.listdir("plantvillage dataset/color")))
print(os.listdir("plantvillage dataset/color")[:5])
```

```
print(len(os.listdir("plantvillage dataset/grayscale")))
print(os.listdir("plantvillage dataset/grayscale")[:5])

print(len(os.listdir("plantvillage dataset/color/Grape___healthy")))
print(os.listdir("plantvillage dataset/color/Grape___healthy")[:5])

base_dir = 'plantvillage dataset/color'

image_path = '/content/plantvillage
dataset/color/Apple___Cedar_apple_rust/025b2b9a-0ec4-4132-96ac-
7f2832d0db4a___FREC_C.Rust 3655.JPG'

# Read the image
img = mpimg.imread(image_path)

print(img.shape)
# Display the image
plt.imshow(img)
plt.axis('off')  # Turn off axis numbers
plt.show()

image_path = '/content/plantvillage
dataset/color/Apple___Cedar_apple_rust/025b2b9a-0ec4-4132-96ac-
7f2832d0db4a___FREC_C.Rust 3655.JPG'

# Read the image
img = mpimg.imread(image_path)

print(img)
```

```
# Image Parameters
img_size = 224
batch_size = 32

# Image Data Generators
data_gen = ImageDataGenerator(
    rescale=1./255,
    validation_split=0.2  # Use 20% of data for validation
)

# Train Generator
train_generator = data_gen.flow_from_directory(
    base_dir,
    target_size=(img_size, img_size),
    batch_size=batch_size,
    subset='training',
    class_mode='categorical'
)

# Validation Generator
validation_generator = data_gen.flow_from_directory(
    base_dir,
    target_size=(img_size, img_size),
    batch_size=batch_size,
    subset='validation',
    class_mode='categorical'
)

# Model Definition
model = models.Sequential()
```

```
model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(img_size, img_size, 3)))
model.add(layers.MaxPooling2D(2, 2))

model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D(2, 2))



model.add(layers.Flatten())
model.add(layers.Dense(256, activation='relu'))
model.add(layers.Dense(train_generator.num_classes, activation='softmax'))

# model summary
model.summary()

# Compile the Model
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])

# Training the Model
history = model.fit(
    train_generator,
    steps_per_epoch=train_generator.samples // batch_size,  # Number of steps per epoch
    epochs=5,  # Number of epochs
    validation_data=validation_generator,
    validation_steps=validation_generator.samples // batch_size  # Validation steps
    )
```

```
# Model Evaluation
print("Evaluating model...")
val_loss, val_accuracy = model.evaluate(validation_generator,
steps=validation_generator.samples // batch_size)
print(f"Validation Accuracy: {val_accuracy * 100:.2f}%")

# Plot training & validation accuracy values
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='upper left')
plt.show()

# Plot training & validation loss values
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='upper left')
plt.show()

# Function to Load and Preprocess the Image using Pillow
def load_and_preprocess_image(image_path, target_size=(224, 224)):
    # Load the image
    img = Image.open(image_path)
```

```
    # Resize the image
    img = img.resize(target_size)
    # Convert the image to a numpy array
    img_array = np.array(img)
    # Add batch dimension
    img_array = np.expand_dims(img_array, axis=0)
    # Scale the image values to [0, 1]
    img_array = img_array.astype('float32') / 255.
    return img_array

# Function to Predict the Class of an Image
def predict_image_class(model, image_path, class_indices):
    preprocessed_img = load_and_preprocess_image(image_path)
    predictions = model.predict(preprocessed_img)
    predicted_class_index = np.argmax(predictions, axis=1)[0]
    predicted_class_name = class_indices[predicted_class_index]
    return predicted_class_name

# Create a mapping from class indices to class names
class_indices = {v: k for k, v in train_generator.class_indices.items()}

class_indices

# saving the class names as json file
json.dump(class_indices, open('class_indices.json', 'w'))

# Example Usage
image_path = '/content/test_apple_black_rot.JPG'
#image_path = '/content/test_blueberry_healthy.jpg'
#image_path = '/content/test_potato_early_blight.jpg'
```

```
predicted_class_name = predict_image_class(model, image_path, class_indices)

# Output the result
print("Predicted Class Name:", predicted_class_name)

model.save('drive/MyDrive/plant_model.h5')
```

ДОДАТОК Б

ВІДГУК КЕРІВНИКА ЕКОНОМІЧНОГО РОЗДІЛУ

ДОДАТОК В

ПЕРЕЛІК ДОКУМЕНТІВ НА ОПТИЧНОМУ НОСІЇ

Ім’я файлу	Опис
Пояснювальні документи	
Кваліфікаційна робота Заіка М.І. 121-20-2.docx	Пояснювальна записка до кваліфікаційної роботи. Документ Word.
Кваліфікаційна робота Заіка М.І. 121-20-2.pdf	Пояснювальна записка до кваліфікаційної роботи в форматі PDF.
Програма	
AVGO VISION +ph	Директорія веб застосунку. Містить коди програми та файли тренування нейронної мережі.
Презентація	
Заіка М.І.pptx	Презентація кваліфікаційної роботи.