

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний аерокосмічний університет ім. М. Є. Жуковського
«Харківський авіаційний інститут»

Факультет програмної інженерії та бізнесу

Кафедра інженерії програмного забезпечення

Пояснювальна записка до дипломного проєкту

(тип кваліфікаційної роботи)

бакалавра

(освітній ступінь)

на тему «Програмне забезпечення попередньої обробки зображень для
визначення PROMPT на основі CLIP моделі»

XAI.603.641п.24В.121.206315 ПЗ

Виконав: студент (ка) 4 курсу групи 641п

Галузь знань 12 Інформаційні технології
(код та найменування)

Спеціальність 121 – Інженерія програмного
забезпечення

(код та найменування)

Освітня програма Інженерія програмного
забезпечення

(найменування)

Пісоцький Д.В.

(прізвище й ініціали студента(ки))

Керівник: Вдовітченко О.В.

(прізвище й ініціали)

Рецензент: Кантемир І. В.

(прізвище й ініціали)

Міністерство освіти і науки України
Національний аерокосмічний університет ім. М. Є. Жуковського
“Харківський авіаційний інститут”

Факультет програмної інженерії та бізнесу
(повне найменування)
Кафедра інженерії програмного забезпечення
(повне найменування)
Рівень вищої освіти перший (бакалаврський)
Галузь знань 12 – інформаційні технології
(код та найменування)
Спеціальність 121 – інженерія програмного забезпечення
(код та найменування)
Освітня програма інженерія програмного забезпечення

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Ігор ТУРКІН
(підпис) (ініціали та прізвище)
« ____ » _____ 202_ р.

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

_____ Пісоцькому Денису Вікторовичу

(прізвище, ім'я та по батькові)

1. Тема кваліфікаційної роботи «Програмне забезпечення попередньої обробки зображень для визначення PROMPT на основі CLIP моделі»
керівник кваліфікаційної роботи Вдовітченко Олександр Валерійович, к.т.н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом Університету № 570 від «25» квітня 2024 року

2. Термін подання здобувачем кваліфікаційної роботи 29.05.2024 р.

3. Вихідні дані до роботи, вимоги замовника до застосунку.

4. Зміст пояснювальної записки (перелік завдань, які потрібно розв'язати)

1) аналіз вимог до програмного забезпечення попередньої обробки зображень для визначення PROMPT на основі CLIP моделі;

2) проєктування програмного забезпечення попередньої обробки зображень для визначення PROMPT на основі CLIP моделі;

3) тестування ПЗ попередньої обробки зображень для визначення PROMPT на основі CLIP моделі;

4) економічне обґрунтування розроблення ПЗ попередньої обробки зображень для визначення PROMPT на основі CLIP моделі.

5. Перелік графічного матеріалу діаграма варіантів використання, діаграма класів, діаграма пакетів, дерево функцій, блок-схеми взаємодії, діаграми бізнес-моделей, діаграми бази даних, макети екранних форм

6. Консультанти розділів кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1	Вдовітченко О.В., доцент каф. 603		
2	Вдовітченко О.В., доцент каф. 603		
3	Вдовітченко О.В., доцент каф. 603		
4	Купріянова В.С., доц. каф. 601		

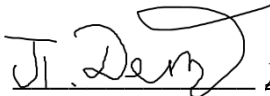
Нормоконтроль _____ Тетяна ДЕГТЯРЬОВА « 12 » 06 2024 р.
(підпис) (ініціали та прізвище)

7. Дата видачі завдання «2» жовтня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів кваліфікаційної роботи	Примітка
1	Отримання і затвердження теми кваліфікаційної роботи	01.07.2023 – 12.10.2023	Виконано
2	Аналіз предметної області	03.10.2023 – 22.10.2023	Виконано
3	Аналіз вимог до програмного забезпечення попередньої обробки зображень для визначення PROMPT на основі CLIP моделі	28.10.2023 – 20.12.2023	Виконано
4	Проектування програмного забезпечення попередньої обробки зображень для визначення PROMPT на основі CLIP моделі	01.01.2024 – 01.03.2024	Виконано
5	Тестування ПЗ попередньої обробки зображень для визначення PROMPT на основі CLIP моделі	08.03.2024 – 01.05.2024	Виконано
6	Економічне обґрунтування розроблення ПЗ попередньої обробки зображень для визначення PROMPT на основі CLIP моделі	08.05.2024 – 18.05.2024	Виконано
7	Оформлення пояснювальної записки	20.05.2024 – 25.05.2024	Виконано
8	Передзахист кваліфікаційної роботи	27.05.2024 - 7.06.2024	Виконано
9	Захист кваліфікаційної роботи	27.05.2024 - 7.06.2024	Виконано

Здобувач


(підпис)

Денис Пісоцький
(ініціали та прізвище)

Керівник кваліфікаційної роботи

Олександр Вдовітченко
(ініціали та прізвище)

підпис)

РЕФЕРАТ

Пояснювальна записка до дипломного проекту бакалавра: 167 сторінок, 60 рисунків, 52 таблиці, 69 посилань на літературні джерела.

Для поліпшення і прискорення роботи фірм, які займаються обробкою зображень для визначення PROMPT на основі CLIP моделі, передбачається створити програмне забезпечення ImgProPlus, яке буде являти собою веб-застосунок для попередньої обробки зображень. Цей застосунок призначений для автоматизації процесу підготовки зображень до визначення PROMPT за допомогою CLIP моделі. Застосунок дозволить прискорити обмін інформацією про обробку зображень між користувачами і системою. Даний продукт дозволить перейти від ручної обробки зображень до автоматизованого процесу. Веб-застосунок являє собою зручний спосіб попередньої обробки зображень з будь-якого пристрою, що має доступ до мережі Інтернет. Метою дипломного проекту є автоматизація, спрощення, а також зменшення витрат людських ресурсів на процес підготовки зображень для визначення PROMPT на основі CLIP моделі.

У процесі виконання дипломного проекту бакалавра був проведений аналіз вимог до програмного забезпечення попередньої обробки зображень, розроблені моделі процесів і даних, розроблені алгоритми для реалізації програмного забезпечення, розроблені модулі, проведено тестування програмного забезпечення, економічно обґрунтована розробка програмного забезпечення.

Проектування програмного забезпечення було виконано в середовищі розробки Visual Studio Code, а розробка програмного забезпечення була реалізована на мові програмування Python.

Для тестування інтернет-магазину з продажу і доставки їжі були розроблені плани системного (41), інтеграційного (14) і автономного (13) тестувань, розроблені специфікації для тестів, також виконано 9 тестів автономного, 6 тестів інтеграційного і 31 тесту системного тестування.

На етапі економічного обґрунтування розроблення програмного забезпечення попередньої обробки зображень для визначення PROMPT на основі CLIP моделі було визначено собівартість розробки, яка склала 182663,42 грн., відпускну ціну продажу всіх підписок з урахуванням ПДВ, яка склала 526848 грн., при цьому прибуток становить 188,42 % від собівартості, що відповідає 344184,57 грн.

Ключові слова: *CLIP, Interrogator, transformer, штучний інтелект, додаток, PROMPT, зображення.*

ABSTRACT

Explanatory note to the bachelor's diploma project: 167 pages, 60 figures, 52 tables, 69 references to literary sources.

In order to improve and speed up the work of image processing firms to determine PROMPT based on the CLIP model, it is planned to create the ImgProPlus software, which will be a web application for image preprocessing. This application is designed to automate the process of image preparation for PROMPT definition using the CLIP model. The application will speed up the exchange of image processing information between users and the system. This product will allow you to move from manual image processing to an automated process. The web application is a convenient way to pre-process images from any device that has access to the Internet. The goal of the diploma project is to automate, simplify, and reduce the cost of human resources for the process of image preparation for determining PROMPT based on the CLIP model.

In the process of completing the bachelor's diploma project, an analysis of requirements for image preprocessing software was carried out, process and data models were developed, algorithms for software implementation were developed, modules were developed, software testing was carried out, and economically justified software development.

Software design was performed in the Visual development environment Studio Code, and software development was implemented in the Python programming language.

To test the online store for food sales and delivery, plans for system (41), integration (14) and autonomous (13) testing were developed, test specifications were developed, 9 autonomous tests, 6 integration tests and 31 system testing tests were also performed.

At the economic substantiation stage of the development of image preprocessing software for determining PROMPT based on the CLIP model, the development cost was determined, which amounted to UAH 182,663.42, the selling price of all subscriptions including VAT, which amounted to UAH 526,848, while the profit is 188.42% of the cost price, which corresponds to UAH 344,184.57.

Keywords: *CLIP, Interrogator, transformer, artificial intelligence, application, PROMPT, image.*

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ ВИМІРЮВАНЬ ФІЗИЧНИХ ВЕЛИЧИН, СКОРОЧЕНЬ І ТЕРМІНІВ	9
ВСТУП	10
1 АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ПОПЕРЕДНЬОЇ ОБРОБКИ ЗОБРАЖЕНЬ ДЛЯ ВИЗНАЧЕННЯ PROMPT НА ОСНОВІ SLIP МОДЕЛІ	12
1.1 Бізнес-вимоги	12
1.1.1 Загальний опис програмного продукту та бізнес-вимоги.....	12
1.1.2 Розробка глосарію	12
1.1.3 Огляд і аналіз існуючих аналогів програмного забезпечення.....	14
1.1.4 Концепція програмного продукту та обсяг проекту.	16
1.1.5 Бізнес-правила	17
1.2 Вимоги користувачів	18
1.2.1 Характеристики класів користувачів	18
1.2.2 Діаграма класів використання	19
1.2.3 Сценарії варіантів використання	20
1.3 Вимоги до програмного забезпечення	24
1.3.1 Функціональні вимоги до програмного забезпечення	24
1.3.1.1 Адміністративні функції.....	25
1.3.1.2 Інформативні функції	26
1.3.1.3 Обробка зображень	26
1.3.1.4 Платіжні функції	27
1.3.2 Нефункціональні вимоги до ПЗ	28
1.3.2.1 Вимоги до зовнішніх інтерфейсів	28
1.3.2.1.1 Інтерфейси користувачів	28
1.3.2.1.2 Інтерфейси програмного забезпечення.....	36
1.3.2.1.3 Комунікаційні інтерфейси.....	37
1.3.2.2 Вимоги до показників якості	37
1.3.2.3 Обмеження.....	39
1.4 Матриця відстеження вимог	41
Висновки до розділу 1	43
2 ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ПОПЕРЕДНЬОЇ ОБРОБКИ ЗОБРАЖЕНЬ ДЛЯ ВИЗНАЧЕННЯ PROMPT НА ОСНОВІ SLIP МОДЕЛІ	43
2.1 Архітектурне проєктування.....	43
2.1.1. Методологія розробки програмного забезпечення	43
2.1.2. Загальна модель/стиль архітектури програмного забезпечення.....	45

2.1.3. Проєктування підсистеми збереження даних	48
2.1.3.1. Вибір моделі даних та підходу для проєктування БД	48
2.1.3.2. Проєктування та розробка моделі даних.....	49
2.1.3.3. Взаємодія з БД	54
2.1.3.4. Розробка запитів до БД	55
2.1.4 Проєктування взаємодії з зовнішніми сервісами	59
2.2. Детальне проєктування	65
2.2.1. Структурні моделі.....	67
2.2.2. Моделювання функціональності та взаємодії	69
2.3. Проєктування інтерфейсу користувача	74
2.4. Реалізація програмного забезпечення	80
Висновки до розділу 2	87
3 ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ПОПЕРЕДНЬОЇ ОБРОБКИ PROMPT НА ОСНОВІ SLIP МОДЕЛІ	88
3.1. Модульне тестування специфікованих модулів.....	88
3.1.1. Стратегія та інструментальні засоби модульного тестування	88
3.1.2. План модульного тестування.....	88
3.1.3. Висновки щодо модульного тестування.....	105
3.2. Інтеграційне тестування специфікованих модулів	106
3.2.1. Програмні та інструментальні засоби інтеграційного тестування	106
3.2.2. План інтеграційного тестування	107
3.2.3. Результати інтеграційного тестування	112
3.2.4. Висновки щодо інтеграційного тестування	113
3.3. Тестування реалізації вимог до ПЗ	113
3.3.1. Програмні та інструментальні засоби.....	113
3.3.2. План тестування реалізації функціональних вимог	113
3.3.3. План тестування реалізації нефункціональних вимог	124
3.3.4. Матриця відповідності вимог та тестів	128
3.3.5. Результати тестування реалізації вимог до ПЗ	132
3.3.6. Висновки щодо тестування реалізації вимог до ПЗ	133
3.4. Тестування реалізації варіантів використання	134
3.4.1. План тестування реалізації варіантів використання	134
3.4.2. Результати тестування реалізації варіантів використання	140
3.4.3. Висновки щодо тестування реалізації варіантів використання	141
Висновки до розділу 3	141
4 ЕКОНОМІЧНЕ ОБҐРУНТУВАННЯ ПРОЕКТУ	142
4.1. Опис, споживацька ціна та аналіз конкурентоспроможності програмного продукту.....	142
4.1.1. Оцінювання ринку збуту та конкуренція	142

4.1.2. Виявлення конкурентів та аналіз конкурентоспроможності програмного продукту	144
4.1.3. Розрахунок споживацької ціни програмного продукту, уточнення цільової місткості ринку	146
4.2. Виробничо-організаційний план проекту. Розрахунок трудомісткості проекту та кількості його виконавців	147
4.2.1. Перелік робіт	147
4.2.2. Розрахунки трудомісткості робіт проекту	148
4.2.3. Виконавці проекту	150
4.3. Фінансовий план проекту	150
4.3.1 Розрахунок потреб в інвестиціях на обладнання.....	150
4.3.2 Розрахунок заробітної плати виконавців проекту	151
4.3.3 Складання кошторису витрат на розроблення проекту програмного продукту	152
4.3.6 План доходів від реалізації програмного продукту та витрат на його виготовлення	157
Висновки до розділу 4	158
ВИСНОВКИ.....	160
ПЕРЕЛІК ПОСИЛАНЬ	161
<i>ДОДАТОК А – КОД ПРОГРАМИ</i>	167
<i>ДОДАТОК Б – ЗАПИТИ ДО БД</i>	335
<i>ДОДАТОК В – КОД ТЕСТІВ</i>	340

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ ВИМІРЮВАНЬ ФІЗИЧНИХ ВЕЛИЧИН, СКОРОЧЕНЬ І ТЕРМІНІВ

3D – Three-Dimensional;
API – Application Programming Interface;
CLIP – Contrastive Language-Image Pretraining;
CPU – Central Processing Unit;
ER – Entity-Relationship;
GPU – Graphical Processing Unit;
HTTPS – Hypertext Transfer Protocol Secure;
JSON – JavaScript Object Notation;
PROMPT – Programmed Retrieval Operations for Machine Processing and Translation;
SQL – Structured Query Language;
SSL – Secure Sockets Layer;
TLS – Transport Layer Security;
UI – User Interface;
БД – база даних;
ООП – об'єктно-орієнтоване програмування;
ОС – операційна система;
ПЗ – програмне забезпечення;
СКБД – система керування базами даних;
ІІ – Штучний інтелект.

ВСТУП

В сучасному інформаційному суспільстві та науці комп'ютерних наук, технології швидко розвиваються, надаючи нові можливості у багатьох аспектах життя людини. Однією з ключових галузей є обробка зображень та розпізнавання тексту, які відіграють важливу роль в таких сферах, як комп'ютерний зір, медицина, автономні автомобілі, мультимедіа та ігрова індустрія. [1]

В рамках цього дослідження, ми розглядаємо тему "Програмне забезпечення попередньої обробки зображень для визначення запитів на основі моделі CLIP" [2] та проводимо обґрунтування її актуальності, значущості та ступеня новизни.

Завдання, пов'язані з розпізнаванням тексту та зображень, набувають все більшого значення в контексті сучасних технологій. Актуальність цієї теми обумовлена рядом факторів.

По-перше, використання моделі CLIP виявляється дуже перспективним у контексті обробки зображень. Модель CLIP дозволяє розуміти семантику зображень, використовуючи текстові описи, і надає можливість широкого спектру застосувань. Вона може бути використана для підвищення точності пошуку та індексації зображень, а також для автоматичної генерації текстових описів для зображень, що важливо в мультимедійних системах. [3]

По-друге, ігрова індустрія є однією з галузей, де розпізнавання тексту та обробка зображень грають ключову роль. Від реалістичних ігрових світів до віртуальних декорацій, ігри вимагають розвитку технологій, які забезпечують високу якість візуалізації та інтеракції. Можливості створення реалістичних об'єктів і сценаріїв у грах дуже значущі. [4]

По-третє, технології «Generative from text to 3D» [5] моделі штучного інтелекту визначають майбутнє. Це означає, що на основі текстових описів можна буде генерувати 3D-моделі об'єктів, і це стане перспективним напрямком для вирішення завдань в ігровій індустрії, виготовленні декорацій та багатьох інших сферах. Такий підхід може значно заощадити час і ресурси, що важливо в конкурентному світі.

Також програмне забезпечення для попередньої обробки зображень має ряд важливих переваг, що робить його привабливим для широкого кола користувачів. Серед особливостей цього програмного забезпечення можна виділити наступні:

Ефективність. Для обробки зображень та визначення PROMPT не потрібно витрачати багато часу на ручну обробку або складні процедури. Програмне забезпечення надає зручні і швидкі інструменти для виконання цих завдань, що дозволяє ефективно використовувати час користувачів.

Зручність. Використання програмного забезпечення для попередньої обробки зображень дозволяє працювати з ними в будь-якому місці та в будь-який час, де є доступ до комп'ютера або мобільного пристрою. Це забезпечує зручність і гнучкість у роботі з зображеннями.

Автоматизація. Програмне забезпечення надає можливість автоматизувати багато рутинних операцій з обробки зображень, що дозволяє значно зекономити час і зусилля користувачів. [6]

Точність. Використання CLIP моделі для визначення PROMPT забезпечує високу точність результатів. Програмне забезпечення допомагає уникнути помилок та забезпечує надійність у визначенні результатів.

Цією технологією користуються як професіонали в області обробки зображень, так і прості користувачі, які потребують швидкого і надійного інструменту для роботи з зображеннями.

1 АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ПОПЕРЕДНЬОЇ ОБРОБКИ ЗОБРАЖЕНЬ ДЛЯ ВИЗНАЧЕННЯ PROMPT НА ОСНОВІ CLIP МОДЕЛІ

1.1 Бізнес-вимоги

1.1.1 Загальний опис програмного продукту та бізнес-вимоги

«ImgProPlus» – це новий веб-додаток, який надає інноваційну платформу для попередньої обробки зображень та визначення PROMPT на основі CLIP моделі. Метою розробки є створення системи, яка дозволить користувачам ефективно використовувати передові технології для аналізу та опрацювання зображень.

Замовником проекту є компанія «Image Intelligence Solutions» з метою покращення якості обробки зображень та розширення можливостей використання CLIP моделі в різних галузях, таких як обробка фотографії, 3D-моделювання, тощо.

Бізнес-вимоги, які стосуються програмного продукту «ImgProPlus», наведено в таблиці 1.1. (див. табл. 1.1)

Таблиця 1.1 – Бізнес-вимоги до додатку.

ID	Опис бізнес-вимог
BR-01	Забезпечення ефективної попередньої обробки зображень для користувачів з різним рівнем експертизи.
BR-02	Можливість отримання прибутку компанією «Image Intelligence Solutions» через надання розширених функцій та послуг, доступних за плату.

1.1.2 Розробка глосарію

Додаток (Application) – інтерактивний програмний продукт, який використовується через комп'ютер користувача для визначення PROMPT.

Попереднє навчання (Pre-training): Процес, під час якого модель вивчає універсальні представлення зображень та тексту перед вирішенням конкретного завдання. [7]

Векторне представлення (Vector Representation): Унікальне числове представлення об'єкта або концепції, яке використовується для подальшого аналізу та порівняння.

Трансформер (Transformer): Архітектура нейронної мережі, яка використовує механізм самоуваги для ефективної обробки послідовностей.

Генерація підказок (Prompt Engineering): Процес пошуку найкращих текстових підказок для досягнення бажаного виведення в алгоритмі PROMPT.

Контрастне попереднє навчання (Contrastive Pre-training): Техніка, де модель вивчає розрізнення між подібними та відмінними об'єктами, щоб отримати більш інформативні представлення.

Мовно-зображувальні представлення (Language-Visual Representations): Векторні представлення, що об'єднують текстові та візуальні концепції для підвищення здатностей алгоритму в розумінні зв'язків між ними.

Адаптивні алгоритми (Adaptive Algorithms): Алгоритми, які можуть автоматично змінювати свою поведінку та параметри згідно з новими даними та умовами для оптимальної ефективності. [8]

Автоматичне оцінювання зображень (Automated Image Assessment): Використання алгоритмів штучного інтелекту для автоматичної оцінки та аналізу зображень з використанням моделі CLIP.

Метод самоуваги (Self-Attention Mechanism): Частина архітектури трансформерів, яка дозволяє моделі взаємодіяти з різними частинами вхідного сигналу на різних рівнях представлення.

Технології обробки природної мови (Natural Language Processing Technologies): Інструменти та методи, що використовуються для аналізу та розуміння людської мови, [9] що покращують взаємодію з текстовою інформацією в контексті вивчення англійської мови за допомогою CLIP.

Штучний інтелект (Artificial Intelligence): Галузь комп'ютерних наук, яка вивчає створення систем, здатних до виконання завдань, які зазвичай вимагають людського інтелекту, таких як розпізнавання образів, мови та прийняття рішень.

Обробка Зображень (Image Processing): Процес застосування різноманітних технік та алгоритмів для оптимізації та покращення зображень перед їх подальшим аналізом.

Модуль Попередньої Обробки (Preprocessing Module): Частина програмного продукту, що відповідає за виконання операцій з обробки та підготовки зображень для подальшого використання в CLIP моделі.

Подання Завдань (Task Representation): Процес конвертації вхідних завдань або запитань у формат, зрозумілий CLIP моделі для ефективного визначення PROMPT.

Векторне Представлення (Vector Representation): Перетворення інформації про зображення в векторний формат для подальшого використання у моделі.

Навчання та Калібрування (Training and Calibration): Процес використання навчальних даних для оптимізації параметрів CLIP моделі та адаптації її до конкретного завдання.

Термін (Term): Умове позначення для слова чи концепції, яке піддається аналізу та визначенню за допомогою CLIP моделі.

Активний Користувач (Active User): Особа, яка використовує програмне забезпечення для попередньої обробки зображень та визначення PROMPT.

1.1.3 Огляд і аналіз існуючих аналогів програмного забезпечення

ImageToPrompt (див. рис. 1.1) – електронна платформа для обробки зображень та визначення PROMPT на основі CLIP моделі, яка пропонує як безкоштовний доступ. Сервіс розроблений так, щоб надавати користувачам можливість створити та оптимізувати відповіді PROMPT для використання в штучному інтелекті.

На відміну від інших сервісів, цей є повністю безкоштовним та працює за лічені секунди.

Проблемами цього сервісу є те, що він дозволяє обробляти картинки лише до 4 MB, та по одній за раз. Також є проблемою, що сайт працює повільно, хоча алгоритм знаходження PROMPT працює швидко, сам сайт відкривається дуже довго та повільно. До уваги також береться те, що PROMPT не завжди якісний, що приводить до проблем в подальшому використанні.

На відміну від imagetoprompt.com у нашому застосунку можна буде використовувати всі можливості вашого комп'ютера, для створення PROMPT для декількох картинок одночасно, також так як наш додаток це локальна версія, то користувачі не будуть переживати проблеми очікування PROMPT або те, що сайт не працює.

ImageToPrompt.com



AI Generated Prompt

a close up of a person wearing a scarf, mr bean face, beans in his eyes sockets, mrs bean, wry smirk, joyful smirk, insane face, bet face, roguish smirk, he looks like a human minion, large eyes and menacing smile, cynical face, very slightly smiling, giddy smirk

[Try another image >](#)

Created by Charlie Holtz



Рисунок 1.1 – Вигляд сайту ImageToPrompt

PromptoMania (див. рис. 1.2) – це інноваційна платформа для обробки зображень та створення PROMPT на основі CLIP моделі. Спеціалізована на вивченні та оптимізації відповідей PROMPT, PromptoMania надає унікальні інструменти для кастомізації та деталізації ваших промптів відповідно до вибраної моделі штучного інтелекту.

Користувачі PromptoMania можуть не лише взаємодіяти з інтуїтивним інтерфейсом, але й створювати свої унікальні PROMPT, використовуючи власні зображення та адаптуючи їх до своїх потреб. Платформа також дозволяє вибирати та кастомізувати PROMPT згідно зі специфікаціями вибраної моделі, надаючи більшу гнучкість у навчанні та використанні штучного інтелекту. [10]

PromptoMania пропонує дві версії: безкоштовну та платну. Безкоштовна версія містить базові функції, але має обмеження на обробку великої кількості зображень та кастомізацію PROMPT. Платна версія, у свою чергу, забезпечує більше можливостей, працює ефективніше та швидше, а також надає доступ до розширених функцій.

Мінусами "promptomania.com" є обмеження безкоштовної версії, яке включає обмеження на кількість оброблених зображень та обмежені можливості кастомізації PROMPT. Також, користувачам безкоштовної версії може бути надано обмежений доступ до деяких розширених функцій.

На відміну від інших аналогів, PromptoMania дає гнучкий інструментарій для створення PROMPT.

У нашому застосунку буде реалізована частково функція кастомізації PROMPT та можливість за додаткову плату отримувати додаткові функції.

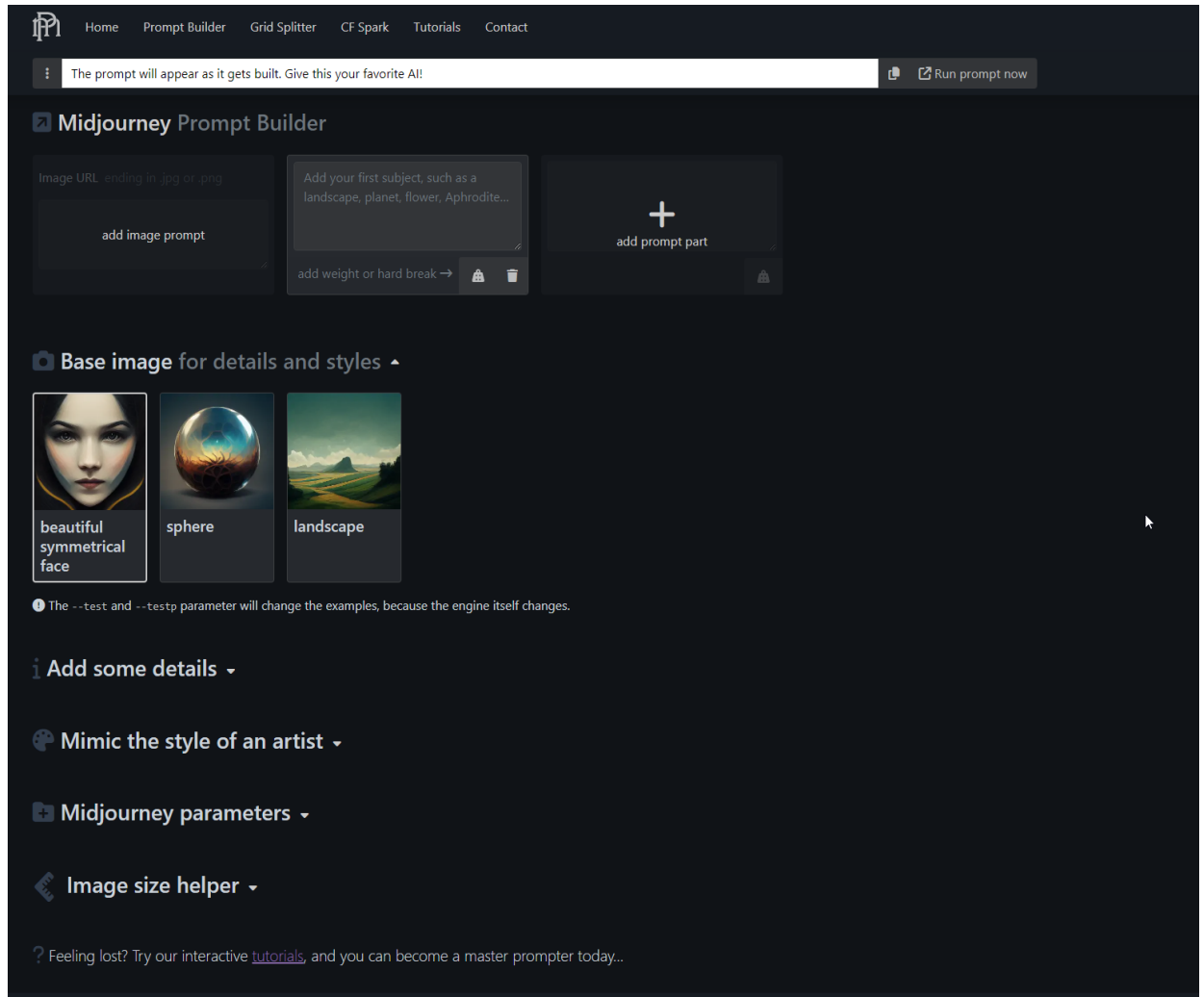


Рисунок 1.2 – Вигляд сайту PromptMania.

1.1.4 Концепція програмного продукту та обсяг проекту.

ImgProPlus є програмним забезпеченням для ПК для компанії «Image Intelligence Solutions», яка прагне покращити обробку зображень і покращити якість своїх послуг за допомогою технології CLIP. Він автоматизує обробку та управління зображеннями. На відміну від існуючих програм, які зараз використовує компанія, ImgProPlus буде інтегровано з провідними технологічними рішеннями та автоматизує весь процес роботи з зображеннями.

Крім того, він надає інструменти для аналізу та вдосконалення роботи з клієнтами, а також систему передбачення та управління ресурсами, щоб максимізувати використання оброблювальної потужності.

Функції які будуть реалізовані:

- Можливість завантаження та програмної обробки зображень для створення вдосконалених PROMPT.
- Візуалізація та аналіз результатів, отриманих від CLIP моделі.
- Використання власних зображень та можливість адаптації їх до потреб користувача.
- Вивчення та оптимізація роботи з PROMPT через розділ правил та документації.

Функції які не будуть реалізовані:

- Кастомізація відповідно до обраної моделі.

1.1.5 Бізнес-правила

Бізнес-правила, які стосуються ПЗ «ImgProPlus» наведено в таблиці. (див. табл. 1.2)

Таблиця 1.2– Бізнес-правила

Ідентифікатор	Визначення правила	Тип правила	Джерело (походження)
BRL-1	Використання зображень та їх попередня обробка для створення PROMPT	Факт	Методи визначення PROMPT
BRL-2	При створенні PROMPT можна використовувати аналіз та обробку зображень.	Факт	Методи визначення PROMPT
BRL-3	Під час процесу визначення PROMPT необхідно використовувати правила та алгоритми на основі CLIP моделі.	Факт	Методи визначення PROMPT

Кінець таблиці 1.2

Ідентифікатор	Визначення правила	Тип правила	Джерело (походження)
BRL-4	При використанні програми можна адаптувати параметри обробки для досягнення оптимальних результатів.	Факт	Методи визначення PROMPT
BRL-5	Використання словника і документації для визначення термінів та понять у контексті визначення PROMPT	Факт	Методи визначення PROMPT
BRL-6	Забезпечення контролю над ефективністю використання PROMPT шляхом включення механізмів оцінки та відстеження результатів	Факт	Методи визначення PROMPT

1.2 Вимоги користувачів

1.2.1 Характеристики класів користувачів

Загальні характеристики продукту:

Продукт є програмним забезпеченням для попередньої обробки зображень для визначення PROMPT на основі CLIP моделі.

Основні функції включають завантаження, обробку зображень та визначення PROMPT з використанням CLIP моделі.

Продукт має інтерактивний інтерфейс для користувачів та інтеграцію з CLIP моделлю для оптимального визначення PROMPT.

Таблиця 1.3 – Класи користувачів.

Класи користувачів	Опис
Неавторизований користувач	Неавторизований користувач – користувач який тільки зайшов в додаток, та який повинен авторизуватись для отримання відповідної ролі.
Користувач без підписки	Користувач без підписки – це користувач додатку, який може користуватись можливостями додатку, якщо використовує ресурси своєї машини. У випадку запитів до сервера, кількість опрацювань обмежена.
Користувач з підпискою	Користувач з підпискою – це користувач додатку, який має доступ до всіх нових оновлень, алгоритмів та має безлімітний доступ до сервера, для опрацювань картинок.
Адміністратор	Адміністратор – це людина, у якої є спеціальна версія програми, що дозволяє взаємодіяти з базою даних, та відповідно до запитів користувачів або розробників, надавати необхідну інформацію (розробник автоматично і є адміністратором).

1.2.2 Діаграма класів використання

Відповідно до програмного забезпечення була розроблена діаграма (див. рис. 1.3) використання в якій задіяно 4 особи (неавторизований користувач, авторизований користувач без підписки, авторизований користувач з підпискою та адміністратор), задіяно 1 API та 7 процесів. Більш детальніше про алгоритми використання буде розписано в наступному розділі.

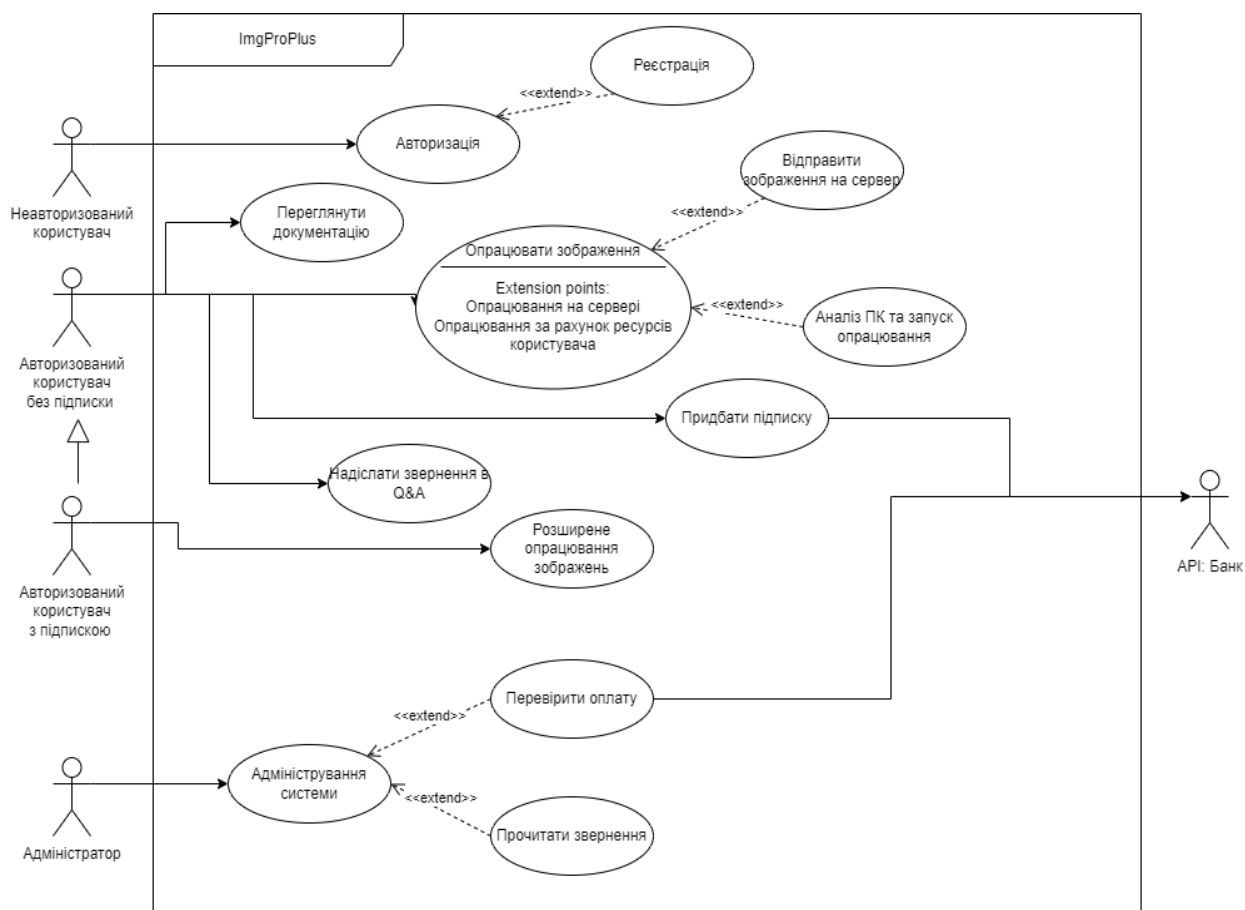


Рисунок 1.3 – Діаграма використання ПЗ для попередньої обробки зображень для визначення PROMPT на основі CLIP моделі

1.2.3 Сценарії варіантів використання

Сценарії використання неавторизованого користувача. (Авторизація)

Таблиця 1.4 – Варіант використання «Авторизація»

ID та назва:	UC-1 Авторизація
Основна дійова особа	Неавторизований користувач
Додаткові дійові особи	–
Тригер:	Ввімкнення програми
Попередні умови:	–
Нормальний напрямок розвитку варіанта використання:	<p>1.0 Авторизуватися</p> <p>1. Користувач вводить логін (пошту) та пароль.</p> <p>2. У випадку відсутності реєстрації, користувач має зареєструватися (див. 1.1)</p> <p>3. При натисканні вхід, додаток надсилає запит до серверу.</p> <p>4. Відповідь від серверу отримана, користувач зайшов в додаток. (див. 1.0.E1)</p>
Альтернативні напрямки розвитку варіанту використання:	<p>1.1 Реєстрація</p> <p>1. Користувач вводить пошту та пароль</p> <p>2. Користувачу приходить код підтвердження електронної пошти</p> <p>3. Користувач вводить код, та успішно реєструється в додаткові.</p>
Винятки:	<p>1.0.E1 Сервер не працює</p> <p>1а. Після запиту авторизації, користувач отримує повідомлення про неможливість авторизації та прохання спробувати пізніше.</p>
Вихідні умови:	POST-1. Користувач зайшов до системи.
Бізнес-правила:	–
Нефункціональні вимоги:	<p>1. Функціонал має бути доступний 24\7.</p> <p>2. Функціонал авторизації має бути захищений від несанкціонованого доступу до даних користувачів.</p>

Таблиця 1.5 – Варіант використання «Опрацювати зображення»

ID та назва:	UC-2 Опрацювати зображення
Основна дійова особа	Авторизований користувач з підпискою або без
Додаткові дійові особи	–

Кінець таблиці 1.5

ID та назва:	UC-2 Опрацювати зображення
Тригер:	Користувач натиснув завантажити зображення.
Попередні умови:	PRE-1. Завантажена картинка відповідає вимогам та користувач обрав метод опрацювання (локальний чи на сервері) (див 2.0.E1)
Нормальний напрямок розвитку варіанта використання:	2.0 Опрацювати зображення 1. Користувач завантажує зображення до додатку. 2. Користувач вибирає варіант опрацювання серверне (див. 2.1) 3. Користувач обирає варіант опрацювання локальне (див. 2.2) 4. Користувач отримав результат у вигляді промптів.
Альтернативні напрямки розвитку варіанту використання:	2.1 Опрацювати зображення на сервері 1. Додаток відправляє зображення на сервер, той у відповідь надсилає опрацьований PROMPT. (див. 2.1.E1) 2. Користувач отримує у додатку необхідний PROMPT. 2.2 Опрацювати зображення локально 1. Додаток опрацьовує зображення користувача і видає необхідний PROMPT.
Винятки:	2.0.E1. Користувач немає підписки і його безкоштовні «кредити» закінчились 1а. Користувач отримує повідомлення про нестачу та пропозицію оформити підписку 2а. Запит на обробку зображень можна перенести на локальну версію, щоб користувач зміг опрацювати на своєму ПК 2.1.E1. Сервер не зміг опрацювати зображення 1а. Користувач отримує повідомлення про неможливість опрацювання зображення на сервері. 2а. Користувачеві надається «міні-рапорт» з помилкою, та можливість відправити її розробникам по натисканню кнопки.
Вихідні умови:	POST-1. Користувач опрацював відправлене зображення та отримав з нього PROMPT. POST-2. У випадку опрацювання на сервері, користувач який не мав підписки втрачає 1 «кредит».
Бізнес-правила:	BRL-2, BRL-3, BRL-4
Нефункціональні вимоги:	1. Функціонал має бути доступний 24\7. 2. Функціонал опрацювання має бути захищений від несанкціонованого доступу до даних користувачів (від перехвату зображень).

Таблиця 1.6 – Варіант використання «Придбати підписку»

ID та назва:	UC-3 Придбати підписку
Основна дійова особа	Авторизований користувач
Додаткові дійові особи	API: Банк
Тригер:	Авторизований користувач натиснув кнопку купити підписку
Попередні умови:	PRE-1. Користувач авторизувався до додатку
Нормальний напрямок розвитку варіанта використання:	3.0 Придбати підписку 1. Користувач отримує інформацію про наявні підписки та їх вартість. 2. Користувач обирає підписку і його термін. Система розраховує вартість. 3. Користувач вводить дані банку, або використовує вбудовану систему (GooglePay, ApplePay). (див. 3.0.E1) 4. Користувач отримує статус підписника на місяць (див. 3.0.E2)
Альтернативні напрямки розвитку варіанту використання:	—
Винятки:	3.0.E1 Користувач ввів не правильні дані 1а. За рахунок API Банку користувач отримає в додаткові повідомлення про неправильність даних 3.0.E2 Користувач не має достатню кількість коштів, та отримує про це повідомлення в додаткові.
Вихідні умови:	POST-1. Користувач опрацював отримав підписку POST-2. В базі даних з'явився користувач з підпискою.
Бізнес-правила:	BRL-5
Нефункціональні вимоги:	1. Функціонал має бути доступний 24\7. 2. Функціонал придбання має бути захищений від несанкціонованого доступу до даних користувачів (номер карти).

1.3 Вимоги до програмного забезпечення

1.3.1 Функціональні вимоги до програмного забезпечення

Я побудував відповідне дерево функцій для «ПЗ попередньої обробки зображень для визначення PROMPT на основі CLIP моделі». Виділив 4 відповідні функції: адміністративні, інформативні, обробки зображення та платіжні. Такий розподіл дозволяє зручно і ефективно розподілити функції програмного забезпечення. [11]

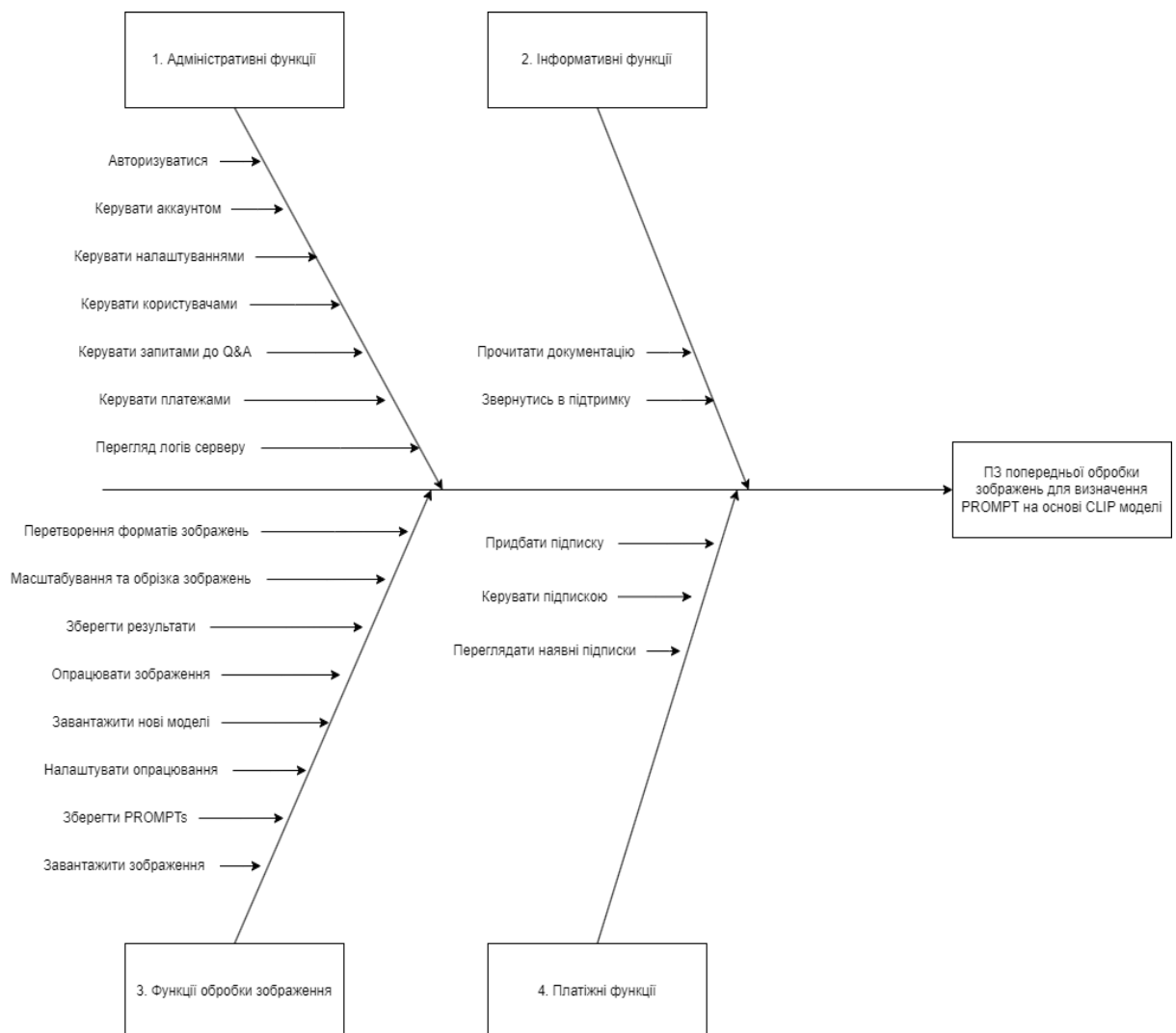


Рисунок 1.4 – Дерево функцій до ПЗ

1.3.1.1 Адміністративні функції

Група функцій "Адміністративні функції" включає в себе функції, які стосуються управління та налаштуванням різних аспектів системи. Ці функції надають можливість користувачам, зокрема адміністраторам та звичайним користувачам, ефективно керувати різними параметрами системи. (наведено в табл. 1.5) [12]

Таблиця 1.5 – Функціональні вимоги групи FR1

ID	Опис вимог
FR1.01	Система повинна мати змогу авторизувати користувачів
FR1.01.01	При вході в додаток користувач має можливість авторизуватись.
FR1.01.02	Система повинна дозволяти користувачеві зареєструватись за допомогою пошти і пароллю. Користувач має отримати код підтвердження.
FR1.01.03	Користувач має отримувати повідомлення або отримувати візуальну помилку (помічати червоним кольором), при неправильному введенні даних.
FR1.02	Система повинна надавати керування аккаунтом користувача.
FR1.03	Система повинна забезпечувати можливість користувачам налаштовувати параметри свого облікового запису.
FR1.04	Система повинна дозволяти адміністраторам керувати користувачами та їх обліковими записами
FR1.04.01	Адміністратор має містити можливість створення «тестового» користувача та перегляду списку існуючих користувачів з можливістю редагування і видалення.
FR1.04.02	При видаленні користувача адміністратором, система повинна відображати підтверджувальне вікно з двома кнопками: «Підтвердити видалення» та «Скасувати»
FR1.05	Система повинна дозволяти адміністраторам керувати запитаннями та відповідями.
FR1.06	Система повинна дозволяти адміністраторам керувати платежами.
FR1.06.01	Сторінка управління платежами має містити можливість перегляду та редагування списку платежів.
FR1.06.02	Сторінка деталізації платежів повинна включати інформацію про сам платіж та всі пов'язані з ним дані.
FR1.07	Система повинна надавати можливість адміністраторам переглядати логи серверу.

1.3.1.2 Інформативні функції

Група функцій "Інформативні функції" включає в себе функції, які стосуються надання інформації та документації щодо різних аспектів системи. Ці функції надають можливість користувача отримувати інформацію та розуміти основні аспекти системи. Деталізація функцій даної групи наведена у таблиці нижче (див. табл. 1.6):

Таблиця 1.6 – Інформативні функції

ID	Опис вимог
FR2.01	Система повинна давати можливість користувачам прочитати документацію додатку, що включає інформацію щодо функціоналу, налаштувань та інструкцій з використання додатку.
FR2.02	Система повинна дозволяти користувачам звертатися в підтримку з доступом до ресурсів отримання допомоги.

1.3.1.3 Обробка зображень

Група функцій "Функції обробки зображень" включає в себе функції, які стосуються опрацювання та трансформації зображень. Ці функції надають можливість користувачам ефективно керувати різними параметрами обробки зображень. Деталізація функцій даної групи наведена у таблиці нижче (див. табл. 1.7):

Таблиця 1.7 – Функції обробки зображень

ID	Опис вимог
FR3.01	Система повинна мати можливість перетворювати різні формати зображень, при завантаженні.
FR3.02	Система повинна дозволяти користувачами зберігати результати обробки зображень
FR3.03	Система повинна мати можливість опрацьовувати зображення за допомогою завантажених моделей обробки (наприклад CLIP v1, v2 і т.д.).
FR3.04	Система має містити можливість вибору тип опрацювання «На сервері» або «На локальній машині»..
FR3.04.01	При виборі «На сервері», якщо не має достатню кількість кредитів і не має підписку, система має відобразити повідомлення про це.

Кінець таблиці 1.7

ID	Опис вимог
FR3.04.02	При виборі «На локальній машині», користувачу відобразиться повідомлення, про те, що на його ПК буде проведено перевірку потужності, та після цього відображено результати.
FR3.05	Система повинна надавати можливість налаштовувати опрацювання зображення. Параметри налаштування, такі як «Кількість кроків», «Негативний PROMPT», «Креативність» тощо.
FR3.06	Система повинна надавати можливість зберегти результати опрацювання зображення згідно з заданими PROMPTs (інструкціями).
FR3.07	Система повинна дозволяти завантажувати нові моделі обробки зображень (наприклад, CLIP v1, v2 і т.д.).
FR3.07.01	Система має надавати вибирати модель із наявних моделей.
FR3.07.02	Система у випадку, якщо користувач не має підписку, то йому при спробі завантажити моделі, відображається повідомлення, яке пояснює, що ця можливість наявна лише з підпискою, та пропонує її купити.
FR3.08	Система повинна надавати можливість завантажувати зображення для подальшої обробки.

1.3.1.4 Платіжні функції

Група функцій "Платіжні функції" включає в себе функції, які стосуються управління та налаштуванням різних аспектів платіжної системи. Ці функції надають можливість користувачам ефективно керувати різними параметрами платіжної системи. Деталізація функцій даної групи наведена у таблиці нижче (див. табл. 1.8): [13]

Таблиця 1.8 – Платіжні функції

ID	Опис вимог
FR4.01	Система повинна надавати можливість користувачам придбати підписку за переліком можливих підписок.
FR4.02	Система повинна давати можливість користувачам переглядати наявні підписки. Користувачеві має відобразитись наявні підписки у вигляді списку, з якого він може вибирати відповідно. Також мають бути вказані ціни, та знижки (якщо такі наявні).
FR4.03	Система повинна дозволяти користувачам керувати своєю підпискою.

1.3.2 Нефункціональні вимоги до ПЗ

1.3.2.1 Вимоги до зовнішніх інтерфейсів

1.3.2.1.1 Інтерфейси користувачів

Відповідно до «ПЗ попередньої обробки зображень на основі CLIP моделі», було побудовано карту діалогових вікон (див. рис. 1.5).

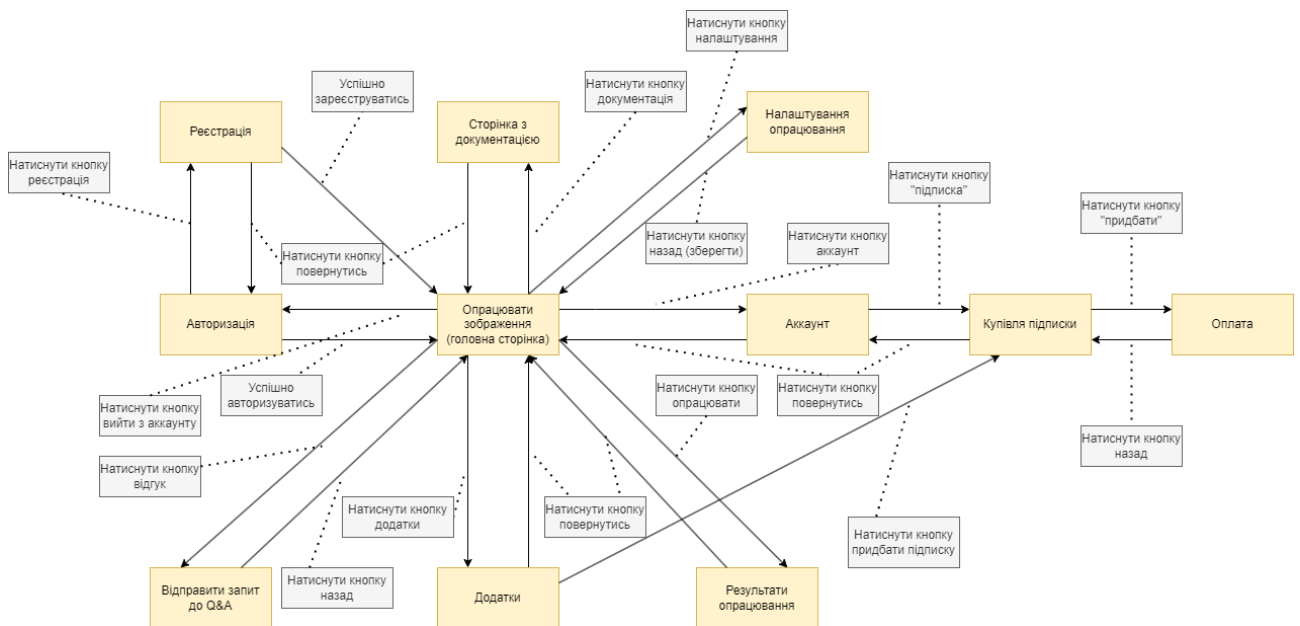
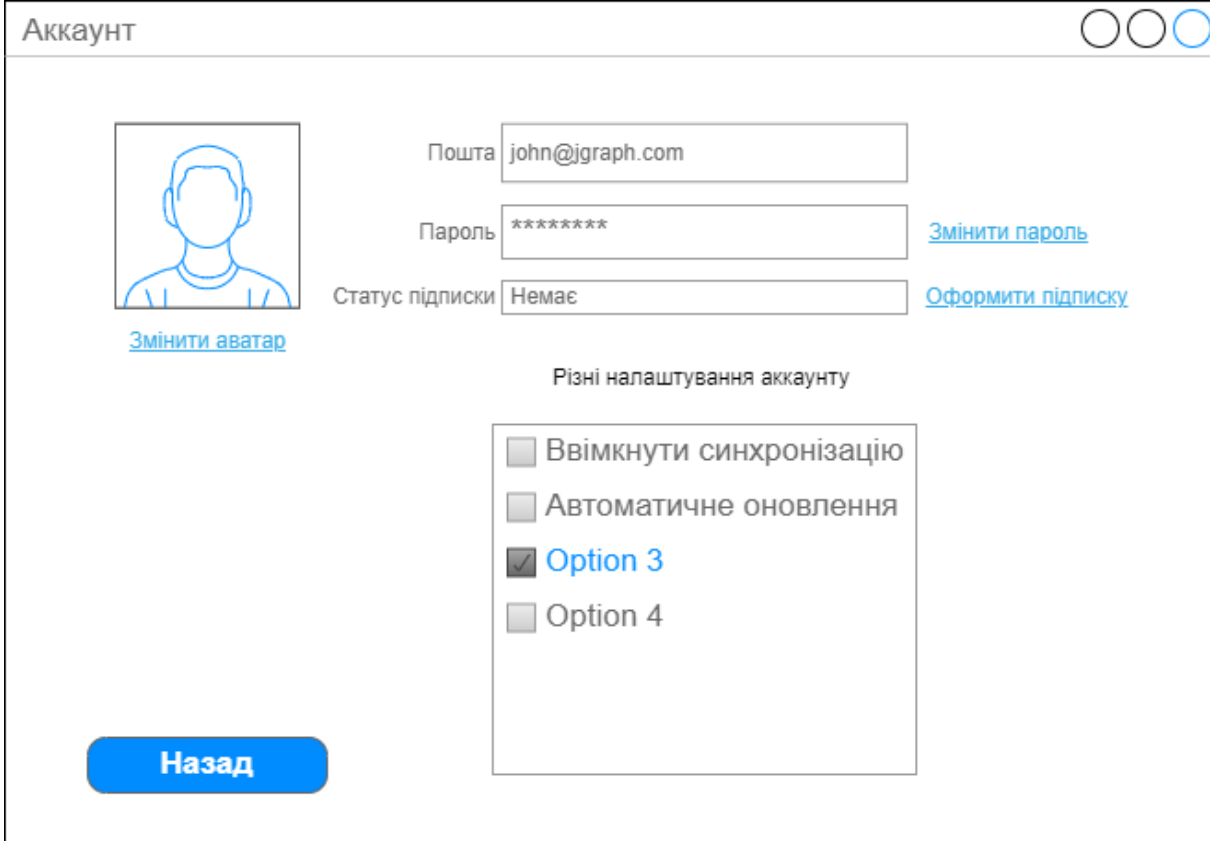


Рисунок 1.5 – Карта діалогових вікон

Одноразові прототипи діалогових вікон:

Відповідно до побудованої карти діалогових вікон (див. рис. 1.5), для кожного діалогового вікна було розроблено прототип які наведені нижче. (див. рис. 1.6 – 1.16).



Аккаунт

Пошта john@jgraph.com

Пароль ***** [Змінити пароль](#)

Статус підписки Немає [Оформити підписку](#)

[Змінити аватар](#)

Різні налаштування аккаунту

- ☐ Ввімкнути синхронізацію
- ☐ Автоматичне оновлення
- ☒ Option 3
- ☐ Option 4

[Назад](#)

Рисунок 1.6 – Прототип діалогового вікна «Аккаунт»

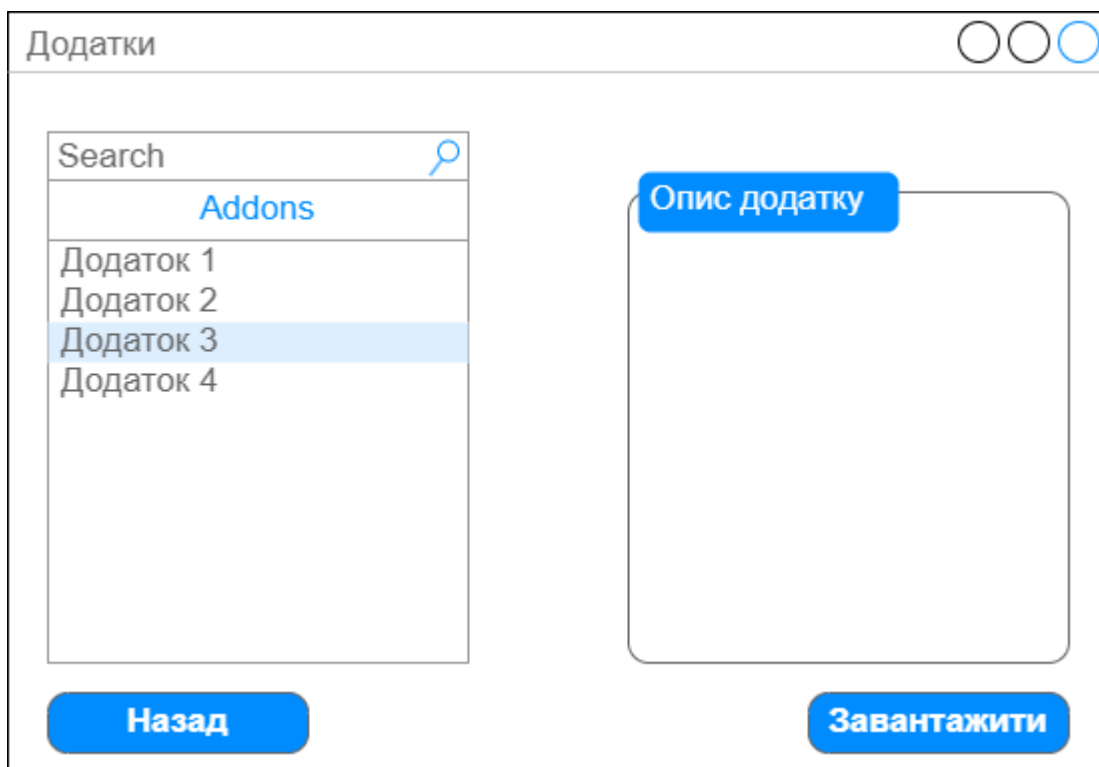


Рисунок 1.7 – Прототип діалогового вікна «Додатки»

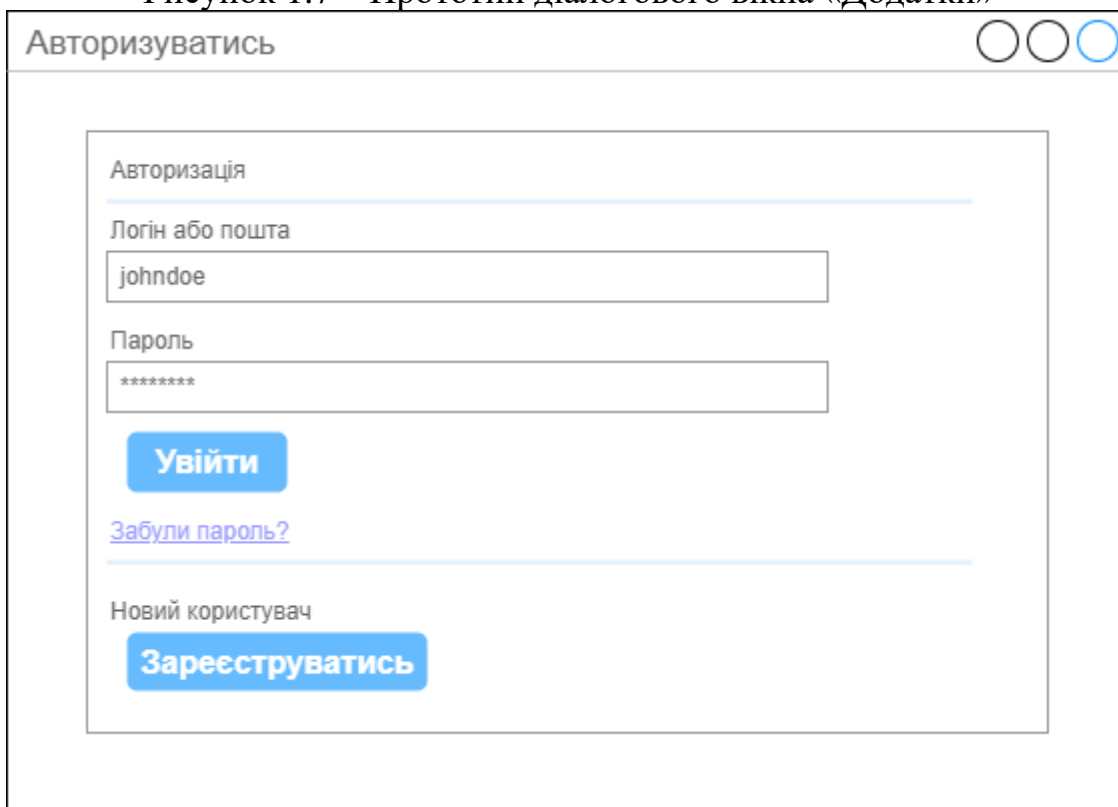


Рисунок 1.8 – Прототип діалогового вікна «Авторизація»

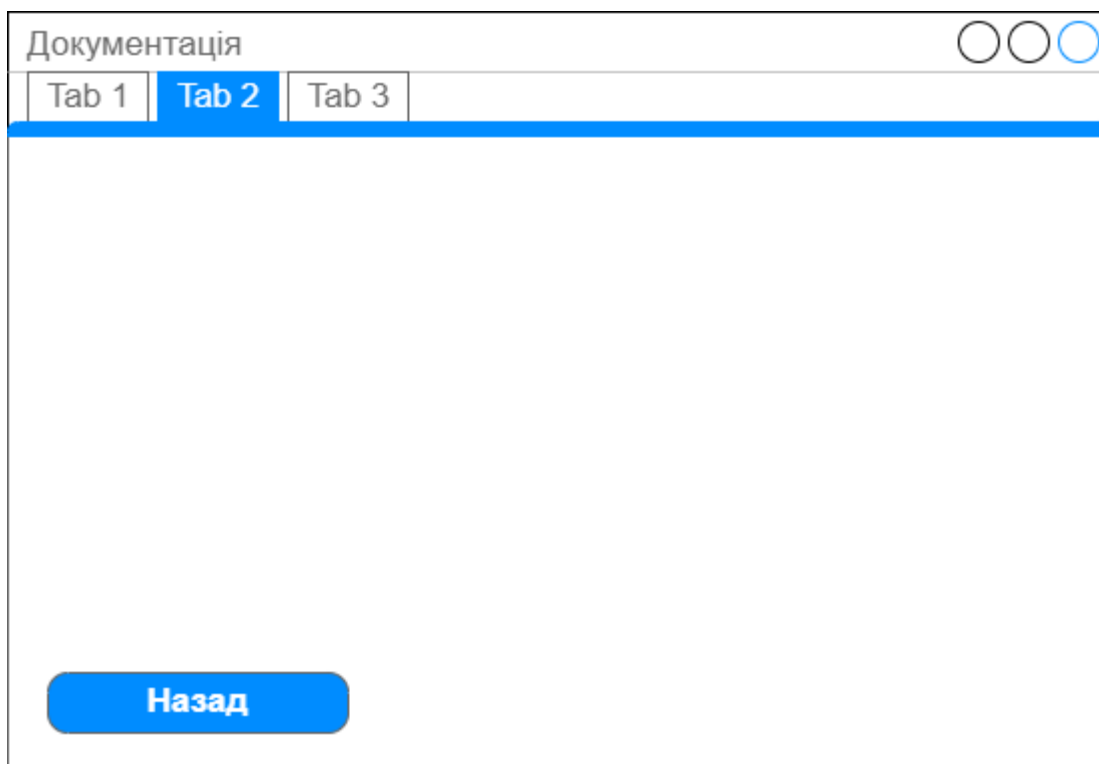


Рисунок 1.9– Прототип діалогового вікна «Документація»

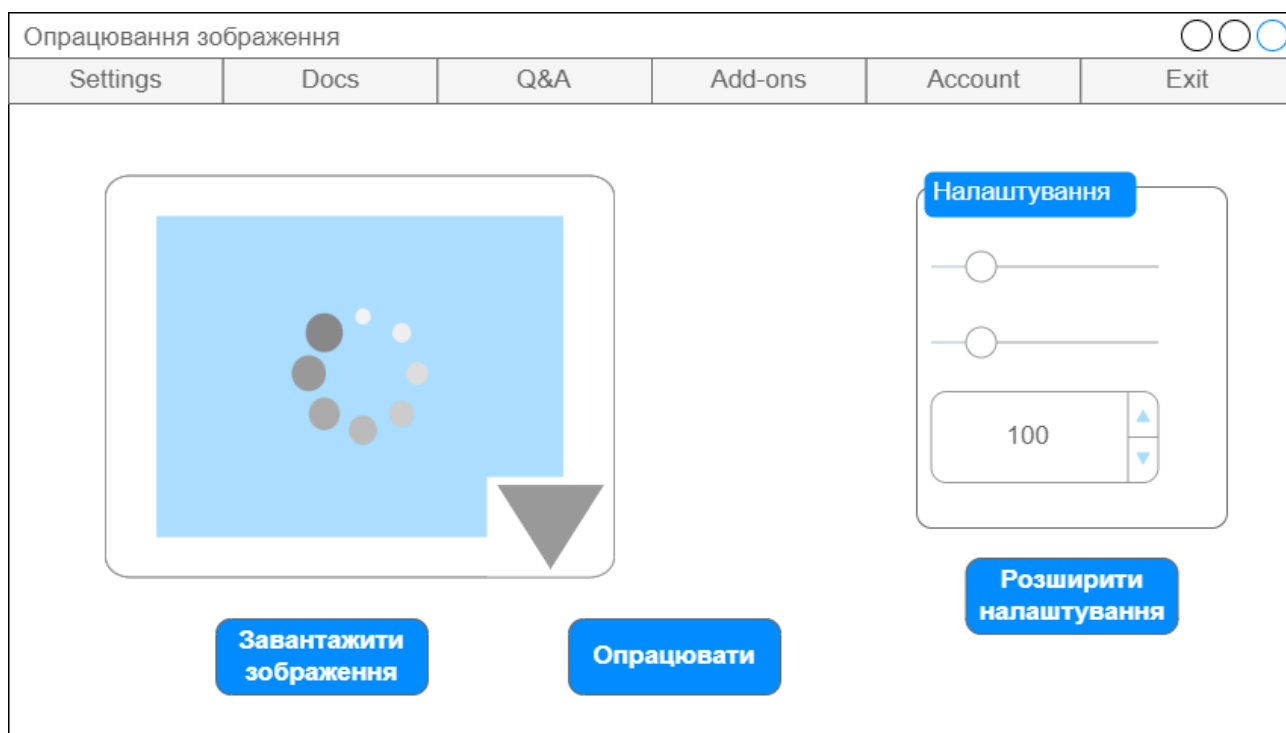
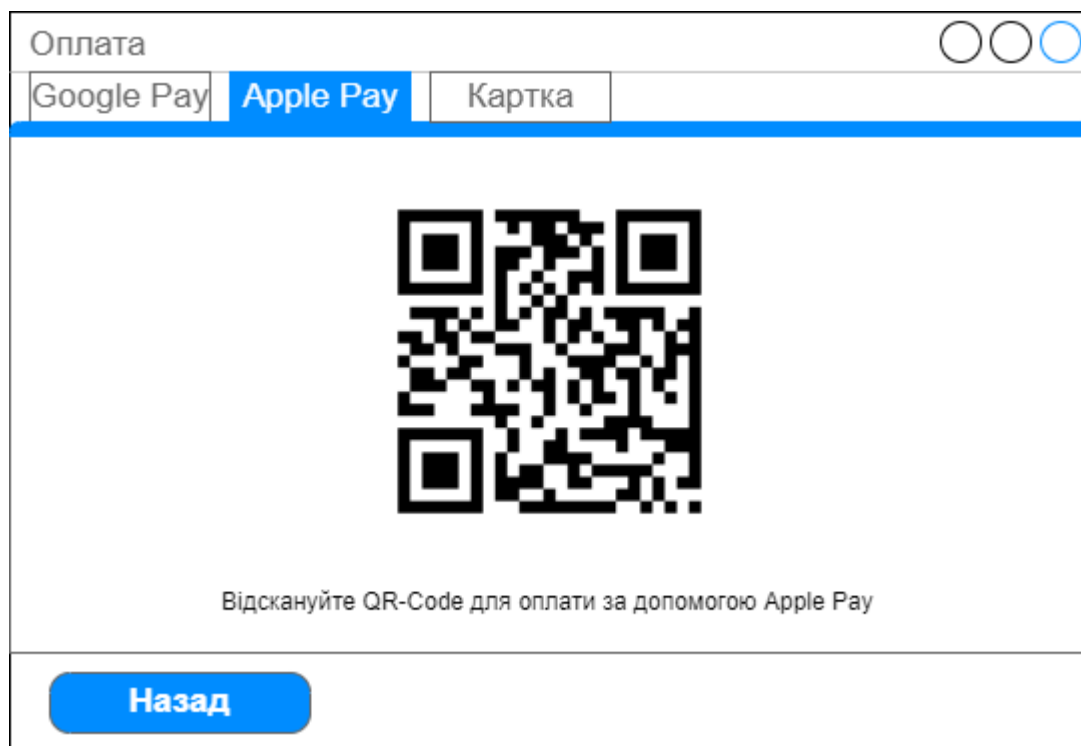
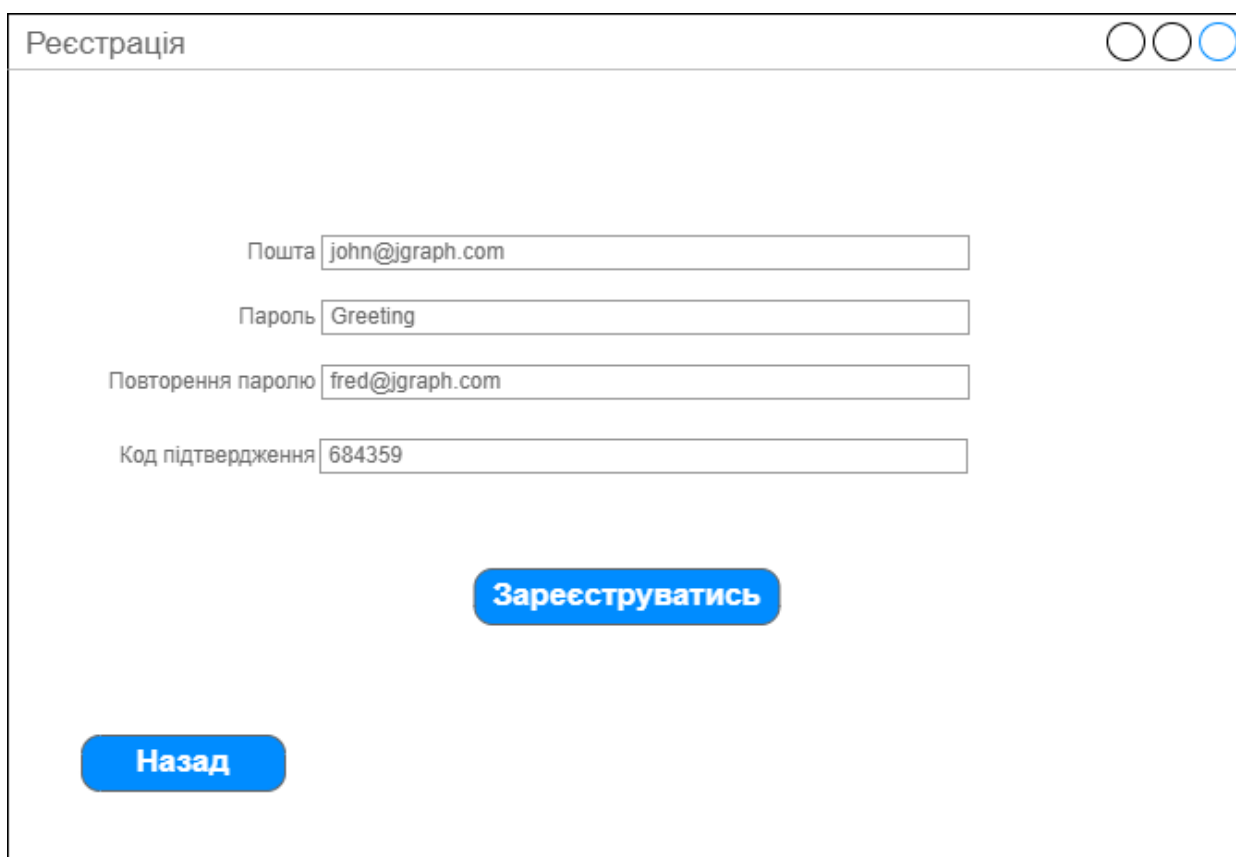


Рисунок 1.10 – Прототип діалогового вікна «Опрацювання зображення»



The image shows a prototype of a 'Payment' dialog window. At the top, the title bar reads 'Оплата' (Payment) with standard window control buttons. Below the title bar is a tabbed interface with three tabs: 'Google Pay', 'Apple Pay' (which is selected and highlighted in blue), and 'Картка' (Card). The main area of the window displays a large QR code. Below the QR code, the text reads 'Відскануйте QR-Code для оплати за допомогою Apple Pay' (Scan the QR code to pay using Apple Pay). At the bottom of the window, there is a blue button labeled 'Назад' (Back).

Рисунок 1.11– Прототип діалогового вікна «Оплата»



The image shows a prototype of a 'Registration' dialog window. The title bar reads 'Реєстрація' (Registration) with standard window control buttons. The main area contains four input fields for registration details: 'Пошта' (Email) with the value 'john@jgraph.com', 'Пароль' (Password) with the value 'Greeting', 'Повторення паролю' (Repeat password) with the value 'fred@jgraph.com', and 'Код підтвердження' (Confirmation code) with the value '684359'. Below these fields is a large blue button labeled 'Зареєструватись' (Register). In the bottom-left corner, there is a blue button labeled 'Назад' (Back).

Рисунок 1.12 – Прототип діалогового вікна «Реєстрація»

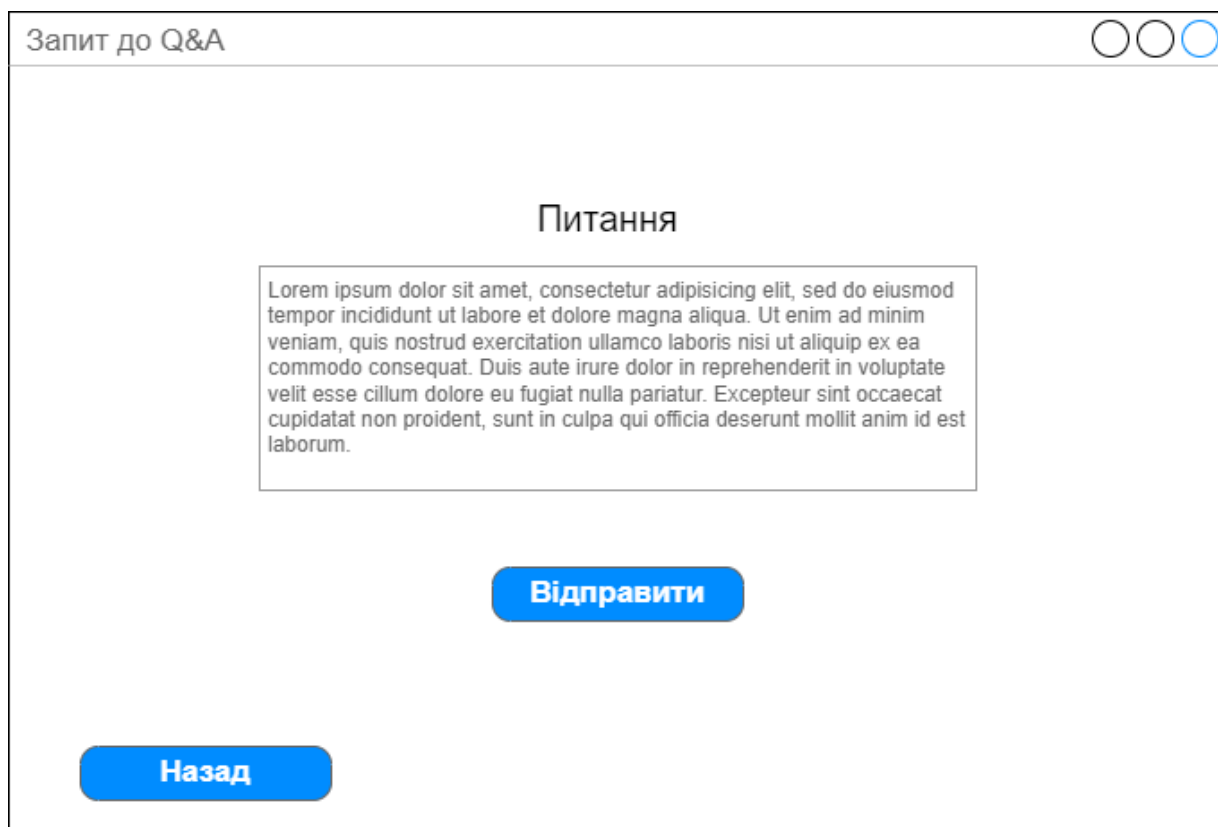


Рисунок 1.13 – Прототип діалогового вікна «Запит до Q&A»

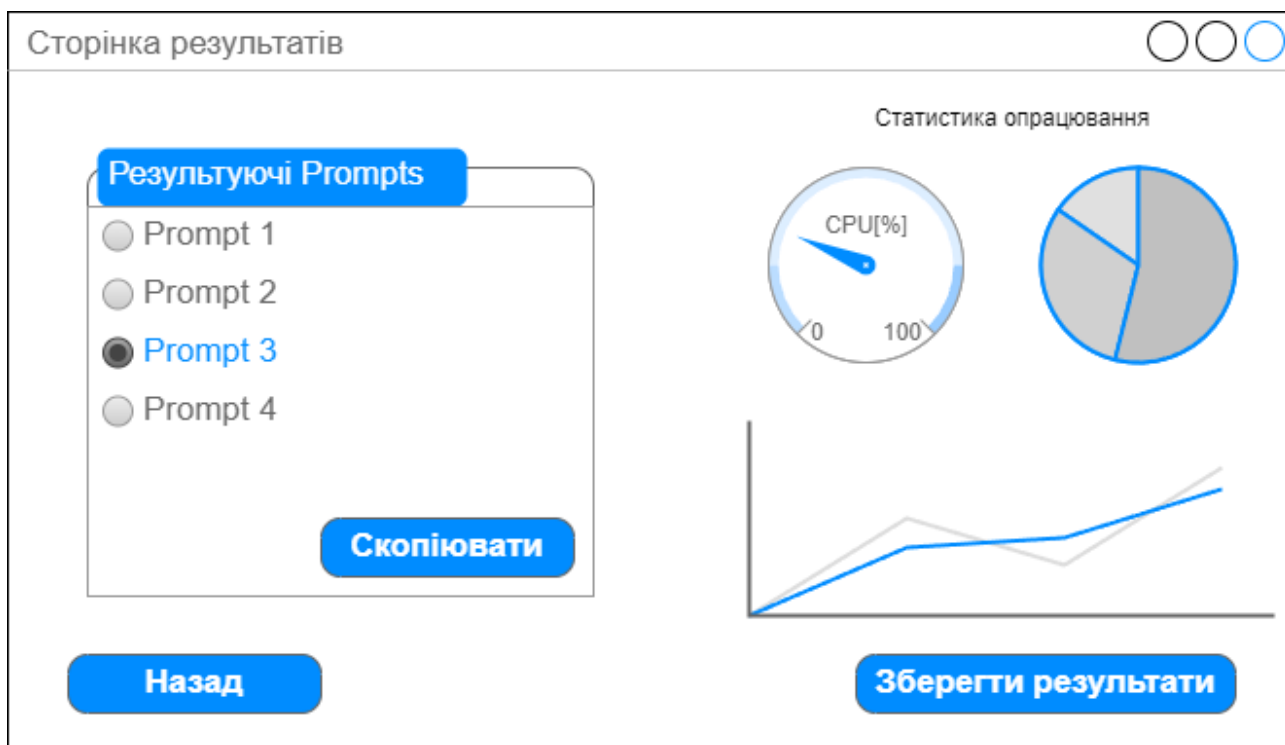


Рисунок 1.14 – Прототип діалогового вікна «Сторінка результатів»

Рисунок 1.15 – Прототип діалогового вікна «Налаштування обробки»

Рисунок 1.16 – Прототип діалогового вікна «Купівля підписки»

Разом з прототипами було розроблено вимоги до інтерфейсу користувача, які наведені в таблиці 1.9. (див. табл. 1.9)

Таблиця 1.9 – Опис вимог інтерфейсу користувача

ID	Опис вимог
UI-01	Кожне вікно, пов'язане із обробкою зображень на основі CLIP моделі, повинно мати назву, що чітко вказує на його призначення та функціонал.
UI-02	Кожне вікно у правому верхньому куті має кнопку закриття вікна.
UI-03	Система повинна забезпечувати посилання на довідку на кожну форму, пояснювальну, як користуватися цією формою.
UI-04	Система повинна надавати посилання на довідку для кожної функції обробки зображень на основі CLIP моделі, пояснюючи користувачам, як використовувати цю функцію.
UI-05	Система повинна дозволяти користувачам легко звертатися до контекстної довідки, навіть під час роботи з конкретною функцією обробки зображень на основі CLIP моделі.
UI-06	Кожен інтерфейс елемент, що взаємодіє з CLIP моделлю, має відображати інтуїтивно зрозумілі підказки для допомоги користувачеві розуміти дії та опції.
UI-07	Система повинна дозволяти користувачам налаштовувати параметри візуалізації результатів обробки зображень на основі CLIP моделі для кращого розуміння вихідних даних.
UI-08	Вікно «Авторизація» має містити поля «Логін», «Пароль» кнопки «Вхід», «Реєстрація», та посилання на відновлення паролю
UI-09	Віно «Реєстрація» має містити поля «Пошта», «Логін», «Пароль», «Повторення паролю», «Код підтвердження», кнопки «Назад» та «Зареєструватись».
UI-10	Вікно «Налаштування обробки», має містити відповідний блок «Параметри», для редагування користувачем, та блок «Вибір моделі». Також вікно має містити кнопки «Назад» та «Зберегти».
UI-11	Вікно «Купівля підписки» має містити кнопки «Назад» та «Придбати». Також має містити відповідний блок «План підписки», в якому буде відображено в зручному для користувача вигляді список наявних підписок.
UI-12	Вікно «Сторінка результатів» має містити список «Результуючі PROMPTS», блок «Статистика опрацювання», в якому буде показано відповідно візуалізовано результати обробки. Також має містити кнопки «Назад» та «Зберегти результати»
UI-13	Вікно «Запит до Q&A» має містити поле «Повідомлення», кнопки «Відправити» та «Назад».

Кінець таблиці 1.9

ID	Опис вимог
UI-14	Вікно «Оплата» має містити меню «Методи оплати» та відображати відповідно до методу оплати наявну інформацію (QR-Codes і так далі), та кнопку «Назад».
UI-15	Вікно «Опрацювання зображення», має містити поле «Зображення», блок «Налаштування», кнопки «Розширити налаштування», «Завантажити зображення», «Опрацювати», та ряд меню «Навігація» зверху вікна. При завантаженні зображення, у користувача з'являється кнопка (прозора) на зображенні, для редагування його (масштабування та обрізки).
UI-16	Вікно «Документація» має містити всю необхідну інформацію для роботи з додатком, та посилання на відповідні теми, статті чи документи в інтернеті.
UI-17	Вікно «Аккаунт» має містити поля «Пошта», «Логін», «Наявність підписки», та кнопки для переходу «Купівля підписки», «Назад», «Змінити пароль».
UI-18	Вікно «Додатки» має містити блок з наявними новими моделями та алгоритмами, поле «Пошук», блок «Інформація про додаток», та кнопки «Завантажити», «Назад».

1.3.2.1.2 Інтерфейси програмного забезпечення

Вимоги відповідно до зовнішнього ПЗ наведені в таблиці 1.10. (див. табл.1.10)

Таблиця 1.10 – Опис вимог до інтерфейсів програмного забезпечення

ID	Зовнішнє ПЗ	Опис вимог
SI 1-01	API Банку	Система має надсилати запит за допомогою API банку, та перевіряти можливість оплати підписку користувача, і відповідно при успішній сплаті, система видає користувачеві статус підписника.
SI 1-02	API Банку	Система має після отримання успішної оплати, надіслати всі дані про оплату до системи, щоб вони збереглись в системі.

1.3.2.1.3 Комунікаційні інтерфейси

ПЗ також буде використовувати певні комунікаційні інтерфейси, для вирішення питань з користувачем та підтвердження його аккаунту. Всі вимоги наведені в таблиці 1.11. (див. табл. 1.11)

Таблиця 1.11 – Опис вимог до комунікаційних інтерфейсів ПЗ

ID	Опис вимог
CI-01	Система має при реєстрації надсилати користувачеві на вказану пошту код підтвердження, який буде діяти декілька годин. [14]
CI-02	Система має після успішного придбання підписки, надіслати користувачеві на пошту вказану при реєстрації, чек та текст з описом всіх можливостей підписки.
CI-03	Система має надсилати користувачеві посилання на зміну паролю, на пошту вказану при реєстрації.
CI-04	Якщо користувач відправив запит до Q&A, всі відповіді звітти мають надходити користувачеві на пошту, вказану при реєстрації.

1.3.2.2 Вимоги до показників якості

Вимоги до показників якості якості можна поділити на зовнішні, які виявляються в процесі виконання, та внутрішні, які не виявляються. Проаналізувавши ПЗ, було побудовано та сформульовано нефункціональні вимоги. (див. табл. 1.12)

Таблиця 1.12 – Опис нефункціональних вимог

ID	Атрибут	Опис вимог
NFR-01	Доступність	Система повинна бути доступною постійно.
NFR-02	Масштабованість	Система повинна бути достатньо масштабованою, щоб підтримувати завантаження 10000 зображень одночасно.

Кінець таблиці 1.12

ID	Атрибут	Опис вимог
NFR-03	Масштабованість	Система повинна бути масштабованою, щоб підтримувати одночасну обробку 1000 зображень.
NFR-04	Переносність	Програма має працювати на різних операційних системах, забезпечуючи стабільну роботу на платформах Windows, macOS і Linux. [15]
NFR-05	Надійність	Система повинна працювати без збоїв у 97% відсотках випадків використання протягом місяця.
NFR-06	Надійність	Середній час відновлення системи (MTTRS) після збою не повинен перевищувати 10 хвилин. [16]
NFR-07	Безпека	Система повинна бути захищена від несанкціонованого доступу.
NFR-08	Продуктивність	Система повинна забезпечувати швидку та передбачувану обробку зображень на основі CLIP моделі, з часом відповіді не більше 5 секунд для стандартних зображень (Full-HD).
NFR-09	Зручність використання	Рівень помилок користувачів при взаємодії з інтерфейсом системи обробки зображень на основі CLIP моделі не повинен перевищувати 5%.
NFR-10	Зручність використання	Система повинна мати простий та інтуїтивно зрозумілий інтерфейс для користувачів, що дозволяє легко встановлювати та використовувати програму без додаткових інструкцій.
NFR-11	Цілісність	Система повинна гарантувати цілісність обробки зображень, запобігаючи втраті або викриванню даних під час обробки та зберігання результатів.
NFR-12	Перевірюваність та тестованість	Розробники та тестувальники повинні мати можливість ефективно перевіряти та тестувати систему обробки зображень, забезпечуючи швидке підтвердження правильної реалізації та виявлення помилок.
NFR-13	Захист	Система повинна мати заходи для захисту від пошкоджень даних та програмного забезпечення внаслідок експлуатації та взаємодії з іншими системами. [17]
NFR-14	Локалізація	Система повинна бути здатною задовольняти мовні, культурні та інші локальні вимоги користувачів в різних регіонах (Українська, Англійська). [18]

1.3.2.3 Обмеження

Оскільки ПЗ має специфічну ідею та реалізацію, було розроблено обмеження, які характеризують необхідні інструменти та технології. Відповідні обмеження наведено в таблиці 1.13. (див. табл. 1.13)

Таблиця 1.13 – Обмеження при реалізації ПЗ

ID	Опис вимог
LI-01	Система повинна використовувати мову програмування Python або Rust [19] для реалізації клієнтської сторони додатку та обробки зображень з використанням системи CLIP.
LI-02	Для роботи з базою даних необхідно використовувати MySQL.
LI-03	Реалізація серверної сторони додатку може бути виконана за допомогою мови програмування Python або Node.js, з урахуванням оптимальності та ефективності для конкретних завдань.
LI-04	Для реалізації системи обробки зображень за допомогою CLIP, рекомендується використовувати бібліотеку OpenCV [20] для роботи з зображеннями та PyTorch для інтеграції з моделлю CLIP.
LI-05	Початкова розробка додатку передбачає оптимізацію для операційних систем Windows 10 та 11, а після успішного впровадження буде здійснено портування на платформи Linux та macOS.
LI-06	Для взаємодії з банківським API використовуйте API monobank або LiqPay для забезпечення ефективного та безпечного обміну фінансовою інформацією.
LI-07	Для взаємодії з банківським API (monobank або LiqPay) необхідно дотримуватися протоколів та стандартів безпеки, зокрема використовувати протокол HTTPS для захищеної передачі даних.
LI-08	Для забезпечення безпеки та стабільності системи рекомендується регулярно оновлювати та використовувати актуальні версії використовуваних бібліотек та фреймворків.
LI-09	Реалізація обробки зображень повинна враховувати можливості апаратної платформи та забезпечувати оптимальне використання ресурсів, зокрема, використання оптимізованих алгоритмів для роботи з графічними даними.

Кінець таблиці 1.13

ID	Опис вимог
LI-10	При розробці необхідно враховувати можливість інтеграції з іншими системами у майбутньому, тому код слід писати з врахуванням стандартів відкритого програмного забезпечення та RESTful API.
LI-11	Забезпечити модульність та розширюваність дизайну, щоб нові функціональність та компоненти можна було додавати без необхідності значних змін у вже існуючому коді.
LI-12	Використовувати стандартизовані підходи до оформлення коду та коментування, щоб полегшити зрозуміння та підтримку програмного забезпечення.

Кінець таблиці 1.14

Функції он- альні вимоги	Варіанти використання							Бізнес-правила					
	U C- 01	U C- 02	U C- 03	U C- 04	U C- 05	U C- 06	U C- 07	BR L- 01	BR L- 02	BR L- 03	BR L- 04	BR L- 05	BR L- 06
FR3.07. 02						+							
FR3.08		+											
FR3.08. 01		+											
FR3.08. 02		+											
FR3.09		+											
FR4.01			+						+				
FR4.02			+						+				
FR4.03			+						+				
NFR-01		+			+							+	
NFR-02		+											
NFR-03									+				
NFR-04												+	
NFR-05										+			
NFR-06											+		
NFR-07	+	+			+								
NFR-08										+			
NFR-09		+											
NFR-10	+				+								
NFR-11		+				+							
NFR-12										+			
NFR-13		+	+										
NFR-14	+	+	+	+	+	+	+	+					+

Висновки до розділу 1

У даному розділі було розроблено бізнес-вимоги, вимоги користувачів, вимоги програмного забезпечення, нефункціональні вимоги до програмного забезпечення та матрицю відстеження вимог до проекту «Програмне забезпечення попередньої обробки PROMPT на основі CLIP моделі».

2 ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ПОПЕРЕДНЬОЇ ОБРОБКИ ЗОБРАЖЕНЬ ДЛЯ ВИЗНАЧЕННЯ PROMPT НА ОСНОВІ CLIP МОДЕЛІ

2.1 Архітектурне проєктування

2.1.1. Методологія розробки програмного забезпечення

Ми вибрали методологію Feature-Driven Development (FDD) для управління процесом розробки. Цей метод дозволяє зосередитися на створенні конкретних функціональних компонентів. Методологія Feature-Driven Development (FDD)

Гнучка методологія розробки програмного забезпечення, Feature-Driven Development (FDD), зосереджена на постійному створенні та вдосконаленні окремих функціональних компонентів системи. Розробка загальної моделі, побудова списку функцій, планування функцій, проєктування функцій і реалізація є основними етапами FDD. [21]

Розробка загальної моделі

На першому етапі ми створили архітектуру системи в цілому. Модуль попередньої обробки зображень, модуль визначення текстових підказок, який базується на моделі CLIP, і інтерфейс користувача для взаємодії з системою є основними компонентами. Загальна модель забезпечує основу для подальшої деталізації та розробки, а також допомагає зрозуміти, як різні компоненти системи взаємодіють один з одним.

Побудова списку функцій

Список функцій системи включає:

- Завантаження та зберігання зображень
- Попередня обробка зображень, що включає зміну розміру, нормалізацію та виділення важливих ознак
- Інтеграція з моделлю CLIP для аналізу зображень
- Генерація текстових підказок на основі результатів аналізу
- Відображення результатів користувачеві через зручний інтерфейс

Кожна функція була деталізована та включена в загальний план розробки.

Планування по функціях

Визначення пріоритетів і визначення порядку їх виконання є частиною планування реалізації функцій. Завантаження та зберігання зображень були найважливішими завданнями для подальшої роботи системи. Розробка модуля попередньої обробки, інтеграція моделі CLIP і функції генерації та відображення текстових підказок були наступними кроками.

Проектування по функціях

Перед реалізацією кожної функції ми розробили детальне технічне завдання. Для функції попередньої обробки зображень було визначено алгоритми нормалізації та зміни розміру, а також параметри, необхідні для подальшого аналізу моделлю CLIP. Також ми врахували можливість масштабування та адаптації під різні типи зображень.

Реалізація по функціях

Ми розробили детальне технічне завдання перед тим, як виконувати кожну функцію. Були визначені алгоритми нормалізації та зміни розміру для функції попередньої обробки зображень, а також параметри, необхідні для подальшого аналізу моделі CLIP. Ми також розглянули можливості масштабування та адаптації до різних типів зображень.

Інтеграція CLIP моделі

Програмування для завантаження та зберігання фотографій було першим кроком до реалізації функцій. Після цього були введені функції попередньої обробки, інтеграції моделі CLIP і генерації та відображення текстових підказок. Кожна функція була протестована окремо та в комплексі після того, як вона була інтегрована в загальну систему.

Тестування та валідація

Розробка програмного забезпечення залежить від тестування. Ми проводимо тестування з реальними користувачами для оцінки ефективності та зручності системи, інтеграційні тести для перевірки взаємодії між різними компонентами системи. Ми також планували постійне вдосконалення та оптимізацію системи на основі зворотного зв'язку від користувачів і результатів тестування.

2.1.2. Загальна модель/стиль архітектури програмного забезпечення

Програмне забезпечення для клієнт-серверного застосування, яке використовує модель CLIP для обробки зображень, може бути описано як розподілена система, що складається з багатьох основних підсистем. Наступне описує основні підсистеми та їхні зв'язки. [22]

Підсистема клієнта: Підсистема клієнта розташована в архітектурі програмного забезпечення для обробки зображень на основі моделі CLIP і складається з багатьох компонентів, які взаємодіють з користувачем і гарантують, що дані надсилаються швидко на сервер для обробки.[23] Давайте поговоримо про деталі цієї підсистеми:

Користувацький інтерфейс (UI).

Користувацький інтерфейс (UI): Графічний інтерфейс є компонентом клієнтської підсистеми, який дозволяє користувачам взаємодіяти з системою. [24] Інтерфейс користувача (UI) повинен бути зручним і простим для розуміння, щоб користувачі могли легко використовувати функціональність програми.

Форма завантаження зображення: користувачеві потрібно мати елементи для завантаження зображення. Це може бути кнопка «Обрати файл» або інший елемент інтерфейсу, схожий на цю.

Модуль передачі зображення: механізм передачі даних відповідає за ефективну передачу зображення з користувача на сервер. Розподіл для ефективного використання мережі, оптимізація передачі даних тощо може бути частиною цього.

Доставка метаданих: Модуль передачі може передавати метадані, такі як ідентифікатор користувача, інформація про час відправки та інші, крім самого зображення.

Взаємодія з іншими підсистемами.

Взаємодія з серверною підсистемою: клієнтська підсистема взаємодіє з серверною частиною, надсилаючи зображення для обробки та отримуючи результати. Цей зв'язок відбувається через мережу за допомогою протоколу, який відповідає цьому, наприклад, HTTP або HTTPS. [25]

Взаємодія з базою даних клієнтів: можливо, що потрібно буде взаємодіяти з базою даних, щоб отримати або оновити інформацію про користувача.

Захист інформації.

Шифрування даних: шифрування може забезпечити безпеку та конфіденційність передачі даних. Це особливо важливо, коли зображення передаються через відкриті мережі, такі як Інтернет. [26]

Аутентифікація користувача: важливий компонент захисту передбачає, що тільки авторизовані користувачі можуть використовувати систему.

Ефективна передача даних для подальшої обробки на серверній стороні та зручність для користувачів повинні бути основними цілями розробки клієнтської підсистеми.

Підсистема сервера для програмного забезпечення обробки зображень на основі моделі CLIP складається з багатьох компонентів, які відповідають за прийом, обробку та аналіз зображень від користувачів. Давайте розглянемо основні компоненти, які складають цю підсистему:

Приймання та обробка зображень.

Приймач зображень: цей компонент приймає зображення з підсистеми клієнта. Він повинен мати методи отримання та часткового розбору файлу.

Модуль обробки: обробляє зображення за допомогою моделі CLIP. Предпроцесинг, подальший аналіз і виведення результатів можуть бути частиною цього етапу.

Взаємодія з базою даних.

Збереження результатів: результати обробки, такі як визначені запити на основі зображення, можна зберігати в базі даних для аналізу або використання в майбутньому.

Запити до бази даних клієнтів: якщо система має можливість зберігати та отримувати інформацію про користувачів, сервер може зв'язатися з базою даних, щоб отримати потрібну інформацію.

Взаємодія з підсистемою обслуговування клієнтів включає модуль обробки запитів, який приймає та обробляє запити клієнтів. Використання цього модуля передбачає передачу зображень для обробки та передачу результатів назад.

Протоколи взаємодії.

Щоб забезпечити ефективний обмін даними з клієнтами, сервер повинен використовувати відповідні протоколи взаємодії, такі як HTTP або HTTPS.

Захист інформації.

Захист з'єднання: забезпечує шифрування та безпеку з'єднань, щоб запобігти несанкціонованому доступу.

Аутентифікація та авторизація: гарантує, що користувачі дійсно ідентифіковані, і дозволяє їм отримувати доступ лише до ресурсів, які необхідні.

Моніторинг і запис подій: Журналювання подій дозволяє записувати події, що відбуваються на сервері, для подальшого аналізу та виявлення потенційних проблем.

Моніторинг ресурсів: система моніторингу, яка відстежує використання ресурсів сервера, щоб швидко виявити проблеми.

Розробка серверної підсистеми полягає в тому, щоб забезпечити ефективну та безпечну обробку зображень, а також надійну мережеву взаємодію з клієнтами.

Зв'язки між підсистемами: Підсистема клієнта взаємодіє з підсистемою сервера через мережу, надсилаючи дані для обробки та отримуючи результати.

Підсистема сервера взаємодіє з базою даних клієнтів, щоб збирати та отримувати дані про користувачів.

За ефективну передачу зображень від клієнта до сервера відповідає модуль передачі зображення.

Ця архітектура дозволяє створювати ефективний та масштабований клієнт-серверний застосунок для обробки зображень на основі моделі CLIP, забезпечуючи зручний інтерфейс користувача та високу обробку серверного боку.

2.1.3. Проектування підсистеми збереження даних

2.1.3.1. Вибір моделі даних та підходу для проектування БД

Враховуючи специфіку нашого проєкту, було вирішено використовувати реляційну модель даних. Це рішення обґрунтоване кількома факторами:

–Структуризація даних: дані нашої системи структуровані та складаються з інформації про зображення, результатів обробки, метаданих і текстових підказок (PROMPT). За допомогою таблиць і зв'язків між ними реляційні бази даних (РБД) дозволяють ефективно керувати такими структурованими даними.

–Цілісність даних: Реляційні бази даних підтримують надійні дані за допомогою зовнішніх ключів та обмежень цілісності. Це важливо для нашої системи, оскільки ми маємо справу з взаємозв'язаними даними, такими як зображення та їх текстові підказки.

Для проектування бази даних було обрано підхід «Database first». Цей метод передбачає створення бази даних зі структурами та таблицями, а потім створення моделей даних на основі існуючої схеми бази даних. Основні переваги використання методу «Database First» у нашому проєкті: [27]

–Контроль над схемою баз даних: коли ми створюємо базу даних, ми маємо повний контроль над її структурою. Це дозволяє ретельно планувати та оптимізувати схему, щоб гарантувати високу продуктивність і ефективне зберігання даних.

–Забезпечення відповідності бізнес-вимогам: спочатку розробляючи схему бази даних, ми можемо переконатися, що дані відповідають вимогам бізнесу та що вони мають правильну структуру, яку можна обробити та аналізувати.

–Легкість інтеграції: використання підходу «База даних перша» полегшує інтеграцію існуючої бази даних з іншими системами та інструментами; це може бути важливим для розширення функціональності або інтеграції з іншими у майбутньому.

Реалізація підходу Database First

Процес реалізації підходу "Database First" включає кілька ключових етапів:

Дизайн схеми бази даних: Наразі створюється схема бази даних, включаючи визначення всіх необхідних таблиць, їх атрибутів і зв'язків між ними. Наприклад, таблиця зображень може містити поля для зберігання даних, таких як дата завантаження файлу та інформація про файл. Щоб дозволити зберігати результати аналізу, таблиця зображень буде пов'язана з таблицею текстових підказок (PROMPT) за допомогою зовнішнього ключа.

Розробка бази даних: Система управління базами даних (СУБД), такі як MySQL і PostgreSQL, використовується для створення реальної бази даних на основі розробленої схеми. Крім того, під час цього процесу можуть бути знайдені індекси, тригери та інші оптимізації, необхідні для забезпечення високої продуктивності. [28]

Генерація моделей даних: Після створення бази даних генеруються моделі даних, які використовуватимуться у програмному забезпеченні. Це можна зробити за допомогою інструментів ORM (Object-Relational Mapping), таких як Entity Framework для C# або SQLAlchemy для Python. [29] Ці інструменти автоматично генерують класи моделей на основі існуючої схеми бази даних.

Інтеграція з програмним забезпеченням: Генеровані моделі даних інтегруються з програмним забезпеченням для забезпечення взаємодії з базою даних. Це включає реалізацію методів для завантаження, зберігання та обробки даних, а також інтеграцію з модулем CLIP для аналізу зображень та генерації текстових підказок.

2.1.3.2. Проєктування та розробка моделі даних

В результаті проведеного аналізу вимог до ПЗ в попередніх пунктах, були виділені наступні узагальнені сутності, опис яких приведено нижче:

- Користувачі
- Платежі
- Пропозиції підписок
- Моделі (CLIP моделі, які можна завантажити)

Після нормалізації бази даних [29] та формування зв'язків між сутностями, була побудована концептуальна модель даних предметної області «Додаток для оброблення зображень», яка наведена на рисунку 2.1.



Рисунок 2.1 – ER-модель даних предметної області для ПЗ «ImgProPlus»

Наведемо опис виділених сутностей та їх атрибутів у табличному вигляді (див. табл. 2.1)

Таблиця 2.1 – Опис сутностей ER-моделі

Сутність	Первинний ключ	Атрибути
Users – користувачі системи	username – логін користувача	password – пароль
Auth Tokens – токени верифікації	username – логін користувача	authority – назва ролі
Payments – платежі	payment_id – код платежу	username – логін користувача, payment_date – дата платежу, promotion_id – код підписки
Confirmation Codes – коди підтвердження	id – ідентифікаційний номер	username – логін користувача, code – код підтвердження, expiration_date – дата знищення

Кінець таблиці 2.1

Сутність	Первинний ключ	Атрибути
Models – моделі	model_name – назва моделі	description – опис моделі, premium – доступність моделі користувачам
Credits – кредити користувачів	username – логін користувача	credits – кількість кредитів користувача без підписки
Promotions – пропозиції	promotion_id – код пропозиції	subscription_length – довжина підписки (в місяцях), cost – вартість підписки (в USD), description – опис пропозиції, subscription_length_ua – довжина підписки українською, description_text_ua – опис підписки українською.
Subscribers – статус підписки	username – логін користувача	end_date – дата закінчення підписки
Requests – запити	request_id – код запиту	username – логін користувача, request_text – текст запиту

Після оцінки сутностей і атрибутів було розроблено логічну модель даних, яка наведена на рисунку. 2.2. (див. рис. 2.2)

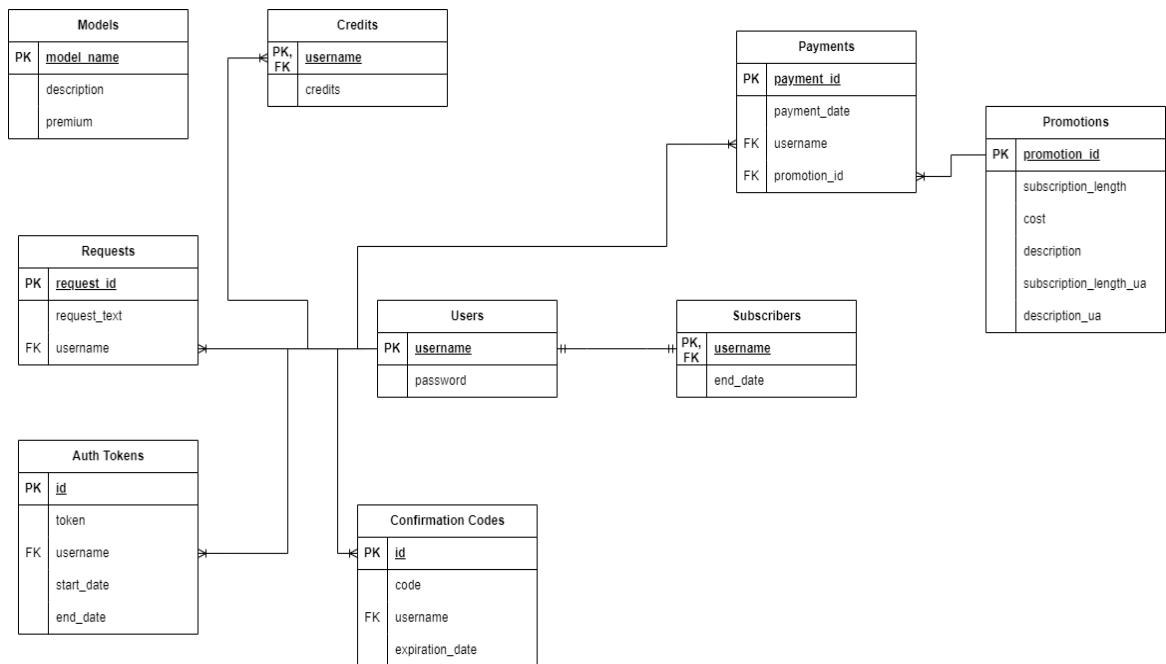


Рисунок 2.2 – Логічна модель даних предметної області «Додаток для опрацювання зображень»

За допомогою ретельно сформованих ER-моделі та логічної моделі даних, ми можемо сформувавши фізичну модель даних, зі зручним розподілом по типам даних, задля економії місця на жорсткому диску. [30]

Для зручності, фізична модель подана зображенням сформованим в MySQL Workbench, який створює діаграму базуючись на існуючій базі даних. Фізична модель подана на рисунку 2.3. (див. рис. 2.3)

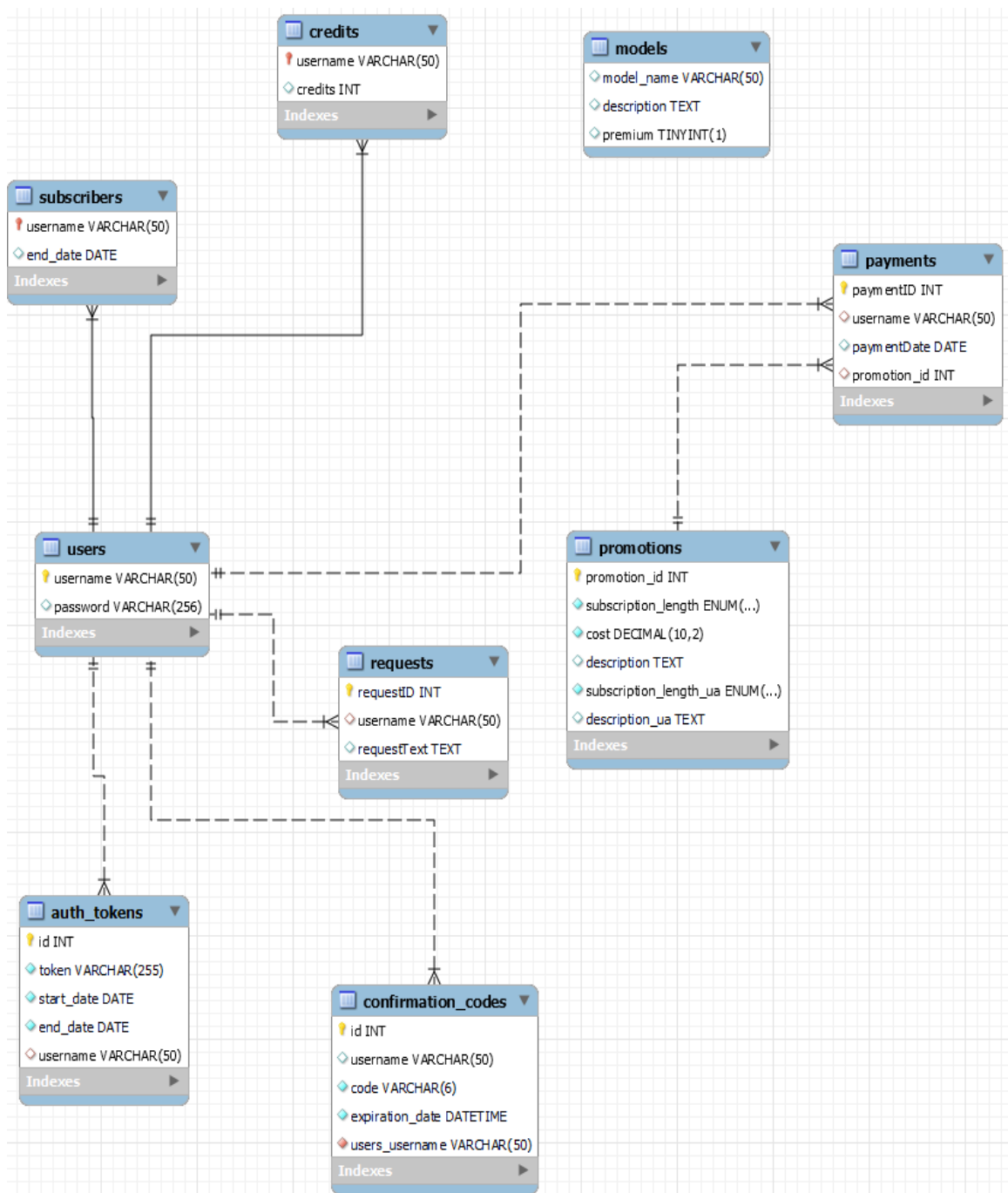


Рисунок 2.3 – Фізична модель даних

2.1.3.3. Взаємодія з БД

Додаток буде взаємодіяти з БД за допомогою `mysql.connector`, бібліотекою, яка написана компанією «Oracle», що дозволяє швидко та без проблем обробляти запити від програми.

Алгоритм роботи БД (який буде написаний в класі `database.py`, та більш детально розглянутий в пункті 2.4), представлено на рисунку 2.4 (див. рис. 2.4) [31]



Рисунок 2.4 – Алгоритм обробки запиту в БД за допомогою `mysql.connector`

Конфігураційний файл буде в собі зберігати дані, для підключення до бази даних. Формат файлу `json`, що дозволяє зручно змінювати дані за потреби, та працювати з ними в програмі.

Для обробки запиту, було написано функцію `execute_query`, яка дозволяє гнучко для розробника виконувати різні виклики до бази даних. Код цієї функції наведено на рисунку 2.5. (див. рис. 2.5.)

```

def execute_query(self, query, params=None) -> list | bool:
    """
    Execute a MySQL query and return the results.

    Args:
        query (str): The SQL query to execute.
        params (tuple, optional): A tuple of parameters to substitute into the query.

    Returns:
        list: A list of tuples containing the rows returned by the query.
        bool: True if the query executed successfully, False otherwise.

    This function executes the provided SQL query with optional parameters and returns
    the results as a list of tuples. It also logs any errors that occur during execution.
    """

    cursor = self.connection.cursor()
    try:
        cursor.execute(query, params)
        results = cursor.fetchall()
        self.connection.commit()
        if query[:6] == "SELECT":
            return results if results else False
        return results if results else True

    except mysql.connector.Error as error:
        self.logger.error(f"Error executing query: {error}")
        return False

    finally:
        cursor.close()

```

Рисунок 2.5 – Код та документація execute_query

Після повернення результатів, розробник взаємодіє з ними вже за допомогою інших функцій, які є частиною database.py, та наведені в Додатку А (див. Додаток А).

2.1.3.4. Розробка запитів до БД

Створивши фізичну модель даних, та опрацювавши, яким чином додаток буде взаємодіяти з БД, в пункті 2.1.3.3, можемо формувати код та запити до бази даних. В цьому пункті наведені лише фрагменти коду, весь код знаходиться в Додатку А (див. Додаток А).

Фрагмент створення таблиць наведено на рисунку 2.6.

```
-- Create Requests table
> CREATE TABLE Requests (
    requestID INT AUTO_INCREMENT PRIMARY KEY,
    username VARCHAR(50),
    requestText TEXT,
    FOREIGN KEY (username) REFERENCES Users(username)
- );

-- Create auth_tokens table
> CREATE TABLE auth_tokens (
    id INT AUTO_INCREMENT PRIMARY KEY,
    token VARCHAR(255) NOT NULL,
    start_date DATE NOT NULL,
    end_date DATE NOT NULL,
    username VARCHAR(50),
    FOREIGN KEY (username) REFERENCES Users(username)
- );

-- Create confirmation_codes table
> CREATE TABLE confirmation_codes (
    id INT AUTO_INCREMENT PRIMARY KEY,
    username VARCHAR(50),
    code varchar(6) NOT NULL,
    expiration_date DATETIME NOT NULL,
    UNIQUE KEY unique_code (code)
- );
```

Рисунок 2.6 – Фрагмент створення таблиць в БД

Я обрав саме ці таблиці, оскільки для них необхідно створити тригери (events), які будуть автоматично видаляти дані, що застаріли. [32] На рисунку 27, наведено код для тригерів.


```

-- EVENTS

-- Delete expired hashes event
DELIMITER $$
CREATE EVENT delete_expired_hashes
ON SCHEDULE
    EVERY 1 DAY
COMMENT 'Delete expired hash records from the auth_tokens table'
DO
BEGIN
    DELETE FROM auth_tokens
    WHERE CURRENT_DATE() NOT BETWEEN start_date AND end_date;
END$$
DELIMITER ;

-- Delete expired codes event
DELIMITER $$
CREATE EVENT delete_expired_codes
ON SCHEDULE EVERY 1 MINUTE
DO
    DELETE FROM confirmation_codes WHERE expiration_date < NOW();
DELIMITER ;

```

Рисунок 2.7 – Створення тригерів для очищення таблиць auth_tokens та confirmation_codes

Для взаємодії з INSERT та SELECT командами, було написано execute_query функцію, про яку йде мова в попередньому пункті, але, для наглядності, буде продемонстровано виклик цієї функції, з командами INSERT та SELECT відповідно. Функції та їх документація, в яких відбувається виклик, наведені на рисунках 2.8-2.9. (див. рис. 2.8-2.9)

Для прикладу приведено функції verify_confirmation_code, яка взаємодіє з таблицею confirmation_code та дістає звітти командою SELECT необхідний код.

```
def verify_confirmation_code(self, username: str, code: str) -> bool:
    """
    Verify a confirmation code for a user.

    Args:
        username (str): The username of the user.
        code (str): The confirmation code to verify.

    Returns:
        bool: True if the confirmation code is valid for the user, False otherwise.

    This function verifies the provided confirmation code for the user specified by the
    username. It checks if the code exists in the 'confirmation_codes' table in the database
    for the given username and returns True if the code is valid, False otherwise.
    """
    try:
        query = "select code from confirmation_codes where username = %s and code = %s"
        results = self.execute_query(query, (username, code))
        if results:
            self.logger.info('verify_confirmation_code executed succesfully!')
            return True
        else: return False
    except Error as e:
        self.logger.error(f"Something went wrong during verify_confirmation_code function. Error - {e}")
```

Рисунок 2.8 – Виклик SELECT в функції verify_confirmation_code

Для наступного прикладу, наведено функцію generate_confirmation_code, яка генерує тимчасовий код, та заносить його до БД. (див. рис. 2.9)

```
def generate_confirmation_code(self, username: str) -> str:
    """
    Generate a confirmation code for a user.

    Args:
        username (str): The username of the user.

    Returns:
        str: The generated confirmation code.

    This function generates a confirmation code for the user specified by the username.
    It inserts the code into the 'confirmation_codes' table in the database and returns
    the generated code.
    """
    try:
        code = str(randint(100000, 999999))
        query = "INSERT INTO confirmation_codes(username, code, expiration_date) VALUES (%s, %s, %s)"
        expiration_date = datetime.now() + timedelta(minutes=10)
        results = self.execute_query(query, (username, code, expiration_date))
        if results:
            self.logger.info('generate_confirmation_code executed succesfully!')
            return code
        else: return False
    except Error as e:
        self.logger.error(f"Something went wrong during generate_confirmation_code function. Error - {e}")
```

Рисунок 2.9 – Виклик INSERT в функції generate_confirmation_code

Виклики UPDATE та REMOVE, в цьому розділі розглядатись не будуть. Вони наявні в функціях delete_user, reset_password, тощо. (див. Додаток А)

2.1.4 Проєктування взаємодії з зовнішніми сервісами

Оскільки додаток використовує клієнт-серверну архітектуру, для зручності було розроблено власну комунікативну модель на основі HTTPS.

Використовуючи вбудовану бібліотеку socket, для класу Server, користувач відправляв дані, які за допомогою SSL та зашифрованого ключа, [33] кодувались та безпечно передавались через інтернет. Базова робота алгоритму зображена на рисунку 2.10. (фрагменти з шифруванням та отриманням ключів, були відкинуті, для наглядної роботи безпосередньо клієнт-серверної системи)

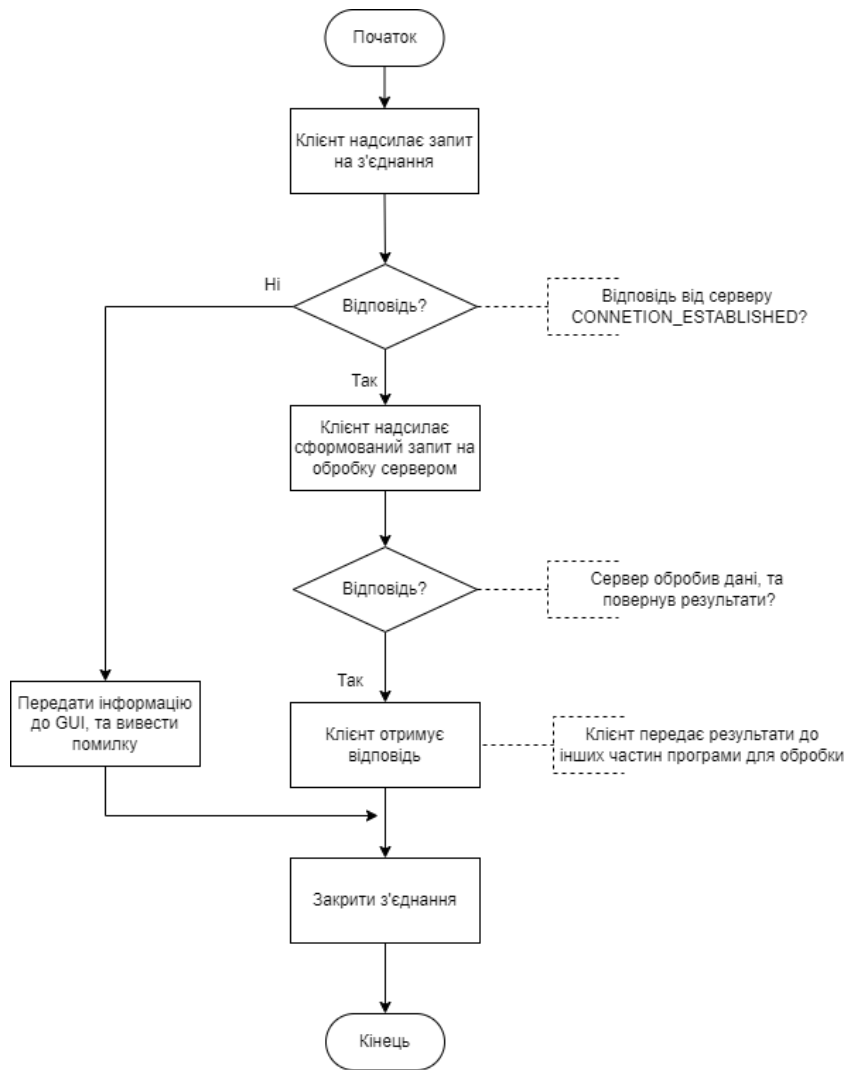


Рисунок 2.10 – Взаємодія серверу з клієнтом

Оскільки GET та POST виклики, було вирішено не використовувати, формат запитів було перероблено, та розроблено власний, який зручніший для розгортання в цьому проекті. [34] Як виглядає виклик зображено на рисунку 2.11. (див. рис. 2.11)



Рисунок 2.11 – Структура запиту до серверу, що використовується в додаткові

Розглянемо складові цього запиту. Для зручності, дані будуть занесені до таблиці 2.2. (див. табл. 2.2)

Таблиця 2.2 – Складові запиту

Складова частина	Довжина	Опис
HEADER	3 символи	Визначає ціль виклику до серверу, за яким сервер буде розподіляти та розуміти, що саме хоче клієнт. Наприклад, «UPD» – запит на перевірку оновлень для додатку.
<SEP>	1 символ	Визначає закінчення однієї частини запиту, та початку іншої. Для цього проекту було вибрано « » (з англ. vertical bar)
DATA	N символів	Складається з даних, які клієнт надсилає на перевірку. В деяких запитах може бути пустим. Як приклад, «EXT», що означає запит на отримання списку наявних моделей, не має отримувати DATA поле.

Так як, в додаткові використовується HTTPS, звернемо увагу на те, як працює TLS-handshake. [35] Алгоритм взаємодії клієнту з сервером під час HTTPS наведено на рисунку 2.12. (див. рис. 2.12)

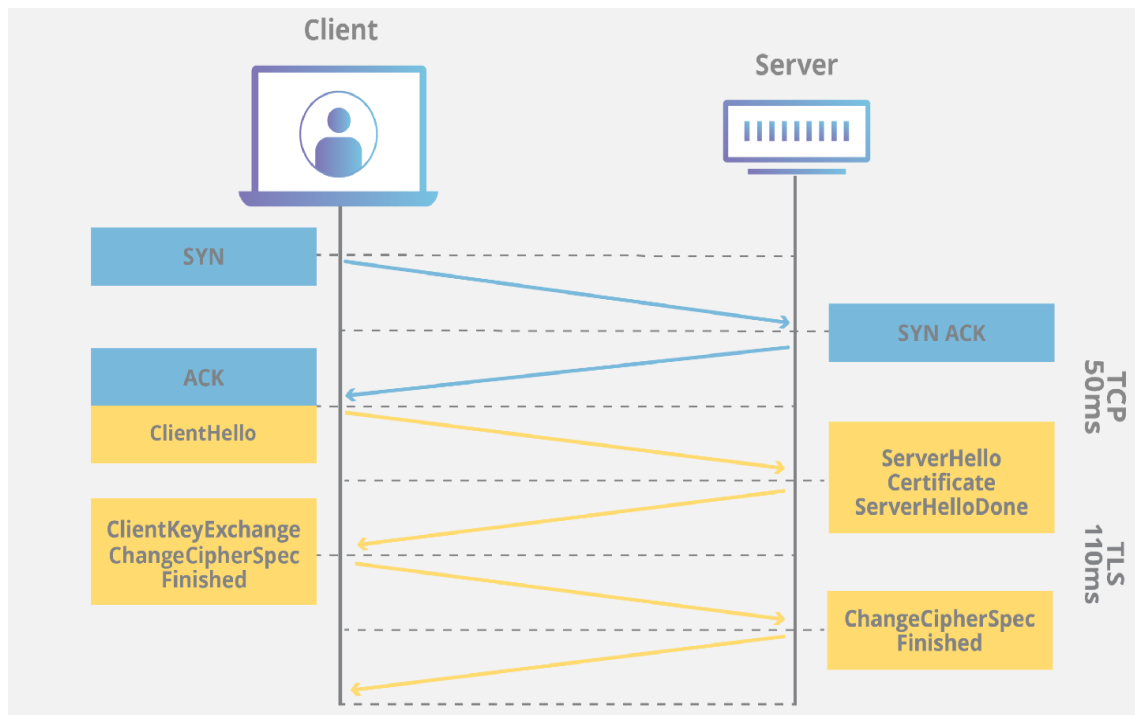


Рисунок 2.12 – Алгоритм роботи TLS-handshake

Клієнт при запускові, надсилає до сервера «hello» повідомлення, в якому є дані про TLS сертифікат, тощо. Також, в цьому повідомленні, буде зберігатись випадковий набір байтів, що називається «random клієнта». У відповідь сервер надсилає SSL-сертифікат. [36]

Після отримання сертифікату, клієнт перевіряє цей сертифікат в центрі сертифікації, підтверджуючи, що він взаємодіє з сервером за дійсним доменом. (в нашому випадку, з сервером додатку «ImgProPlus»)

Якщо сертифікат успішно перевірений, відбувається створення сеансових ключів, які перевіряють правильність генерації ключів. Після цього етапу, сервер та клієнт готові до обміну даними.

Під час всього алгоритму, використовується асиметрична криптографія (відкритий та закритий ключі). Тому важливо при розгортанні роботи сервера, згенерувати пару ключів, для подальшої роботи з клієнтами. Генерація ключів описана в Додатку А.

Робота з WebAPI (сервіс «LiqPay»)

Оскільки створення власної системи платежів та транзакцій, неймовірно складна та відповідальна робота, було прийнято рішення використовувати зовнішню фінансову систему (таку практику використовують компанії Steam, OpenAI, X, тощо). Вибір пав на сервіс «LiqPay», що є українським платіжним сервісом, інтегрованим з «ПриватБанк». [37]

Безпосередньо в самій БД будуть зберігатись лише квитанції про платіж, щоб у випадку проблем отримання преміуму, адміністратор зміг вирішити та зв'язувався з користувачем, так кажучи був посередником (з англ. middle-man).

API в «LiqPay» також використовує асиметричне шифрування, [38] що дозволяє за допомогою отриманих ключів у власному кабінеті, формувати посилання на необхідні платежні «міні-сайти», в яких користувач вже сам зможе оплатити зручним йому засобом. За допомогою зручної документації це реалізувати доволі легко. Нижче наведено інтерфейс додатку ImgProPlus та як працює оплата. (див. рис. 2.13-2.14)

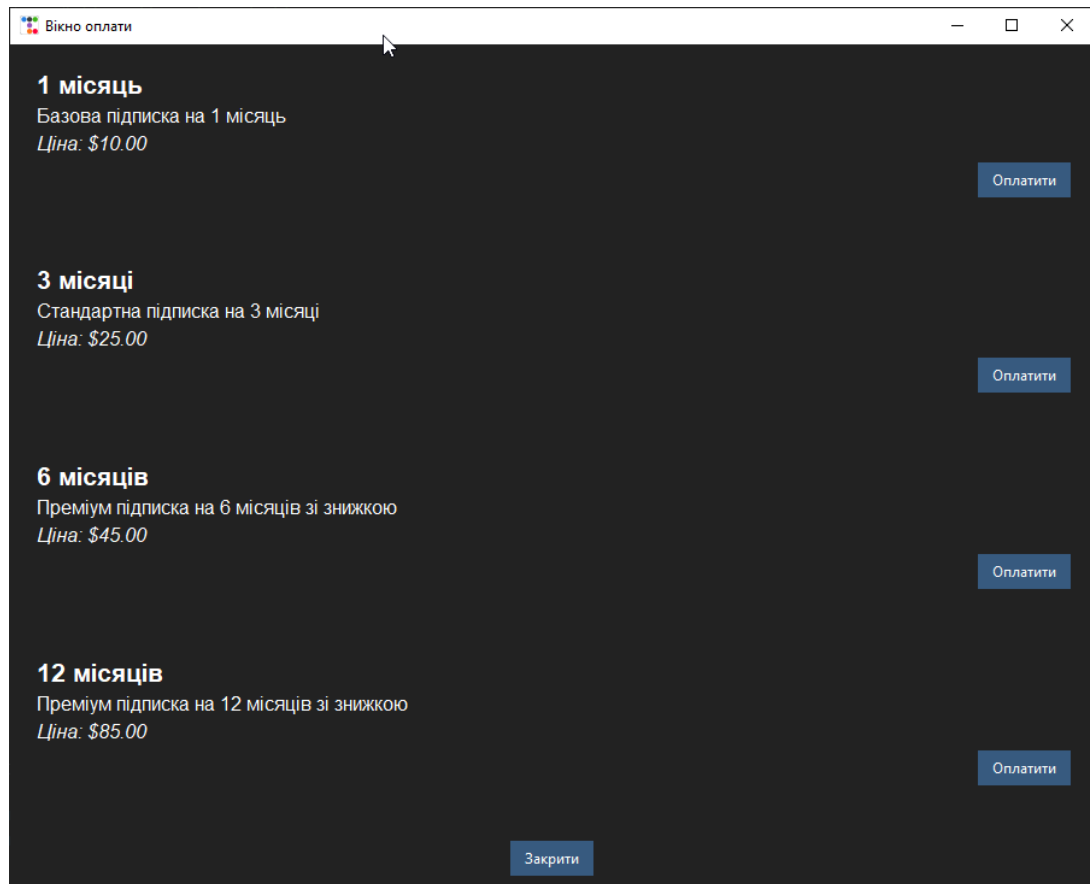


Рисунок 2.13 – Вікно оплати

Після натиснення на кнопку «Оплатити», користувач побачить наступне (див. рис. 2.14)

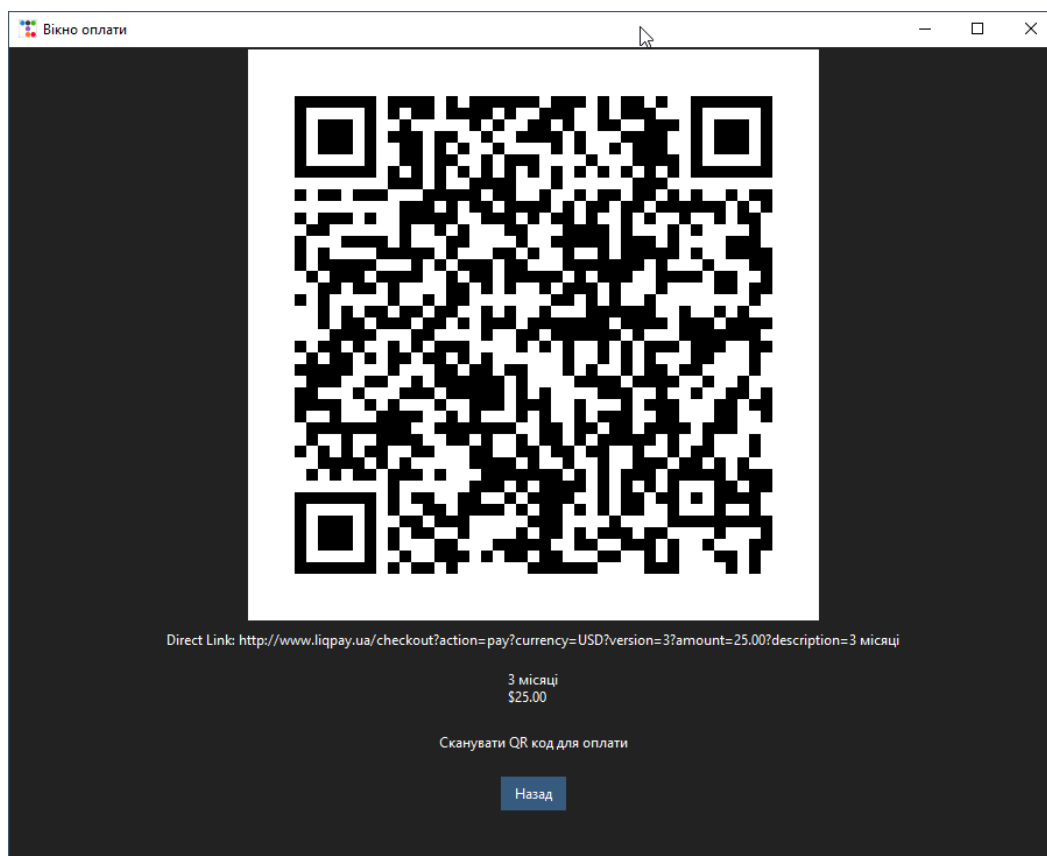


Рисунок 2.14 – згенерований код для трьох-місячної оплати

Користувач може відсканувати код, або натиснути на посилання нижче, і перейти в браузер (якщо користувачу зручніше оплатити з комп'ютера).

Генерується код наступним чином – за допомогою LiqPayAPI ми генеруємо форму оплати (див. рис. 2.15), яка використовує тег `<form>` в HTML, та передаємо її в браузер користувача або користувач сканує QR-код та оплачує з телефону. [38]

```
liqpay = LiqPay(public_key, private_key)
html = liqpay.cnb_form({
    'action': 'pay',
    'amount': '1',
    'currency': 'USD',
    'description': 'description text',
    'order_id': 'order_id_1',
    'version': '3'
})
```

Рисунок 2.15 – Приклад використання LiqPayAPI

За допомогою швидкості роботи, та зручності LiqPay, розгортати додаток стало набагато легше. (Не кажучи про те, що податок автоматично вираховується з доходів, що полегшує роботу ФОП).

2.2. Детальне проєктування

Модель звертання в службу підтримки

Одним з бізнес процесів, які будуть використані в додаткові, можливість звертання до служби підтримки. В цьому процесі задіяні користувач та служба підтримки.

Користувач надсилає запит до служби підтримки, і вона відповідає, якщо користувача відповідь не втішила, кейс не закривається до моменту вирішення проблеми. [39] У випадку, якщо користувач не відповідає після 48 годин, то кейс автоматично зачиняється.

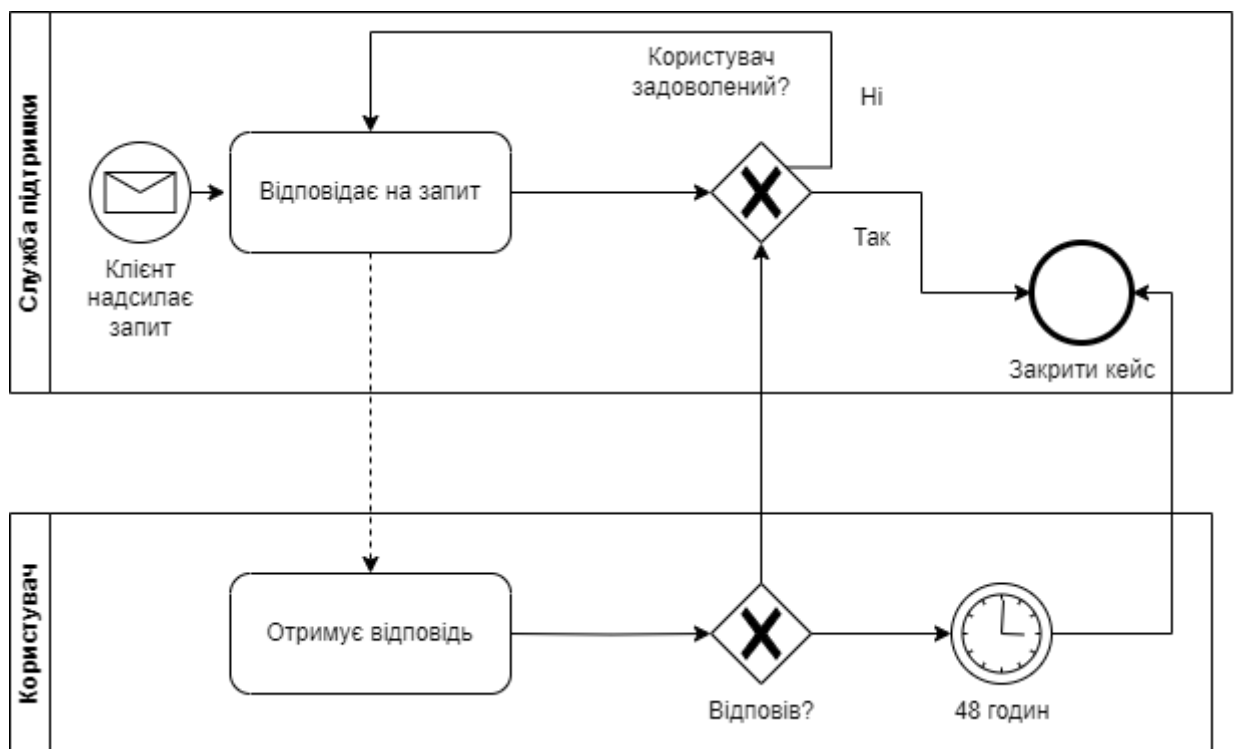


Рисунок 2.16 – Модель звертання в службу підтримки

Модель оформлення підписки

Ще одним з бізнес процесів, які будуть використані в додаткові, можливість оформлювати підписку. В цьому процесі задіяні користувач, сервер додатку та API банку.

Користувач хоче придбати підписку, сервер надає список всіх наявних підписок, після вибору підписки, може купити або відмовитись від цього. При купівлі задіяний також API банку, який опрацьовує платіж, та видає відповідну інформацію в додаток.

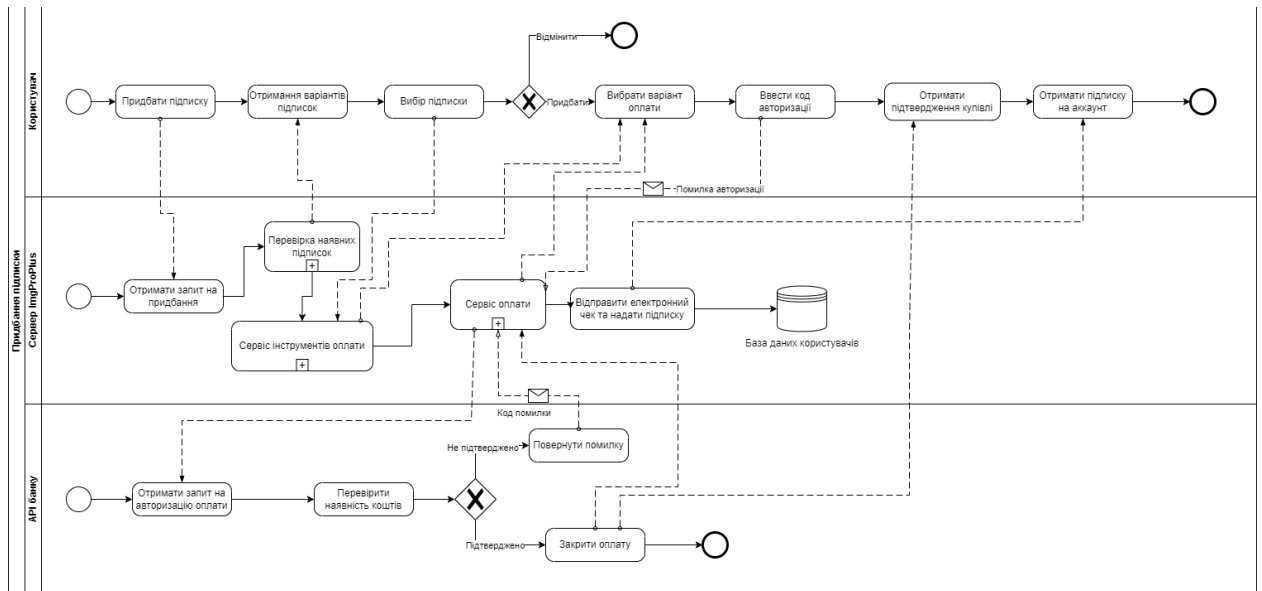


Рисунок 2.17 – Модель придбання підписки

Модель реєстрації

В додаткові буде використаний також бізнес процес реєстрації користувача. Задіяними в процесі є користувач та сервер. Користувач намагається зареєструватись та вводить дані, сервер отримує цей запит та відповідає (в різних випадках по різному). В результаті, якщо все вірно, користувач зареєстрований і сервер зберіг його дані в себе в базі даних.

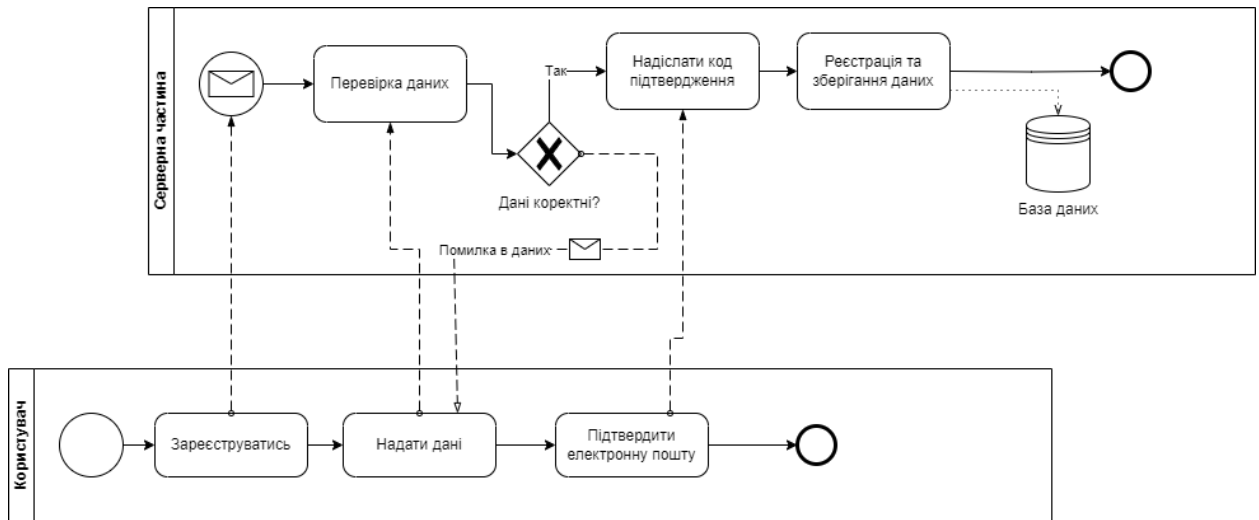


Рисунок 2.18 – Модель реєстрації

2.2.1. Структурні моделі

За клієнт-серверною моделлю, додаток складається з пакетів `app`, `server-side` та доданим пакетом `clip`, який має в собі всі необхідні утиліти для обробки зображень. [40] Більше детально структурну модель подано на рисунку 2.19. (див. рис. 2.19)

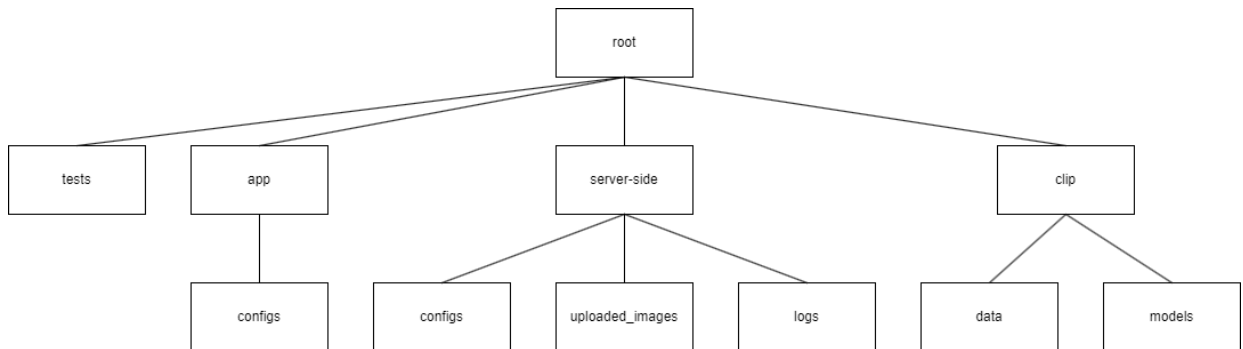


Рисунок 2.19 – Структурна модель пакетів застосунку ImgProPlus

В пакеті `clip` знаходяться файли коду `clip_interrogator.py` (зміст наведено в Додаток А), та додаткові директорії `data` і `models`. В них зберігаються дані, необхідні для роботи моделей CLIP (.safetensors файли та словники, тобто .txt файли).

В пакеті `server-side` знаходяться файли для роботи серверу, тобто `server.py`, `database.py` та додаток для адміністраторів, `admin_app.py`. Додаткові

директорії `configs`, `uploaded_images` та `logs` мають в собі конфігураційні файли для роботи програми та серверу, завантажені користувачем зображення, які знаходяться в даний момент в обробці та логи серверу і БД.

В пакеті `app` знаходиться додаток користувача – `app.py`, `client.py` та інші дочірні файли для роботи програми. Також існує директорія з конфігураційними файлами для роботи програми (налаштування користувача, токен, кеш, тощо).

В пакеті `tests` знаходяться тести до всіх інших пакетів, та вони не є частиною поширюваного ПЗ, оскільки їх використовують лише розробники для перевірки правильності написання коду.

Наведемо UML діаграми для класу `clip_interrogator.py`. (діаграми для GUI елементів в цьому пункті наводитись не будуть оскільки взаємодія між компонентами там мінімальна, але вони будуть наведені в пункті 2.3). [41]

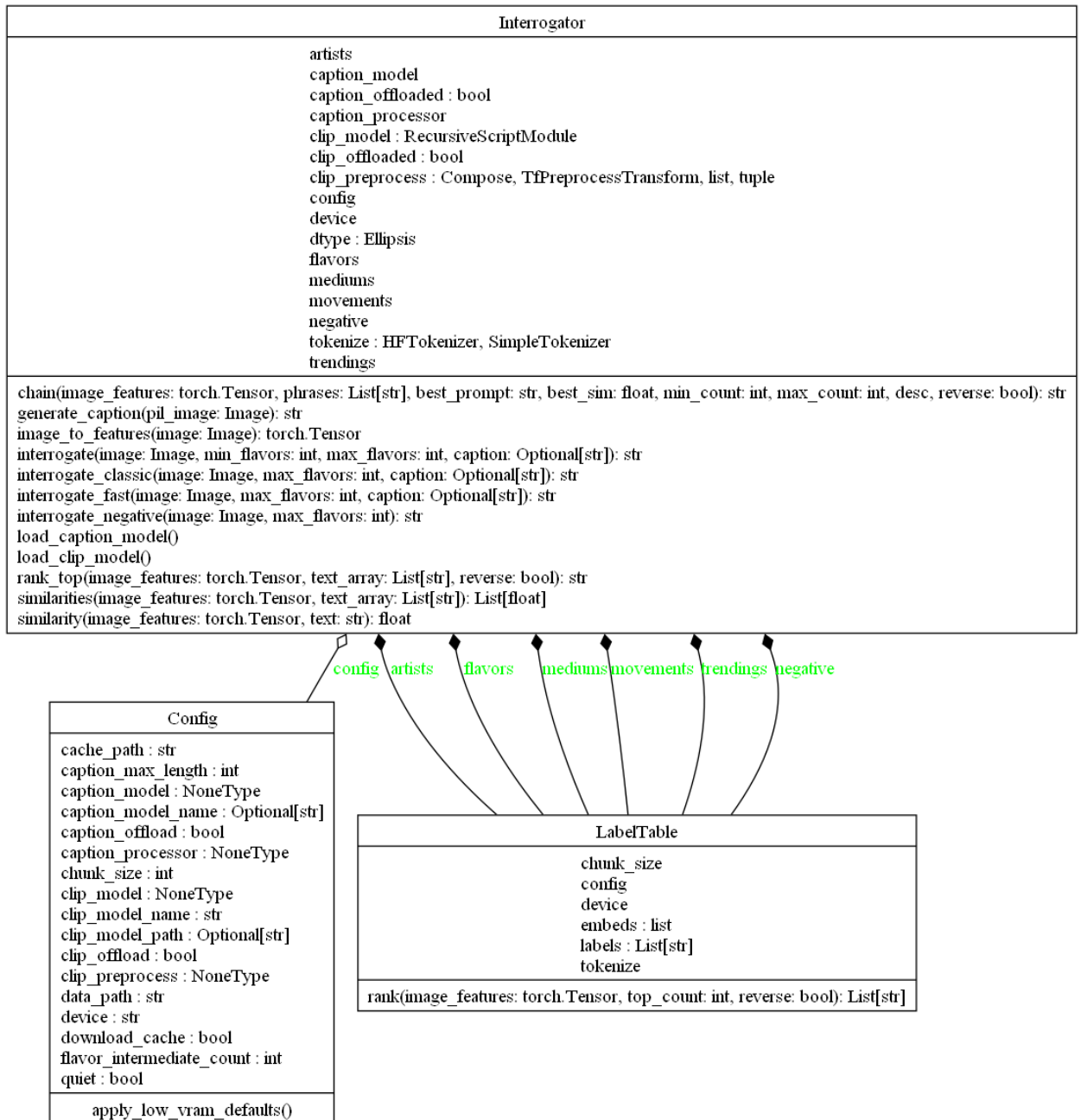


Рисунок 2.20 – UML діаграма clip_interrogator.py

2.2.2. Моделювання функціональності та взаємодії

Основну функціональність додатку «ImgProPlus» буде подано у вигляді блок-схем, що відображають покрокову логіку та роботу програми. Ключові діаграми подані на рисунках 2.21 – 2.24.

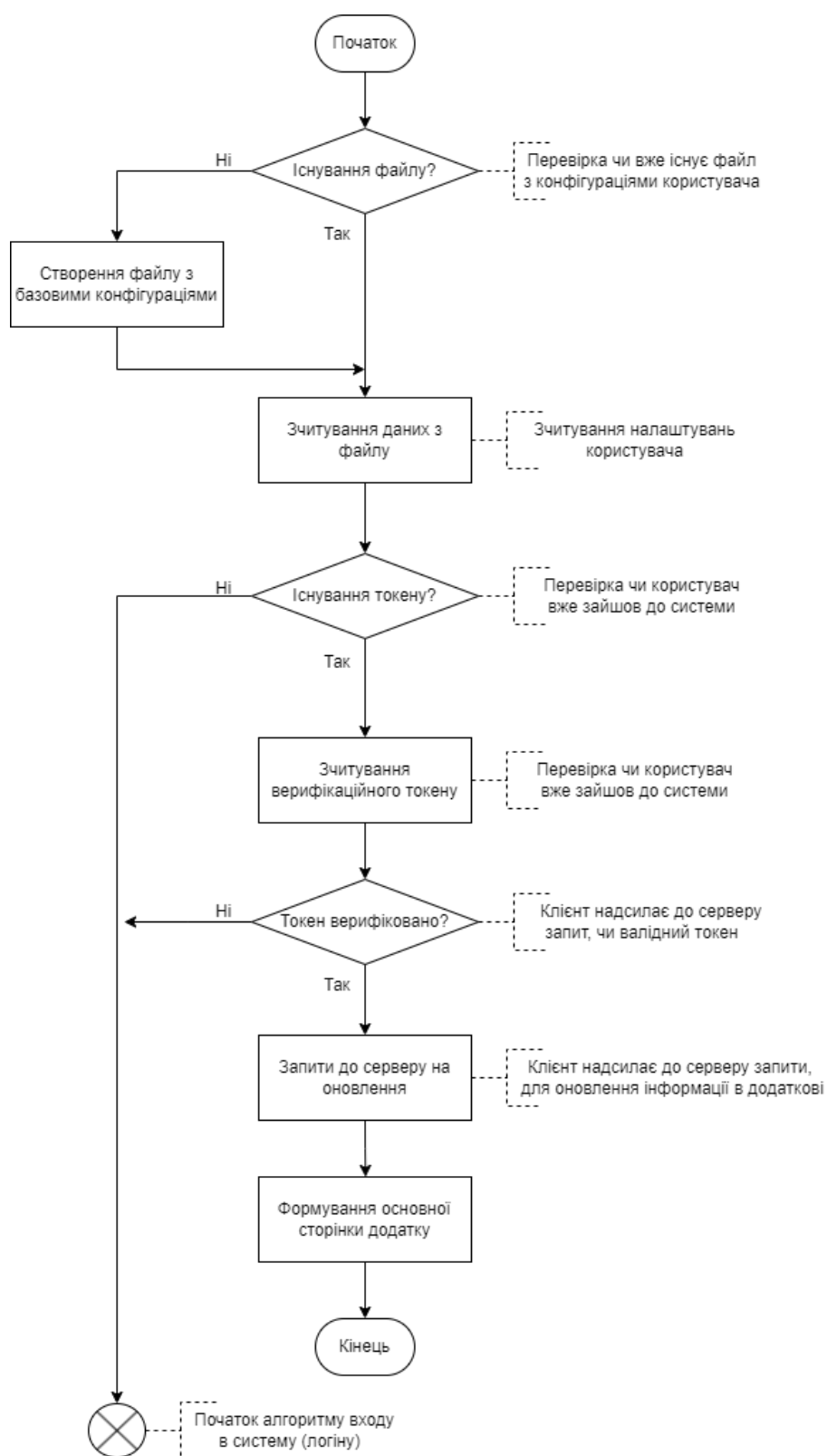


Рисунок 2.21 – Діаграма моделювання роботи додатку при запуску програми

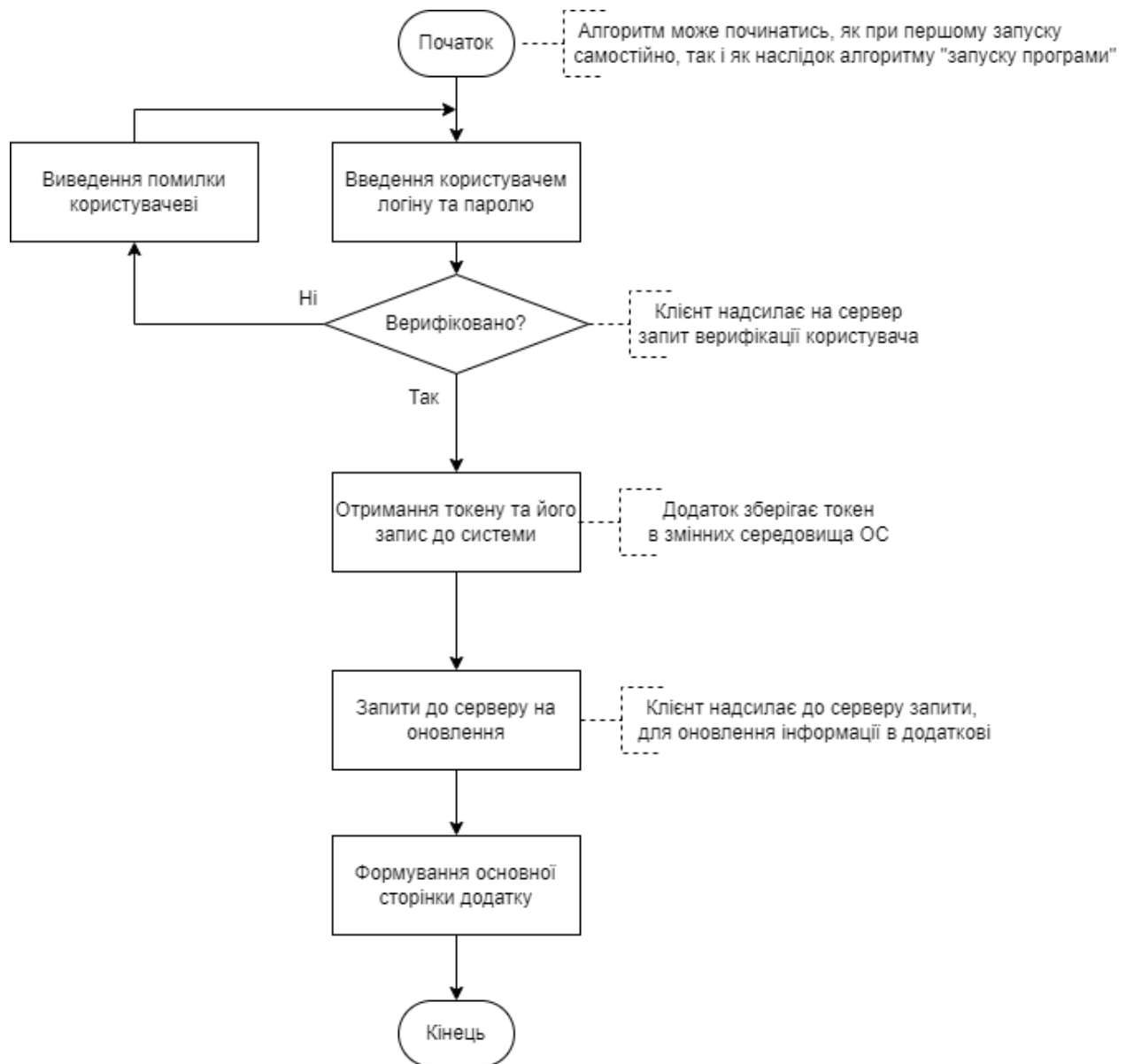


Рисунок 2.22 – Діаграма моделювання входу в додаток (логіну)

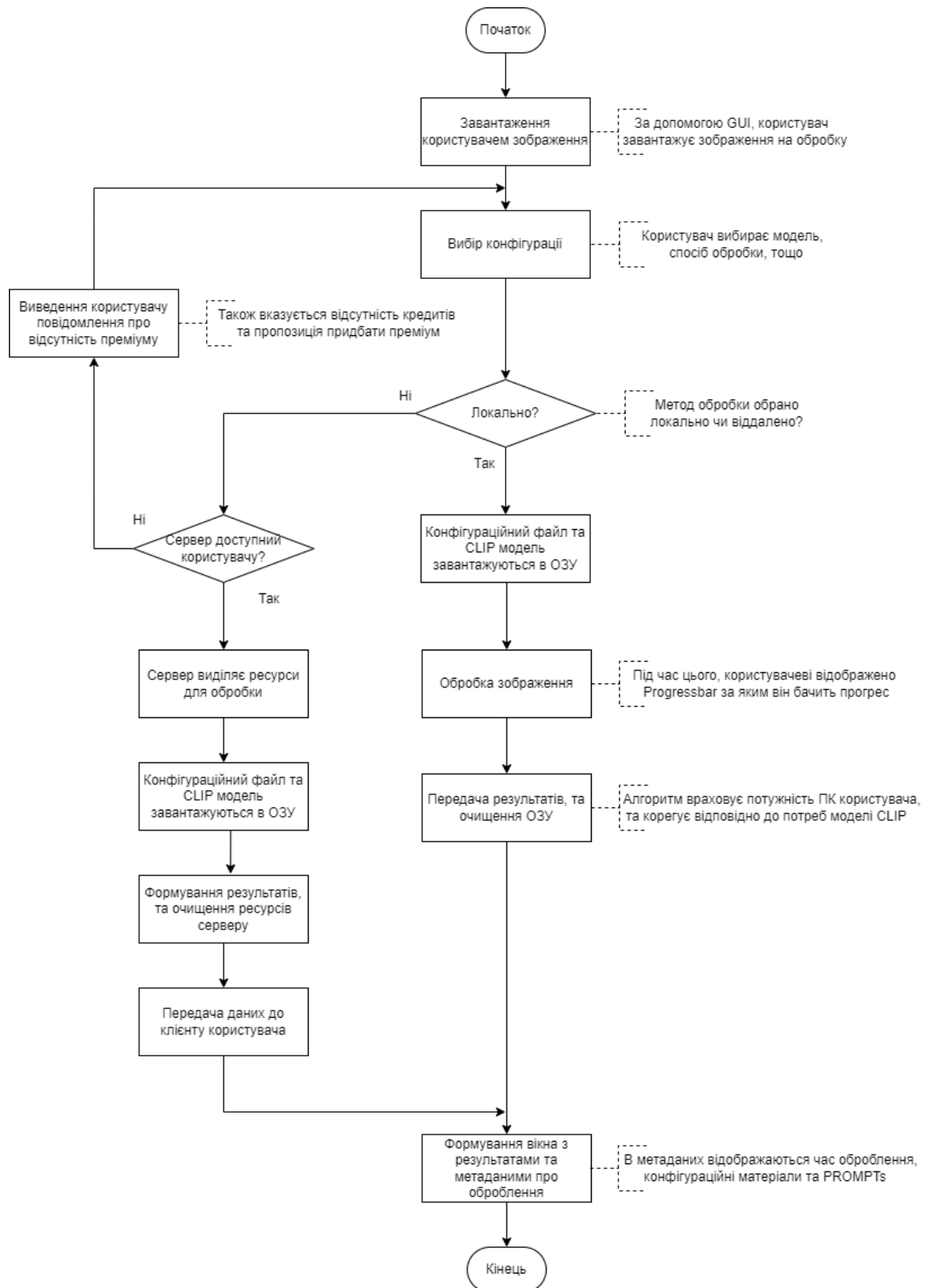


Рисунок 2.23 – Діаграма моделювання обробки зображення

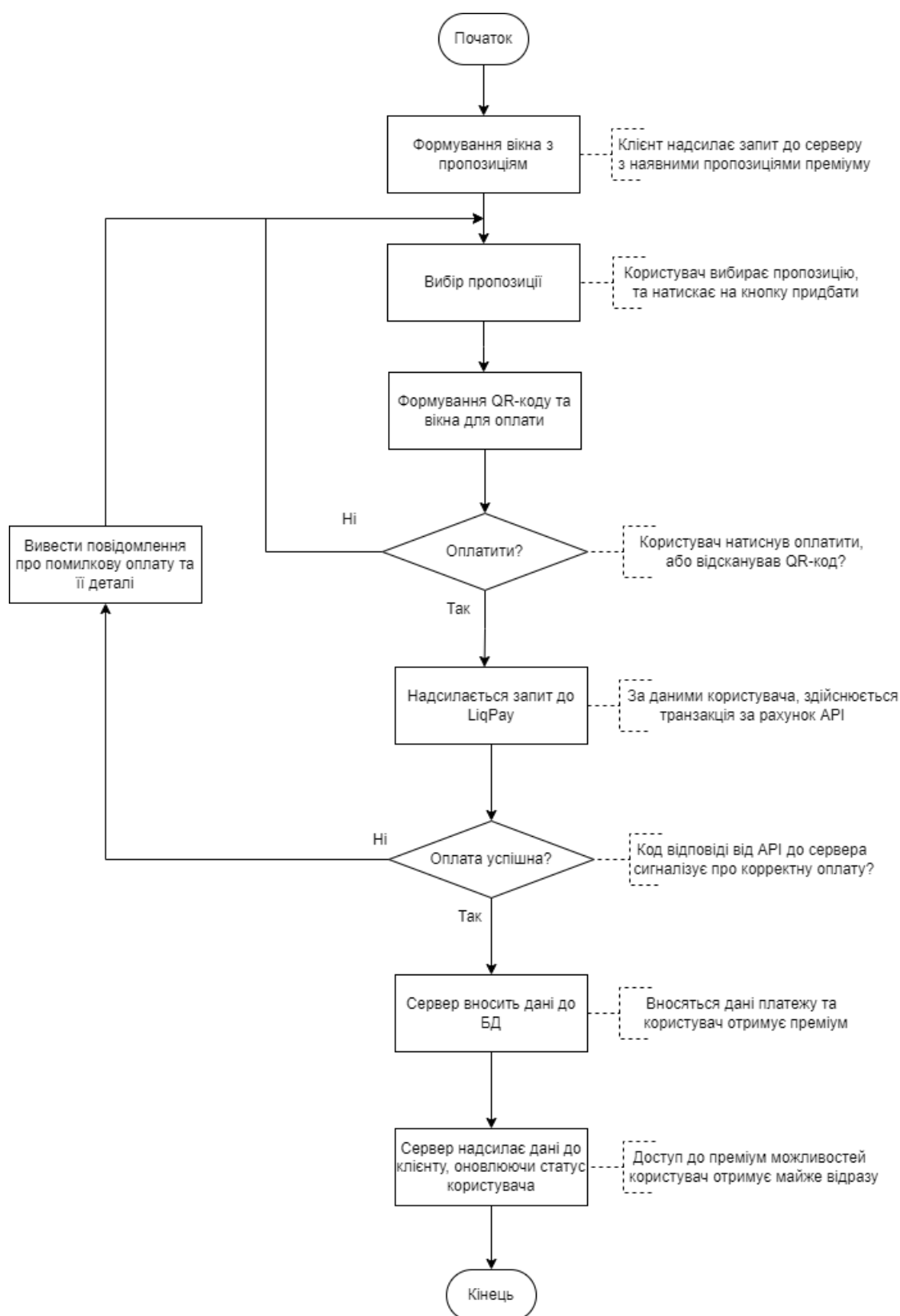


Рисунок 2.24 – Діаграма моделювання придбання підписки

2.3. Проектування інтерфейсу користувача

Відповідно до пункту 1.3.2.1.1, було написано інтерфейс користувача. Карта переходів (Transition Map) [42] реалізована відповідно до діаграми на рисунку 1.5. Для економії місця було вирішено відобразити інтерфейс лише українською мовою (додаток підтримує англійську та українські мови, як зазначено у вимогах). Спроектвані інтерфейси подані на рисунках 2.25 – 2.36.

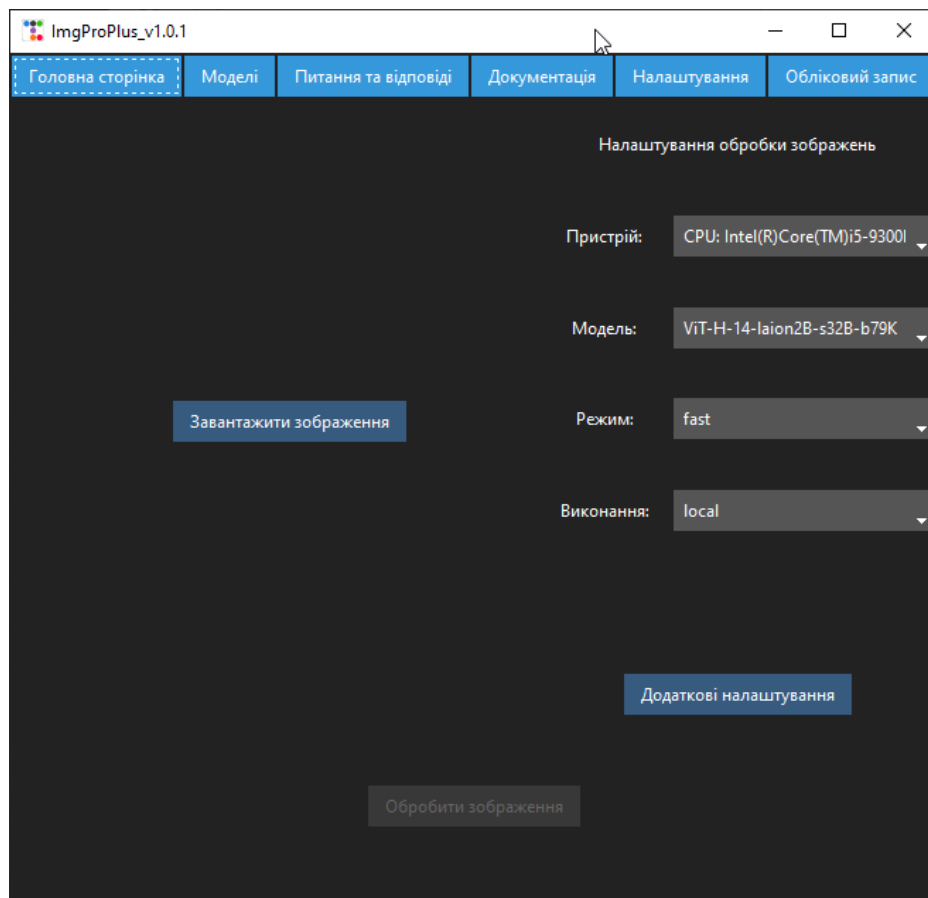


Рисунок 2.25 – Інтерфейс вікна «Головна сторінка»

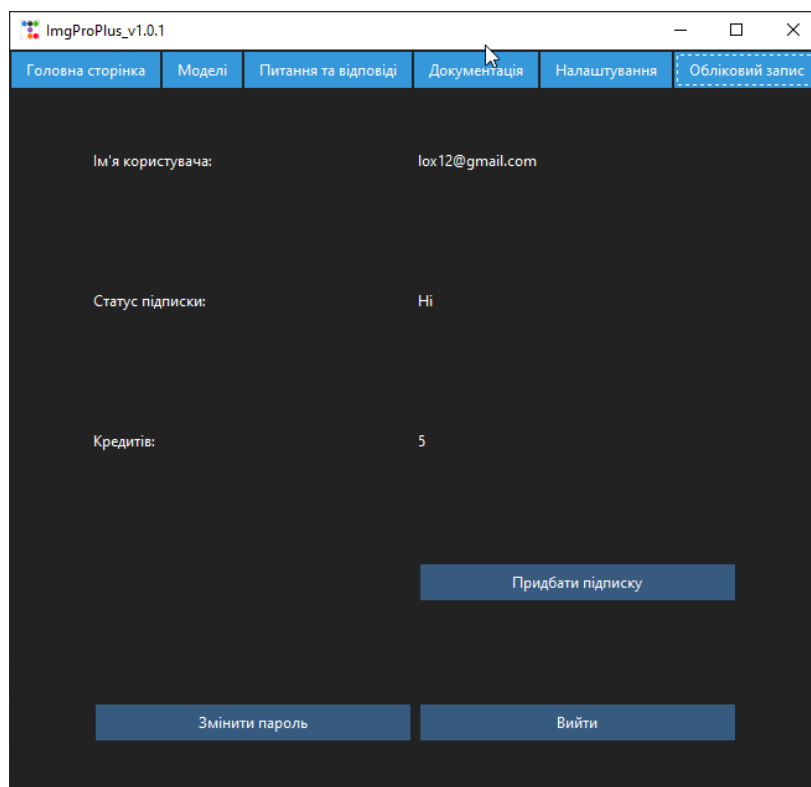


Рисунок 2.26 – Інтерфейс вікна «Аккаунт»

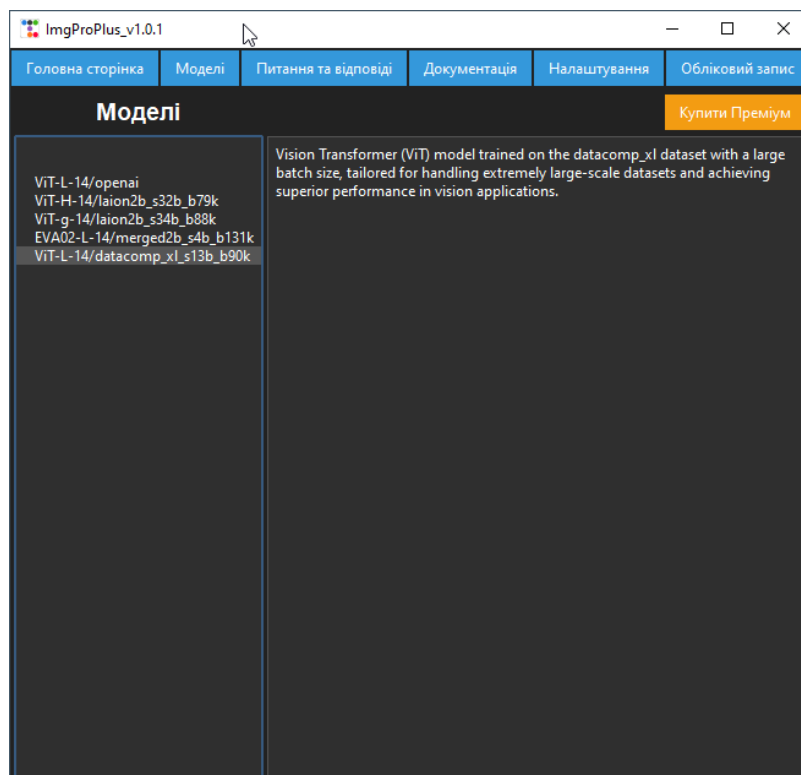


Рисунок 2.27 – Інтерфейс вікна «Моделі»

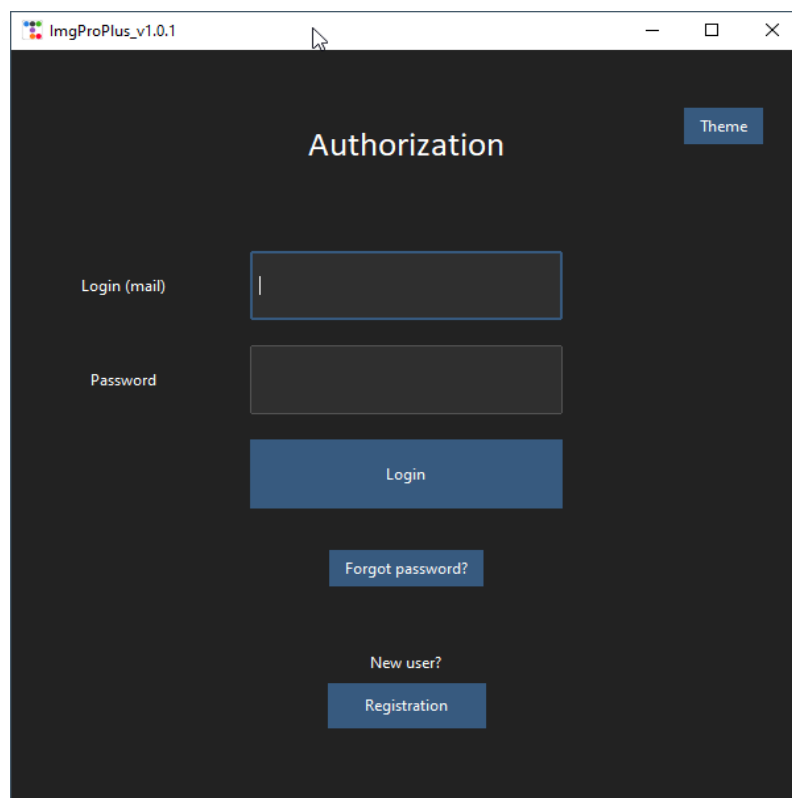


Рисунок 2.28 – Інтерфейс вікна «Авторизація»

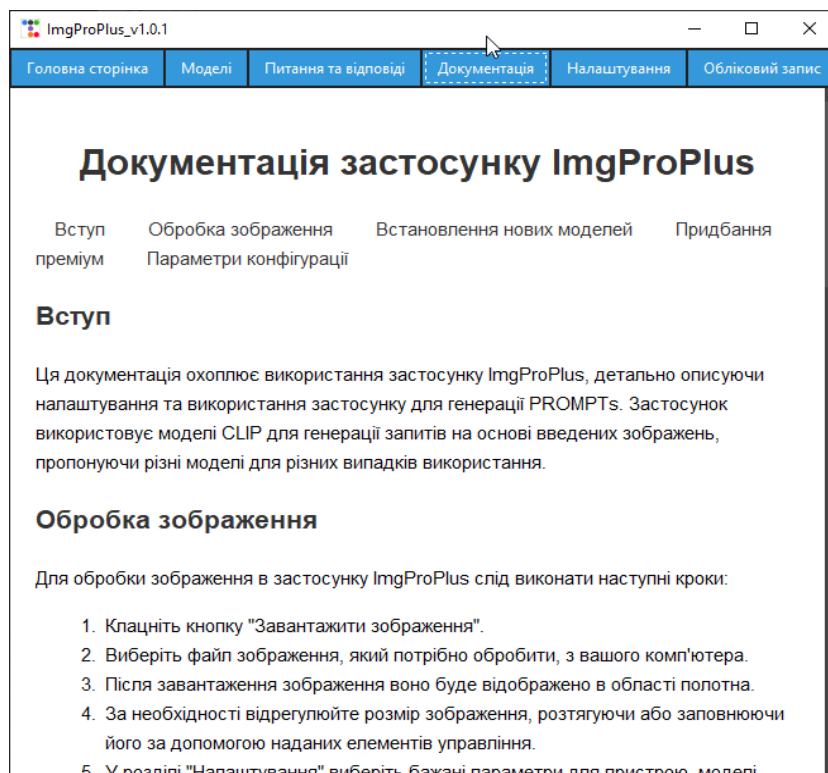


Рисунок 2.29 – Інтерфейс вікна «Документація»

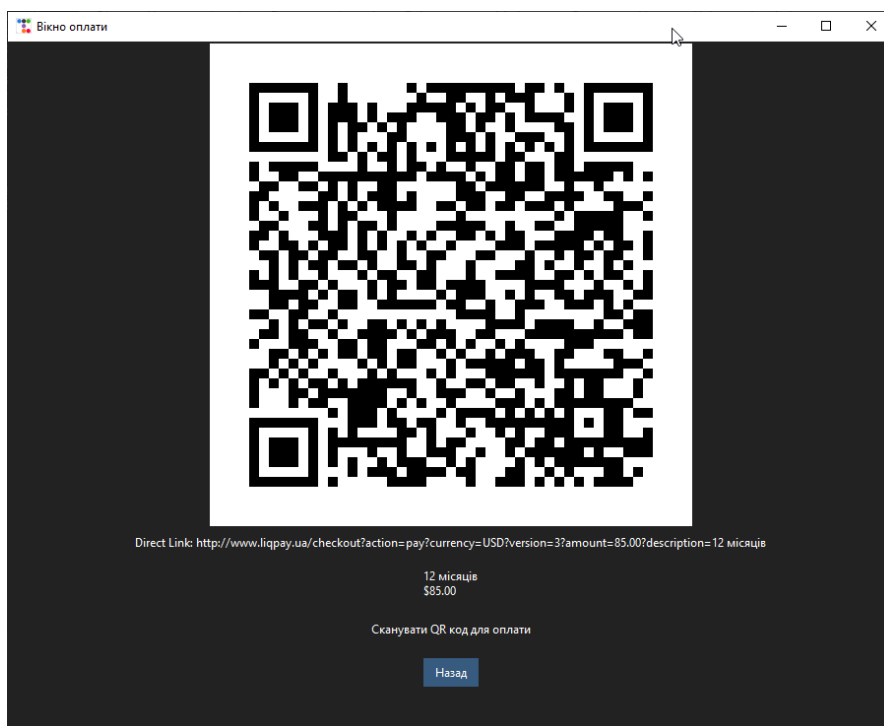


Рисунок 2.30 – Інтерфейс вікна «Оплата»

Рисунок 2.31 – Інтерфейс вікна «Реєстрація»

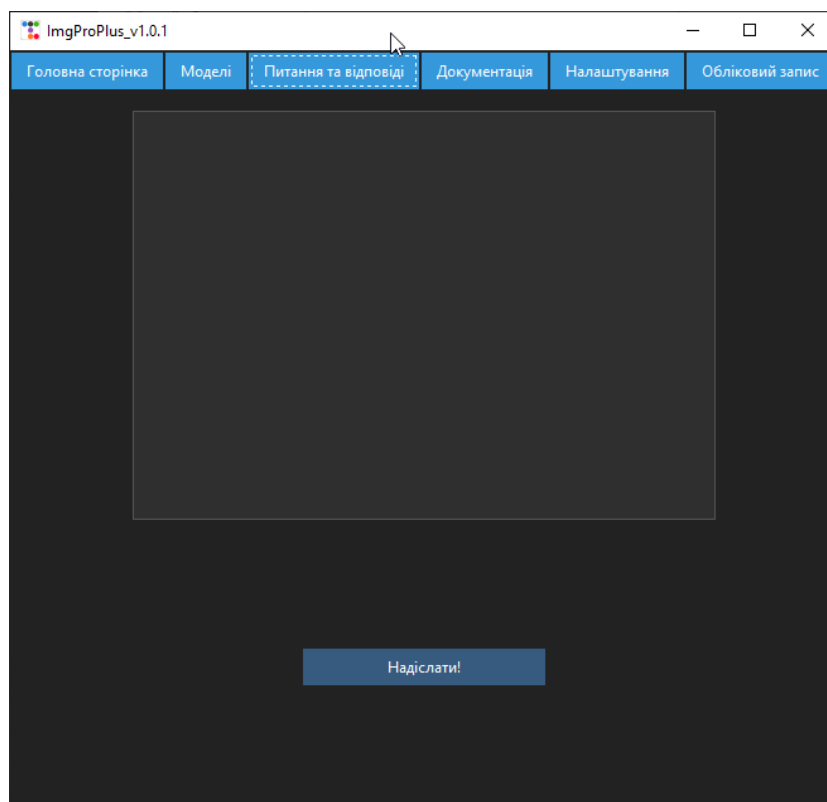


Рисунок 2.32 – Інтерфейс вікна «Запит до Q&A»

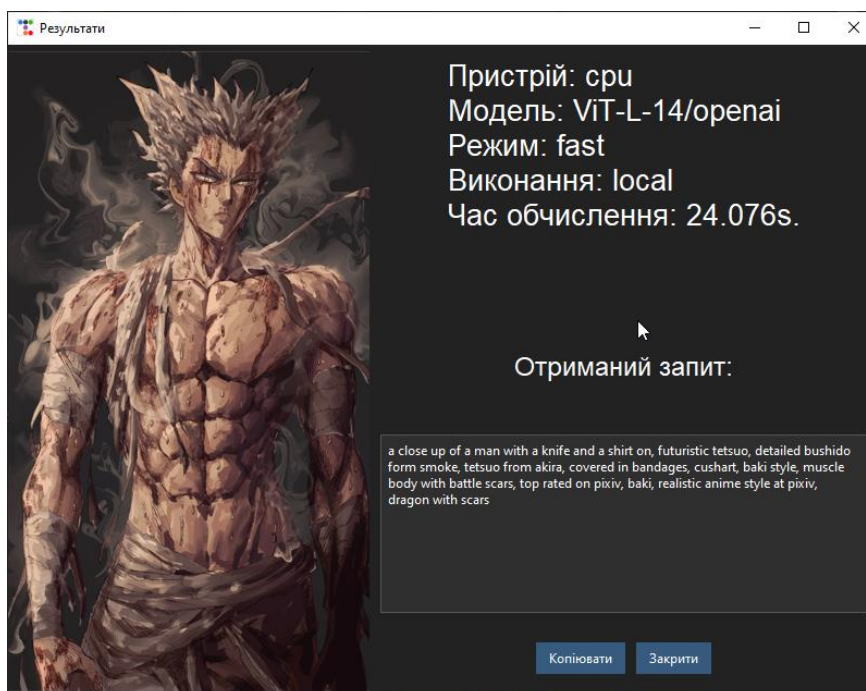


Рисунок 2.33 – Інтерфейс вікна «Сторінка результатів»

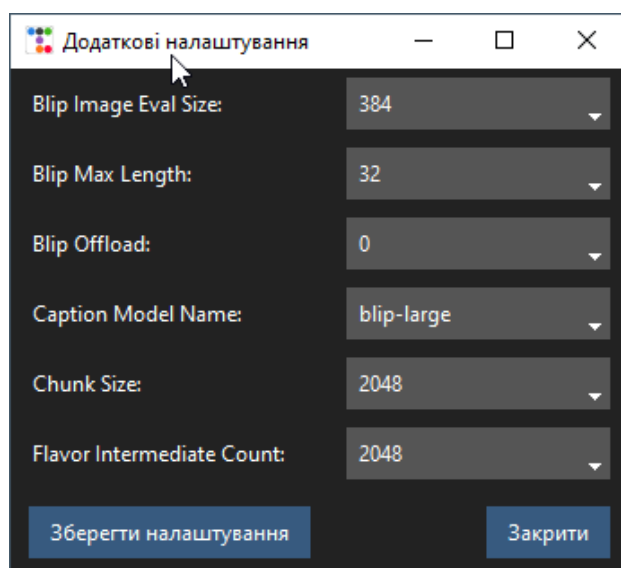


Рисунок 2.34 – Інтерфейс вікна «Налаштування обробки»

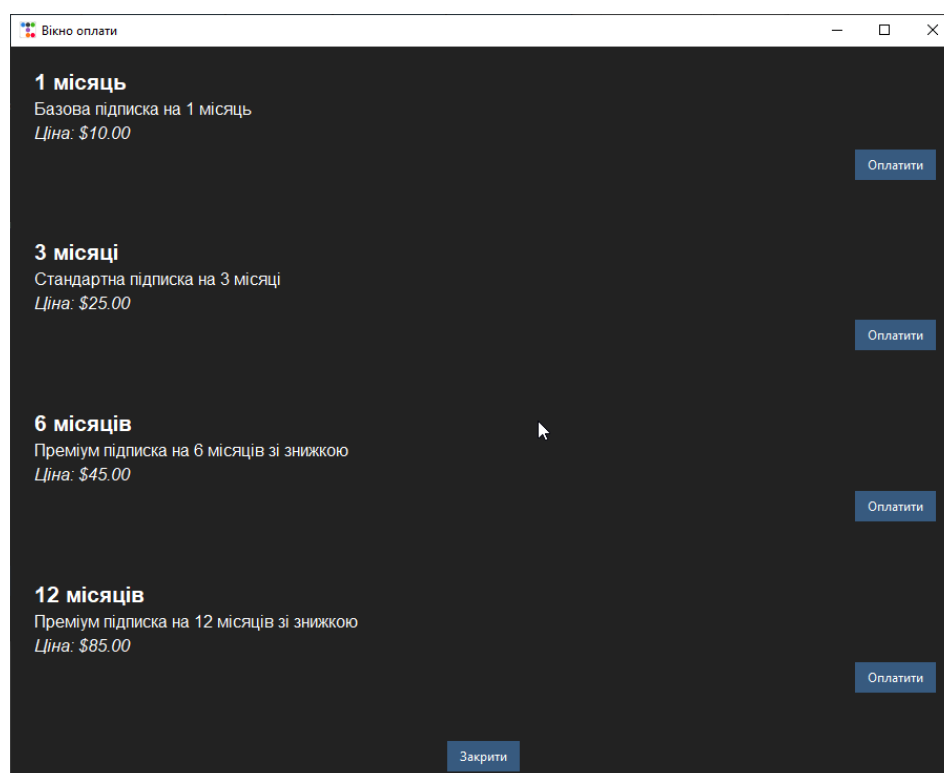


Рисунок 2.35 – Інтерфейс вікна «Купівля підписки»

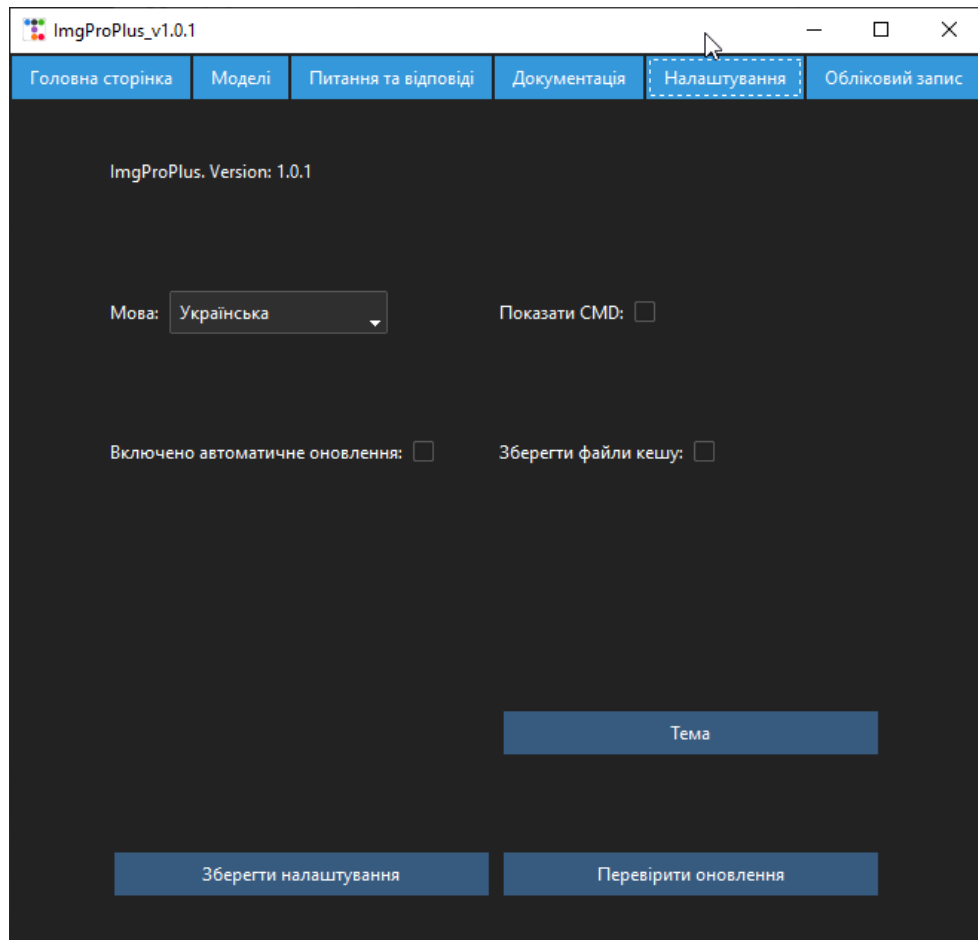


Рисунок 2.36 – Інтерфейс вікна «Налаштування додатку»

Оскільки, при першому запуску та реєстрації, додаток автоматично створює налаштування для користувача, базуючись на його системі, то вікна «Реєстрація» та «Авторизація» надані англійською мовою, так як комп'ютер, на якому проходила розробка використовує англійську мову, як основну.

Як бачимо з рисунків 2.25 – 2.36 інтерфейс розроблено відповідно до вимог в розділі 1.3.2.1.

2.4. Реалізація програмного забезпечення

Метрики коду

Після написання програми, за допомогою утиліти «VS Code Counter» [43] ми отримали відповідні метрики по кожній з директорій додатку.

В директорії clip ми маємо лише один файл clip_interrogator.py. Метрики директорії подані на рисунку 2.37. (див. рис. 2.37)

Files					
filename	language	code	comment	blank	total
clip/clip_interrogator.py	Python	396	5	74	475

Рисунок 2.37 – Метрики директорії clip

Як бачимо коду та коментарів в сумі 401 рядок.

В директорії app ми маємо ряд файлів, які подано разом з метриками на рисунку 2.38. (див. рис. 2.38)

filename	language	code	comment	blank	total
app/addons.py	Python	64	7	16	87
app/app.py	Python	823	125	290	1,238
app/client.py	Python	57	18	23	98
app/docs.html	HTML	169	0	6	175
app/docs_u.html	HTML	169	0	7	176
app/localization.py	Python	116	9	21	146
app/login_page.py	Python	534	21	158	713
app/main.py	Python	73	15	16	104
app/my_logger.py	Python	10	3	5	18
app/settings.py	Python	46	3	19	68
app/slide_widget.py	Python	15	0	4	19

Рисунок 2.38 – метрики директорії app

Директорія app має в собі файли, як мови Python так і HTML. В сумі коду та коментарів директорії app отримаємо 2277 рядків.

Директорія server-side має ряд файлів які подано разом з метриками на рисунку 2.39. (див. рис. 2.39)

filename	language	code	comment	blank	total
server-side/admin_app.py	Python	592	73	191	856
server-side/database.py	Python	488	17	103	608
server-side/mail_serv.py	Python	21	1	5	27
server-side/my_logger.py	Python	10	3	5	18
server-side/server.py	Python	275	31	61	367

Рисунок 2.39 – Метрики директорії server-side

Всі файли написані мовою Python (не враховуючи configuration file, який займає 4 рядки та поданий у форматі JSON). В сумі коду та коментарів директорії app отримаємо 2277 рядки.

Реалізація вимог

Після написання та реалізація програмного забезпечення, необхідно перевірити якість виконання заданих вимог. Почнемо розгляд з груп функціональних вимог.

Виконання функціональних вимог

FR1.01:

–Вхід до додатку, реєстрація та отримання коду підтвердження реалізовані в файлі login_page.py (див. Додаток А.13).

–Реєстрація користувачів здійснюється через електронну пошту та пароль, що реалізовано в тому ж файлі (див. Додаток А.13).

–Перевірка правильності введених даних та відображення повідомлень про помилку здійснюється в файлах login_page.py та slide_widget.py (див. Додаток А.13, та див. Додаток А.12).

–Відправка коду підтвердження здійснюється через сервер, описаний в файлі mail_serv.py (див. Додаток А.5).

FR1.02:

–Керування аккаунтом користувача реалізоване в файлі app.py (див. Додаток А.6).

FR1.03:

–Налаштування параметрів облікового запису користувача виконується також в файлі `app.py` (див. Додаток А.6).

FR1.04:

–Управління користувачами та їх обліковими записами виконується в адмінському додатку, реалізованому в файлі `admin_app.py` (див. Додаток А.4).

–Створення, перегляд, редагування та видалення користувачів здійснюється через цей адмінський додаток. (див. Додаток А.4)

FR1.05:

–Керування запитаннями та відповідями реалізоване в адмінському додатку в файлі `admin_app.py` (див. Додаток А.4).

FR1.06:

–Управління платежами виконується в адмінському додатку в файлі `admin_app.py` (див. Додаток А.4).

–Сторінка управління платежами та деталізація платежів реалізовані в цьому ж файлі (див. Додаток А.4).

FR1.07:

–Перегляд логів серверу здійснюється через адмінський додаток в файлі `admin_app.py` (див. Додаток А.4).

FR2.01:

–Можливість користувачам прочитати документацію додатку, що включає інформацію щодо функціоналу, налаштувань та інструкцій з використання додатку, реалізована у файлі `docs.html` (див. Додаток А.11), а функціонал відображення у файлі `app.py` (див. Додаток А.6).

FR2.02:

–Можливість користувачам звертатися в підтримку з доступом до ресурсів отримання допомоги, реалізована у файлі `app.py` (див. Додаток А.6), а частина обробки запитів реалізована в `admin_app.py` (див. Додаток А.4) та `server.py` (див. Додаток А.3).

FR3.01:

–Система має можливість перетворювати різні формати зображень при завантаженні, що реалізовано у файлі `clip_interrogator.py` (див. Додаток А.1).

FR3.02:

–Система дозволяє користувачам зберігати результати обробки зображень, що виконується в файлі `clip_interrogator.py` (див. Додаток А.1).

FR3.03:

–Система має можливість опрацювати зображення за допомогою завантажених моделей обробки, таких як CLIP v1, v2 і т.д., що реалізовано в файлі `clip_interrogator.py` (див. Додаток А.1).

FR3.04:

–Користувач має можливість вибору типу опрацювання «На сервері» або «На локальній машині», що реалізовано в файлі `app.py` (див. Додаток А.6).

FR3.04.01:

–При виборі «На сервері», якщо не має достатню кількість кредитів і не має підписки, система відображає повідомлення про це, що реалізовано в файлах `client.py` (див. Додаток А.9), `database.py` (див. Додаток А.2) та `server.py` (див. Додаток А.3).

FR3.04.02:

–При виборі «На локальній машині», користувачу відобразиться повідомлення про те, що на його ПК буде проведено перевірку потужності, та після цього відобразено результати, що виконується в файлах `app.py` (див. Додаток А.6) та `addons.py` (див. Додаток А.10).

FR3.05:

–Система має надавати можливість налаштовувати опрацювання зображення, параметри налаштування які «Кількість кроків», «Негативний PROMPT», «Креативність» тощо, реалізовано в файлі `app.py` (див. Додаток А.6)

FR3.06:

–Система має можливість зберегти результати опрацювання зображення згідно з заданими PROMPTS (інструкціями), що виконується в файлі app.py (див. Додаток А.6).

FR3.07:

–Система має можливість завантажувати нові моделі обробки зображень, наприклад, CLIP v1, v2 і т.д., що реалізовано в файлах app.py (див. Додаток А.6), server.py (див. Додаток А.3), clip_interrogator (див. Додаток А.1).

FR3.08.01:

–Система має надавати вибирати модель із наявних моделей, що виконується в файлі app.py (див. Додаток А.6).

FR3.08.02:

–Система у випадку, якщо користувач не має підписку, то йому при спробі завантажити моделі, відображається повідомлення, яке пояснює, що ця можливість наявна лише з підпискою, та пропонує її купити, що реалізовано в файлах app.py (див. Додаток А.6) та clip_interrogator.py (див. Додаток А.1).

FR3.09:

–Система має надавати можливість завантажувати зображення для подальшої обробки, що виконується в файлі app.py (див. Додаток А.6).

FR4.01:

–Система має надавати можливість користувачам придбати підписку за переліком можливих підписок, що реалізовано в файлі app.py (див. Додаток А.6). Серверна частина реалізована в server.py (див. Додаток А.3)

FR4.02:

–Система має дозволяти користувачам переглядати наявні підписки у вигляді списку, з якого вони можуть вибирати відповідно, а також вказувати ціни та знижки (якщо такі наявні), що реалізовано в файлі app.py (див. Додаток А.6).

Вимоги, які наведені в таблиці 1.9 реалізовані відповідно, їх виконання зображене на рисунках 2.25 – 2.36.

Вимоги SI 1-01 та SI 1-02 реалізовані в файлах server.py (див. Додаток А.3), app.py (див. Додаток А.6). Для виконання цих вимог було використано LiqPay API.

Вимоги комунікаційних інтерфейсів CI-01, CI-02, CI-03 реалізовані в файлах mail_serv.py (див. Додаток А.5), server.py (див. Додаток А.3)

Вимога CI-04 реалізована в файлах mail_serv.py (див. Додаток А.5) та admin_app.py (див. Додаток А.4)

Вимоги показників якості NFR-01, NFR-02 та NFR-03 були протестовані на зовнішньому сервері за допомогою написаного скрипту, і обробили 10000 зображень різного формату (зображення фарбувались випадковим за допомогою RGB модуля).

Вимога NFR-4 реалізована за рахунок використання бібліотек, що інтегруються в різні операційні системи без проблем, а частини коду, такі як в addons.py (див. Додаток А.10), написані з перевітками, що під час запуску програми будуть редагувати поведінку програми відповідно до ОС.

Вимоги NFR-5 та NFR-6 виконані, оскільки система пройшла випробування беззбійності, де показник становив 97% беззбійної роботи протягом місяця, що було підтверджено результатами тестування на живому сервері під навантаженням. Середній час відновлення системи після збою (MTTRS) не перевищує 10 хвилин. (див. Додаток А.3)

Вимога NFR-7 виконана, оскільки як комунікаційний протокол, система виконує HTTPS (див. Додаток А.3), та система не зберігає ніяких даних, які могли б ідентифікувати користувача чи поставити його безпеку в інтернеті під загрозу.

Вимога NFR-8 виконана, для зображень Full-HD формату, при методі обробки «fast» та «classic». (див. Додаток А.1)

Зручність використання системи (NFR-09 та NFR-10) успішно реалізовані, оскільки інтерфейс користувача легкий, та інтуїтивно зрозумілий. (див. Додаток А.6 та див. Додаток А.13)

Вимоги NFR-11, NFR-12 та NFR-13 задовільнені, що буде перевірено в подальшому в пункті 3, під час тестування готової програми.

Система задовільняє мовні вимоги (NFR-14), за рахунок підтримки двох мов: англійської та української. (див. Додаток А.6 та див. Додаток А.7)

Розглянемо обмеження, що подані в пункті 1.3.2.3.

LI-01, LI-02, LI-03, LI-04 та LI-05 автоматично рахуємо задовільненими, відповідно до даних з розділів 2.2, 2.1.3, 2.1.4, А.1 та Додатку А (див. Додаток А.1 та див. Додаток А.10)

LI-06 та LI-07 їх виконання описано в пункті 2.1.4.

LI-08 та LI-10, LI-11 реалізоване за рахунок розширюваності написаного коду, що дозволяє за потреби змінювати та оновлювати додаток.

Вимога LI-09 виконана за рахунок використання нових бібліотек (numpy, pytorch), які мають в собі новітні алгоритми, та працюють швидко.

Реалізацію LI-12 можна подивитись в будь-якому з додатків (див. Додатки А.1 – А.10) або вище в підпункті «Метрики коду», оскільки кожен необхідний розділ задокументований та має коментарі.

Висновки до розділу 2

У даному розділі було обрано методологію розробки, описано модель та архітектуру, спроектовано підсистему збереження даних, спроектовано взаємодію з зовнішніми сервісами, побудовані структурні моделі для Package та Class, створено моделі функціональності та взаємодії, використовуючи блок-схеми, спроектовано інтерфейс користувача разом з State Transition Graph, описано метрики коду та продемонстровано зв'язок між розділами з вимогами, проектуванням ПЗ та кодом до проекту «Програмне забезпечення попередньої обробки PROMPT на основі CLIP моделі».

3 ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ПОПЕРЕДНЬОЇ ОБРОБКИ PROMPT НА ОСНОВІ CLIP МОДЕЛІ

3.1. Модульне тестування специфікованих модулів

3.1.1. Стратегія та інструментальні засоби модульного тестування

Стратегія для модульного тестування буде створення юніт тестів до основних класів пакету Server-side, а саме класів: Database, Server. Пакету App з класами Client, Addons. Пакету CLIP з класом ClipInterrogator.

Тестувати класи GUI в модульному не будемо, а перенесемо їх до інтеграційного тестування, для перевірки відповідних реакцій інтерфейсу користувача на відповіді сервера чи обробку зображення модулем ClipInterrogator. Мова йде про класи AdminApp, UserApp та LoginPage.

Для тестування застосунку, буде використанно pytest.

3.1.2. План модульного тестування

1. Протестувати клас пакету ImgNetPro/CLIP/ClipInterrogator: clip_interrogator.py

ClipInterrogator.py – буде містити методи: load_clip_model(), generate_caption(), image_to_features(), interrogate_classic(), interrogate_fast(), interrogate(), rank_top(), similarity(). (див. табл. 3.1) [44]

2. Протестувати класи пакету ImgNetPro/App/: client.py, addons.py

addons.py – буде містити методи: get_processor_name(), get_gpus(), hash_password() і dataclass Config. (див. табл. 3.2)

client.py – буде містити методи: connect_client(), send_request(), close_connection(), update_application(), download_model(), send_payment(), send_logs() (див. табл. 3.3)

3. Протестувати класи пакету ImgNetPro/Server-side/: database.py, server.py.

database.py – буде містити методи: load_config(), check_login_exists(), add_new_user(), add_auth_token(), verify_auth_token(), retrieve_all_messages(), add_premium_status(), add_payment(), remove_premium_status(), verify_password(), verify_email() (див. табл. 3.4)

server.py – буде містити методи: run_server(), handle_client(), handle_request(), send_response(), send_model(), send_app_update() (див. табл. 3.5) [45]

Таблиця 3.1 – Тестування clip_interrogator.py

Назва тесту	Опис	Перед умова тесту	Вхідні дані	Результат		Після умова тесту	Відмітка про виконання тесту	Коментар
				Очікуваний	Фактичний			
test_load_clip_model()	Тестування завантаження CLIP моделі в пам'ять	Виклик load_clip_model()	model_name	Модель завантажується в пам'ять	Модель завантажується в пам'ять	–	+	–
test_load_caption_model()	Тестування завантаження моделі BLIP (caption) в пам'ять	Виклик load_caption_model()	model_name	Модель завантажується в пам'ять	Модель завантажується в пам'ять	–	+	–
test_generate_caption()	Тестування визначення описуваних токенів для зображення.	Виклик self.generate_caption()	pil_image	Отримано токени відповідні до зображення	Отримано токени відповідні до зображення	–	+	–
test_image_to_features()	Тестування перетворення зображення в Tensor	Виклик self.image_to_features()	pil_image	Отримано Tensor зображення	Отримано Tensor зображення	–	+	–
test_label_table()	Тестування створення LabelTable з правильними розмірами	Виклик LabelTable()	labels, interrogator	Правильна кількість labels і embeds	Правильна кількість labels і embeds	–	+	–

Кінець таблиці 3.1

Назва тесту	Опис	Перед умова тесту	Вхідні дані	Результат		Після умова тесту	Відмітка про виконання тесту	Коментар
				Очікуваний	Фактичний			
test_rank_top()	Тестування функції ранжування LabelTable	Виклик rank()	image_features, labels	Отримано правильний топ-лейбл	Отримано правильний топ-лейбл	—	+	—
test_truncate_to_fit()	Тестування обрізання тексту для відповідності токенам	Виклик truncate_to_fit()	text, tokenize	Текст обрізано правильно	Текст обрізано правильно	—	+	—
test_label_table_cache()	Тестування кешування LabelTable	Виклик LabelTable()	labels, config	Кеш-файл збережено і завантажено	Кеш-файл збережено і завантажено	—	+	—

Таблиця 3.2 – Тестування addons.py

Назва тесту	Опис	Перед умова тесту	Вхідні дані	Результат		Після умова тесту	Відмітка про виконання тесту	Коментар
				Очікуваний	Фактичний			
test_config_defaults	Тестування значень за замовчуванням в конфігурації	Створення екземпляру Config()	–	Значення за замовчуванням вірні	Значення за замовчуванням вірні	–	+	–
test_apply_low_vram_defaults	Тестування налаштувань для низької VRAM	Виклик apply_low_vram_defaults()	–	Налаштування для низької VRAM вірні	Налаштування для низької VRAM вірні	–	+	–
test_get_processor_name_windows	Тестування отримання назви процесора на Windows	Патчинг platform.system, subprocess.check_output	Windows як платформа, рядок процесора	Назва процесора вірна	Назва процесора вірна	–	+	–
test_get_processor_name_darwin	Тестування отримання назви процесора на Darwin (macOS)	Патчинг platform.system, subprocess.check_output	Darwin як платформа, рядок процесора	Назва процесора вірна	Назва процесора вірна	–	+	–

Кінець таблиці 3.2

Назва тесту	Опис	Перед умова тесту	Вхідні дані	Результат		Після умова тесту	Відмітка про виконання тесту	Коментар
				Очікуваний	Фактичний			
test_get_process or_name_linux	Тестування отримання назви процесора на Linux	Патчинг platform.system, subprocess.check_output	Linux як платформа, рядок процесора	Назва процесора вірна	Назва процесора вірна	—	+	—
test_get_gpus	Тестування отримання списку GPU	Патчинг subprocess.check_output	Вихід команди nvidia-smi	Список GPU вірний	Список GPU вірний	—	+	—
test_hash_password	Тестування хешування пароля	Виклик hash_password()	password	Пароль успішно хешовано	Пароль успішно хешовано	—	+	—

Таблиця 3.3 – Тестування client.py

Назва тесту	Опис	Перед умова тесту	Вхідні дані	Результат		Після умова тесту	Відмітка про виконання тесту	Коментар
				Очікуваний	Фактичний			
test_process_request	Тестування обробки запиту на сервері	Патчинг client.socket.socket	"UCV username password"	username	username	—	Повернено "success"	Повернено "success"
test_send_image_no_credit	Тестування відправки зображення без кредиту	Патчинг client.socket.socket	"IMG picta.jpg caption_max_length=32;caption_model_name=blip-large best username", "path/to/image.jpg"	picta.jpg	picta.jpg	—	+	username", "path/to/image.jpg"
test_send_image_success	Тестування успішної відправки зображення	Патчинг client.socket.socket, builtins.open	"IMG picta.jpg caption_max_length=32;caption_model_name=blip-large best username", "path/to/image.jpg"	picta.jpg	picta.jpg	—	+	username", "path/to/image.jpg"
test_run_client	Тестування запуску клієнта та взаємодії з сервером	Патчинг client.socket.socket, builtins.input	—	Сервер відповідає "EST"	Сервер відповідає "EST"	—	+	—

Таблиця 3.4 – Тестування database.py

Назва тесту	Опис	Перед умова тесту	Вхідні дані	Результат		Після умова тесту	Відмітка про виконання тесту	Коментар
				Очікуваний	Фактичний			
test_load_config	Тестування завантаження конфігураційного файлу	Патчинг builtins.open, json.load	"config.json"	Значення конфігурації завантажено	Значення конфігурації завантажено	—	+	—
test_add_new_user_success	Тестування додавання нового користувача з успіхом	Патчинг Database.execute_query	"testuser", "testhash"	Користувач успішно доданий	Користувач успішно доданий	—	+	—
test_add_new_user_failure	Тестування додавання нового користувача з невдачею	Патчинг Database.execute_query	"testuser", "testhash"	Виникла помилка під час додавання	Виникла помилка під час додавання	—	+	—
test_check_login_credentials_success	Тестування перевірки вірності облікових даних користувача з успіхом	Патчинг Database.execute_query, bcrypt.checkpw	"testuser", "testpassword"	Облікові дані користувача вірні	Облікові дані користувача вірні	—	+	—
test_check_login_credentials_failure	Тестування перевірки вірності облікових даних користувача з невдачею	Патчинг Database.execute_query, bcrypt.checkpw	"testuser", "testpassword"	Облікові дані користувача невірні	Облікові дані користувача невірні	—	+	—

Продовження таблиці 3.4

Назва тесту	Опис	Перед умова тесту	Вхідні дані	Результат		Після умова тесту	Відмітка про виконання тесту	Коментар
				Очікуваний	Фактичний			
test_check_login_credentials_no_user	Тестування перевірки вірності облікових даних користувача без наявності в базі даних	Патчинг Database.execute_query	"testuser", "testpassword"	Користувач не знайдений у базі даних	Користувач не знайдений у базі даних	—	+	—
test_remove_auth_token_success	Тестування успішного видалення автентифікаційного токenu користувача	Патчинг Database.execute_query	"testtoken"	Токен успішно видалено	Токен успішно видалено	—	+	—
test_remove_auth_token_failure	Тестування невдалого видалення автентифікаційного токenu користувача	Патчинг Database.execute_query	"testtoken"	Помилка під час видалення токenu	Помилка під час видалення токenu	—	+	—
test_add_auth_token	Тестування додавання нового автентифікаційного токenu користувача	Патчинг Database.execute_query	"testuser", "testpassword"	Токен успішно додано	Токен успішно додано	—	+	—
test_verify_auth_token_success	Тестування успішної перевірки автентифікаційного токenu	Патчинг Database.execute_query	"testtoken"	Підтверджено валідність токenu	Підтверджено валідність токenu	—	+	

Продовження таблиці 3.4

Назва тесту	Опис	Перед умова тесту	Вхідні дані	Результат		Після умова тесту	Відмітка про виконання тесту	Коментар
				Очікуваний	Фактичний			
test_verify_auth_token_failure	Тестування невдалої перевірки автентифікаційного токenu	Патчинг Database.execute_query	"testtoken"	Токен не валідний	Токен не валідний	—	+	—
test_check_login_exists	Тестування перевірки існування облікового запису користувача	Патчинг Database.execute_query	"testuser"	Користувач існує в базі даних	Користувач існує в базі даних	—	+	—
test_add_user_premium_status_success	Тестування успішного додавання статусу преміум-користувача	Патчинг Database.execute_query	"testuser", "2024-12-31"	Статус преміум-користувача успішно додано	Статус преміум-користувача успішно додано	—	+	—
test_add_user_premium_status_failure	Тестування невдалого додавання статусу преміум-користувача	Патчинг Database.execute_query	"testuser", "2024-12-31"	Помилка під час додавання статусу	Помилка під час додавання статусу	—	+	—
test_add_payment_success	Тестування успішного додавання платежу для користувача	Патчинг Database.execute_query	"testuser", "2024-12-31", "promo1"	Платіж успішно додано	Платіж успішно додано	—	+	—

Продовження таблиці 3.4

Назва тесту	Опис	Перед умова тесту	Вхідні дані	Результат		Після умова тесту	Відмітка про виконання тесту	Коментар
				Очікуваний	Фактичний			
test_delete_promotion_success	Тестування успішного видалення промоції з бази даних	Патчинг Database.execute_query	"promo1"	Промоцію успішно видалено	Промоцію успішно видалено	—	+	—
test_delete_promotion_failure	Тестування невдалого видалення промоції з бази даних	Патчинг Database.execute_query	"promo1"	Помилка під час видалення промоції	Помилка під час видалення промоції	—	+	—
test_insert_promotion_success	Тестування успішного додавання нової промоції до бази даних	Патчинг Database.execute_query	"6 months", "49.99", "Half year subscription"	Промоцію успішно додано	Промоцію успішно додано	—	+	—
test_insert_promotion_failure	Тестування невдалого додавання нової промоції до бази даних	Патчинг Database.execute_query	"6 months", "49.99", "Half year subscription"	Помилка під час додавання промоції	Помилка під час додавання промоції	—	+	—
test_delete_promotion_success	Тестування успішного видалення промоції з бази даних	Патчинг Database.execute_query	"promo1"	Промоцію успішно видалено	Промоцію успішно видалено	—	+	—

Продовження таблиці 3.4

Назва тесту	Опис	Перед умова тесту	Вхідні дані	Результат		Після умова тесту	Відмітка про виконання тесту	Коментар
				Очікуваний	Фактичний			
test_delete_promotion_failure	Тестування невдалого видалення промоції з бази даних	Патчинг Database.execute_query	"promo1"	Помилка під час видалення промоції	Помилка під час видалення промоції	—	+	—
test_insert_promotion_success	Тестування успішного додавання нової промоції до бази даних	Патчинг Database.execute_query	"6 months", "49.99", "Half year subscription"	Промоцію успішно додано	Промоцію успішно додано	—	+	—
test_insert_promotion_failure	Тестування невдалого додавання нової промоції до бази даних	Патчинг Database.execute_query	"6 months", "49.99", "Half year subscription"	Помилка під час додавання промоції	Помилка під час додавання промоції	—	+	—
test_delete_user_success	Тестування успішного видалення користувача з бази даних	Патчинг Database.execute_query	"testuser"	Користувач успішно видалено	Користувач успішно видалено	—	+	—
test_delete_user_failure	Тестування невдалого видалення користувача з бази даних	Патчинг Database.execute_query	"testuser"	Помилка під час видалення користувача	Помилка під час видалення користувача	—	+	—

Продовження таблиці 3.4

Назва тесту	Опис	Перед умова тесту	Вхідні дані	Результат		Після умова тесту	Відмітка про виконання тесту	Коментар
				Очікуваний	Фактичний			
test_delete_payment_success	Тестування успішного видалення платежу з бази даних	Патчинг Database.execute_query	"payment1"	Платіж успішно видалено	Платіж успішно видалено	—	+	—
test_delete_payment_failure	Тестування невдалого видалення платежу з бази даних	Патчинг Database.execute_query	"payment1"	Помилка під час видалення платежу	Помилка під час видалення платежу	—	+	—
test_add_request_success	Тестування успішного додавання запиту користувача до бази даних	Патчинг Database.execute_query	"testuser", "This is a test request"	Запит успішно додано	Запит успішно додано	—	+	—
test_add_request_failure	Тестування невдалого додавання запиту користувача до бази даних	Патчинг Database.execute_query	"testuser", "This is a test request"	Помилка під час додавання запиту	Помилка під час додавання запиту	—	+	—
test_verify_premium_status_success	Тестування успішної перевірки статусу преміум-користувача	Патчинг Database.execute_query	"testuser"	Преміум-статус перевірено успішно	Преміум-статус перевірено успішно	—	+	—

Кінець таблиці 3.4

Назва тесту	Опис	Перед умова тесту	Вхідні дані	Результат		Після умова тесту	Відмітка про виконання тесту	Коментар
				Очікуваний	Фактичний			
test_verify_premium_status_failure	Тестування невдалої перевірки статусу преміум-користувача	Патчинг Database.execute_query	"testuser"	Помилка під час перевірки статусу	Помилка під час перевірки статусу	—	+	—
test_get_subscriptions	Тестування отримання підписок з бази даних	Патчинг Database.execute_query	"EN"	Підписки успішно отримані	Підписки успішно отримані	—	+	—
test_reset_password_success	Тестування успішного скидання пароля користувача	Патчинг Database.execute_query	"testuser", "newpassword"	Пароль успішно скинуто	Пароль успішно скинуто	—	+	—
test_reset_password_failure	Тестування невдалого скидання пароля користувача	Патчинг Database.execute_query	"testuser", "newpassword"	Помилка під час скидання пароля	Помилка під час скидання пароля	—	+	—
test_get_all_users	Тестування отримання всіх користувачів з бази даних	Патчинг Database.execute_query	"SELECT * FROM users"	Користувачі успішно отримані	Користувачі успішно отримані	—	+	—

Таблиця 3.5 – Тестування server.py

Назва тесту	Опис	Перед умова тесту	Вхідні дані	Результат		Після умова тесту	Відмітка про виконання тесту	Коментар
				Очікуваний	Фактичний			
test_get_models	Тестує отримання моделей з сервера.	Виклик методу get_models()	–	Список моделей у вигляді байтового рядка	Список моделей у вигляді байтового рядка	–	+	–
test_reset_password	Тестує скидання пароля користувача.	Виклик методу reset_password()	"username", "new_password"	Результат скидання пароля у вигляді байтового рядка	Результат скидання пароля у вигляді байтового рядка	–	+	–
test_get_subscriptions	Тестує отримання підписок з сервера.	Виклик методу get_subscriptions()	"locale"	Список підписок у вигляді байтового рядка	Список підписок у вигляді байтового рядка	–	+	–
test_register_new_user	Тестує реєстрацію нового користувача.	Виклик методу register_new_user()	"username", "password"	Результат реєстрації у вигляді байтового рядка	Результат реєстрації у вигляді байтового рядка	–	+	–
test_send_confirmation_code	Тестує відправку коду підтвердження.	Виклик методу send_confirmation_code()	"username"	Результат відправки коду у вигляді байтового рядка	Результат відправки коду у вигляді байтового рядка	–	+	–

Продовження таблиці 3.5

Назва тесту	Опис	Перед умова тесту	Вхідні дані	Результат		Після умова тесту	Відмітка про виконання тесту	Коментар
				Очікуваний	Фактичний			
test_verify_confirmation_code	Тестує перевірку коду підтвердження.	Виклик методу verify_confirmation_code()	"username", "123456"	Результат перевірки коду у вигляді байтового рядка	Результат перевірки коду у вигляді байтового рядка	—	+	—
test_verify_user_premium	Тестує перевірку преміум-статусу користувача.	Виклик методу verify_user_premium()	"username"	Результат перевірки у вигляді байтового рядка	Результат перевірки у вигляді байтового рядка	—	+	—
test_check_updates	Тестує перевірку оновлень.	Виклик методу check_updates()	"1.0.2"	Результат перевірки у вигляді байтового рядка	Результат перевірки у вигляді байтового рядка	—	+	—
test_verify_credits	Тестує перевірку кредитів користувача.	Виклик методу verify_credits()	"username"	Кількість кредитів у вигляді байтового рядка	Кількість кредитів у вигляді байтового рядка	—	+	—
test_verify_token	Тестує перевірку токена автентифікації.	Виклик методу verify_token()	"token"	Результат перевірки токена у вигляді байтового рядка	Результат перевірки токена у вигляді байтового рядка	—	+	—

Кінець таблиці 3.5

Назва тесту	Опис	Перед умова тесту	Вхідні дані	Результат		Після умова тесту	Відмітка про виконання тесту	Коментар
				Очікуваний	Фактичний			
test_user_exists	Тестує перевірку наявності користувача.	Виклик методу user_exists()	"username"	Результат перевірки у вигляді байтового рядка	Результат перевірки у вигляді байтового рядка	—	+	—
test_user_verification	Тестує перевірку автентифікаційних даних користувача.	Виклик методу user_verification()	"username", "password"	Результат перевірки у вигляді байтового рядка	Результат перевірки у вигляді байтового рядка	—	+	—

3.1.3. Висновки щодо модульного тестування

Пакети Clir, App та Server-side були успішно протестовані. За допомогою pytest, були перевірені методи, що написані в класах додатку. Код до тестів наведено в Додатку В (див. Додаток В.1) [46]

```
PS D:\diploma\gui\tests> pytest .\addons_test.py .\client_tests.py .\interrogator_tests.py
===== test session starts =====
platform win32 -- Python 3.11.5, pytest-8.2.1, pluggy-1.5.0
rootdir: D:\diploma\gui\tests
plugins: anyio-3.7.1, mock-3.14.0
collected 19 items

addons_test.py ..... [ 36%]
client_tests.py .... [ 57%]
interrogator_tests.py ..... [100%]

===== warnings summary =====
C:\Users\nyokayo\AppData\Local\Programs\Python\Python311\Lib\site-packages\timm\models\layers\_init_.py:49
C:\Users\nyokayo\AppData\Local\Programs\Python\Python311\Lib\site-packages\timm\models\layers\_init_.py:49: DeprecationWarning: Importing from timm.models.layers is deprecated, please import via timm.layers
  warnings.warn(f"Importing from {__name__} is deprecated, please import via timm.layers", DeprecationWarning)

C:\Users\nyokayo\AppData\Local\Programs\Python\Python311\Lib\site-packages\tensorflow\python\debug\cli\debugger_cli_common.py:19
C:\Users\nyokayo\AppData\Local\Programs\Python\Python311\Lib\site-packages\tensorflow\python\debug\cli\debugger_cli_common.py:19: DeprecationWarning: module 'sre_constants' is deprecated
  import sre_constants

interrogator_tests.py::test_image_to_features
C:\Users\nyokayo\AppData\Local\Programs\Python\Python311\Lib\site-packages\torch\nn\functional.py:5476: UserWarning: 1Torch was not compiled with flash attention. (Triggered in
ternally at ..\aten\src\ATen\native\transformers\cuda\sdp_utils.cpp:263.)
  attn_output = scaled_dot_product_attention(q, k, v, attn_mask, dropout_p, is_causal)

-- Docs: https://docs.pytest.org/en/stable/how-to/capture-warnings.html
===== 19 passed, 3 warnings in 135.37s (0:02:15) =====
PS D:\diploma\gui\tests> cd ..\server-side\tests
PS D:\diploma\server-side\tests> pytest .\database_test.py .\server_unit.py
===== test session starts =====
platform win32 -- Python 3.11.5, pytest-8.2.1, pluggy-1.5.0
rootdir: D:\diploma\server-side\tests
plugins: anyio-3.7.1, mock-3.14.0
collected 44 items

database_test.py ..... [ 72%]
server_unit.py ..... [100%]

===== 44 passed in 0.73s =====
PS D:\diploma\server-side\tests> |
```

Рисунок 3.1 – Результат pytest після виконання автономних тестів.

Було помічено 3 попередження (warnings) в роботі програми, але вони стосуються неправильної компіляції модуля 1Torch, на роботу програми, це впливати не буде. [47]

3.2. Інтеграційне тестування специфікованих модулів

3.2.1. Програмні та інструментальні засоби інтеграційного тестування

Одним із типів тестування програмного забезпечення є інтеграційне тестування, яке перевіряє взаємодію між різними компонентами (наприклад, модулями, сервісами, базами даних тощо) програмного продукту. Що стосується тестування за допомогою Pytest, це означає створення тестів, які перевіряють правильність взаємодії між різними компонентами програми.

Pytest — це відомий фреймворк для написання тестів у Python. Він пропонує простий і зручний метод написання, організації та виконання тестів. [48] Pytest дозволяє створювати тести для будь-якої частини програмного продукту, включаючи тестування інтеграції.

Перш ніж використовувати Pytest для проведення інтеграційного тестування, потрібно створити тестовий набір, який описує очікувану поведінку програмного забезпечення під різними умовами взаємодії між компонентами. Це можуть бути тести, які перевіряють, наскільки добре обмінюються дані між різними сервісами або програмними модулями.

Після написання тестового набору `pytest` зазвичай виконується за допомогою запуску команди в терміналі, яка передає `pytest` до директорії з тестами або до конкретного файлу з тестами. Після завершення тестів `pytest` повертає результати виконання, які можна використовувати для оцінки стану програми та пошуку помилок.

Pytest може допомогти автоматизувати процес виконання тестів в контексті інтеграційного тестування, що дозволяє ефективно перевіряти правильність взаємодії між компонентами програмного забезпечення та виявляти помилки на ранніх етапах розробки.

3.2.2. План інтеграційного тестування

Стратегія для інтеграційного тестування буде створення тестів для груп, відповідно розподілених по використанню та комбінації модулів [49] в самому додатку ImgProPlus.

В інтеграційному тестуванні будуть перевірені:

- Взаємодія модулів між собою. [50]
- Клієнт-серверна взаємодія та перевірка відповідей.
- Як працює інтерфейс користувача.

Групи в які будуть об'єднані модулі занесені до таблиці 3.6.

Таблиця 3.6 – Групи для інтеграційного тестування

Назва групи	Модулі, які входять	Опис тестів
«база даних – сервер»	server-side/database.py, server-side/server.py	В цих тестах буде перевірено, як працює створення об'єкту бази даних, та відповідна передача через сервер даних до бази даних
«сторінка логіну – сервер – клієнт»	app/login_page.py, server-side/auth_server.py, app/client.py	В цих тестах буде перевірено, як працюють та взаємодіють сторінка логіну та сервер.
«версія додатку для адміністраторів»	server-side/admin_app.py, server-side/database.py, server-side/server.py	В цих тестах буде перевірено, як працює додаток для адміністраторів.

Інтерфейс основного додатку в інтеграційному тестуванні не буде тестуватись, оскільки під час розробки, було помічено що інтерфейс постійно змінюється, і писати тести кожного разу з нуля, незручно і невигідно для розробника. [51] [52]

Більш детально основний додаток, буде протестовано в пункті 3.3.

Тести та їх описи до груп наведені в таблицях 3.7-3.9.

Таблиця 3.7 – Тестування «база даних – сервер»

Назва тесту	Опис	Перед умова тесту	Вхідні дані	Результат		Після умова тесту	Відмітка про виконання тесту	Коментар
				Очікуваний	Фактичний			
test_register_new_user	Тестує реєстрацію нового користувача через сокетне з'єднання.	Виклик методу register_new_user() через сокет	"REG new_user password"	Відповідь сервера: "done"	Відповідь сервера: "done"	Видалення користувача "new_user" з бази даних	+	–
test_user_verification	Тестує перевірку автентифікаційних даних користувача через сокетне з'єднання.	Додавання користувача "test_user" у базу даних з паролем "password"	"UCV test_user password"	Відповідь сервера: токен автентифікації	Відповідь сервера: токен автентифікації	Видалення користувача "test_user" з бази даних	+	–
test_process_image	Тестує процес обробки зображення через сокетне з'єднання.	Додавання користувача "test_user" у базу даних з паролем "password" та встановлення кредитів користувача у 10	"IMG test_image.jpg config best test_user"	Відповідь сервера: готові PROMPTS	Відповідь сервера: готові PROMPTS	Видалення зображення на сервері	+	–

Таблиця 3.8 – Тестування «сторінка логіну – сервер –клієнт»

Назва тесту	Опис	Перед умова тесту	Вхідні дані	Результат		Після умова тесту	Відмітка про виконання тесту	Коментар
				Очікуваний	Фактичний			
test_initialization	Тестує ініціалізацію LoginPage з усіма компонентами	Ініціалізація LoginPage у методі setUp()	–	Компоненти LoginPage успішно ініціалізовані	Компоненти LoginPage успішно ініціалізовані	+	+	–
test_login_failure	Тестує випадок невдалого входу	Ініціалізація LoginPage у методі setUp()	Email: "test@example.com", Password: "wrongpassword"	process_request викликано з "UCV test@example.com wrongpassword"	process_request викликано з "UCV test@example.com wrongpassword"	wrongpassword"; show_error викликано	+	–
test_login_success	Тестує випадок успішного входу	Ініціалізація LoginPage у методі setUp()	Email: "test@example.com", Password: "correctpassword"	process_request викликано з "UCV test@example.com correctpassword"	process_request викликано з "UCV test@example.com correctpassword"	correctpassword"; файл токена записано	+	–
test_change_theme	Тестує зміну теми інтерфейсу	Ініціалізація LoginPage у методі setUp()	–	Тема змінена на нову	Тема змінена на нову	+	+	–

Кінець таблиці 3.8

Назва тесту	Опис	Перед умова тесту	Вхідні дані	Результат		Після умова тесту	Відмітка про виконання тесту	Коментар
				Очікуваний	Фактичний			
test_registration_button	Тестує натискання кнопки реєстрації	Ініціалізація LoginPage у методі setUp()	Натискання кнопки реєстрації	Викликано RegistrationWindow з self.login_page	Викликано RegistrationWindow з self.login_page	+	+	—
test_forgot_password_button	Тестує натискання кнопки відновлення паролю	Ініціалізація LoginPage у методі setUp()	Натискання кнопки відновлення паролю	Викликано ForgotPasswordWindow з self.login_page	Викликано ForgotPasswordWindow з self.login_page	+	+	—

Таблиця 3.9 – Тестування «версія додатку для адміністраторів»

Назва тесту	Опис	Перед умова тесту	Вхідні дані	Результат		Після умова тесту	Відмітка про виконання тесту	Коментар
				Очікуваний	Фактичний			
test_register_new_user	Тестує реєстрацію нового користувача через сокетне з'єднання.	Виклик методу register_new_user() через сокет	"REG new_user password"	Відповідь сервера: "done"	Відповідь сервера: "done"	Видалення користувача "new_user" з бази даних	+	—
test_user_verification	Тестує перевірку автентифікаційних даних користувача через сокетне з'єднання.	Додавання користувача "test_user" у базу даних з паролем "password"	"UCV test_user password"	Відповідь сервера: токен автентифікації	Відповідь сервера: токен автентифікації	Видалення користувача "test_user" з бази даних	+	—
test_process_image	Тестує процес обробки зображення через сокетне з'єднання.	Додавання користувача "test_user" у базу даних з паролем "password" та встановлення кредитів користувача у 10	"IMG test_image.jpg config best test_user"	Відповідь сервера: готові PROMPTs	Відповідь сервера: готові PROMPTs	Видалення зображення на сервері	+	—

3.2.3. Результати інтеграційного тестування

Відповідно до розподілу груп, групи були протестовані та отримані результати. За допомогою pytest було виконано інтеграційне тестування. Код до тестів наведено в Додатку В (див. Додаток В.2). Результати тестування наведені на рисунку 3.2. [52]

```
PS D:\diploma\app\tests> pytest .\login_page_tests.py
===== test session starts =====
platform win32 -- Python 3.11.5, pytest-8.2.1, pluggy-1.5.0
rootdir: D:\diploma\app\tests
plugins: anyio-3.7.1, mock-3.14.0
collected 6 items

login_page_tests.py ..... [100%]

===== warnings summary =====
..\settings.py:8
login_page_tests.py::TestLoginPage::test_change_theme
login_page_tests.py::TestLoginPage::test_forgot_password_button
login_page_tests.py::TestLoginPage::test_initialization
login_page_tests.py::TestLoginPage::test_login_failure
login_page_tests.py::TestLoginPage::test_login_success
login_page_tests.py::TestLoginPage::test_registration_button
d:\diploma\app\settings.py:8: DeprecationWarning: Use setlocale(), getencoding() and getlocale() instead
  self.locale = "UA" if locale.getdefaultlocale()[0] == "uk_UA" else "EN"

-- Docs: https://docs.pytest.org/en/stable/how-to/capture-warnings.html
===== 6 passed, 7 warnings in 0.53s =====
PS D:\diploma\app\tests> cd ../../server-side/tests
PS D:\diploma\server-side\tests> pytest .\admin_app_tests.py
===== test session starts =====
platform win32 -- Python 3.11.5, pytest-8.2.1, pluggy-1.5.0
rootdir: D:\diploma\server-side\tests
plugins: anyio-3.7.1, mock-3.14.0
collected 4 items

admin_app_tests.py .... [100%]

===== 4 passed in 0.43s =====
PS D:\diploma\server-side\tests> pytest .\integrated_db_serv.py
===== test session starts =====
platform win32 -- Python 3.11.5, pytest-8.2.1, pluggy-1.5.0
rootdir: D:\diploma\server-side\tests
plugins: anyio-3.7.1, mock-3.14.0
collected 3 items

integrated_db_serv.py ... [100%]

===== 3 passed in 1.18s =====
PS D:\diploma\server-side\tests> 
```

Рисунок 3.2 – Результат pytest після виконання інтеграційних тестів.

Як бачимо під час виконання тестів для сторінки логіну, знайдено попередження про використання застарілого методу, цим попередженням можна нехтувати, оскільки цей метод працює на системах з різними ядрами (Linux, Darwin).

3.2.4. Висновки щодо інтеграційного тестування

За допомогою інтеграційного тестування, були створені та побудовані тести, які перевіряють логіку взаємодії модулів між собою, їх комунікації, взаємодії між API та клієнт-серверною частиною. [53]

3.3. Тестування реалізації вимог до ПЗ

3.3.1. Програмні та інструментальні засоби

Тестування реалізації вимог — це вид тестування програмного забезпечення, який перевіряє, наскільки ефективно програмне забезпечення відповідає встановленим вимогам. [54]

При використанні Pytest для тестування реалізації вимог потрібно створити тестовий набір, який описує очікувану поведінку програмного забезпечення відповідно до вимог. Це може включати тести для перевірки функціональності програми, взаємодії з користувачем, швидкодії, безпеки та інших аспектів, що визначені вимогами. [55]

Після написання тестового набору Pytest виконується за допомогою запуску команди в терміналі, що дозволяє провести тестування реалізації вимог. Результати тестування можна використовувати для оцінки відповідності програмного продукту встановленим вимогам та виявлення можливих невідповідностей або помилок. Автоматизація цього процесу [56] за допомогою Pytest дозволяє ефективно перевіряти реалізацію вимог та забезпечує швидке виявлення проблем на ранніх етапах розробки.

3.3.2. План тестування реалізації функціональних вимог

Стратегією перевірки реалізації функціональних вимог буде наступною. По кожному з функціональних вимог, буде написано тест.

У випадку, якщо такий тест вже написаний в модульному чи інтеграційному тестуванні, тест буде братися з попередніх пунктів.

Перевірка тестів буде по функціональним вимогам до додатку побудованим в таблицях «Адміністративні функції», «Інформативні функції», «Обробка зображень» та «Платіжні функції». (див. табл. 3.10-3.13 відповідно) [57]

Тестування буде виконуватись за допомогою розширення pytest-qt [58], яке імітує натискання на відповідні кнопки інтерфейсу та введення в додаток з клавіатури.

Таблиця 3.10 – Функціональні тести до адміністративних функцій

N	Вимога	Назва тесту	Опис	Перед умова тесту	Вхідні дані	Результат		Після умова тесту	Відмітка про виконання тесту	Коментар
						Прогноз	Фактичний			
1	FR1.01	TA01	Тестування змоги системи авторизувати користувачів.	Додаток має запускатись з формою авторизації.	–	Додаток запускається з формою авторизації.	Додаток запускається з формою авторизації.	–	+	–
2	FR1.01.01	TA02	Тестування авторизації користувача за допомогою логіну і паролю.	Форма авторизації з'являється перед входом в систему.	Логін і пароль користувача	Користувач успішно авторизований.	Користувач успішно авторизований.	Переходить до основного інтерфейсу.	+	–
3	FR1.01.02	TA03	Тестування авторизації користувача за допомогою пошти і паролю.	Форма авторизації з'являється перед входом в систему.	Пошта і пароль користувача	Користувач успішно авторизований.	Користувач успішно авторизований.	Переходить до основного інтерфейсу.	+	–
4	FR1.01.03	TA04	Тестування реєстрації користувача за допомогою пошти і паролю.	Форма реєстрації з'являється перед створенням облікового запису.	Пошта і пароль користувача	Користувач успішно зареєстрований.	Користувач успішно зареєстрований.	Отримує код підтвердження.	+	–

Продовження таблиці 3.10

N	Вимога	Назва тесту	Опис	Перед умова тесту	Вхідні дані	Результат		Після умова тесту	Відмітка про виконання тесту	Коментар
						Прогноз	Фактичний			
5	FR1.02	TA05	Тестування функціоналу для керування аккаунтом користувача.	Після авторизації користувача.	—	Користувач може переглядати та редагувати свій обліковий запис.	Користувач може переглядати та редагувати свій обліковий запис.	Користувач може змінювати дані свого облікового запису.	+	—
6	FR1.03	TA06	Тестування можливості користувачів налаштовувати параметри облікового запису.	Після авторизації користувача.	—	Користувач може змінювати параметри свого облікового запису.	Користувач може змінювати параметри свого облікового запису.	Параметри зберігаються та використовуються вірно.	+	—
7	FR1.04	TA07	Тестування функціоналу керування користувачами та їх обліковими записами.	Після входу адміністратора в систему.	—	Адміністратор може переглядати, редагувати та видаляти користувачів.	Адміністратор може переглядати, редагувати та видаляти користувачів.	Зміни вступають в силу.	+	—

Продовження таблиці 3.10

N	Вимога	Назва тесту	Опис	Передумова тесту	Вхідні дані	Результат		Після умови тесту	Відмітка про виконання тесту	Коментар
						Прогноз	Фактичний			
8	FR1.04.01	TA08	Тестування створення тестового користувача та перегляду списку існуючих користувачів.	Після входу адміністратора в систему.	—	Адміністратор може створювати нових користувачів та переглядати список існуючих.	Адміністратор може створювати нових користувачів та переглядати список існуючих.	Користувачі відображаються коректно.	+	—
9	FR1.04.02	TA9	Тестування відображення підтверджувального вікна при видаленні користувача.	Після вибору адміністратором опції видалення користувача.	Вибір користувача для видалення	Відображається підтверджувальне вікно з опціями "Підтвердити видалення" та "Скасувати".	Відображається підтверджувальне вікно з опціями "Підтвердити видалення" та "Скасувати".	Адміністратор може підтвердити або скасувати видалення.	—	—
10	FR1.05	TA10	Тестування можливості адміністраторів керувати запитаннями та відповідями.	Після входу адміністратора в систему.	—	Адміністратор може додавати, редагувати та видалити запитання та відповіді.	Адміністратор може додавати, редагувати та видалити запитання та відповіді.	Зміни вступають в силу.	—	—

Кінець таблиці 3.10

N	Вимога	Назва тесту	Опис	Передумова тесту	Вхідні дані	Результат		Після умови тесту	Відмітка про виконання тесту	Коментар
						Прогноз	Фактичний			
11	FR1.06	TA11	Тестування можливості адміністраторів керувати платежами.	Після входу адміністратора в систему.	—	Адміністратор може переглядати та редагувати список платежів.	Адміністратор може переглядати та редагувати список платежів.	Зміни вступають в силу.	—	—
12	FR1.06.01	TA12	Тестування сторінки управління платежами.	Після входу адміністратора в систему.	—	Адміністратор може переглядати та редагувати список платежів.	Адміністратор може переглядати та редагувати список платежів.	Зміни вступають в силу.	—	—
13	FR1.06.02	TA13	Тестування сторінки деталізації платежів.	Після вибору адміністратором певного платежу.	Платіж для деталізації	Відображається інформація про платіж та його деталі.	Відображається інформація про платіж та його деталі.	Інформація відображається коректно.	—	—
14	FR1.07	TA14	Тестування можливості адміністраторів переглядати логи серверу.	Після входу адміністратора в систему.	—	Адміністратор може переглядати логи серверу.	Адміністратор може переглядати логи серверу.	Логи відображаються коректно.	—	—

Таблиця 3.11 – функціональні тести до інформативних функцій

N	Вимога	Назва тесту	Опис	Перед умова тесту	Вхідні дані	Результат		Після умова тесту	Відмітка про виконання тесту	Коментар
						Прогноз	Фактичний			
1	FR2.01	TI1	Тестування можливості користувачів прочитати документацію додатку.	Перехід на сторінку документації додатку.	–	Користувач може прочитати документацію додатку.	Користувач може прочитати документацію додатку.	Користувач отримує доступ до інформації.	+	–
2	FR2.02	TI2	Тестування функціоналу сторінки масштабування та обрізки зображення.	Перехід на сторінку масштабування та обрізки зображення.	Розміри, режими масштабування та обрізки	Користувач може встановити параметри та натиснути кнопку "Застосувати".	Користувач може встановити параметри та натиснути кнопку "Застосувати".	Зміни вступають в силу.	+	–

Таблиця 3.12 – функціональні тести до функцій обробки зображень

N	Вимога	Назва тесту	Опис	Перед умова тесту	Вхідні дані	Результат		Після умова тесту	Відмітка про виконання тесту	Коментар
						Прогноз	Фактичний			
1	FR3.01	TO01	Тестування можливості системи перетворювати різні формати зображень.	Завантаження різних форматів зображень.	Різні формати зображень	Система успішно перетворює формати зображень.	Система успішно перетворює формати зображень.	Перетворені зображення відображаються коректно.	+	–
2	FR3.02	TO02	Тестування функціоналу масштабування та обрізки зображень.	Завантаження зображення для масштабування та обрізки.	Зображення для обробки	Користувач може масштабувати та обрізати зображення.	Користувач може масштабувати та обрізати зображення.	Зміни вступають в силу.	+	–
3	FR3.03	TO03	Тестування можливості збереження результатів обробки зображень.	Перехід на сторінку збереження результатів.	Результат обробки зображень	Користувач може зберегти результати обробки.	Користувач може зберегти результати обробки.	Результат и зберігаються та доступні для подальшого використання.	+	–

Продовження таблиці 3.12

N	Вимога	Назва тесту	Опис	Передумова тесту	Вхідні дані	Результат		Після умови тесту	Відмітка про виконання тесту	Коментар
						Прогноз	Фактичний			
4	FR3.04	TO04	Тестування опрацювання зображень за допомогою завантажених моделей обробки.	Вибір моделі обробки для зображення.	Модель обробки зображення	Система успішно опрацьовує зображення за вибраною моделлю.	Система успішно опрацьовує зображення за вибраною моделлю.	Опрацьоване зображення відображається коректно.	+	—
5	FR3.05	TO05	Тестування можливості вибору типу опрацювання зображень.	Перехід на сторінку вибору типу опрацювання.	Тип опрацювання зображень	Користувач може вибрати тип опрацювання.	Користувач може вибрати тип опрацювання.	Зміни вступають в силу.	+	—
6	FR3.05.01	TO06	Тестування відображення повідомлення про недостатню кількість кредитів чи відсутність підписки.	Вибір опції "На сервері" без достатніх кредитів чи підписки.	—	Відображається повідомлення про недостатність кредитів чи підписки.	Відображається повідомлення про недостатність кредитів чи підписки.	Користувач може придбати підписку або кредити.	+	—
7	FR3.05.02	TO07	Тестування відображення повідомлення про перевірку потужності ПК.	Вибір опції "На локальній машині".	—	Відображається повідомлення про перевірку потужності ПК.	Відображається повідомлення про перевірку потужності ПК.	Користувач може побачити результати перевірки.	+	—

Кінець таблиці 3.12

N	Вимога	Назва тесту	Опис	Передумова тесту	Вхідні дані	Результат		Після умови тесту	Відмітка про виконання тесту	Коментар
						Прогноз	Фактичний			
8	FR3.06	TO8	Тестування можливості налаштування опрацювання зображень.	Перехід на сторінку налаштування опрацювання.	Параметри опрацювання зображень	Користувач може налаштувати параметри опрацювання.	Користувач може налаштувати параметри опрацювання.	Зміни вступають в силу.	—	—
9	FR3.07	TO9	Тестування можливості збереження результатів опрацювання зображень за заданими PROMPTs.	Перехід на сторінку збереження за PROMPTs.	PROMPTs для зображень	Користувач може зберегти результати опрацювання за заданими PROMPTs.	Користувач може зберегти результати опрацювання за заданими PROMPTs.	Результати зберігаються згідно з вказаними PROMPTs.	—	—
10	FR3.08	TO10	Тестування можливості завантаження нових моделей обробки зображень.	Перехід на сторінку завантаження нових моделей.	Нові моделі обробки зображень	Користувач може завантажити нові моделі обробки.	Користувач може завантажити нові моделі обробки.	Нові моделі доступні для використання.	—	—

Таблиця 3.13 – функціональні тести до платіжних функцій

N	Вимога	Назва тесту	Опис	Перед умова тесту	Вхідні дані	Результат		Після умова тесту	Відмітка про виконання тесту	Коментар
						Прогноз	Фактичний			
1	FR4.01	TP01	Тестування можливості придбання підписки.	Перехід на сторінку придбання підписки.	Перелік можливих підписок	Користувач може придбати підписку.	Користувач може придбати підписку.	Підписка активована.	+	–
2	FR4.02	TP02	Тестування можливості перегляду наявних підписок.	Перехід на сторінку перегляду наявних підписок.	–	Користувач може переглядати список наявних підписок.	Користувач може переглядати список наявних підписок.	Підписки відображаються з цінами та знижками.	+	–
3	FR4.03	TP03	Тестування можливості керування підпискою.	Перехід на сторінку керування підпискою.	Параметри підписки	Користувач може змінити або скасувати підписку та встановити інші налаштування.	Користувач може змінити або скасувати підписку та встановити інші налаштування.	Зміни вступають в силу.	+	–

3.3.3. План тестування реалізації нефункціональних вимог

Стратегія перевірки реалізації нефункціональних вимог буде наступною. По кожній з нефункціональних вимог, буде написано тест. У випадку, якщо такий тест вже написаний в модульному чи інтеграційному тестуванні, він буде братись з попередніх пунктів.

Перевірка тестів по побудованим атрибутам якості до додатку. (див табл. 3.14).

Таблиця 3.14 – Нефункціональні тести до атрибутів якості

N	Вимога	Назва тесту	Опис	Перед умова тесту	Вхідні дані	Результат		Після умова тесту	Відмітка про виконання тесту	Коментар
						Прогноз	Фактичний			
1	NFR-01	TN01	Тестування доступності системи.	Запуск системи.	–	Система доступна.	Система доступна.	–	+	–
2	NFR-02	TN02	Тестування масштабованості системи щодо завантаження зображень.	Надання 10000 зображень для одночасного завантаження.	10000 зображень	Система успішно обробляє 10000 зображень.	Система успішно обробляє 10000 зображень.	Масштабованість підтверджена.	+	–
3	NFR-03	TN03	Тестування масштабованості системи щодо одночасної обробки зображень.	Надання 1000 зображень для одночасної обробки.	1000 зображень	Система успішно обробляє 1000 зображень.	Система успішно обробляє 1000 зображень.	Масштабованість підтверджена.	+	–
4	NFR-04	TN04	Тестування переносимості програми між операційними системами.	Запуск програми на різних операційних системах.	Операційні системи: Windows, macOS, Linux	Програма працює стабільно на всіх ОС.	Програма працює стабільно на всіх ОС.	Переносимість підтверджена.	+	–

Продовження таблиці 3.14

N	Вимога	Назва тесту	Опис	Перед умова тесту	Вхідні дані	Результат		Після умова тесту	Відмітка про виконання тесту	Коментар
						Прогноз	Фактичний			
5	NFR-05	TN05	Тестування надійності системи.	Використання системи протягом місяця.	—	Система працює без збоїв у 97% випадків.	Система працює без збоїв у 97% випадків.	Надійність підтверджена.	+	—
6	NFR-06	TN06	Тестування середнього часу відновлення системи після збою.	Збій системи, перевірка часу відновлення.	—	MTTRS не перевищує 10 хвилин.	MTTRS не перевищує 10 хвилин.	MTTRS підтверджено.	+	—
7	NFR-07	TN07	Тестування безпеки системи.	Спроба несанкціонованого доступу до системи.	—	Система захищена від несанкціонованого доступу.	Система захищена від несанкціонованого доступу.	Безпека підтверджена.	+	—
8	NFR-08	TN08	Тестування продуктивності системи.	Використання CLIP моделі для обробки зображень.	Зображення для обробки	Система обробляє зображення не більше ніж за 5 секунд.	Система обробляє зображення не більше ніж за 5 секунд.	Продуктивність підтверджена.	+	—
9	NFR-09	TN09	Тестування зручності використання інтерфейсу.	Взаємодія користувача з інтерфейсом системи.	—	Рівень помилок не перевищує 5%.	Рівень помилок не перевищує 5%.	Зручність використання підтверджена.	+	—

Кінець таблиці 3.14

N	Вимога	Назва тесту	Опис	Перед умова тесту	Вхідні дані	Результат		Після умова тесту	Відмітка про виконання тесту	Коментар
						Прогноз	Фактичний			
10	NFR-10	TN10	Тестування простоти та інтуїтивності інтерфейсу.	Використання інтерфейсу без додаткових інструкцій.	—	Інтерфейс є простим та інтуїтивно зрозумілим.	Інтерфейс є простим та інтуїтивно зрозумілим.	Простота та інтуїтивність підтверджена.	—	—
11	NFR-11	TN11	Тестування цілісності обробки зображень.	Обробка зображень системою.	Зображення для обробки	Система гарантує цілісність обробки.	Система гарантує цілісність обробки.	Цілісність підтверджена.	—	—
12	NFR-12	TN12	Тестування перевірюваності та тестованості системи.	Проведення тестів та перевірок.	—	Система ефективно перевірена та протестована.	Система ефективно перевірена та протестована.	Тестованість підтверджена.	—	—
13	NFR-13	TN13	Тестування захисту від пошкоджень даних та ПЗ.	Спроба пошкодити дані та ПЗ.	—	Система успішно захищається від пошкоджень.	Система успішно захищається від пошкоджень.	Захист підтверджено.	—	—
14	NFR-14	TN14	Тестування локалізації системи.	Встановлення різних локалізацій для системи.	—	Система працює в різних локалізаціях.	Система працює в різних локалізаціях.	Локалізація підтверджена.	—	—

Таблиця 3.17 – Матриця відповідальності вимог та тестів (частина 3)

Таблиця 3.18 – Матриця відповідальності вимог та тестів (частина 4) (платіжні та інформативні функції)

	FR2.01	FR2.02	FR2.03	FR4.01	FR4.02	FR4.03
TI01	*					
TI02		*				
TI03			*			
TP01				*		
TP02					*	
TP03						*

3.3.5. Результати тестування реалізації вимог до ПЗ

Відповідно до розподілу груп, групи були протестовані та отримані результати. За допомогою pytest було виконано функціональне тестування. Код до тестів наведено в Додатку В (див. Додаток В.3)

Результати тестування наведені на рисунку 3.3.

```
PS D:\diploma\functional_tests> python .\image_tests.py
===== test session starts =====
platform win32 -- Python 3.11.5, pytest-8.2.1, pluggy-1.5.0
PyQt6 6.7.0 -- Qt runtime 6.7.1 -- Qt compiled 6.7.1
rootdir: D:\diploma\functional_tests\
plugins: anyio-3.7.1, mock-3.14.0, qt-4.4.0
collected 12 items
image_tests.py .....
===== 12 passed in 21.04s ===== [100%]

PS D:\diploma\functional_tests> python .\infor_tests.py
===== test session starts =====
platform win32 -- Python 3.11.5, pytest-8.2.1, pluggy-1.5.0
PyQt6 6.7.0 -- Qt runtime 6.7.1 -- Qt compiled 6.7.1
rootdir: D:\diploma\functional_tests\
plugins: anyio-3.7.1, mock-3.14.0, qt-4.4.0
collected 2 items
infor_tests.py ..
===== 2 passed in 0.22s ===== [100%]

PS D:\diploma\functional_tests> python .\admin_tests.py
d:\diploma\server-side\config\db_config.json
===== test session starts =====
platform win32 -- Python 3.11.5, pytest-8.2.1, pluggy-1.5.0
PyQt6 6.7.0 -- Qt runtime 6.7.1 -- Qt compiled 6.7.1
rootdir: D:\diploma\functional_tests\
plugins: anyio-3.7.1, mock-3.14.0, qt-4.4.0
collected 14 items
admin_tests.py .....
===== 14 passed in 13.28s ===== [100%]

PS D:\diploma\functional_tests> python .\payme_tests.py
===== test session starts =====
platform win32 -- Python 3.11.5, pytest-8.2.1, pluggy-1.5.0
PyQt6 6.7.0 -- Qt runtime 6.7.1 -- Qt compiled 6.7.1
rootdir: D:\diploma\functional_tests\
plugins: anyio-3.7.1, mock-3.14.0, qt-4.4.0
collected 3 items
payme_tests.py ...
===== 3 passed in 3.01s ===== [100%]

PS D:\diploma\functional_tests> |
```

Рисунок 3.3 – Результат pytest після виконання функціональних тестів.

В результаті виконання функціональних тестів, помилок знайдено не було.

3.3.6. Висновки щодо тестування реалізації вимог до ПЗ

Наступні висновки можна отримати після проведення тестування реалізації вимог до програмного забезпечення (ПЗ), яке базується на використанні pytest і програми для обробки зображень, яка визначає PROMPT на основі моделі CLIP:

- Стабільність програмного забезпечення: під час тестування програма для обробки зображень продемонструвала високий рівень стабільності. Програма продемонструвала надійну працездатність без помітних випадкових збоїв під час кількох тестів, що включали модульні, інтеграційні тести та приймально-здавальні тести.

- Відповідність вимогам функціональності: ПЗ успішно відповідає всім вимогам щодо функціональності, таким як здатність обробляти зображення та визначати PROMPT на основі CLIP моделі. Програмне забезпечення виконало всі вимоги, визначені на етапі аналізу та проектування.

- Ефективність та продуктивність: під час тестування було проведено аналіз ефективності та продуктивності програми. ПЗ показав високу швидкість обробки зображень і визначення PROMPT, при цьому ресурси були оптимізовані.

- Відсутність помилок і дефектів: під час тестування не виявлено жодних критичних помилок або дефектів, які могли значно вплинути на роботу програмного забезпечення або його функціональність.

- Відповідність стандартам безпеки: ПК дотримується стандартів безпеки обробки зображень і зберігання даних, що гарантує, що інформація залишається конфіденційною та безпечною.

Загалом результати тестування показують, що вимоги до програмного забезпечення виконуються з високою якістю та надійністю. Рекомендується продовжувати моніторинг роботи програми та вживати заходів для забезпечення того, щоб вона продовжувала працювати ефективно та функціонально в майбутньому.

3.4. Тестування реалізації варіантів використання

3.4.1. План тестування реалізації варіантів використання

Тестування реалізації варіантів використання – це вид тестування програмного забезпечення, який перевіряє, наскільки ефективно програмне забезпечення відповідає встановленим варіантам використання. Це охоплює перевірку того, чи відповідає програмний продукт потребам користувачів, які визначені на етапі аналізу та специфікації варіантів використання програмного продукту. [58]

При використанні Pytest для тестування реалізації варіантів використання потрібно створити тестовий набір, який описує очікувану поведінку програмного забезпечення відповідно до варіантів використання. Це може включати тести для перевірки функціональності програми, взаємодії з користувачем, швидкодії, безпеки та інших аспектів, що визначені варіантами використання.

Після написання тестового набору Pytest виконується за допомогою запуску команди в терміналі, що дозволяє провести тестування реалізації варіантів використання. [59] Результати тестування можна використовувати для оцінки відповідності програмного продукту встановленим варіантам використання та виявлення можливих невідповідностей або помилок. Автоматизація цього процесу за допомогою Pytest дозволяє ефективно перевіряти реалізацію варіантів використання та забезпечує швидке виявлення проблем на ранніх етапах розробки.

Таблиця 4.1 – Тестування варіанта використання «Авторизація»

Базовий потік	Крок	Опис
А – Користувач S – Додаток	1	А: користувач вводить логін (пошту) та пароль.
	2	S: перевіряє логін та пароль
	3	S: виводить повідомлення про правильний логін
	4	S: додаток переносить користувача на основну сторінку
Розширення	1a	А: вводить неправильні дані
	2a	S: виводить повідомлення про помилку
	2b	S: сервер не працює
Винятки	1b	А: за відсутності реєстрації користувач має зареєструватись
	1c	S: система дає можливість зареєструватись користувачу

Таблиця 4.2 – Тестування варіанта використання «Опрацювання зображення»

Базовий потік	Крок	Опис
А – Користувач	1	А: користувач завантажує зображення до додатку
S – Додаток	2	S: перевіряє, чи завантажене зображення відповідає вимогам та чи обрав користувач метод опрацювання (локальний чи на сервері)

Продовження таблиці 4.2

Базовий потік	Крок	Опис
	3	S: видає користувачу варіанти опрацювання: серверне (див. 2.1) або локальне (див. 2.2)
	4	A: користувач обирає метод опрацювання (локальний чи на сервері)
	5	Якщо обрано серверне опрацювання:
		S: відправляє зображення на сервер
		S: сервер обробляє зображення та надсилає опрацьований PROMPT (див. 2.1.E1)
		A: користувач отримує у додатку опрацьований PROMPT
	6	Якщо обрано локальне опрацювання:
		S: додаток опрацьовує зображення користувача локально
		A: користувач отримує у додатку опрацьований PROMPT
Розширення	1a	A: користувач отримує повідомлення про помилку (неправильні дані)
		S: додаток пропонує користувачеві оформити підписку або перейти до локального опрацювання
	2a	S: сервер не зміг опрацювати зображення

Кінець таблиці 4.2

Базовий потік	Крок	Опис
		A: користувач отримує повідомлення про неможливість опрацювання на сервері та можливість відправити помилку розробникам
Винятки	1с	S: система дає можливість зареєструватись користувачеві

Таблиця 4.3 – Тестування варіанта використання «Купити платіжну підписку»

Базовий потік	Крок	Опис
A – Користувач	1	A: користувач натискає кнопку купити підписку.
S – Додаток	2	S: додаток відображає інформацію про наявні підписки та їх вартість
API – Банк	3	A: користувач обирає підписку та її термін
	4	A: користувач вводить дані банку або використовує вбудовану систему платежів (GooglePay, ApplePay)
		API: перевіряє правильність введених даних
	5	Якщо дані введено правильно:
		A: користувач отримує статус підписника на місяць.
Винятки	4a	A: ввів неправильні дані.
		API: банк повідомляє про помилку
	4b	A: користувач отримує повідомлення про кошти.

Таблиця 4.4 – Тестування варіанта використання «написати звернення в Q&A»

Базовий потік	Крок	Опис
А – Користувач	1	А: користувач натискає кнопку "надіслати звернення".
S – Додаток	2	А: користувач пише тему та повідомлення.
	3	А: користувач натискає кнопку "відправити" для надсилання звернення до Q&A.
	4	S: система відправляє повідомлення на сервер
Розширення	—	—
Винятки	4а	А: користувач отримує повідомлення з поштою, на яку можна надіслати звернення вручну.

Таблиця 4.5 – Тестування варіанта використання «Переглянути документацію»

Базовий потік	Крок	Опис
А – Користувач S – Додаток	1	А: користувач натискає кнопку "переглянути документацію"
	2	А: користувач отримує доступ до документації.
	3	S: користувач отримує посилання на відповідні інтернет-джерела.
Розширення	—	—
Винятки	—	—

Таблиця 4.6 – Тестування варіанта використання «написати звернення в Q&A»

Базовий потік	Крок	Опис
A – Користувач S – Додаток	1	A: користувач зайшов в вкладку для підписників
	2	A: користувач читає про нові алгоритми або функції, які доступні.
	3	A: після натискання кнопки "завантажити", користувач отримує доступ до цих функцій
	4	S: додаток завантажує відповідне оновлення або відповідну генеративну модель
Розширення	—	—
Винятки	4a	A: користувач отримує повідомлення про неможливість завантаження оновлень через проблеми з сервером.

Таблиця 4.7 – Тестування варіанта використання «адміністрування системи»

Базовий потік	Крок	Опис
А – Користувач S – Додаток	1	А: адміністратор зайшов у відповідний додаток
	2	А: адміністратор переглядає базу даних та рапорти по системі.
	3	А: адміністратор працює з переліком користувачів (видаляє, редагує відповідні акаунти).
	4	S: система реагує на зміни та вносить їх
Розширення	2a	А: адміністратор перевіряє оплату
	2b	А: адміністратор читає звернення в Q&A.
Винятки	2c	S: адміністратор отримує повідомлення та рапорт про те, що не можна було сформулювати статистичні відомості.

3.4.2. Результати тестування реалізації варіантів використання

Для переконання в коректності функціональності та відповідності вимогам результати тестування реалізації варіантів використання систематично збиралися та аналізувалися. Тестування допомогло знайти та виправити потенційні помилки та недоліки в роботі кожного варіанту використання. Реалізація варіантів забезпечила високу надійність на ефективність програми. [60]

3.4.3. Висновки щодо тестування реалізації варіантів використання

Було проведено тестування реалізації варіантів використання, і всі висновки слід розбити за такими критеріями:

- Якість впровадження: Більшість варіантів використання було реалізовано відповідно до специфікацій і вимог.

- Програмне забезпечення працює надійно без будь-яких збоїв або помилок під час виконання вашого сценарію використання.

- Виявлені проблеми: Тестування виявило кілька незначних проблем із правильним відображенням деяких елементів інтерфейсу під час аутентифікації та обробки зображень.

Крім того затримки у відповіді сервера під час запуску деяких альтернативних маршрутів, були ліквідовані. [61]

Висновки до розділу 3

У даному розділі були розроблені плани для тестувань: інтеграційного, модульного, функціонального, нефункціонального та тестування варіантів використання. Відповідно до створених планів, було написано та відредаговано тести до проекту «Програмне забезпечення попередньої обробки PROMPT на основі CLIP моделі». Всі дані для звіту були взяті з відкритих джерел.

Всі баги та проблеми знайдені під час тестування, були виправлені. [62]

4 ЕКОНОМІЧНЕ ОБҐРУНТУВАННЯ ПРОЕКТУ

4.1. Опис, споживацька ціна та аналіз конкурентоспроможності програмного продукту

До реалізації пропонується продукт, який надає систему доступу до обробки зображень та отримання з них PROMPT. Головною метою цієї системи є забезпечення можливості опрацювання зображень зі сторони користувача: визначення PROMPT, купівля підписки. Зі сторони адміністратора: змінювати дані в базі даних, відповідати на звернення користувачів.

Ця система враховує вельми нюансів у роботі з користувачем і базою даних. У додатку реалізовано створення різних ролей користувача завдяки використанню додаткових компонентів. Користувач має зареєструватись, інакше він не матиме доступу до сервісу. У випадку реєстрації та відсутності підписки, користувач матиме обмежений доступ.

Систему слід використовувати у роботі з зображеннями, які в подальшому будуть оброблюватись іншими моделями штучного інтелекту.

4.1.1. Оцінювання ринку збуту та конкуренція

Для оцінювання ринку збуту використаємо метод сегментування ринку.

Сегментування ринку – це процес поділу користувачів на групи з урахуванням різних чинників та принципів.

Формула для обрахування визначена в 4.1.

$$S_{\text{повн.}i} = N_i * K_{\text{ЕОМ}i} * K_{\text{охоп.}i} * m_j; \quad (4.1)$$

$$S_i = S_{\text{повн.}i} / T, \quad (4.2)$$

де N_i – кількість населення в i -му сегменті;

$K_{\text{ЕОМ}i}$ – частка власників ПК;

$K_{\text{охоп.}i}$ – коефіцієнт охоплення, тобто частка власників ПК, які бажають придбати ПП в i -му сегменті;

m_j – комплект поставки для одного ПК, зазвичай $m_j = 1$ шт.

T – строк повторної купівлі ПП.

Для визначення сегментів ринку, звернемося до відкритих джерел в інтернеті. Першим сегментом будуть користувачі віком від 15 до 64 років, які працюють в сфері штучного інтелекту, 3D моделювання або оброблюють зображення (графічні дизайнери, митці, тощо) або просто використовують штучний інтелект в своєму житті та проживають на території України. Другим сегментом будуть користувачі з англомовних країн, які мають однаковий опис з першим сегментом.

Для отримання відповідних числових даних для першого сегменту, звернемося до даних Державної Служби Статистики України. [63] Маємо, що кількість жителів віком від 15 до 64 станом на 1 січня 2022 – 27 646 700 осіб. Дослідження, проведене Центром Разумкова на замовлення видання «Дзеркало тижня», показало, що українці не особливо обізнані з інструментами штучного інтелекту та майже не використовують його в роботі та повсякденному житті, використовує лише 9% [64], це і буде використано як $K_{\text{охоп.}i}$. Видання «Mind» провело дослідження для виявлення кількості українців, що мають ПК або ноутбук [65], як результат лише 59%, що буде використано як $K_{\text{ЕОМ}i}$.

Використовуючи формулу 4.1, обраховуємо $S_{\text{повн.}i}$.

$$S_{\text{повн.}i} = 27\,646\,700 * 0.59 * 0.14 * 1 = 1\,468\,039 \text{ осіб}$$

Строком придбання підписки в рік (T) буде 30 днів, або 0.08 року.

Використовуючи формулу 1.2, обраховуємо S_i

$$S_i = 1\,468\,039 / 0.08 = 18\,350\,487$$

Другий сегмент – жителі сполучених штатів Америки, які розбиваються по таким самим критеріям, як перший сегмент. Кількість жителів за даними «Державної Статистики США», становить 339 996 563 жителів в 2023 році. [66] За даними розслідування видання «YouGov» 24% жителів використовують штучний інтелект в своєму житті. [67] Центр досліджень «Pew Research Center» в 2023 році підвів статистику, що кількість жителів, які мають ПК чи ноутбук становить 73%.

Варто також зазначити, що конкуренто-спроможність на ринку США в сфері «AI-based Applications» дуже велика, тому $S_{повн.i}$ буде помножено на коефіцієнт 0.2, щоб сегментування було більш реальним.

Використовуючи формулу 4.1, обраховуємо $S_{повн.i}$.

$$S_{повн.i} = 339\,996\,563 * 0.24 * 0.73 * 0.2 = 11\,913\,479 \text{ осіб}$$

Використовуючи формулу 4.2, обраховуємо S_i

$$S_i = 11\,913\,479 / 0.08 = 148\,918\,500 \text{ осіб}$$

Таблиця 4.1 – Сегментування та місткість ринку ПП

Регіон	Кількість N_i	Частка власників ПК $K_{ЕОМi}$	Кількість ПК, шт.	Коефіцієнт охоплення $K_{охоп.i}$	Потреба в ПП $S_{повн.i}$ шт.	Строк повторного придбання T , рік	Річна місткість S_i , шт./рік
Перший сегмент	27 646 700	0.59	1	0.14	1 468 039	0.08	18 350 487
Другий сегмент	339 996 563	0.73	1	0.24	11 913 479	0.08	148 918 500
Всього	-	-	-	-	13 381 518	-	167 268 987

4.1.2. Виявлення конкурентів та аналіз конкурентоспроможності програмного продукту

Використання програмного забезпечення помітно спрощує комунікацію та зменшує кількість годин, необхідних для прийняття рішень в усіх сферах життя людей. Перш за все це бізнес-процеси, де від кількості часу залежить прибуток.

Desktop Application – розподілений додаток, в якому клієнтом є окремий додаток з графічним інтерфейсом, а сервером – окремо написаний сервер (можливо навіть термінального інтерфейсу).

Штучний інтелект, кожного дня охоплює все більшу кількість людей та галузей, тому що вони пришвидшують роботу та зручні в використанні.

Магазини додатків на різних платформах (телефони, ПК, тощо) дозволяють встановлювати додатки розробників зі всього світу. Незважаючи на конкуренцію і мільйони додатків, що базуються на штучному інтелекті, можна заснувати свій невеличкий стартап і заробляти на цьому.

Стрімкий розвиток мережі Інтернет та застосування новітніх технологій у комерційній діяльності та повсякденному житті зумовили виникнення нових економічних явищ, серед яких можна виділити «self-hosted companies», спрямовані на обробку зображень.

Цей сегмент стартапів, націлений на розробку ПЗ та інструментів, що дають можливість аналізувати та обробляти візуальні дані.

Завдяки доступності технології та зростаючому інтересу до візуального аналізу даних, компанії у цій галузі мають широкі можливості розвитку та успіху. Вони можуть залучати інвестиції, співпрацювати з іншими компаніями та впроваджувати рішення на ринку, які дозволять стати ключовими гравцями у сфері оброблення зображень.

За даними зібраними в інтернеті, та аналізі відкритих українських джерел, в Україні лише 2 компанії, що займаються обробкою зображень. Тому для визначення конкурентної спроможності, додаток буде орієнтуватись на аналоги з Євросоюзу та Америки (США, Канада).

До реалізації пропонується програмний продукт, що не має аналогів в Україні.

З метою просування товару маркетинговою компанією передбачаються:

- використання елементів соціальної комерції на основі соціальних мереж (TikTok, Facebook, Instagram, Telegram);
- публікація на популярних форумах відповідно до тематики продукту (Reddit, X, тощо);
- регулярний пошук клієнтів серед популярних блогерів чи дизайнерів, які за спеціальні умови використання, будуть готові прорекламувати товар.

Потреба в розробленому програмному продукті буде підвищуватися у зв'язку:

- 1) з недостатньою кількістю готових рішень;
- 2) підвищенням популярності моделей ШІ та моделей створення зображень на основі PROMPT;

3) розробкою нових технологій в сфері відео, аудіо та композиції віртуальних світів.

4.1.3. Розрахунок споживацької ціни програмного продукту, уточнення цільової місткості ринку

За результатами проведеного аналізу конкурентоспроможності можна прогнозувати споживацьку ціну товару та позиціонувати його на ринку з уточненнями цільової місткості ринку.

Формуємо споживацьку ціну нового продукту, що є передбачувана максимально можлива ціна, яку зможуть заплатити покупці, враховуючи переваги нового продукту порівняно з продуктами найкращих конкурентів. Споживацька ціна визначається за формулою (див. формула 4.3):

$$C_{\text{спож}} = 0,9 * C_{\text{б}} * Y_{\text{н-б}} , \quad (4.3)$$

де $C_{\text{спож}}$ – споживацька (максимально можлива) ціна нового продукту;

$C_{\text{б}}$ – ціна реалізації базового продукту (найкращого конкурента);

коефіцієнт 0,9 враховує моральне старіння базового продукту до початку реалізації нового;

$Y_{\text{н-б}}$ – рівень якості нового продукту відносно базового.

Отже, з проведеного аналізу ми маємо, що ціна найкращого конкурента становить 9,99 \$ за один місяць підписки. За курсом 20 травня 2024 року 1 долар дорівнює 39,83 гривні. [68] Оскільки новий продукт поки що, відстає від конкурентів за реалізацією, тому, рівень якості нового продукту оцінюємо за коефіцієнтом 0,50.

Обраховуємо споживацьку ціну:

$$C_{\text{спож}} = 0,9 * 9,99 * 0,5 * 39,83 = 179,20 \text{ UAH}$$

4.2. Виробничо-організаційний план проекту. Розрахунок трудомісткості проекту та кількості його виконавців

4.2.1. Перелік робіт

Роботу буде виконувати один працівник – системний інженер, який здійснює господарську діяльність в організаційно-правовій формі – фізична особа-підприємець (ФОП). Вид діяльності фізичної особи – підприємця вибирається відповідно до *Класифікатора видів економічної діяльності*: 62.0 – Комп'ютерне програмування, консультування та пов'язана з ними діяльність.

Цей розділ містить види діяльності з надання експертної оцінки у сфері інформаційних технологій, а саме: розроблення, модифікацію, тестування і технічну підтримку програмного забезпечення, планування та проектування інтегрованих комп'ютерних систем, які поєднують апаратні засоби, програмне забезпечення та комунікаційні технології; керування й обслуговування комп'ютерних систем клієнтів і/або оброблення даних та іншу професійну діяльність у сфері інформаційних технологій.

Визначимо перелік робіт за проектом і розрахуємо загальну трудомісткість виконання робіт (див. табл. 4.2)

Таблиця 4.2 – Перелік робіт

Номер етапу	Назва стадій та етапів	Трудомісткість, люд.-дні
		Системний інженер
I	Аналіз вимог до програмного забезпечення	10
II	Підготовка до розробки (створення макетів інтерфейсів, архітектури, тощо)	14
III	Розроблення програмного продукту	22
IV	Тестування та реліз продукту	13
Разом		59

Загальна трудомісткість за проектом становить 59 нормо-днів, що співпадає з тривалістю проекту - 59 днів, тому що проект виконує 1 особа ФОП.

За даними таблиці будуємо діаграму Ганта за допомогою Microsoft Excel. (див. рис. 4.1.)

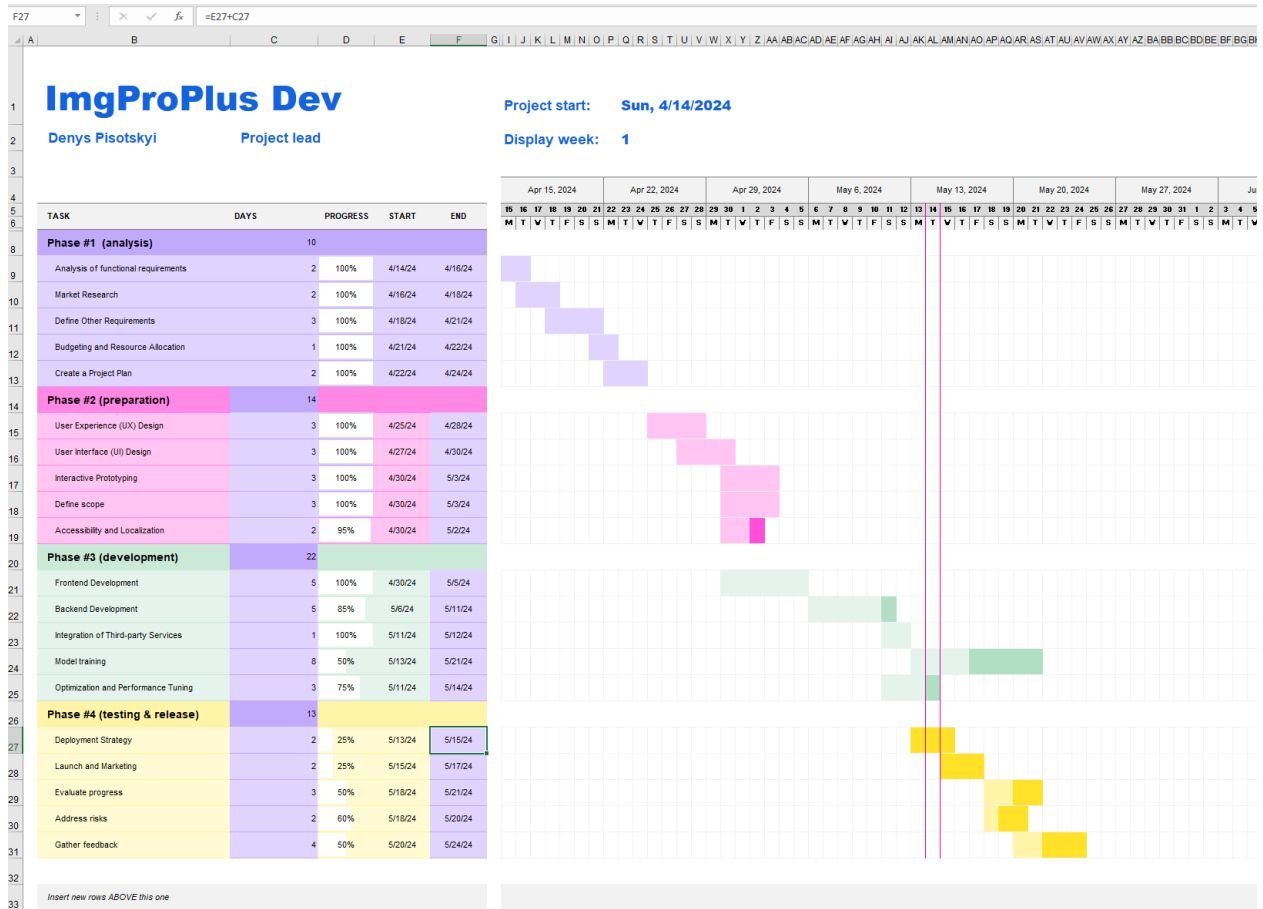


Рисунок 4.1 – Діаграма Ганта побудована за переліком робіт

4.2.2. Розрахунки трудомісткості робіт проекту

Законодавством не встановлено єдиної норми тривалості робочого часу на рік. Ця норма може бути різною залежно від того, який робочий тиждень встановлено в організації (п'ятиденний або шестиденний), яка тривалість щоденної роботи, коли встановлено вихідні дні. У зв'язку з цим на підприємствах, в установах і організаціях норма тривалості робочого часу на рік визначається

самостійно з дотриманням вимог статей 50-53, 67 і 73 Кодексу законів про працю (КЗпП).

Як передбачено ст. 50 КЗпП, нормальна тривалість робочого часу працівників не може перевищувати 40 годин на тиждень. Відповідно до ч. 1 ст. 51 КЗпП скорочена тривалість робочого часу встановлюється:

Розрахунок норми тривалості робочого часу на 2024 рік за календарем п'ятиденного робочого тижня з двома вихідними днями в суботу та неділю при однаковій тривалості часу роботи за день протягом робочого тижня та відповідним зменшенням тривалості роботи напередодні святкових і неробочих днів наводиться у листі Міністерства соціальної політики України від 29.07.2019 № 1133/0/206-19 «Про розрахунок норми тривалості робочого часу на 2024 рік».

Таким чином, місячний фонд робочого часу в 2024 році становить 21 день у місяць і 251 день у рік.

Нормування праці в процесі створення ПЗ істотно ускладнене через творчий характер праці програміста.

Тому трудомісткість розроблення ПЗ можна розраховувати на основі системи моделей з різною точністю оцінювання за формулою (1.4):

$$\text{ПЗ} = t_o + t_u + t_a + t_n + t_m \text{ (людино-годин)}, \quad (4.4)$$

де t_o – споживацька (максимально можлива) ціна нового продукту;

t_o – витрати праці на аналіз і опис предметної галузі;

t_u – витрати праці на визначення мети дослідження;

t_a – витрати праці на дослідження інструментальних засобів і створення моделі;

t_n – витрати праці на програмування за готовою моделлю;

t_m – витрати праці на підготовку документації та маркетинг.

В нашому випадку $t_o = 0$, оскільки мета та ідея розроблення сформована спочатку та входить в інші витрати.

Маємо, що

$$\text{ПЗ} = 10 + 0 + 14 + 22 + 13 = 59 \text{ (людино-годин)}$$

4.2.3. Виконавці проекту

Як зазначено в пункті 4.2.1, виконавцем проекту буде одна особа, а саме системний інженер.

Визначаємо, що заробітна плата системного інженера становить 25000 грн за місяць. Ураховуємо, що місячний фонд в 2024 році становить 21 день на місяць.

$$L_{\text{доб}} = O_{\text{м}} / N_{\text{роб. днів}}, \quad (4.5)$$

де $O_{\text{м}}$ – оклад місячний, грн;

$N_{\text{роб.днів}}$ – кількість робочих днів у місяці;

Отже, розрахуємо поденний оклад системного інженера за формулою (4.5):

$$L = \frac{25000}{21} = 1190,50(\text{грн}).$$

Таблиця 4.3 – Виконавці проекту

Код виконавця	Посада	Місячний розмір* посадового окладу, грн	Денний розмір* посадового окладу, грн
01	Системний інженер	25000	1190, 50

4.3. Фінансовий план проекту

4.3.1 Розрахунок потреб в інвестиціях на обладнання

Для друку необхідних документів, звітів було використано різне канцелярське приладдя. Перелік та вартість матеріалів наведено в таблиці 4.4. (див. табл. 4.4)

Таблиця 4.4 – Вартість канцелярського приладдя

№ п/п	Матеріали	Кількість	Ціна, грн	Вартість, грн
1	Папір офісний форматом А4 (500 шт./уп.)	1	185	185
2	Матові файли форматом А4 (100 шт./уп.)	1	120	120
3	Канцелярське знаряддя в асортименті, шт.	2	250	500
4	Наповнювачі в принтер для друку, шт.	1	1200	1200
Разом				2005

Разом вартість канцелярського приладдя для оформлення документації, та друку становить 2005 грн.

4.3.2 Розрахунок заробітної плати виконавців проекту

Роботу буде виконувати самозайнята особа, зареєстрована у формі фізичної особи-підприємця.

Фізична особа-підприємець – це зручна організаційно-правова форма для невеликого бізнесу. Переваги такої форми:

- відносна простота реєстрації та ведення обліку;
- відсутність потреби наймати директора;
- не потрібно виплачувати заробітну плату/дивіденди (кошти фізичної особи підприємця є автоматично коштами фізичної особи);
- можливість вибрати будь-яку групу спрощеної системи оподаткування (за певних умов).

Для ведення господарської діяльності вибирається II група спрощеної системи оподаткування зі сплатою єдиного податку. Ставка єдиного податку розраховується виходячи з розміру мінімальної заробітної плати 8000 грн і з 1 квітня 2024 року становить 1420 грн/місяць.

Таким чином, щомісячно потрібно буде сплачувати єдиний податок у розмірі 1420 грн.

Отже,

$$\text{ЄП}_{\text{проект}} = 1420 * 5 = 7100 \text{ грн.}$$

Визначимо заробітну плату виконавця роботи. В пункті 4.2.3, було обраховано поденний оклад. Маємо:

$$L_0 = T * L, \quad (4.6)$$

де T – трудомісткість дослідження.

Після підстановки числових значень отримаємо

$$L_0 = 59 * 1190,50 = 70239,50 \text{ грн}$$

Обов'язкове нарахування на заробітну плату – єдиний соціальний внесок ЄСВ – становить 22 % на суму заробітної плати. Розрахуємо ЄСВ:

$$L_{add} = L * 0,22 = 70239,50 * 0,22 = 15452,69 \text{ грн}$$

Загальну суму заробітної плати і нарахування на заробітну плату визначаємо за формулою

$$L_{\text{заг}} = L_0 + L_{add} = 70239,5 + 15452,69 = 85692,19 \text{ грн}$$

4.3.3 Складання кошторису витрат на розроблення проекту програмного продукту

У межах підготовки до роботи необхідно знайти приміщення для розгортання інфраструктури. З цією метою було орендовано приміщення на підприємстві площею 15 кв. м. Приміщення обладнано освітленням, мережею електричного живлення та іншими комунікаціями. Орендна плата становить 5100 грн/місяць. Вартість оплати інтернету і мобільного зв'язку – 280 грн/місяць. Вартість опалення у зимовий період – 2500 грн/місяць.

Розрахунок витрат на інтернет, опалення на аренду наведено в таблиці 4.5. (див. табл. 4.5.)

Таблиця 4.5 – Розрахунок витрат на оренду, інтернет та опалення

№ п/п	Найменування	Ціна за місяць, грн	Кількість місяців використання послуги	Вартість на весь період, грн
1	Оренда	5100	5	25500
2	Інтернет та мобільний зв'язок	280	5	1200
3	Опалення	2500	3	7500
Разом				34200

Розрахунок споживання електроенергії на 59 робочих дні при восьми годинах робочого дня наведено в таблиці 4.6.

Ноутбук споживає

$$P_{Laptop} = 1 * 0,18 * 472 = 84,96 \left(\frac{\text{кВт}}{\text{год}} \right),$$

використання зовнішнього монітору (1 шт.) –

$$P_{monitor} = 1 * 0,2 * 472 = 94,4 \left(\frac{\text{кВт}}{\text{год}} \right),$$

освітлення приміщення лампами (2 шт.) –

$$P_{leds} = 2 * 0,04 * 472 = 37,76 \left(\frac{\text{кВт}}{\text{год}} \right),$$

роутер –

$$P_{router} = 1 * 0,06 * 472 = 28,32 \left(\frac{\text{кВт}}{\text{год}} \right).$$

Таблиця 4.6 – Розрахунок споживання електроенергії для виконання роботи

№ п/п	Найменування	Потужність, кВт/год	Кількість відпрацьованого часу, год	Кількість спожитої електроенергії, кВт/год
1	Ноутбук Lenovo	0,18	472	84,96
2	Монітор Dell	0,2	472	94,4
3	Освітлення (лампи)	0,04	472	37,76
4	Роутер TP-Link	0,06	472	28,32
Разом				245,44

Відповідно до інформації, розміщеної на сайті Національної комісії, що здійснює державне регулювання у сферах енергетики та комунальних послуг, вартість електроенергії в АТ «Полтаваобленерго» для непобутових споживачів 2-го класу напруги становить 1,68 грн за 1 кВт/год. [69]

Розрахуємо вартість спожитої електроенергії:

$$P_{TOTAL} = P * C, \quad (4.7)$$

де P – кількість спожитої електроенергії, кВт/год;

C – ціна за кВт/год, грн.

Обчислимо вартість спожитої електроенергії за весь термін проведення дослідження за формулою (4.7):

$$P_{TOTAL} = 245,44 * 1,68 = 412,33 \text{ (грн)}$$

Реалізація проекту не потребує купівлі нової обчислюваної техніки, яка сьогодні є в наявності. Однак, для тренування моделей штучного інтелекту, було прийнято рішення орендувати сервер з потужними відеокартами. Розрахунок споживання обчислювальної потужності на сервері з необхідними GPU наведено в таблиці 4.7. (див. табл. 4.7)

Вибір пав на тариф, що надає доступ до двох відеокарт Nvidia RTX 4090 TI, які мають достатню потужність для тренування моделей, з ціною \$0.905/годину. За курсом Мінфіну переводимо в гривні:

$$C_{GPU} = 0,905 * 39,83 = 36,04 \left(\frac{\text{грн}}{\text{год}} \right)$$

Відповідно тренування відбувалось майже весь час розробки, тому час тренування складає:

$$T_{GPU} = 24 * 59 = 1416 \text{ години.}$$

Розраховуємо потрачені кошти на хмарний сервіс оренди відеокарт:

$$P_{GPU} = C_{GPU} * T_{GPU} = 36,04 * 1416 = 51032,64 \text{ грн}$$

Таблиця 4.7 – Розрахунок витрат на хмарний сервіс оренди GPU

№ п/п	Найменування	Вартість, грн/год	Кількість відпрацьованого часу, год	Кількість витрачених коштів, грн
1	Оренда 2 відеокарт Nvidia RTX 4090 TI	36,04	1416	51032,64
Разом				51032,64

Розрахуємо загальну суму амортизаційних відрахувань (за проект).

Суму амортизаційних відрахувань обчислюємо за формулою

$$A = F * N_A / 100\%, \quad (4.8)$$

де F – первісна вартість основних засобів за видами, грн;

N_A – річна норма амортизації за видами основних засобів, %.

Отже,

$$A_{T_{laptop}} = 28000 * \frac{20\%}{100\%} = 5600 \text{ (грн),}$$

$$A_{T_{monitor}} = 18000 * \frac{20\%}{100\%} = 3600 \text{ (грн),}$$

$$A_{T_{router}} = 4800 * \frac{20\%}{100\%} = 960 \text{ (грн).}$$

Суму амортизаційних відрахувань за місяць визначаємо за формулою

$$A_M = A_T * N / T, \quad (4.9)$$

де A_T – річна сума амортизаційних відрахувань, грн;

N – тривалість проекту, дні;

T – тривалість робочого року, дні.

Звідси

$$A_{M_{laptop}} = \frac{5600 * 59}{251} = 1316,30(\text{грн}),$$

$$A_{M_{monitor}} = \frac{360 * 59}{251} = 846,20(\text{грн}),$$

$$A_{M_{router}} = \frac{960 * 59}{251} = 225,65(\text{грн}),$$

Суму амортизаційних відрахувань за рік і за місяць наведено в таблиці 4.8. (див. табл. 4.8)

Таблиця 4.8 – Розрахунок річної суми амортизаційних відрахувань

№ п/п	Елементи основних засобів	Кількість	Вартість, грн	Сума, грн	Норма амортизації, %	Амортизаційні відрахування за рік, грн	Амортизаційні відрахування за місяць, грн
1	Ноутбук	1	28000	28000	20%	5600	1316,3
2	Монітор	1	18000	18000	20%	3600	846,2
3	Роутер	1	4800	4800	20%	960	225,65
Разом							2388,15

Діяльність з реалізації програмного забезпечення передбачає відрядження для встановлення та технічної підтримки програми. Мінімальна заробітна плата у розрахунку на місяць з 01.04.2024 року становить 8000 грн (ст. 8 Закону України "Про Державний бюджет України на 2024 рік"). Отже, максимальний розмір неоподатковуваних добових у 2024 році у межах території України становить 800 грн у розрахунку за кожен календарний день такого відрядження.

Унаслідок урахування всіх перелічених вище статей витрат можна визначити собівартість проекту. Розрахунок статей витрат на розроблення програмного продукту наведено в таблиці 4.9. Тривалість проекту – 59 днів.

Таблиця 4.9 – Кошторис витрат за проектом

№ п/п	Стаття калькуляції	Сума, грн
1	Заробітна плата	70239,50
2	Єдиний соціальний внесок	15452,69
3	Єдиний податок (сплачується з моменту реєстрації)	7100,00
4	Інформаційні послуги: інтернет і мобільний зв'язок	1200,00
5	Комунальні послуги (електроенергія)	245,44
6	Оренда приміщення	25500,00
7	Опалення	7500,00
8	Амортизація основних засобів	2388,15
9	Канцелярське приладдя	2005,00
10	Витрати на хмарні сервіси для розвитку проекту	51032,64
11	Витрати на відрядження	0
Разом: собівартість власних робіт		182663,42

Таким чином, унаслідок виконаних розрахунків витрати на дослідження й створення програмного забезпечення для обробки зображень визначення PROMPT на основі CLIP моделі становлять 182663,42 грн.

4.3.6 План доходів від реалізації програмного продукту та витрат на його виготовлення

Клієнти, які виявили бажання користуватись продуктом і придбали преміум-підписку, – 2450 користувачів. Очікуваний дохід від реалізації програмного продукту при вартості 179,20 грн визначається за формулою:

$$P = N_s * C,$$

де N – кількість споживачів;

C – вартість одиниці програмного продукту, грн.

Також варто зазначити необхідність обрахування вартості разом з ПДВ. (20%)

Отже,

$$P = 2450 * 179,20 = 439040 \text{ (грн)}.$$

Економічну ефективність пропонованого впровадження оцінено за системою показників, які використовуються у міжнародній і вітчизняній практиці (див. табл. 4.10). Під час оцінювання економічної ефективності проекту використано такі показники:

- прибуток, що розраховується за формулою;
- рентабельність витрат;
- рентабельність доходу;
- термін окупності.

Таблиця 4.10 – Розрахунок економічної ефективності проекту

Показник	Методика розрахунку	Сума, грн
Дохід, грн		526848
Витрати, грн		182663,42
Прибуток, грн	Дохід – Витрати	344184,57
Рентабельність витрат, %	(Прибуток / Витрати) * 100%	188,42
Рентабельність доходу, %	(Прибуток / Дохід) * 100%	65,32

Висновки до розділу 4

В економічному розділі було виконано наступні завдання:

- проведено сегментацію ринку та визначено конкурентоспроможність пропонованого проекту. Зроблено висновок, що програмний продукт є конкурентоспроможним;
- проведено розрахунок трудомісткості виконаної роботи за проектом – 59 людино-днів;

- призначено виконавців проекту та його оклад;
- складено кошторис витрат за проектом – 182663,42 грн;
- розраховано очікуваний прибуток за результатами реалізації проекту – 344184,57 грн;
- розрахована рентабельність витрат становить 188,42 %, а рентабельність доходу – 65,32 %.

ВИСНОВКИ

У даному дипломному проєкті було розроблено програмне забезпечення попередньої обробки зображень для визначення PROMPT на основі CLIP моделі. Це рішення дозволяє автоматизувати процес опису зображень, який є важливим етапом у багатьох галузях, таких як комп'ютерний зір, обробка природної мови та генерація зображень.

Ключові результати та висновки даної роботи:

1. Проаналізовано існуючі підходи та методи для опису зображень, зокрема, моделі на основі глибокого навчання, такі як CLIP (Contrastive Language-Image Pre-training).

2. Розроблено програмне забезпечення, що інтегрує CLIP модель та забезпечує попередню обробку зображень, включаючи завантаження, масштабування, нормалізацію та перетворення в необхідний формат для подальшого аналізу.

3. Реалізовано алгоритм для генерації PROMPT (текстового опису), який максимально відповідає змісту зображення, використовуючи CLIP модель та методи пошуку оптимального рішення.

4. Проведено тестування розробленого програмного забезпечення на різноманітних зображеннях, що дозволило оцінити його ефективність та точність у генерації PROMPT.

5. Продемонстровано практичну застосовність розробленого рішення в задачах опису зображень, що може бути використано в різних галузях, таких як маркетинг, реклама, пошукові системи, медицина та інші.

Розроблене програмне забезпечення є гнучким та масштабованим, що дозволяє його подальшу інтеграцію з іншими системами та додатками. Воно може бути корисним інструментом для автоматизації процесу опису зображень, економії часу та ресурсів, а також підвищення точності та релевантності отриманих результатів.

ПЕРЕЛІК ПОСИЛАНЬ

1. Image processing and its future implications - magnimind academy. Magnimind Academy - Launch a new career with our programs. URL: <https://magnimindacademy.com/blog/image-processing-and-its-future-implications/> (date of access: 10.06.2024).
2. CLIP: connecting text and images. OpenAI. URL: <https://openai.com/research/clip> (date of access: 21.04.2024).
3. Active image indexing - meta research | meta research. Meta Research. URL: <https://research.facebook.com/publications/active-image-indexing/> (date of access: 10.06.2024).
4. The future of gaming: ai's transformation - whimsy games. Whimsy Games. URL: <https://whimsygames.co/blog/the-future-of-gaming-ais-transformation/> (date of access: 10.06.2024).
5. Magic3D: High-Resolution Text-to-3D Content Creation. Research at NVIDIA | Advancing the Latest Technology | NVIDIA. URL: <https://research.nvidia.com/labs/dir/magic3d/> (date of access: 10.06.2024).
6. Machine learning-based automated image processing for quality management in industrial Internet of Things / N. Rahmatov et al. International journal of distributed sensor networks. 2019. Vol. 15, no. 10. P. 155014771988355. URL: <https://doi.org/10.1177/1550147719883551> (date of access: 11.06.2024).
7. Sester M., Feng Y., Thiemann F. Building generalization using deep learning. ISPRS - international archives of the photogrammetry, remote sensing and spatial information sciences. 2018. Vol. XLII-4. P. 565–572. URL: <https://doi.org/10.5194/isprs-archives-xlii-4-565-2018> (date of access: 11.06.2024).
8. Wang S., Wang H., Huang L. Adaptive algorithms for multi-armed bandit with composite and anonymous feedback. Proceedings of the AAAI conference on artificial intelligence. 2021. Vol. 35, no. 11. P. 10210–10217. URL: <https://doi.org/10.1609/aaai.v35i11.17224> (date of access: 11.06.2024).
9. The power of natural language processing. Harvard Business Review. URL: <https://hbr.org/2022/04/the-power-of-natural-language-processing> (date of access: 10.06.2024).
10. The art of AI prompt crafting: a comprehensive guide for enthusiasts. OpenAI Developer Forum. URL: <https://community.openai.com/t/the-art-of-ai-prompt-crafting-a-comprehensive-guide-for-enthusiasts/495144> (date of access: 10.06.2024).
11. Horning J. Software fundamentals. ACM SIGSOFT software engineering notes. 2001. Vol. 26, no. 4. P. 91. URL: <https://doi.org/10.1145/505482.505498> (date of access: 11.06.2024).

12. Davis J., Sable T., Devers C. Modern system administration: building and maintaining reliable systems. O'Reilly Media, Incorporated, 2022. 300 p.
13. Processing payments | PortaOne Documentation | May 3rd, 2024 |. PortaOne Documentation. URL: <https://docs.portaone.com/docs/mr114-payments> (date of access: 10.06.2024).
14. Configuring and managing B2B messaging. Moved. URL: <https://docs.oracle.com/en/cloud/saas/supply-chain-and-manufacturing/24a/facmm/manage-confirmation-codes.html> (date of access: 10.06.2024).
15. Sheldon A. Developing cross-platform applications with python: pros and cons. LinkedIn: Log In or Sign Up. URL: <https://www.linkedin.com/pulse/developing-cross-platform-applications-python-pros-cons-amyra-sheldon> (date of access: 10.06.2024).
16. Frank A. Designing Data-Intensive. Independently Published, 2021.
17. Anderson K. Hacking: web application security. Independently Published, 2020.
18. The ultimate guide to python localization. Phrase. URL: <https://phrase.com/blog/posts/python-localization/> (date of access: 10.06.2024).
19. McKee A. Rust vs python: choosing the right language for your data project. Learn Data Science and AI Online | DataCamp. URL: <https://www.datacamp.com/blog/rust-vs-python> (date of access: 10.06.2024).
20. OpenCV: OpenCV modules. OpenCV documentation index. URL: <https://docs.opencv.org/4.10.0/> (date of access: 10.06.2024).
21. Contributors to Wikimedia projects. Feature-driven development - Wikipedia. Wikipedia, the free encyclopedia. URL: https://en.wikipedia.org/wiki/Feature-driven_development (date of access: 10.06.2024).
22. Real Python. Socket programming in python (guide) – real python. Python Tutorials – Real Python. URL: <https://realpython.com/python-sockets/> (date of access: 10.06.2024).
23. Software architecture: the hard parts / M. Richards et al. O'Reilly Media, Incorporated, 2021.
24. freeCodeCamp.org. Python tkinter GUI design using ttkbootstrap - complete course, 2023. YouTube. URL: https://www.youtube.com/watch?v=0tM-1_ZsxjU (date of access: 10.06.2024).
25. How HTTPS Works. How HTTPS works - How HTTPS works. URL: <https://howhttps.works/> (date of access: 10.06.2024).
26. Contributors to Wikimedia projects. Public-key cryptography - Wikipedia. Wikipedia, the free encyclopedia. URL: https://en.wikipedia.org/wiki/Public-key_cryptography (date of access: 10.06.2024).

27. Database-first and model-first - llblgen pro designer v5.8 documentation. LLBLGen Pro | Entity Modeling Solution and ORM framework for .NET. URL: <https://www.llblgen.com/Documentation/5.8/Designer/Concepts/DatabaseFirstModelFirst.htm> (date of access: 10.06.2024).
28. MySQL documentation. Moved. URL: https://docs.oracle.com/cd/E17952_01/index.html (date of access: 10.06.2024).
29. Contributors to Wikimedia projects. Database normalization - Wikipedia. Wikipedia, the free encyclopedia. URL: https://en.wikipedia.org/wiki/Database_normalization (date of access: 10.06.2024).
30. Introduction of ER Model - GeeksforGeeks. GeeksforGeeks. URL: <https://www.geeksforgeeks.org/introduction-of-er-model/> (date of access: 10.06.2024).
31. Database design basics - Microsoft Support. Microsoft Support. URL: <https://support.microsoft.com/en-gb/office/database-design-basics-eb2159cf-1e30-401a-8084-bd4f9c9ca1f5> (date of access: 10.06.2024).
32. Events in MySQL - GeeksforGeeks. GeeksforGeeks. URL: <https://www.geeksforgeeks.org/events-in-mysql/> (date of access: 10.06.2024).
33. How does SSL work? | entrust. Entrust | Identities, secure payments, and protected data. URL: <https://www.entrust.com/resources/learn/how-does-ssl-work> (date of access: 10.06.2024).
34. IBM i 7.4. IBM - United States. URL: <https://www.ibm.com/docs/en/i/7.4?topic=communications-socket-programming> (date of access: 10.06.2024).
35. SSL/TLS-рукописання – що це? Все про SSL/TLS handshake | блог ssl.com.ua. Блог SSL.com.ua. URL: <https://ssl.com.ua/blog/ukr/what-is-ssl-tls-handshake/> (дата звернення: 10.06.2024).
36. What is an SSL Certificate? | DigiCert. SSL Digital Certificate Authority | Encryption & Authentication | DigiCert.com. URL: <https://www.digicert.com/what-is-an-ssl-certificate> (date of access: 10.06.2024).
37. Підключити LiqPay (ЛікПей) до інтернет-магазину | налаштування способів приймання платежів. privatbank.ua. URL: <https://privatbank.ua/business/business-connect-liqpay> (дата звернення: 10.06.2024).
38. GitHub - liqpay/sdk-python: LiqPay python sdk. GitHub. URL: <https://github.com/liqpay/sdk-python> (date of access: 10.06.2024).
39. Contributors to Wikimedia projects. Business process - Wikipedia. Wikipedia, the free encyclopedia. URL: https://en.wikipedia.org/wiki/Business_process (date of access: 10.06.2024).
40. Engineering systems design rhapsody. IBM - United States. URL: <https://www.ibm.com/docs/en/engineering-lifecycle-management-suite/design-rhapsody/9.0.1?topic=rhapsody-structural-model> (date of access: 10.06.2024).

41. Pyreverse - Pylint 3.3.0-dev0 documentation. Pylint 3.2.3 documentation. URL: <https://pylint.readthedocs.io/en/latest/pyreverse.html> (date of access: 10.06.2024).
42. What is the most effective way to design a user interface for a desktop application?. LinkedIn: Log In or Sign Up. URL: <https://www.linkedin.com/advice/3/what-most-effective-way-design-user-interface-desktop> (date of access: 10.06.2024).
43. VS code counter - visual studio marketplace. Extensions for Visual Studio family of products | Visual Studio Marketplace. URL: <https://marketplace.visualstudio.com/items?itemName=uctakeoff.vscode-counter> (date of access: 10.06.2024).
44. Paper D. Advanced transfer learning. State-of-the-Art deep learning models in tensorflow. Berkeley, CA, 2021. P. 171–199. URL: https://doi.org/10.1007/978-1-4842-7341-8_7 (date of access: 21.04.2024).
45. Леонов О. Автоматичне тестування ПЗ (визначення, процес створення). Друкарня. URL: <https://drukarnia.com.ua/articles/avtomatichne-testuvannya-pz-viznachennya-proces-stvorenniya-WWNqn> (дата звернення: 21.04.2024).
46. Design Patterns: Elements of Reusable Object-Oriented Software (Addison-Wesley Professional Computing Series) / R. Helm et al. Addison-Wesley Professional, 1995. 395 p.
47. Flash attention compilation warning?. PyTorch Forums. URL: <https://discuss.pytorch.org/t/flash-attention-compilation-warning/196692> (date of access: 10.06.2024).
48. Pytest: helps you write better programs – pytest documentation. pytest: helps you write better programs – pytest documentation. URL: <https://docs.pytest.org/en/8.0.x/> (date of access: 21.04.2024).
49. Тестування веб-проектів: основні етапи та поради - QALight. QALight. URL: <https://qalight.ua/baza-znaniy/testuvannya-veb-proektiv-osnovni-etapi-ta-poradi/> (дата звернення: 21.04.2024).
50. Axelrod A. Complete guide to test automation: techniques, practices, and patterns for building and maintaining effective software projects. Apress, 2018. 560 p.
51. Mohan G. Full stack testing: a practical guide for delivering high quality software. O'Reilly Media, Incorporated, 2022.
52. McLeod R. J., Everett G. D. Software testing: testing across the entire software development life cycle. Wiley-IEEE Computer Society Pr, 2007. 280 p.
53. Urma R.-G., Warburton R. Real-World software development. O'Reilly Media, Incorporated, 2019. 325 p.
54. Wiegers K. E. Software development pearls: lessons from fifty years of software experience. Pearson Education, Limited, 2000.

55. Singureanu C. Динамічне тестування програмного забезпечення - типи, процес, інструменти та інше!. ZAPTEST. URL: <https://www.zaptest.com/uk/динамічне-тестування-в-тестуванні-пр> (дата звернення: 21.04.2024).
56. Ramalho L. Fluent Python: Clear, Concise, and Effective Programming. O'Reilly Media, 2021. 850 p.
57. Pytest-qt. PyPI. URL: <https://pypi.org/project/pytest-qt/> (date of access: 10.06.2024).
58. Що таке тестування програмного забезпечення: види, етапи, інструменти. Sigma Software University. URL: <https://university.sigma.software/what-is-software-testing/> (дата звернення: 21.04.2024).
59. Beck K. Test driven development: by example (the addison-wesley signature series). Addison-Wesley Professional, 2002. 240 p.
60. Siddiqui S. Learning test-driven development: a polyglot guide to writing uncluttered code. O'Reilly Media, Incorporated, 2021. 275 p.
61. Forgács I., Kovács A. Modern software testing techniques. Berkeley, CA : Apress, 2024. URL: <https://doi.org/10.1007/978-1-4842-9893-0> (date of access: 21.04.2024).
62. McLeod R. J., Everett G. D. Software testing: testing across the entire software development life cycle. Wiley-IEEE Computer Society Pr, 2007. 280 p.
63. Стандартизована інформація. Задачева незмінність стандартів ЄС. URL: https://www.ukrstat.gov.ua/operativ/oper_new.html (date of access: 11.06.2024).
64. Лише 14% українців використовують штучний інтелект. Інтернет Свобода. URL: <https://netfreedom.org.ua/article/lishe-14-ukrayinciv-vikoristovuyut-shtuchnij-intelekt#:~:text=Лише%2014%%20українців%20використовують%20штучний,%20Інтернет%20Свобода> (дата звернення: 10.06.2024).
65. Майже 23 млн українців регулярно користуються Інтернетом – дослідження. Mind.ua. URL: <https://mind.ua/news/20204323-majzhe-23-mln-ukrayinciv-regulyarno-koristuyutsya-internetom-doslidzhennya> (дата звернення: 11.06.2024).
66. Population clock. Census.gov. URL: <https://www.census.gov/popclock/> (date of access: 10.06.2024).
67. Ballard J. Americans' top feeling about AI: caution | YouGov. YouGov | What the world thinks. URL: <https://today.yougov.com/technology/articles/49099-americans-2024-poll-ai-top-feeling-caution> (date of access: 10.06.2024).
68. Доллар США (USD) - Гривна (UAH) ➤ Перевести доллар в гривну. Мінфін - все про фінанси: новини, курси валют, банки. URL: <https://minfin.com.ua/currency/converter/usd-uah/> (дата звернення: 10.06.2024).

69. Діючі | ТОВ «ПОЛТАВАЕНЕРГОЗБУТ». ТОВ «ПОЛТАВАЕНЕРГОЗБУТ» | постачання електричної енергії споживачу. URL: <https://www.energo.pl.ua/diyuchi/> (дата звернення: 11.06.2024).

ДОДАТОК А – КОД ПРОГРАМИ

A.1. clip_interrogator.py

```

import hashlib
import math
import numpy as np
import open_clip
import os
import requests
import time
import torch

from dataclasses import dataclass
from PIL import Image
from transformers import AutoProcessor, AutoModelForCausalLM, BlipForConditionalGeneration,
Blip2ForConditionalGeneration
from tqdm import tqdm
from typing import List, Optional

from safetensors.numpy import load_file, save_file

CAPTION_MODELS = {
    'blip-base': 'Salesforce/blip-image-captioning-base', # 990MB
    'blip-large': 'Salesforce/blip-image-captioning-large', # 1.9GB
    'blip2-2.7b': 'Salesforce/blip2-opt-2.7b',          # 15.5GB
    'blip2-flan-t5-xl': 'Salesforce/blip2-flan-t5-xl',   # 15.77GB
    'git-large-coco': 'microsoft/git-large-coco',        # 1.58GB
}

```



```

CACHE_URL_BASE = 'https://huggingface.co/pharma/ci-preprocess/resolve/main/'

@dataclass
class Config:
    # models can optionally be passed in directly
    caption_model = None
    caption_processor = None
    clip_model = None
    clip_preprocess = None

    # blip settings
    caption_max_length: int = 32
    caption_model_name: Optional[str] = 'blip-large' # use a key from CAPTION_MODELS or None
    caption_offload: bool = False

    # clip settings
    clip_model_name: str = 'ViT-L-14/openai'
    clip_model_path: Optional[str] = None
    clip_offload: bool = False

    # interrogator settings
    cache_path: str = 'cache' # path to store cached text embeddings
    download_cache: bool = True # when true, cached embeds are downloaded from huggingface
    chunk_size: int = 2048 # batch size for CLIP, use smaller for lower VRAM
    data_path: str = os.path.join(os.path.dirname(__file__), 'data')
    device: str = ("mps" if torch.backends.mps.is_available() else "cuda" if torch.cuda.is_available() else
"cpu")
    flavor_intermediate_count: int = 2048
    quiet: bool = False # when quiet progress bars are not shown

```



```

def apply_low_vram_defaults(self):
    self.caption_model_name = 'blip-base'
    self.caption_offload = True
    self.clip_offload = True
    self.chunk_size = 1024
    self.flavor_intermediate_count = 1024

class Interrogator():
    def __init__(self, config: Config):
        self.config = config
        self.device = config.device
        self.dtype = torch.float16 if self.device == 'cuda' else torch.float32
        self.caption_offloaded = True
        self.clip_offloaded = True
        self.load_caption_model()
        self.load_clip_model()

    def load_caption_model(self):
        if self.config.caption_model is None and self.config.caption_model_name:
            if not self.config.quiet:
                print(f"Loading caption model {self.config.caption_model_name}...")

            model_path = CAPTION_MODELS[self.config.caption_model_name]
            if self.config.caption_model_name.startswith('git-'):
                caption_model = AutoModelForCausalLM.from_pretrained(model_path,
torch_dtype=torch.float32)
            elif self.config.caption_model_name.startswith('blip2-'):
                caption_model = Blip2ForConditionalGeneration.from_pretrained(model_path,
torch_dtype=self.dtype)
            else:

```

```

        caption_model = BlipForConditionalGeneration.from_pretrained(model_path,
torch_dtype=self.dtype)

        self.caption_processor = AutoProcessor.from_pretrained(model_path)

    caption_model.eval()

    if not self.config.caption_offload:
        caption_model = caption_model.to(self.config.device)
        self.caption_model = caption_model
    else:
        self.caption_model = self.config.caption_model
        self.caption_processor = self.config.caption_processor

def load_clip_model(self):
    start_time = time.time()

    config = self.config

    clip_model_name, clip_model_pretrained_name = config.clip_model_name.split('/', 2)

    if config.clip_model is None:
        if not config.quiet:
            print(f"Loading CLIP model {config.clip_model_name}...")

        self.clip_model, _, self.clip_preprocess = open_clip.create_model_and_transforms(
            clip_model_name,
            pretrained=clip_model_pretrained_name,
            precision='fp16' if config.device == 'cuda' else 'fp32',
            device=config.device,
            jit=False,
            cache_dir=config.clip_model_path
        )

```

```

        self.clip_model.eval()
    else:
        self.clip_model = config.clip_model
        self.clip_preprocess = config.clip_preprocess
    self.tokenize = open_clip.get_tokenizer(clip_model_name)

    sites = ['Artstation', 'behance', 'cg society', 'cgsociety', 'deviantart', 'dribble',
            'flickr', 'instagram', 'pexels', 'pinterest', 'pixabay', 'pixiv', 'polycount',
            'reddit', 'shutterstock', 'tumblr', 'unsplash', 'zbrush central']
    trending_list = [site for site in sites]
    trending_list.extend(["trending on "+site for site in sites])
    trending_list.extend(["featured on "+site for site in sites])
    trending_list.extend([site+" contest winner" for site in sites])

    raw_artists = load_list(config.data_path, 'artists.txt')
    artists = [f"by {a}" for a in raw_artists]
    artists.extend([f"inspired by {a}" for a in raw_artists])

    self._prepare_clip()
    self.artists = LabelTable(artists, "artists", self)
    self.flavors = LabelTable(load_list(config.data_path, 'flavors.txt'), "flavors", self)
    self.mediums = LabelTable(load_list(config.data_path, 'mediums.txt'), "mediums", self)
    self.movements = LabelTable(load_list(config.data_path, 'movements.txt'), "movements", self)
    self.trendings = LabelTable(trending_list, "trendings", self)
    self.negative = LabelTable(load_list(config.data_path, 'negative.txt'), "negative", self)

    end_time = time.time()
    if not config.quiet:
        print(f"Loaded CLIP model and data in {end_time-start_time:.2f} seconds.")

```

```

def chain(
    self,
    image_features: torch.Tensor,
    phrases: List[str],
    best_prompt: str="",
    best_sim: float=0,
    min_count: int=8,
    max_count: int=32,
    desc="Chaining",
    reverse: bool=False
) -> str:
    self._prepare_clip()

    phrases = set(phrases)
    if not best_prompt:
        best_prompt = self.rank_top(image_features, [f for f in phrases], reverse=reverse)
        best_sim = self.similarity(image_features, best_prompt)
        phrases.remove(best_prompt)
    curr_prompt, curr_sim = best_prompt, best_sim

    def check(addition: str, idx: int) -> bool:
        nonlocal best_prompt, best_sim, curr_prompt, curr_sim
        prompt = curr_prompt + ", " + addition
        sim = self.similarity(image_features, prompt)
        if reverse:
            sim = -sim

        if sim > best_sim:
            best_prompt, best_sim = prompt, sim
        if sim > curr_sim or idx < min_count:

```

```

        curr_prompt, curr_sim = prompt, sim
        return True
    return False

for idx in tqdm(range(max_count), desc=desc, disable=self.config.quiet):
    best = self.rank_top(image_features, [f"{curr_prompt}, {f}" for f in phrases], reverse=reverse)
    flave = best[len(curr_prompt)+2:]
    if not check(flave, idx):
        break
    if _prompt_at_max_len(curr_prompt, self.tokenize):
        break
    phrases.remove(flave)

return best_prompt

def generate_caption(self, pil_image: Image) -> str:
    assert self.caption_model is not None, "No caption model loaded."
    self._prepare_caption()
    inputs = self.caption_processor(images=pil_image, return_tensors="pt").to(self.device)
    if not self.config.caption_model_name.startswith('git-'):
        inputs = inputs.to(self.dtype)
    tokens = self.caption_model.generate(**inputs, max_new_tokens=self.config.caption_max_length)
    return self.caption_processor.batch_decode(tokens, skip_special_tokens=True)[0].strip()

def image_to_features(self, image: Image) -> torch.Tensor:
    self._prepare_clip()
    images = self.clip_preprocess(image).unsqueeze(0).to(self.device)
    with torch.no_grad(), torch.cuda.amp.autocast():
        image_features = self.clip_model.encode_image(images)
        image_features /= image_features.norm(dim=-1, keepdim=True)

```

```
return image_features
```

```
def interrogate_classic(self, image: Image, max_flavors: int=3, caption: Optional[str]=None) -> str:
```

```
    """Classic mode creates a prompt in a standard format first describing the image,
    then listing the artist, trending, movement, and flavor text modifiers."""
```

```
    caption = caption or self.generate_caption(image)
```

```
    image_features = self.image_to_features(image)
```

```
    medium = self.mediums.rank(image_features, 1)[0]
```

```
    artist = self.artists.rank(image_features, 1)[0]
```

```
    trending = self.trendings.rank(image_features, 1)[0]
```

```
    movement = self.movements.rank(image_features, 1)[0]
```

```
    flaves = ", ".join(self.flavors.rank(image_features, max_flavors))
```

```
    if caption.startswith(medium):
```

```
        prompt = f"{caption} {artist}, {trending}, {movement}, {flaves}"
```

```
    else:
```

```
        prompt = f"{caption}, {medium} {artist}, {trending}, {movement}, {flaves}"
```

```
    return _truncate_to_fit(prompt, self.tokenize)
```

```
def interrogate_fast(self, image: Image, max_flavors: int=32, caption: Optional[str]=None) -> str:
```

```
    """Fast mode simply adds the top ranked terms after a caption. It generally results in
    better similarity between generated prompt and image than classic mode, but the prompts
    are less readable."""
```

```
    caption = caption or self.generate_caption(image)
```

```
    image_features = self.image_to_features(image)
```

```
    merged = _merge_tables([self.artists, self.flavors, self.mediums, self.movements, self.trendings],
```

```
self)
```

```
    tops = merged.rank(image_features, max_flavors)
```

```
return _truncate_to_fit(caption + ", " + ", ".join(tops), self.tokenize)
```

```
def interrogate_negative(self, image: Image, max_flavors: int = 32) -> str:
```

```
    """Negative mode chains together the most dissimilar terms to the image. It can be used
    to help build a negative prompt to pair with the regular positive prompt and often
    improve the results of generated images particularly with Stable Diffusion 2."""
```

```
    image_features = self.image_to_features(image)
```

```
    flaves = self.flavors.rank(image_features, self.config.flavor_intermediate_count, reverse=True)
```

```
    flaves = flaves + self.negative.labels
```

```
    return self.chain(image_features, flaves, max_count=max_flavors, reverse=True, desc="Negative
chain")
```

```
def interrogate(self, image: Image, min_flavors: int=8, max_flavors: int=32, caption:
Optional[str]=None) -> str:
```

```
    caption = caption or self.generate_caption(image)
```

```
    image_features = self.image_to_features(image)
```

```
    merged = _merge_tables([self.artists, self.flavors, self.mediums, self.movements, self.trendings],
self)
```

```
    flaves = merged.rank(image_features, self.config.flavor_intermediate_count)
```

```
    best_prompt, best_sim = caption, self.similarity(image_features, caption)
```

```
    best_prompt = self.chain(image_features, flaves, best_prompt, best_sim, min_count=min_flavors,
max_count=max_flavors, desc="Flavor chain")
```

```
    fast_prompt = self.interrogate_fast(image, max_flavors, caption=caption)
```

```
    classic_prompt = self.interrogate_classic(image, max_flavors, caption=caption)
```

```
    candidates = [caption, classic_prompt, fast_prompt, best_prompt]
```

```
    return candidates[np.argmax(self.similarities(image_features, candidates))]
```

```
def rank_top(self, image_features: torch.Tensor, text_array: List[str], reverse: bool=False) -> str:
```

```
    self._prepare_clip()
```

```

text_tokens = self.tokenize([text for text in text_array]).to(self.device)
with torch.no_grad(), torch.cuda.amp.autocast():
    text_features = self.clip_model.encode_text(text_tokens)
    text_features /= text_features.norm(dim=-1, keepdim=True)
    similarity = text_features @ image_features.T
    if reverse:
        similarity = -similarity
return text_array[similarity.argmax().item()]

def similarity(self, image_features: torch.Tensor, text: str) -> float:
    self._prepare_clip()
    text_tokens = self.tokenize([text]).to(self.device)
    with torch.no_grad(), torch.cuda.amp.autocast():
        text_features = self.clip_model.encode_text(text_tokens)
        text_features /= text_features.norm(dim=-1, keepdim=True)
        similarity = text_features @ image_features.T
    return similarity[0][0].item()

def similarities(self, image_features: torch.Tensor, text_array: List[str]) -> List[float]:
    self._prepare_clip()
    text_tokens = self.tokenize([text for text in text_array]).to(self.device)
    with torch.no_grad(), torch.cuda.amp.autocast():
        text_features = self.clip_model.encode_text(text_tokens)
        text_features /= text_features.norm(dim=-1, keepdim=True)
        similarity = text_features @ image_features.T
    return similarity.T[0].tolist()

def _prepare_caption(self):
    if self.config.clip_offload and not self.clip_offloaded:
        self.clip_model = self.clip_model.to('cpu')

```



```

        self.clip_offloaded = True
    if self.caption_offloaded:
        self.caption_model = self.caption_model.to(self.device)
        self.caption_offloaded = False

    def _prepare_clip(self):
        if self.config.caption_offload and not self.caption_offloaded:
            self.caption_model = self.caption_model.to('cpu')
            self.caption_offloaded = True
        if self.clip_offloaded:
            self.clip_model = self.clip_model.to(self.device)
            self.clip_offloaded = False

class LabelTable():
    def __init__(self, labels:List[str], desc:str, ci: Interrogator):
        clip_model, config = ci.clip_model, ci.config
        self.chunk_size = config.chunk_size
        self.config = config
        self.device = config.device
        self.embeds = []
        self.labels = labels
        self.tokenize = ci.tokenize

        hash = hashlib.sha256(".".join(labels).encode()).hexdigest()
        sanitized_name = self.config.clip_model_name.replace('/', '_').replace('@', '_')
        self._load_cached(desc, hash, sanitized_name)

        if len(self.labels) != len(self.embeds):
            self.embeds = []
            chunks = np.array_split(self.labels, max(1, len(self.labels)/config.chunk_size))

```

```

        for chunk in tqdm(chunks, desc=f"Preprocessing {desc}" if desc else None,
                           disable=self.config.quiet):
            text_tokens = self.tokenize(chunk).to(self.device)
            with torch.no_grad(), torch.cuda.amp.autocast():
                text_features = clip_model.encode_text(text_tokens)
                text_features /= text_features.norm(dim=-1, keepdim=True)
                text_features = text_features.half().cpu().numpy()
            for i in range(text_features.shape[0]):
                self.embeds.append(text_features[i])

    if desc and self.config.cache_path:
        os.makedirs(self.config.cache_path, exist_ok=True)
        cache_filepath = os.path.join(self.config.cache_path, f"{sanitized_name}_{desc}.safetensors")
        tensors = {
            "embeds": np.stack(self.embeds),
            "hash": np.array([ord(c) for c in hash], dtype=np.int8)
        }
        save_file(tensors, cache_filepath)

    if self.device == 'cpu' or self.device == torch.device('cpu'):
        self.embeds = [e.astype(np.float32) for e in self.embeds]

def _load_cached(self, desc:str, hash:str, sanitized_name:str) -> bool:
    if self.config.cache_path is None or desc is None:
        return False

    cached_safetensors = os.path.join(self.config.cache_path, f"{sanitized_name}_{desc}.safetensors")

    if self.config.download_cache and not os.path.exists(cached_safetensors):
        download_url = CACHE_URL_BASE + f"{sanitized_name}_{desc}.safetensors"

```

```

try:
    os.makedirs(self.config.cache_path, exist_ok=True)
    _download_file(download_url, cached_safetensors, quiet=self.config.quiet)
except Exception as e:
    print(f"Failed to download {download_url}")
    print(e)
    return False

if os.path.exists(cached_safetensors):
    try:
        tensors = load_file(cached_safetensors)
    except Exception as e:
        print(f"Failed to load {cached_safetensors}")
        print(e)
        return False

    if 'hash' in tensors and 'embeds' in tensors:
        if np.array_equal(tensors['hash'], np.array([ord(c) for c in hash], dtype=np.int8)):
            self.embeds = tensors['embeds']
            if len(self.embeds.shape) == 2:
                self.embeds = [self.embeds[i] for i in range(self.embeds.shape[0])]
            return True

    return False

def _rank(self, image_features: torch.Tensor, text_embeds: torch.Tensor, top_count: int=1, reverse:
bool=False) -> str:
    top_count = min(top_count, len(text_embeds))
    text_embeds = torch.stack([torch.from_numpy(t) for t in text_embeds]).to(self.device)
    with torch.cuda.amp.autocast():
        similarity = image_features @ text_embeds.T

```

```

        if reverse:
            similarity = -similarity
        _, top_labels = similarity.float().cpu().topk(top_count, dim=-1)
        return [top_labels[0][i].numpy() for i in range(top_count)]

def rank(self, image_features: torch.Tensor, top_count: int=1, reverse: bool=False) -> List[str]:
    if len(self.labels) <= self.chunk_size:
        tops = self._rank(image_features, self.embeds, top_count=top_count, reverse=reverse)
        return [self.labels[i] for i in tops]

    num_chunks = int(math.ceil(len(self.labels)/self.chunk_size))
    keep_per_chunk = int(self.chunk_size / num_chunks)

    top_labels, top_embeds = [], []
    for chunk_idx in tqdm(range(num_chunks), disable=self.config.quiet):
        start = chunk_idx*self.chunk_size
        stop = min(start+self.chunk_size, len(self.embeds))

        tops = self._rank(image_features, self.embeds[start:stop], top_count=keep_per_chunk,
reverse=reverse)

        top_labels.extend([self.labels[start+i] for i in tops])
        top_embeds.extend([self.embeds[start+i] for i in tops])

    tops = self._rank(image_features, top_embeds, top_count=top_count)
    return [top_labels[i] for i in tops]

def _download_file(url: str, filepath: str, chunk_size: int = 4*1024*1024, quiet: bool = False):
    r = requests.get(url, stream=True)
    if r.status_code != 200:
        return

```

```

file_size = int(r.headers.get("Content-Length", 0))
filename = url.split("/")[-1]
progress = tqdm(total=file_size, unit="B", unit_scale=True, desc=filename, disable=quiet)
with open(filepath, "wb") as f:
    for chunk in r.iter_content(chunk_size=chunk_size):
        if chunk:
            f.write(chunk)
            progress.update(len(chunk))
progress.close()

def _merge_tables(tables: List[LabelTable], ci: Interrogator) -> LabelTable:
    m = LabelTable([], None, ci)
    for table in tables:
        m.labels.extend(table.labels)
        m.embeds.extend(table.embeds)
    return m

def _prompt_at_max_len(text: str, tokenize) -> bool:
    tokens = tokenize([text])
    return tokens[0][-1] != 0

def _truncate_to_fit(text: str, tokenize) -> str:
    parts = text.split(', ')
    new_text = parts[0]
    for part in parts[1:]:
        if _prompt_at_max_len(new_text + part, tokenize):
            break
        new_text += ', ' + part
    return new_text

```

```

def list_caption_models() -> List[str]:
    return list(CAPTION_MODELS.keys())

def list_clip_models() -> List[str]:
    return ['/'.join(x) for x in open_clip.list_pretrained()]

def load_list(data_path: str, filename: Optional[str] = None) -> List[str]:
    """Load a list of strings from a file."""
    if filename is not None:
        data_path = os.path.join(data_path, filename)
    with open(data_path, 'r', encoding='utf-8', errors='replace') as f:
        items = [line.strip() for line in f.readlines()]
    return items

def download_model_gui(model: str, result: list) -> None:
    try:
        config = Config()
        config.clip_model_name = model
        config.clip_model_path = os.path.dirname(os.path.realpath(__file__)) + "\\models"
        clip_model_name, clip_model_pretrained_name = config.clip_model_name.split('/', 2)
        clip_model, _, clip_preprocess = open_clip.create_model_and_transforms(
            clip_model_name,
            pretrained=clip_model_pretrained_name,
            precision='fp16' if config.device == 'cuda' else 'fp32',
            device=config.device,
            jit=False,
            cache_dir=config.clip_model_path
        )
        result.append(True)
    except Exception as e:

```

```

        result.append(False)

if __name__ == '__main__':
    #download_model_gui("ViT-L-14/datacomp_xl_s13b_b90k")
    print("Hi!")

```

A.2. database.py

```

import mysql.connector
from mysql.connector import Error
import json
from my_logger import setup_logger
from hashlib import sha256
from datetime import date, timedelta, datetime
import bcrypt
from random import randint

class Database:
    def __init__(self, config_file) -> None:
        """
        Initialize the Database class.

        Args:
            config_file (str): The path to the JSON configuration file.

        This function initializes the Database class by loading the database configuration
        from a JSON file and setting up logging.
        """
        self.host = None
        self.user = None
        self.password = None

```

```

self.database = None

self.connection = None

self.logger = setup_logger("DatabaseLogger", "server_side_logs.log")

self.load_config(config_file)

def load_config(self, config_file) -> None:
    """
    Load database configuration from a JSON file.

    Args:
        config_file (str): The path to the JSON configuration file.

    This function reads the database configuration from a JSON file and sets the
    'host', 'user', 'password', and 'database' attributes accordingly.
    """
    try:
        with open(config_file, 'r') as file:
            config = json.load(file)
            self.host = config.get('host')
            self.user = config.get('user')
            self.password = config.get('password')
            self.database = config.get('database')
    except Exception as e:
        self.logger.error(f"Error loading configuration from JSON file: {e}")

def add_new_user(self, username: str, password_hash: str) -> bool:
    """
    Add a new user to the database.

```


Args:

username (str): The username of the new user.

password_hash (str): The hashed password for the new user.

Returns:

bool: True if the new user was added successfully, False otherwise.

This function adds a new user to the 'users' table in the database and creates a corresponding record in the 'Credits' table. It performs the following steps:

1. Inserts a new row into the 'users' table with the provided username and password hash.
2. Logs a message indicating the new user's login and password hash.
3. Inserts a new row into the 'Credits' table with the new user's username.
4. Logs a message indicating that a record was added to the credit system for the new user.
5. Returns True if both insertions were successful, False otherwise.

If an error occurs during the execution of the SQL queries, it logs an error message with the query and the error details, and returns False.

"""

try:

Insert the new user into the 'users' table

query = "INSERT INTO users (username, password) VALUES (%s, %s)"

self.execute_query(query, (username, password_hash))

self.logger.info(f"Added new user with Login: {username} and Pass_Hash: {password_hash}")

Add a record for the new user in the 'Credits' table

query = "INSERT INTO Credits(username) VALUES (%s)"

self.execute_query(query, (username,))

self.logger.info(f"Added record in credit system for user: {username}")

```
return True
```

```
except Error as e:
```

```
    self.logger.error(f"Error executing adding new user. Query - {query}: {e}")
```

```
return False
```

```
def get_models(self) -> list:
```

```
    """
```

Retrieve a list of models from the database.

Returns:

list: A list of models retrieved from the database.

This function retrieves a list of models from the 'models' table in the database and returns them as a list.

```
    """
```

```
try:
```

```
    query = "SELECT * from models"
```

```
    models = self.execute_query(query)
```

```
    return models
```

```
except Error as e:
```

```
    self.logger.error(f"Error executing get_models(). Query - {query}: {e}")
```

```
    return []
```

```
def check_login_credentials(self, username: str, password: str) -> bool | str:
```

```
    """
```

Check the login credentials of a user.

Args:

username (str): The username of the user.

password (str): The password of the user.

Returns:

bool: True if the login credentials are valid, False otherwise.

This function checks the provided username and password against the 'users' table in the database. It returns True if the credentials are valid, False otherwise.

"""

try:

```
query = "SELECT password FROM users WHERE username = %s"
```

```
results = self.execute_query(query, (username,))
```

```
if type(results) == list:
```

```
    stored_hash = results[0][0].encode('utf-8')
```

```
    if bcrypt.checkpw(password.encode('utf-8'), stored_hash):
```

```
        self.logger.info(f"Login credentials for user {username} verified.")
```

```
        return True
```

```
    else:
```

```
        self.logger.info(f"Login credentials for user {username} not verified. Password hashes aren't the same.")
```

```
        return False
```

```
else:
```

```
    self.logger.info(f"Login credentials for user {username} not verified. No record found")
```

```
    return False
```

```
except Error as e:
```

```
    self.logger.error(f"Error executing login credentials check for user {username}: {e}")
```

```
    return False
```

```
def remove_auth_token(self, token: str) -> bool:
```

"""

Remove an authentication token from the database.

Args:

token (str): The authentication token to remove.

Returns:

bool: True if the token was removed successfully, False otherwise.

This function removes the specified authentication token from the 'auth_tokens' table in the database. It returns True if the token was removed successfully, False otherwise.

"""

try:

query = "DELETE FROM auth_tokens WHERE token = %s"

if self.execute_query(query, (token,)):

return True

else:

return False

except Error as e:

self.logger.error(f"Error executing remove_auth_token(): {e}")

return False

create table for this function

def add_auth_token(self, username: str, password: str) -> str:

"""

Add an authentication token for a user.

Args:

username (str): The username of the user.

password (str): The password of the user.

Returns:

str or None: The authentication token if added successfully, None otherwise.

This function generates an authentication token for the user based on the provided username and password. It inserts the token into the 'auth_tokens' table in the database and returns the token if successful, None otherwise.

"""

try:

 today_date = date.today().strftime('%Y-%m-%d')

 reference_string = username + password + today_date

 hashed_token = sha256(reference_string.encode()).hexdigest()

 query = "INSERT INTO auth_tokens (token, start_date, end_date, username) VALUES (%s,%s,%s,%s) ON DUPLICATE KEY UPDATE username=username"

 if self.execute_query(query, (hashed_token, today_date, (date.today() + timedelta(days=21)).strftime('%Y-%m-%d'), username)):

 return hashed_token

 else:

 return None

except Error as e:

 self.logger.error(f"Something went wrong during add_auth_token function. Error - {e}")

def verify_auth_token(self, token: str) -> bool:

"""

Verify the authenticity of an authentication token.

Args:

token (str): The authentication token to verify.

Returns:

bool: True if the token is authentic, False otherwise.

This function checks if the provided authentication token exists in the 'auth_tokens' table in the database. It returns True if the token is found, indicating its authenticity, False otherwise.

```

"""
try:
    query = "SELECT COUNT(*) FROM auth_tokens WHERE token = %s"
    results = self.execute_query(query, (token,))
    if results[0][0]:
        self.logger.info('verify_auth_token executed succesfully!')
        return True
    else: return False
except Error as e:
    self.logger.error(f"Something went wrong during verify_auth_token function. Error - {e}")

```

def check_login_exists(self, username: str) -> bool:

"""

Check the login credentials of a user.

Args:

username (str): The username of the user.

password (str): The password of the user.

Returns:

bool: True if the login credentials are valid, False otherwise.

This function checks the provided username and password against the 'users' table in the database. It returns True if the credentials are valid, False otherwise.

```

"""

try:
    query = "SELECT username FROM users WHERE username = %s"
    results = self.execute_query(query, (username,))
    if not results:
        self.logger.info('Check_login_exists executed succesfully!')
        return True
    else:
        return False
except Error as e:
    self.logger.error(f'Something went wrong during Check_login_exists function. Error - {e}')

# i need subscription table for monitoring subscriptions of users
# userid + subscription id + beginning date + cancelling date
def add_user_premium_status(self, username: str, end_date: str) -> bool:
    try:
        query = "INSERT INTO Subscribers (username, end_date) VALUES (%s, %s)"
        results = self.execute_query(query, (username, end_date))
        self.logger.info('add_user_premium_status executed succesfully!')
        if results:
            return True
        else:
            return False
    except Error as e:
        self.logger.error(f'Something went wrong during add_user_premium_status function. Error - {e}')

# for this i need to check monobank api
def add_payment(self, username: str, payment_date: str, promotion_id: str) -> bool:
    try:
        query = "INSERT INTO add_payment (username, end_date) VALUES (%s, %s)"

```

```

        results = self.execute_query(query, (username, payment_date, promotion_id))
        self.logger.info('add_payment executed succesfully!')

        if results:
            return True
        else:
            return False

    except Error as e:
        self.logger.error(f"Something went wrong during add_payment function. Error - {e}")

def delete_promotion(self, promotion_id) -> bool:
    try:
        query = "DELETE FROM promotions where promotion_id = %s"
        result = self.execute_query(query, (promotion_id, ))
        if result:
            self.logger.info('delete_promotion executed succesfully!')
            return True
        else:
            return False
    except Error as e:
        self.logger.error(f"Something went wrong during delete_promotion function. Error - {e}")

def insert_promotion(self, subscription_length: str, cost: str, description: str) -> bool:
    try:
        query = "INSERT INTO promotions (subscription_length, cost, description) VALUES (%s, %s, %s)"
        result = self.execute_query(query, (subscription_length, cost, description))
        if result:
            self.logger.info('insert_promotion executed succesfully!')
            return True
        else:

```



```

        return False

    except Error as e:

        self.logger.error(f"Something went wrong during insert_promotion function. Error - {e}")

def delete_user(self, username: str) -> bool:

    try:

        tables = ("auth_tokens", "confirmation_codes", "credits", "payments", "requests", "subscribers",
"users")

        results = []

        for table in tables:

            query = f"DELETE FROM {table} where username = '{username}'"

            result = self.execute_query(query)

            results.append(result)

        if all(results):

            self.logger.info('delete_user executed succesfully!')

            return True

        else:

            return False

    except Error as e:

        self.logger.error(f"Something went wrong during delete_user function. Error - {e}")

def delete_payment(self, payment_id: str) -> bool:

    try:

        query = "DELETE FROM payments where paymentID = %s"

        result = self.execute_query(query, (payment_id, ))

        if result:

            self.logger.info('delete_payment executed succesfully!')

            return True

        else:

            return False

```

```
except Error as e:
```

```
    self.logger.error(f"Something went wrong during delete_payment function. Error - {e}")
```

```
def add_request(self, username: str, requestText: str) -> bool:
```

```
    """
```

Add a request to the database.

Args:

username (str): The username of the user making the request.

requestText (str): The content of the request.

Returns:

bool: True if the request was added successfully, False otherwise.

This function inserts a new request into the 'requests' table in the database with the provided username and request text. It returns True if the insertion was successful, False otherwise.

```
    """
```

```
    try:
```

```
        query = "INSERT INTO Requests (username, requestText) VALUES (%s, %s)"
```

```
        results = self.execute_query(query, (username, requestText))
```

```
        if results:
```

```
            self.logger.info('add_request executed succesfully!')
```

```
            return True
```

```
        else:
```

```
            return False
```

```
    except Error as e:
```

```
        self.logger.error(f"Something went wrong during add_request function. Error - {e}")
```

```
def verify_premium_status(self, username: str) -> str | bool:
```

"""

Verify the premium status of a user.

Args:

username (str): The username of the user.

Returns:

str or bool: The end date of the premium subscription if active, False otherwise.

This function checks the premium status of the user specified by the username by querying the 'Subscribers' table in the database. If the user has an active premium subscription, it returns the end date of the subscription; otherwise, it returns False.

"""

try:

query = "SELECT end_date FROM Subscribers WHERE username = %s"

results = self.execute_query(query, (username,))

if results:

self.logger.info('verify_premium_status executed succesfully!')

return results[0][0]

else: return False

except Error as e:

self.logger.error(f"Something went wrong during verify_premium_status function. Error - {e}")

def get_subscriptions(self, locale) -> str:

try:

if locale == 'EN':

query = "SELECT subscription_length, cost, description from promotions"

else:

query = "SELECT subscription_length_ua, cost, description_ua from promotions"

```

        results = self.execute_query(query, )
    if results:
        self.logger.info('get_subscription executed succesfully!')
        return results
    else: return False
except Error as e:
    self.logger.error(f'Something went wrong during get_subscription function. Error - {e}')

def reset_password(self, username: str, new_password: str) -> bool:
    try:
        query = "UPDATE Users SET password = %s WHERE username = %s"
        results = self.execute_query(query, (username, new_password))
        if results:
            self.logger.info('reset_password executed succesfully!')
            return results
        else: return False
    except Error as e:
        self.logger.error(f'Something went wrong during reset_password function. Error - {e}')

def generate_confirmation_code(self, username: str) -> str:
    """
    Generate a confirmation code for a user.

    Args:
        username (str): The username of the user.

    Returns:
        str: The generated confirmation code.
    """

```

This function generates a confirmation code for the user specified by the username.

It inserts the code into the 'confirmation_codes' table in the database and returns the generated code.

```

"""
try:
    code = str(randint(100000, 999999))
    query = "INSERT INTO confirmation_codes(username, code, expiration_date) VALUES (%s, %s, %s)"
    expiration_date = datetime.now() + timedelta(minutes=10)
    results = self.execute_query(query, (username, code, expiration_date))
    if results:
        self.logger.info('generate_confirmation_code executed succesfully!')
        return code
    else: return False
except Error as e:
    self.logger.error(f"Something went wrong during generate_confirmation_code function. Error - {e}")

```

def verify_user_credits(self, username: str) -> str | bool:

"""

Verify the credits balance of a user.

Args:

username (str): The username of the user.

Returns:

str or bool: The credits balance of the user, or False if an error occurs.

This function checks the credits balance of the user specified by the username by querying the 'Credits' table in the database. It returns the credits balance if successful, or False if an error occurs.

"""

```

try:
    query = "SELECT credits FROM Credits WHERE username = %s"
    results = self.execute_query(query, (username,))
    self.logger.info('verify_user_credits executed succesfully!')
    if results[0][0]:
        return results[0][0]
    else: return False
except Error as e:
    self.logger.error(f"Something went wrong during verify_user_credits function. Error - {e}")

```

```
def decrement_user_credits(self, username: str) -> bool:
```

```
    """
```

Decrement the credits balance of a user.

Args:

username (str): The username of the user.

Returns:

bool: True if the credits were decremented successfully, False otherwise.

This function decrements the credits balance of the user specified by the username by updating the 'Credits' table in the database. It returns True if the operation was successful, False otherwise.

```
    """
```

```
try:
```

```

    query = "UPDATE credits SET credits = credits - 1 WHERE username = %s"
    results = self.execute_query(query, (username,))
    self.logger.info('decrement_user_credits executed succesfully!')
    if results:
        return True

```

```

        else: return False

    except Error as e:
        self.logger.error(f"Something went wrong during decrement_user_credits function. Error - {e}")

def verify_confirmation_code(self, username: str, code: str) -> bool:
    """
    Verify a confirmation code for a user.

    Args:
        username (str): The username of the user.
        code (str): The confirmation code to verify.

    Returns:
        bool: True if the confirmation code is valid for the user, False otherwise.

    This function verifies the provided confirmation code for the user specified by the
    username. It checks if the code exists in the 'confirmation_codes' table in the database
    for the given username and returns True if the code is valid, False otherwise.
    """
    try:
        query = "select code from confirmation_codes where username = %s and code = %s"
        results = self.execute_query(query, (username, code))

        if results:
            self.logger.info('verify_confirmation_code executed succesfully!')
            return True
        else: return False
    except Error as e:
        self.logger.error(f"Something went wrong during verify_confirmation_code function. Error - {e}")

def delete_request(self, req_id: str) -> bool:

```

```

try:
    query = "DELETE FROM requests where requestID = %s"
    result = self.execute_query(query, (req_id, ))
    if result:
        self.logger.info('delete_request executed succesfully!')
        return True
    else:
        return False
except Error as e:
    self.logger.error(f"Something went wrong during delete_request function. Error - {e}")

```

```
def connect(self):
```

```
    """
```

```
    Connect to the MySQL database.
```

This function establishes a connection to the MySQL database using the configured host, user, password, and database.

```
    """
```

```
    try:
```

```

        self.connection = mysql.connector.connect(
            host=self.host,
            user=self.user,
            password=self.password,
            database=self.database
        )
        if self.connection.is_connected():
            self.logger.info("Connected to MySQL database")

```

```
    except Error as e:
```

```

        self.logger.error(f"Error connecting to MySQL database: {e}")

```



```

def close(self):
    """
    Close the connection to the MySQL database.

    This function closes the connection to the MySQL database if it is currently open.
    """
    if self.connection.is_connected():
        self.connection.close()
        self.logger.info("Connection to MySQL database closed")

def execute_query(self, query, params=None) -> list | bool:
    """
    Execute a MySQL query and return the results.

    Args:
        query (str): The SQL query to execute.
        params (tuple, optional): A tuple of parameters to substitute into the query.

    Returns:
        list: A list of tuples containing the rows returned by the query.
        bool: True if the query executed successfully, False otherwise.

    This function executes the provided SQL query with optional parameters and returns
    the results as a list of tuples. It also logs any errors that occur during execution.
    """
    cursor = self.connection.cursor()
    try:
        cursor.execute(query, params)
        results = cursor.fetchall()

```

```

        self.connection.commit()

        if query[:6] == "SELECT":
            return results if results else False
        return results if results else True

    except mysql.connector.Error as error:
        self.logger.error(f"Error executing query: {error}")
        return False

    finally:
        cursor.close()

# Example usage:
if __name__ == "__main__":
    config_file = 'server-side/config/db_config.json'
    db = Database(config_file)

    db.connect()

    # pass_hash = "$2b$12$N7hbRSm84721Lqslr29i.eEvqcxNA.WA6anBPNnniiNnnFYDD/KkO"

    # # query = "SELECT * FROM Users"
    username = 'rapperorwhat@gmail.com'
    password = "Suma1124"
    # results = db.check_login_exists(username)

    # print(results)

    #db.get_models()

```

```

# print(db.check_login_credentials(username, password))

sex = "fa1b7992eb6cf603f9234cbee589a9171711ccc9c3dfcd4acf3a5b70ba4b4bea"
print(db.check_login_credentials("seo23ij4", "sdfklj"))

# print(db.check_login_credentials(username, pass_hash))


#query = "select end_date from Subscribers"
#db.execute_query(query, (username, ))


# print(db.verify_premium_status(username))


db.close()

```

A.3. server.py

```

import socket
import threading
from my_logger import setup_logger
from database import Database
import os
import bcrypt
import pathlib
from PIL import Image


CONFIG_FILE = str(pathlib.Path(__file__).parent) + '\config\db_config.json'
print(CONFIG_FILE)
APP_V = "1.0.2"
"""
    answer format
    {

```

```

    "status": ("failure", "success")
    "data": ("")
    "error_msg": "MESSAGE"          # in case of failure
}
"""

```

```
# UCV|rapperorwhat@gmail.com|Suma1124_
```

```

"""request headers
    UPD - update check
    CLS - check login in system
    UCV - user credentials verification (login)
    CSV - credit status verification
    TKN - token verification
    SSV - user premium status verification
    LOG - user send log with error
    SCC - send confirmation code
    IMG - send image for processing
    VCC - verify confirmation code
    MSG - user question
    EXT - info about extensions

    REG - register new user
    SUB - info about subscriptions
    PAY - payment info
    DUP - download update
    RMV - remove token

    RES - reset password
"""

```

```

class Server():
    def __init__(self):

        self.addr_ip = "127.0.0.1"
        self.addr_port = 65432
        self.buffer = 4096

        self.logger = setup_logger('ServerLogger', "server_side_logs.log")

        self.images_dir = "uploaded_images"

        self.server_database = Database(CONFIG_FILE)
        self.server_database.connect()

    def handle_client(self, client_socket: socket.socket, addr) -> None:
        try: # receive and print client messages
            request = client_socket.recv(self.buffer).decode("utf-8")
            request_list = request.split(sep = "|")
            if request_list[0] == "close":
                client_socket.send("closed".encode("utf-8"))
            elif request_list[0] == 'CLS':
                client_socket.send(self.user_exists(request_list[1]))
            elif request_list[0] == 'UCV':
                client_socket.send(self.user_verification(request_list[1], request_list[2]))
            elif request_list[0] == 'IMG':
                prompts = self.process_image(request_list[1], request_list[2], request_list[3], request_list[4],
client_socket)

```

```

        client_socket.send(prompts)
    elif request_list[0] == 'TKN':
        client_socket.send(self.verify_token(request_list[1]))
    elif request_list[0] == 'SSV':
        client_socket.send(self.verify_user_premium(request_list[1]))
    elif request_list[0] == 'CSV':
        client_socket.send(self.verify_credits(request_list[1]))
    elif request_list[0] == 'UPD':
        client_socket.send(self.check_updates(request_list[1]))
    elif request_list[0] == 'REG':
        client_socket.send(self.register_new_user(request_list[1], request_list[2]))
    elif request_list[0] == 'SCC':
        client_socket.send(self.send_confirmation_code(request_list[1]))
    elif request_list[0] == 'VCC':
        client_socket.send(self.verify_confirmation_code(request_list[1], request_list[2]))
    elif request_list[0] == 'MSG':
        client_socket.send(self.apply_request(request_list[1], request_list[2]))
    elif request_list[0] == 'EXT':
        client_socket.send(self.get_models())
    elif request_list[0] == 'SUB':
        client_socket.send(self.get_subscriptions(request_list[1]))
    elif request_list[0] == 'PAY':
        client_socket.send(self.get_payment_info(request_list[1]))
    elif request_list[0] == 'DUP':
        client_socket.send(self.download_update())
    elif request_list[0] == 'RMV':
        client_socket.send(self.remove_token(request_list[1]))
    elif request_list[0] == 'RES':
        client_socket.send(self.reset_password(request_list[1], request_list[2]))
    else:

```

```

        client_socket.send("Error".encode("utf-8"))
    self.logger.info(f"Received: {request}")
    client_socket.close()
except Exception as e:
    self.logger.error(f"Error when handling client: {e}")
finally:
    self.logger.info(f"Connection to client ({addr[0]}:{addr[1]}) closed")

def get_models(self) -> bytes:
    try:
        models = self.server_database.get_models()
        if not models:
            return "Error".encode('utf-8')
        answer = ""
        for model in models:
            answer = answer + f"{model[0]};{model[1]};{model[2]}" + "|"
        return answer[:-1].encode("utf-8")
    except Exception as e:
        self.logger(f"Error during models - {e}.")
        return ""

def reset_password(self, username: str, new_password: str) -> bytes:
    try:
        hashed = bcrypt.hashpw(new_password.encode('utf-8'), bcrypt.gensalt())
        return str(self.server_database.reset_password(username, hashed)).encode('utf-8')
    except Exception as e:
        self.logger(f"Error during reset password - {e}.")

def get_subscriptions(self, locale) -> bytes:
    subscriptions = self.server_database.get_subscriptions(locale)

```

```
if not subscriptions:
```

```
    return "Error".encode('utf-8')
```

```
answer = ""
```

```
for subscription in subscriptions:
```

```
    answer = answer + f"{subscription[0]};{subscription[1]};{subscription[2]}" + "|"

```

```
return answer[:-1].encode("utf-8")
```

```
def get_payment_info(self, username: str) -> bytes:
```

```
    return str(self.server_database.add_payment(username)).encode('utf-8')
```

```
def remove_token(self, token: str) -> bytes:
```

```
    return str(self.server_database.remove_auth_token(token)).encode('utf-8')
```

```
def process_image(self, image_name: str, config_dict: str, interrogate_method: str, username: str,
socket: socket.socket) -> bytes:
```

```
    """
```

```
    Process an image using the specified configuration and interrogation method.
```

```
    Args:
```

```
        image_name (str): The name of the image file to be processed.
```

```
        config_dict (str): A string containing configuration settings in the format 'key=value;key=value;...'.
```

```
        interrogate_method (str): The interrogation method to use ('best', 'classic', 'fast', or 'negative').
```

```
        username (str): The username of the user requesting the image processing.
```

```
        socket (socket.socket): The socket object used for communication.
```

```
    Returns:
```

```
        bytes: The prompts generated by the image interrogation process, encoded as bytes.
```

```
    Raises:
```


Exception: If an error occurs during the image processing.

This function performs the following steps:

1. Verifies the user's premium status and available credits.
2. Receives the image file from the client.
3. Parses the configuration settings from the config_dict string.
4. Creates an instance of the Interrogator class with the specified configuration.
5. Loads the image file and performs the specified interrogation method.
6. Removes the temporary image file.
7. Returns the generated prompts as bytes.

If the user is not a premium user and has no credits, it returns "no credit" as bytes.

If the user is not a premium user but has credits, it decrements the user's credits.

If an error occurs during the process, it logs the error message.

"""

try:

Verify user's premium status and available credits

premium = self.server_database.verify_premium_status(username)

credits = self.server_database.verify_user_credits(username)

if not premium and credits == 0:

return "no credit".encode('utf-8')

elif not premium and credits > 0:

credits = credits - 1

if not self.server_database.decrement_user_credits(username):

return "no credit".encode('utf-8')

Receive the image file from the client

self.receive_image(client_socket=socket, filename=image_name)

from clip_interrogator import Interrogator, Config

```

# Parse configuration settings
config_file = Config()
config_file.clip_model_path = os.path.join(os.path.dirname(__file__), 'models')
list_with_configs = config_dict.split(sep=";")
for pair in list_with_configs:
    key, value = pair.split(sep "=")
    if key == 'caption_max_length' or key == "chunk_size" or key == "flavor_intermediate_count":
        setattr(config_file, key, int(value))
    else:
        setattr(config_file, key, value)

# Create an instance of the Interrogator class
ci = Interrogator(config_file)

# Load the image and perform the specified interrogation method
image_path = os.path.join(self.images_dir, image_name)
image = Image.open(image_path)
if interrogate_method == 'best':
    prompts = ci.interrogate(image)
elif interrogate_method == 'classic':
    prompts = ci.interrogate_classic(image)
elif interrogate_method == 'fast':
    prompts = ci.interrogate_fast(image)
elif interrogate_method == 'negative':
    prompts = ci.interrogate_negative(image)

# Remove the temporary image file
os.remove(image_path)

```

```

        # Return the generated prompts as bytes
        return prompts.encode('utf-8')

    except Exception as e:
        self.logger.error(f"Error when processing image: {e}")

def apply_request(self, message: str, username: str) -> bytes:
    try:
        result = self.server_database.add_request(username, message)
        return str(result).encode('utf-8')
    except Exception as e:
        self.logger.error(f"Error when adding user request: {e}")
        return "False"

def run_server(self):
    try:
        server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        # bind the socket to the host and port
        server.bind((self.addr_ip, self.addr_port))
        # listen for incoming connections
        server.listen()
        self.logger.info(f"Listening on {self.addr_ip}:{self.addr_port}")
        print("Server running!")

    while True:
        # accept a client connection
        client_socket, addr = server.accept()
        self.logger.info(f"Accepted connection from {addr[0]}:{addr[1]}")

```

```

        # start a new thread to handle the client
        thread = threading.Thread(target=self.handle_client, args=(client_socket, addr,))
        self.logger.info(f"{addr[0]}:{addr[1]} received {thread.name} for processing!")
        thread.start()
    except Exception as e:
        self.logger.error(f"Error: {e}")
    finally:
        server.close()

def register_new_user(self, username: str, password: str) -> bytes:
    salt = bcrypt.gensalt()
    pass_hash = bcrypt.hashpw(password.encode('utf-8'), salt)
    result = self.server_database.add_new_user(username, pass_hash)
    if result:
        return "done".encode('utf-8')
    return "error"

def send_confirmation_code(self, username: str) -> bytes:
    try:
        result = self.server_database.generate_confirmation_code(username)
        # import smtplib

        # sender = "Private Person <mailtrap@demomailtrap.com>"
        # receiver = f"A Test User <{username}>"

        # message = f"""\
        # Subject: Hi Mailtrap
        # To: {receiver}
        # From: {sender}

```

```

        # Your confirmation code - {result}.It lasts 10 minutes.
        # Don't send it to anybody. """

        # with smtplib.SMTP("live.smtp.mailtrap.io", 587) as server:
        #     server.starttls()
        #     server.login("api", "87f8f8a3c4eb07c2ce13f7485cdd9ae0")
        #     server.sendmail(sender, receiver, message)
        # return "sent".encode('utf-8')
        return "send".encode('utf-8')
    except Exception as e:
        self.logger.error(f"Error sending email: {e}")
        return f"Error sending email: {e}".encode('utf-8')

def verify_confirmation_code(self, username: str, code: str) -> bytes:
    result = self.server_database.verify_confirmation_code(username, code)
    return str(result).encode('utf-8')

def verify_user_premium(self, username: str) -> bytes:
    result = str(self.server_database.verify_premium_status(username))
    return result.encode('utf-8')

def check_updates(self, app_version: str) -> bytes:
    if app_version == APP_V:
        return "True".encode('utf-8')
    else:
        return APP_V.encode('utf-8')

def download_update(self, app_version: str) -> None:
    pass

```

```
def verify_credits(self, username: str) -> bytes:
    result = str(self.server_database.verify_user_credits(username)).encode('utf-8')
    return result
```

```
def verify_token(self, token: str) -> bytes:
    if self.server_database.verify_auth_token(token):
        return "True".encode('utf-8')
    else:
        return "False".encode('utf-8')
```

requests : password confirm, email in the system, image for processing, buy premium, download model, send message to our team, update an app

#

```
def user_exists(self, username: str) -> bytes:
    return str(self.server_database.check_login_exists(username)).encode('utf-8')
```

```
def user_verification(self, username: str, password: str) -> bytes:
    result = self.server_database.check_login_credentials(username=username, password = password)
    if result:
        token = self.server_database.add_auth_token(username=username, password=password)
        return token.encode('utf-8')
    else:
        return "False".encode('utf-8')
```

TODO: check verification

```
def receive_image(self, client_socket: socket.socket, filename: str) -> None:
    try:
        os.makedirs(self.images_dir, exist_ok=True)
```

```

file_path = os.path.join(self.images_dir, filename)
client_socket.send("Got your image".encode('utf-8'))
with open(file_path, 'wb') as f:
    while True:
        # Receive image data chunk from the client
        image_chunk = client_socket.recv(self.buffer)
        if image_chunk.endswith(b"<END>"):
            f.write(image_chunk[:-5]) # Remove the <END> marker
            break
        # Write the received image data chunk to the file
        f.write(image_chunk)
    self.logger.info(f"Image {filename} saved successfully.")
    client_socket.send(f"Image {filename} received and saved.".encode("utf-8"))
except Exception as e:
    self.logger.error(f"Error receiving image: {e}")
    client_socket.send(f"Error receiving image: {e}".encode("utf-8"))

```

```

if __name__ == '__main__':

```

```

    serv = Server()

```

```

    serv.run_server()

```

A.4. admin_app.py

```

import ttkbootstrap as ttk

```

```

from ttkbootstrap.dialogs import Messagebox, DatePickerDialog

```

```

import socket

```

```

import os

```

```

import database as db

```

```

import bcrypt

```

```

from datetime import date, timedelta

```

```

import time, os

```

```

from pathlib import Path
from ttkbootstrap.validation import validator, add_regex_validation, add_validation
import threading
import smtplib
from server import Server

mail_token = "87f8f8a3c4eb07c2ce13f7485cdd9ae0"
SERV_IP = {"IP": "127.0.0.1", "PORT": 65432}
CONFIG_FILE = 'D:\\diploma\\server-side\\config\\db_config.json'

database = db.Database(CONFIG_FILE)
database.connect()

class Client():
    def __init__(self):

        self.addr = (SERV_IP["IP"], SERV_IP['PORT'])
        self.buffer = 4096
        #self.run_client()

    def run_client(self):
        # create a socket object
        client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

        server_ip = "127.0.0.1" # replace with the server's IP address connect_client(), send_request(),
close_connection(), update_application(), download_model(), send_payment(), send_logs()
        server_port = 65432 # replace with the server's port number
        # establish connection with server

```



```

client.connect((server_ip, server_port))

try:
    while True:
        # get input message from user and send it to the server
        msg = input("Enter message: ")
        client.send(msg.encode("utf-8")[:self.buffer])

        # receive message from the server
        response = client.recv(self.buffer)
        response = response.decode("utf-8")

        # if server sent us "closed" in the payload, we break out of
        # the loop and close our socket
        if response.lower() == "closed":
            break

    except Exception as e:
        print(f"Error: {e}")
    finally:
        # close client socket (connection to the server)
        client.close()

def process_request(self, request: str, image_path=None) -> str:
    client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

    client.connect(self.addr)

    try:
        client.send(request.encode("utf-8")[:self.buffer])

```

```

        response = client.recv(self.buffer)
        response = response.decode("utf-8")

    return response

except Exception as e:
    print(f"Error: {e}")
finally:
    # close client socket (connection to the server)
    client.close()

# done
class AdminApp(ttk.Window):
    def __init__(self):
        super().__init__()

        self.geometry = ("1100x600")
        self.minsize(600,600)
        self.maxsize(1400,1000)
        self.menu = MenuApp(self)
        self.title("Admin ImgProPlus")

        self.menu.pack_propagate(0)
        self.menu.pack(expand=True, fill='both')

    def run(self):
        self.mainloop()

```

```

# done

class MenuApp(ttk.Frame):
    def __init__(self, master):
        super().__init__(master = master)
        self.frames = {
            "Promotions" : PromotionTab(self), #PromotionTab(self),
            "Users": UsersTab(self),
            "Logs": LogsTab(self),
            "Payments": PaymentsTab(self),
            "Server": ServerTab(self),
            "Questions": QuestionsTab(self)
        }
        self.create_widgets()

    def create_widgets(self):
        # Menu frame at the top
        menu_frame = ttk.Frame(self)
        menu_frame.pack(side='top', fill='x')

        # Container frame to hold the different content frames
        # Dictionary to store the content frames

        # List of menu buttons and corresponding frame content
        buttons = [
            ("Promotions", self.create_frame1),
            ("Users", self.create_frame2),
            ("Logs", self.create_frame3),
            ("Payments", self.create_frame4),

```

```

        ("Server", self.create_frame5),
        ("Questions", self.create_frame6)
    ]

    # Create buttons
    for text, frame_func in buttons:
        button = ttk.Button(menu_frame, text=text, command=frame_func, bootstyle='info')
        button.pack(side='left', padx=1, pady=2, fill='x', expand=True)

    # Initialize with the first frame
    self.create_frame1()

def create_frame1(self):
    self.show_frame("Promotions")

def create_frame2(self):
    self.show_frame("Users")

def create_frame3(self):
    self.show_frame("Logs")

def create_frame4(self):
    self.show_frame("Payments")

def create_frame5(self):
    self.show_frame("Server")

def create_frame6(self):
    self.show_frame("Questions")

```

```

def show_frame(self, frame_name):
    # Hide all frames
    for frame in self.frames.values():
        frame.pack_forget()

    # Show the requested frame
    self.frames[frame_name].pack_propagate(0)
    self.frames[frame_name].pack(expand=True, fill='both')

# done
class PromotionTab(ttk.Frame):
    def __init__(self, master):
        super().__init__(master=master)

        self.create_widgets()

    def create_widgets(self):
        # Create Treeview widget
        self.tree = ttk.Treeview(self, columns=("id", "subscription_length", "price", "desc"),
show="headings")

        self.tree.heading("id", text="ID")
        self.tree.heading("subscription_length", text="Subscription Duration")
        self.tree.heading("price", text="Price")
        self.tree.heading("desc", text="Description")

        self.tree.column("id", width=50)
        self.tree.column("subscription_length", width=150)

```

```

self.tree.column("price", width=100)
self.tree.column("desc", width=300)

self.refresh_table()

self.tree.pack(expand=True, fill='both')

# Delete button
delete_button = ttk.Button(self, text="Delete", command = self.delete_promotion, bootstyle='danger')
delete_button.pack(side='left', pady=10, padx=30, fill='both', expand=True)

insert_button = ttk.Button(self, text="New Offer", command = lambda:
self.PromotionCreateWindow(master = self), bootstyle='warning')
insert_button.pack(side='left', pady=10, padx=30, fill='both', expand=True)

refresh_button = ttk.Button(self, text="Refresh", command=self.refresh_table)
refresh_button.pack(side='left', pady=10, padx=30, fill='both', expand=True)

def refresh_table(self):
    # Clear existing items in the Treeview
    self.tree.delete(*self.tree.get_children())

    # Sample data (replace with your data)
    query = "SELECT * from promotions"
    users = database.execute_query(query)
    for user in users:
        modified_user = list(user)
        modified_user[2] = str(modified_user[2]) + "$"

```

```

        self.tree.insert("", "end", values=modified_user)

def delete_promotion(self):
    selected_item = self.tree.selection()
    if not selected_item:
        messagebox.show_error("No item selected", "Error", parent=self)
        return

    confirm = messagebox.yesno("Confirm Delete", "Are you sure you want to delete this
promotions(s)?", parent=self)
    test = True
    if confirm:
        for item in selected_item:
            promotion = self.tree.item(item, "values")[0]
            res = database.delete_promotion(promotion)
            if not res:
                messagebox.show_error(f"Error during deleting promotion - {promotion}.", "Error")
                test = False
            self.tree.delete(item)
    if test:
        messagebox.ok("All selected promotions are deleted!\nPage automatically refreshed.", "Info")
        self.refresh_table()

class PromotionCreateWindow(ttk.Toplevel):
    def __init__(self, master):
        super().__init__(master)
        self.title("Create New Promotion Window")

```

Login Entry

```
self.sub_label = ttk.Label(self, text="Sub duration:")
self.sub_label.grid(row=0, column=0, padx=5, pady=5, sticky="w")
self.sub_entry = ttk.Entry(self)
self.sub_entry.grid(row=0, column=1, padx=5, pady=5)
```

```
add_regex_validation(self.sub_entry, '^d{1,2}\s+months?$')
```

Password Entry

```
self.price_label = ttk.Label(self, text="Price:")
self.price_label.grid(row=1, column=0, padx=5, pady=5, sticky="w")
self.price_entry = ttk.Entry(self)
self.price_entry.grid(row=1, column=1, padx=5, pady=5)
```

```
add_validation(self.price_entry, validate_price)
```

Description Label

```
self.desc_label = ttk.Label(self, text="Description:")
self.desc_label.grid(row=2, column=0, padx=5, pady=5, sticky="nw")
```

Description Text Widget

```
self.desc_text = ttk.Text(self, height=4, width=30)
self.desc_text.grid(row=2, column=1, padx=5, pady=5, sticky="w")
```

Buttons

```
self.close_button = ttk.Button(self, text="Close", command=lambda : self.destroy())
self.close_button.grid(row=3, column=0, padx=5, pady=5, sticky="e")
self.create_button = ttk.Button(self, text="Create", command=self.create_promotion)
self.create_button.grid(row=3, column=1, padx=5, pady=5, sticky="w")
```



```

def show_date_picker(self, *args) -> str:
    result = DatePickerDialog(self, title="Select Sub. End Date", startdate=date.today() +
timedelta(days=1))
    self.user_sub_date = result.date_selected

def create_promotion(self):
    sub = self.sub_entry.get()
    price = self.price_entry.get()
    description = self.desc_text.get("1.0", ttk.END)
    # Add code to create promotion here

    result = database.insert_promotion(sub, price, description)
    if result:
        MessageBox.ok(f"Promotion added to database!\n Page automatically refreshed.", "Info")
        self.master.refresh_table()
    else:
        MessageBox.show_error("Promotion wasn't added to database! Something is wrong", "Error")

#done
class LogsTab(ttk.Frame):
    def __init__(self, master):
        super().__init__(master=master)
        self.log_file = str(Path(__file__).parent.parent) + "\\server_side_logs.log"
        self.last_line_inserted = 0

        self.text_widget = ttk.Text(self, wrap="none") # Set wrap to "none"
        self.text_widget.pack(fill="both", expand=True)
        self.rowconfigure(0, weight=1) # Assign weight to the row

```

```

refresh_button = ttk.Button(self, text="Refresh Logs", command=self.refresh_logs)
refresh_button.pack(side="bottom", pady=5)

self.text_widget.bind("<Enter>", lambda event: self.text_widget.tag_remove("last_added_text",
"1.0", "end"))

self.read_log_file(self.log_file)
self.start_listening_file_changes()

def read_log_file(self, file_path):
    with open(file_path, "r") as file:
        file.seek(self.last_line_inserted) # Move the file pointer to the last line inserted
        log_contents = file.read()
        self.last_line_inserted = file.tell()
    if log_contents:
        self.text_widget.configure(state='normal')
        var1 = self.text_widget.index('end-1c')
        self.text_widget.insert(ttk.END, log_contents)
        var2 = self.text_widget.index('end-1c')
        self.text_widget.see(ttk.END)
        self.text_widget.tag_add("last_added_text", var1, var2)
        self.text_widget.tag_config("last_added_text", background="red")
        self.text_widget.tag_config("last_added_text", foreground="white")

        self.text_widget.configure(state='disabled')

def refresh_logs(self):
    self.read_log_file(self.log_file)

```

```

def detect_file_changes(self, file_path, interval=1):
    last_modified = os.path.getmtime(file_path)
    while True:
        current_modified = os.path.getmtime(file_path)
        if current_modified != last_modified:
            self.read_log_file(file_path)
            last_modified = current_modified
        time.sleep(interval)

def start_listening_file_changes(self) -> None:
    threading.Thread(target=self.detect_file_changes, args=(self.log_file, 1)).start()

#done

class ServerTab(ttk.Frame):
    def __init__(self, master):
        super().__init__(master=master)

        self.server_thread = None

        self.server_running = False

        self.create_widgets()

        self.auto_refresh_interval = 1000 # in milliseconds

    def create_widgets(self):
        self.server_status_label = ttk.Label(self, text="Server Status: Not Running")
        self.server_status_label.grid(row=0, column=0, columnspan=2, pady=10, padx=10)

        self.request_label = ttk.Label(self, text="Input Client Request Above:")
        self.request_label.grid(row=1, column=0, columnspan=2, pady=10, padx=10)

        self.request_entry = ttk.Entry(self)

```

```

self.request_entry.grid(row=2, column=0, columnspan=2, pady=10, padx=10, sticky="ew")

self.send_button = ttk.Button(self, text="Send Request", command=self.send_request)
self.send_button.grid(row=3, column=0, pady=10, padx=10, sticky="ew")

self.run_server_button = ttk.Button(self, text="Run Server", command=self.run_server,
bootstyle="success")
self.run_server_button.grid(row=3, column=1, pady=10, padx=10, sticky="ew")

self.stop_server_button = ttk.Button(self, text="Stop Server", command=self.stop_server,
bootstyle="danger")
self.stop_server_button.grid(row=4, column=0, pady=10, padx=10, sticky="ew")

self.refresh_button = ttk.Button(self, text="Refresh", command=self.refresh_server_status,
bootstyle="warning")
self.refresh_button.grid(row=4, column=1, pady=10, padx=10, sticky="ew")

self.tip_button = ttk.Button(self, text="Show Tip", command=self.show_tip)
self.tip_button.grid(row=5, column=0, pady=10, padx=10, sticky="ew")

self.mini_logs_label = ttk.Label(self, text="Server output:")
self.mini_logs_label.grid(row=6, column=0, columnspan=2, pady=10, padx=10)

self.output_text = ttk.Text(self, state="disabled")
self.output_text.grid(row=7, column=0, columnspan=2, pady=10, padx=10, sticky="nsew")

# Configure column weights for proper resizing
self.grid_columnconfigure(0, weight=1)
self.grid_columnconfigure(1, weight=1)

```

```

def run_server(self):
    if self.server_thread is None or not self.server_thread.is_alive():
        self.server_thread = threading.Thread(target=self.run_server_thread)
        self.server_thread.start()
        self.server_running = True
        self.auto_refresh()
    else:
        MessageBox.show_info("Server", "Server is already running.")

def run_server_thread(self):
    try:
        self.serv = Server()
        self.serv.run_server()
        self.server_status_label.config(text="Server Status: Running")
    except Exception as e:
        print(f"Error: {e}")
    finally:
        self.server_running = False
        self.server_status_label.config(text="Server Status: Not Running")

def stop_server(self):
    if self.server_thread is not None and self.server_thread.is_alive():
        # implement logic here
        self.server_running = False
        self.server_thread = None
        if self.server_thread.is_alive():
            MessageBox.show_error("Server", "Failed to stop the server properly.")
        else:
            self.server_thread = None

```

```

        self.server_status_label.config(text="Server Status: Not Running")
    else:
        MessageBox.show_info("Server is not running.", "Server")

def refresh_server_status(self):
    if self.server_thread is None or not self.server_thread.is_alive():
        self.server_status_label.config(text="Server Status: Not Running")
    else:
        self.server_status_label.config(text="Server Status: Running")

def auto_refresh(self):
    self.refresh_server_status()
    if self.server_running:
        self.after(self.auto_refresh_interval, self.auto_refresh)

def send_request(self):
    request = self.request_entry.get()
    if request:
        client = Client()
        response = client.process_request(request)
        self.output_text['state'] = 'normal'
        self.output_text.insert("1.0", response + "\n-----\n")
        self.output_text.see("1.0")
        self.output_text['state'] = 'disabled'
    else:
        MessageBox.show_warning("Input Error", "Please enter a request.")

def show_tip(self):
    requests = [
        "UPD - update check (UPD|variable app_version)",

```

```

"CLS - check login in system (CLS|variable username)",
"UCV - user credentials verification (login) (UCV|variable username, variable password)",
"CSV - credit status verification (CSV|variable username)",
"TKN - token verification (TKN|variable token)",
"SSV - user premium status verification (SSV|variable username)",
"LOG - user send log with error (LOG|variable log_message)",
"SCC - send confirmation code (SCC|variable username)",
"IMG - send image for processing (IMG|variable image_name, variable config_dict, variable
interrogate_method, variable username)",
"VCC - verify confirmation code (VCC|variable username, variable code)",
"MSG - user question (MSG|variable username, variable question)",
"EXT - info about extensions (EXT)",
"REG - register new user (REG|variable username, variable password, variable email)",
"SUB - info about subscriptions (SUB|variable username)",
"PAY - payment info (PAY|variable username, variable payment_info)",
"DUP - download update (DUP|variable app_version)"
]

```

```

tip_window = self.RequestTipWindow(self, requests)
tip_window.mainloop()

```

```

class RequestTipWindow(ttk.Toplevel):
    def __init__(self, master, requests):
        super().__init__(master)
        self.title("Request Tip")
        self.geometry("300x500")

        self.requests_text = ttk.Text(self, wrap="word")
        self.requests_text.pack(fill="both", expand=True, padx=10, pady=10)

```

```

        for request in requests:

            self.requests_text.insert(ttk.END, request + "\n")


        self.requests_text.configure(state="disabled")


# done

class PaymentsTab(ttk.Frame):

    def __init__(self, master):

        super().__init__(master=master)

        self.create_widgets()


    def create_widgets(self):

        # Create Treeview widget

        self.tree = ttk.Treeview(self, columns=("id", "username", "payment_date", "promotion_id"),
show="headings")

        self.tree.heading("id", text="ID")

        self.tree.heading("username", text="Username")

        self.tree.heading("payment_date", text="Payment Date")

        self.tree.heading("promotion_id", text="Promotion Id")


        self.tree.column("id", width=50)

        self.tree.column("username", width=150)

        self.tree.column("payment_date", width=100)

        self.tree.column("promotion_id", width=100)


        self.refresh_table()


        self.tree.pack(expand=True, fill='both')

```



```

# Delete button

delete_button = ttk.Button(self, text="Delete", command=self.delete_payment, bootstyle='danger')

delete_button.pack(side='left', pady=10, padx=30, fill='both', expand=True)


create_button = ttk.Button(self, text="Create Payment", command= lambda :
self.PaymentCreateWindow(self), bootstyle='warning')

create_button.pack(side='left', pady=10, padx=30, fill='both', expand=True)


refresh_button = ttk.Button(self, text="Refresh", command=self.refresh_table, )

refresh_button.pack(side='left', pady=10, padx=30, fill='both', expand=True)


def refresh_table(self):

    # Clear existing items in the Treeview

    self.tree.delete(*self.tree.get_children())


    # Sample data (replace with your data)

    query = "SELECT * from payments"

    users = database.execute_query(query)

    for user in users:

        self.tree.insert("", "end", values=user)


def delete_payment(self):

    selected_item = self.tree.selection()

    if not selected_item:

        messagebox.show_error("No item selected", "Error", parent=self)

        return


    confirm = messagebox.yesno("Confirm Delete", "Are you sure you want to delete this
payments(s)?", parent=self)

```

```

test = True

if confirm:

    for item in selected_item:

        payment = self.tree.item(item, "values")[0]

        res = database.delete_payment(payment)

        if not res:

            messagebox.show_error(f"Error during deleting payment - {payment}.", "Error")

            test = False

        self.tree.delete(item)

if test:

    messagebox.ok("All selected payments are deleted!\nPage automatically refreshed.", "Info")

    self.refresh_table()

```

```

class PaymentCreateWindow(ttk.Toplevel):

    def __init__(self, master):

        super().__init__(master)

        self.title("Create Payment Window")


        self.payment_date = None

        # Login Entry

        self.username_label = ttk.Label(self, text="Username:")

        self.username_label.grid(row=0, column=0, padx=5, pady=5, sticky="w")

        self.username_entry = ttk.Entry(self)

        self.username_entry.grid(row=0, column=1, padx=5, pady=5)


        # Password Entry

        self.date_label = ttk.Label(self, text="Payment Date:")

        self.date_label.grid(row=1, column=0, padx=5, pady=5, sticky="w")

```

```

self.date_button = ttk.Button(self, text='Pick a date', command = self.show_date_picker)
self.date_button.grid(row=1, column=1, padx=5, pady=5)

self.promotion_label = ttk.Label(self, text="Promotion ID:")
self.promotion_label.grid(row=2, column=0, padx=5, pady=5, sticky="w")
self.promotion_entry = ttk.Entry(self)
self.promotion_entry.grid(row=2, column=1, padx=5, pady=5)

# Buttons
self.close_button = ttk.Button(self, text="Close", command=lambda : self.destroy())
self.close_button.grid(row=3, column=0, padx=5, pady=5, sticky="e")
self.create_button = ttk.Button(self, text="Create", command=self.create_payment)
self.create_button.grid(row=3, column=1, padx=5, pady=5, sticky="w")

def show_date_picker(self, *args) -> str:
    result = DatePickerDialog(self, title="Select Sub. End Date", startdate=date.today() +
timedelta(days=1))
    self.user_sub_date = result.date_selected

def create_payment(self):
    username = self.username_entry.get()
    promotion = self.promotion_entry.get()

    result = database.add_payment(username, self.payment_date, promotion)
    if result:
        messagebox.ok(f"Payment added to database!\nPage automatically refreshed.", "Info")
        self.master.refresh_table()

```

```

        else:
            messagebox.show_error("User wasn't added to database! Something is wrong", "Error")

# done
class UsersTab(ttk.Frame):
    def __init__(self, master):
        super().__init__(master=master)
        self.create_widgets()

    def create_widgets(self):
        # Create Treeview widget
        self.tree = ttk.Treeview(self, columns=("id", "login", "sub_status"), show="headings")
        self.tree.heading("id", text="ID")
        self.tree.heading("login", text="Login")
        self.tree.heading("sub_status", text="Subscription Status")

        self.refresh_table()

        self.tree.pack(expand=True, fill='both')

        # Delete button
        delete_button = ttk.Button(self, text="Delete", command=self.delete_user, bootstyle='danger')
        delete_button.pack(side='left', pady=10, padx=30, fill='both', expand=True)

        create_user_button = ttk.Button(self, text="Create User", command = lambda:
self.UserCreateWindow(self), bootstyle='warning')
        create_user_button.pack(side='left', pady=10, padx=30, fill='both', expand=True)

        self.refresh_button = ttk.Button(self, text="Refresh", command=self.refresh_table)
        self.refresh_button.pack(side='left', pady=10, padx=30, fill='both', expand=True)

```

```

def refresh_table(self):

    # Clear existing items in the Treeview
    self.tree.delete(*self.tree.get_children())

    # Sample data (replace with your data)

    query = "SELECT u.username, CASE WHEN s.username IS NULL THEN 'No Subscription' ELSE
'Has Subscription' END AS Subscription_Status FROM Users u LEFT JOIN Subscribers s ON u.username =
s.username;"

    users = database.execute_query(query)

    i = 1

    for user in users:

        self.tree.insert("", "end", values=(i, *user))

        i += 1


def delete_user(self):

    selected_item = self.tree.selection()

    if not selected_item:

        MessageBox.show_error("No item selected", "Error", parent=self)

        return

    confirm = MessageBox.yesno("Confirm Delete", "Are you sure you want to delete this user(s)?",
parent=self)

    test = True

    if confirm:

        for item in selected_item:

            username = self.tree.item(item, "values")[1]

            res = database.delete_user(username)

            if not res:

                MessageBox.show_error(f"Error during deleting user - {username}.", "Error")

```

```

        test = False

        self.tree.delete(item)

    if test:

        MessageBox.ok("All selected users are deleted!", "Info")


class UserCreateWindow(ttk.Toplevel):

    def __init__(self, master):

        super().__init__(master)

        self.title("Create User Window")


        self.user_sub_date = None

        # Login Entry

        self.login_label = ttk.Label(self, text="Login:")

        self.login_label.grid(row=0, column=0, padx=5, pady=5, sticky="w")

        self.login_entry = ttk.Entry(self)

        self.login_entry.grid(row=0, column=1, padx=5, pady=5)


        # Password Entry

        self.password_label = ttk.Label(self, text="Password:")

        self.password_label.grid(row=1, column=0, padx=5, pady=5, sticky="w")

        self.password_entry = ttk.Entry(self)

        self.password_entry.grid(row=1, column=1, padx=5, pady=5)


        # Subscription Checkbutton

        self.subscription_var = ttk.IntVar()

        self.subscription_var.set(0)

        self.subscription_checkbutton = ttk.Checkbutton(self, text="Subscription",
variable=self.subscription_var)

```

```

self.subscription_checkbutton.grid(row=2, column=0, columnspan=2, padx=5, pady=5,
sticky="w")

self.subscription_var.trace_add('write', self.show_date_picker)


# Buttons

self.close_button = ttk.Button(self, text="Close", command=lambda : self.destroy())
self.close_button.grid(row=3, column=0, padx=5, pady=5, sticky="e")
self.create_button = ttk.Button(self, text="Create", command=self.create_account)
self.create_button.grid(row=3, column=1, padx=5, pady=5, sticky="w")


def show_date_picker(self, *args) -> str:

    result = DatePickerDialog(self, title="Select Sub. End Date", startdate=date.today() +
timedelta(days=1))

    self.user_sub_date = result.date_selected


def create_account(self):

    login = self.login_entry.get()
    password = self.password_entry.get()
    subscribed = self.subscription_var.get()

    # Add code to create account here

    salt = bcrypt.gensalt()
    hash = bcrypt.hashpw(password.encode('utf-8'), salt)
    result = database.add_new_user(login, hash)
    added_sub = None

    if subscribed:

        added_sub = database.add_user_premium_status(login, self.user_sub_date)

    if result and added_sub:

        messagebox.ok(f"User added to database!\nLogin: {login}\nPassword:
{password}\nSubscribed: {'yes' if subscribed else 'no'}", "Info")

    else:

        messagebox.show_error("User wasn't added to database! Something is wrong", "Error")

```

```

# done

class QuestionsTab(ttk.Frame):
    def __init__(self, master):
        super().__init__(master=master)

        self.create_widgets()

    def create_widgets(self):
        # Create Treeview widget
        self.tree = ttk.Treeview(self, columns=("id", "username", "request_text"), show="headings")
        self.tree.heading("id", text="ID")
        self.tree.heading("username", text="Username")
        self.tree.heading("request_text", text="Request Text")

        self.tree.column("id", width=50)
        self.tree.column("username", width=100)
        self.tree.column("request_text", width=300)

        self.refresh_table()

        self.tree.pack(expand=True, fill='both')

        # Delete button
        delete_button = ttk.Button(self, text="Delete", command=self.delete_question, bootstyle='danger')
        delete_button.pack(side='left', pady=10, padx=30, fill='both', expand=True)

        create_user_button = ttk.Button(self, text="Write Response", command =
self.create_writing_window, bootstyle='warning')
        create_user_button.pack(side='left', pady=10, padx=30, fill='both', expand=True)

```



```

self.refresh_button = ttk.Button(self, text="Refresh", command=self.refresh_table)
self.refresh_button.pack(side='left', pady=10, padx=30, fill='both', expand=True)

def create_writing_window(self) -> None:
    try:
        email = self.tree.item(self.tree.selection(), "values")[1]
        self.ResponseCreateWindow(self, email)
    except IndexError:
        MessageBox.show_info("Select request first!", "Info")

def refresh_table(self):
    try:
        # Clear existing items in the Treeview
        self.tree.delete(*self.tree.get_children())

        # Sample data (replace with your data)
        query = "select * from requests"
        users = database.execute_query(query)
        for user in users:
            self.tree.insert("", "end", values=user)
    except TypeError:
        pass

def delete_question(self):
    selected_item = self.tree.selection()
    if not selected_item:
        MessageBox.show_error("No item selected", "Error", parent=self)
    return

```

```

        confirm = messagebox.yesno("Confirm Delete", "Are you sure you want to delete this question(s)?",
parent=self)

        test = True

        if confirm:

            for item in selected_item:

                req_id = self.tree.item(item, "values")[0]

                res = database.delete_request(req_id)

                if not res:

                    messagebox.show_error(f"Error during deleting request - {req_id}.", "Error")

                    test = False

                    self.tree.delete(item)

            if test:

                messagebox.ok("All selected users are deleted!\nPage automatically refreshed.", "Info")

                self.refresh_table()

```

```

class ResponseCreateWindow(ttk.Toplevel):

    def __init__(self, master, email):

        super().__init__(master = master)

        self.title("Send Response Window")

        self.email = email

        # Text Entry

        self.text_label = ttk.Label(self, text=f"Response Text to - {self.email}:")

        self.text_label.grid(row=0, column=0, padx=5, pady=5, sticky="nw")

        self.text_entry = ttk.Text(self, height=25, width=75)

        self.text_entry.grid(row=1, column=0, columnspan=2, padx=5, pady=5, sticky="nsew")

```

```

# Buttons

self.send_button = ttk.Button(self, text="Send", command=self.send_response, bootstyle =
'warning')

self.send_button.grid(row=2, column=0, padx=30, pady=5, sticky="nsew")


self.close_button = ttk.Button(self, text="Close", command=self.destroy)
self.close_button.grid(row=2, column=1, padx=30, pady=5, sticky="nsew")


# Configure column and row weights to make text_entry expandable
self.grid_columnconfigure(0, weight=1)
self.grid_columnconfigure(1, weight=1)
self.grid_rowconfigure(1, weight=1)


def send_response(self) -> None:
    result = self.text_entry.get("1.0", ttk.END)

    # sender = "Private Person <mailtrap@demomailtrap.com>"
    # receiver = f"A Test User <{username}>"

    # message = f"""\
    # Subject: Hi Mailtrap
    # To: {receiver}
    # From: {sender}

    #{result}

    # Best wishes. ImgPlusPro Team."""

    # with smtplib.SMTP("live.smtp.mailtrap.io", 587) as server:
    #     server.starttls()
    #     server.login("api", "{mail_token}")

```

```

        # server.sendmail(sender, receiver, message)

        send = True

        #

        if send:

            answer = Messagebox.yesno(f"Response to {self.email} was sent. Do you want to delete this
record?.", "Info")

            if answer:

                self.master.delete_question()

                Messagebox.ok("Record was successfully deleted.\nPage was automatically refreshed.",
"Info")

                self.master.refresh_table()

            self.destroy()

        else:

            Messagebox.show_error(f"Response to {self.email} wasn't sent. Something is wrong.", "Error")

```

```
@validator
```

```
def validate_price(event):
```

```
    try:
```

```
        x = float(event.postchangetext)
```

```
        if x > 0:
```

```
            return True
```

```
        return False
```

```
    except ValueError:
```

```
        return False
```

```
if __name__ == '__main__':
```

```
    AdminApp().run()
```

A.5. mail_serv.py

```

import smtplib

def prompt(prompt):
    return input(prompt).strip()

fromaddr = prompt("From: ")
toaddrs = prompt("To: ").split()
print("Enter message, end with ^D (Unix) or ^Z (Windows):")

# Add the From: and To: headers at the start!
msg = ("From: %s\r\nTo: %s\r\n\r\n"
       % (fromaddr, ", ".join(toaddrs)))
while True:
    try:
        line = input()
    except KeyboardInterrupt:
        break
    if not line:
        break
    msg = msg + line

print("Message length is", len(msg))

server = smtplib.SMTP('localhost')
server.set_debuglevel(1)
server.sendmail(fromaddr, toaddrs, msg)
server.quit()

```

A.6. app.py

```

from settings import ApplicationConfiguration, APP_INIT
from localization import MAIN_WINDOW, CLOSE_BUTTON
import ttkbootstrap as ttk

from PIL import Image, ImageTk, ImageSequence
from tkinter import filedialog, Canvas

from addons import get_gpus, get_processor_name
from ttkbootstrap.tooltip import ToolTip
from ttkbootstrap.constants import INFO, INVERSE
import os

from client import Client

from login_page import ForgotPasswordWindow
from ttkbootstrap.dialogs import Messagebox
from ttkbootstrap.scrolled import ScrolledText
from addons import Config
import threading

from pathlib import Path
from itertools import cycle
import time
import pyperclip
import tkinterweb

from my_logger import setup_logger
import qrcode
import qrcode.image.pil

CAPTION_MODELS = {
    'blip-base': 'Salesforce/blip-image-captioning-base', # 990MB
    'blip-large': 'Salesforce/blip-image-captioning-large', # 1.9GB

```

```

'blip2-2.7b': 'Salesforce/blip2-opt-2.7b',      # 15.5GB
'blip2-flan-t5-xl': 'Salesforce/blip2-flan-t5-xl',  # 15.77GB
'git-large-coco': 'microsoft/git-large-coco',      # 1.58GB
}

```

```

app_configuration = ApplicationConfiguration()
logger = setup_logger(logger_name="User Application", logger_file="logs.log")

```

```

class MenuApp(ttk.Frame):
    def __init__(self, master):
        super().__init__(master = master)

        global USERNAME

        try:
            with open("token", 'r') as f:
                token = f.read()
        except FileNotFoundError:
            token = None

        if token is not None:
            logger.info(f"User: {token.split(sep='|')[0]}")
            USERNAME = token.split(sep='|')[0]
        else:
            USERNAME = None

        self.frames = {

            MAIN_WINDOW['notebook_mainpage_tab'][app_configuration.locale]: MainTab(self),
            #MainTab(self),

```

```

        MAIN_WINDOW['notebook_extensions_tab'][app_configuration.locale]: ExtensionsTab(self),
        MAIN_WINDOW['notebook_qa_tab'][app_configuration.locale]: SendQuestionTab(self),
        MAIN_WINDOW['notebook_docs_tab'][app_configuration.locale]: DocumentationTab(self),
        MAIN_WINDOW['notebook_settings_tab'][app_configuration.locale]: SettingsTab(self),
        MAIN_WINDOW['notebook_account_tab'][app_configuration.locale]: AccountTab(self),
    }

    self.create_widgets()

def create_widgets(self):
    # Menu frame at the top
    menu_frame = ttk.Frame(self)
    menu_frame.pack(side='top', fill='x')

    # Container frame to hold the different content frames
    # Dictionary to store the content frames

    # List of menu buttons and corresponding frame content
    buttons = [
        (MAIN_WINDOW['notebook_mainpage_tab'][app_configuration.locale], self.create_frame1),
        (MAIN_WINDOW['notebook_extensions_tab'][app_configuration.locale], self.create_frame2),
        (MAIN_WINDOW['notebook_qa_tab'][app_configuration.locale], self.create_frame3),
        (MAIN_WINDOW['notebook_docs_tab'][app_configuration.locale], self.create_frame4),
        (MAIN_WINDOW['notebook_settings_tab'][app_configuration.locale], self.create_frame5),
        (MAIN_WINDOW['notebook_account_tab'][app_configuration.locale], self.create_frame6)
    ]

    # Create buttons
    for text, frame_func in buttons:
        button = ttk.Button(menu_frame, text=text, command=frame_func, bootstyle='info')

```



```

        button.pack(side='left', padx=1, pady=2, fill='x', expand=True)

    # Initialize with the first frame
    self.pack_propagate(0)
    self.create_frame1()

def create_frame1(self):
    self.show_frame(MAIN_WINDOW['notebook_mainpage_tab'][app_configuration.locale])

def create_frame2(self):
    self.show_frame(MAIN_WINDOW['notebook_extensions_tab'][app_configuration.locale])

def create_frame3(self):
    self.show_frame(MAIN_WINDOW['notebook_qa_tab'][app_configuration.locale])

def create_frame4(self):
    self.show_frame(MAIN_WINDOW['notebook_docs_tab'][app_configuration.locale])

def create_frame5(self):
    self.show_frame(MAIN_WINDOW['notebook_settings_tab'][app_configuration.locale])

def create_frame6(self):
    self.show_frame(MAIN_WINDOW['notebook_account_tab'][app_configuration.locale])

def show_frame(self, frame_name):
    # Hide all frames
    for frame in self.frames.values():
        frame.pack_forget()

    # Create the frame if it doesn't exist

```

```
# Show the requested frame
self.frames[frame_name].pack(expand=True, fill='both')
```

```
class AccountTab(ttk.Frame):
    def __init__(self, master):
        super().__init__(master)

        # what can user see in account tab?
        # amount of credits
        # subscription status
        # login
        # ability to change password
        # button to logout
        # buy subscription button

        # dotenv.load_dotenv()

        self.columnconfigure((0,4), weight=1, uniform='a')
        self.columnconfigure((1,2), weight=4, uniform='a')
        self.rowconfigure((0,1,2,3,4), weight=1, uniform='a')

        self.username = USERNAME

        self.username_label_name = ttk.Label(self,
                                              text = MAIN_WINDOW["username_label"][app_configuration.locale])
        self.username_label_name.grid(column=1, row=0, sticky='nsew')
```

```

self.username_label = ttk.Label(self,
                                text = self.username)
self.username_label.grid(column=2, row=0, sticky='nsew')

# subscription
self.subscription_status_label = ttk.Label(self,
                                            text = MAIN_WINDOW["subscription_label"][app_configuration.locale])
self.subscription_status_label.grid(column=1, row=1, sticky='nsew')

self.subscription_status_variable = subscription_status_verification()
self.subscription_status = ttk.Label(self)
if self.subscription_status_variable == "No":
    self.subscription_status['text'] = MAIN_WINDOW['sub_status_no'][app_configuration.locale]
else:
    self.subscription_status['text'] =
f"{MAIN_WINDOW['sub_status_yes'][app_configuration.locale]} (up to {self.subscription_status_variable})"
self.subscription_status.grid(row = 1, column = 2, sticky='nsew')

# credits
self.credit_status_label = ttk.Label(self,
                                      text = MAIN_WINDOW["credit_label"][app_configuration.locale])
self.credit_status_label.grid(column=1, row=2, sticky='nsew')

self.credit_status_variable = self.credit_status_verification(self.username)
self.credit_status = ttk.Label(self,
                                text = self.credit_status_variable)
self.credit_status.grid(row = 2, column = 2, sticky='nsew')

# logout button

```

```

self.logout_button = ttk.Button(self,
                                text=MAIN_WINDOW["logout_button"][app_configuration.locale],
                                command=self.logout)
self.logout_button.grid(row = 4, column = 2, sticky='ew', padx=4)

# change password button

self.change_password_button = ttk.Button(self,
                                          text=MAIN_WINDOW["change_password_button"][app_configuration.locale],
                                          command= lambda: ForgotPasswordWindow(self))
self.change_password_button.grid(row = 4, column = 1, sticky='ew', padx=4)

# buy subscription button

self.purchase_subscription_button = ttk.Button(self,

text=MAIN_WINDOW["purchase_subscription_button"][app_configuration.locale],
                                          command= lambda: PaymentWindow(self.master, True))
self.purchase_subscription_button.grid(row = 3, column = 2, sticky='ew', padx=4)


def logout(self) -> None:
    client = Client()
    respond = client.process_request(f"RMV|{self.username}")
    if respond == "True":
        Messagebox.ok("You have successfully logged out!", "Logout")
        os.remove("token")

```

```

os.remove(os.path.dirname(os.path.realpath(__file__)) + "\\user_settings")
self.master.master.destroy()

```

```

else:

```

```

    MessageBox.show_error("Something went wrong!", "Logout")

```

```

def credit_status_verification(self, username) -> str:

```

```

    if self.subscription_status_variable != 'No':

```

```

        return "∞" #MAIN_WINDOW["unlimited_credits_label"][app_configuration.locale]

```

```

    else:

```

```

        client = Client()

```

```

        result = client.process_request(f"CSV|{username}")

```

```

        return result

```

```

class MainTab(ttk.Frame):

```

```

    def __init__(self, master):

```

```

        super().__init__(master)

```

```

        self.master = master

```

```

        self.PICTURE_EXISTS = False

```

```

        self.image = None

```

```

        self.image_file = None

```

```

        self.config_file = Config()

```

```

        if app_configuration.cmd:

```

```

            self.config_file.quiet = False

```

```

        else:

```

```

self.config_file.quiet = True

self.columnconfigure(0, weight=4, uniform='a')
self.columnconfigure(1, weight=3, uniform='a')
self.rowconfigure(0, weight=4, uniform='a')
self.rowconfigure(1, weight=1, uniform='a')

self.upload_image_frame = ttk.Frame(self,
                                     )

self.upload_image_frame.columnconfigure(0, weight=1, uniform='a',)
self.upload_image_frame.rowconfigure(0, weight=1, uniform='a')

self.upload_button = ttk.Button(self,
                                text = MAIN_WINDOW['upload_image_button'][app_configuration.locale], #
                                change for app_configuration.locales
                                command=self.upload_image)

self.upload_button.place(relx = 0.3,
                          rely = 0.4,
                          anchor = 'center',
                          )

self.upload_image_frame.grid(row = 0, column = 0, sticky='nsew')

# widgets for image_frame

```

```

self.canvas_for_image = Canvas(self.upload_image_frame,
                                height=200,
                                width=200,
                                borderwidth=0,
                                highlightthickness=0,
                                relief='ridge')
self.canvas_for_image.grid(row = 0, column = 0, sticky = 'nsew', padx=0, pady=0)

self.canvas_for_image.bind('<Configure>', self.stretch_image)
self.canvas_for_image.bind('<Configure>', self.fill_image)

# widgets for settings_frame

self.settings_frame = ttk.Frame(self,)

self.settings_device_variable = ttk.StringVar()
self.settings_model_variable = ttk.StringVar()
self.settings_model_mode_variable = ttk.StringVar()
self.settings_compute_variable = ttk.StringVar()
# self.settings_var5 = ttk.StringVar()

self.settings_frame.columnconfigure((0), weight=1, uniform='a')
self.settings_frame.columnconfigure((1), weight=2, uniform='a')
self.settings_frame.rowconfigure((0,1,2,3,4,5,6), weight=1, uniform='a')

self.settings_frame_label = ttk.Label(self.settings_frame,
                                       text =
MAIN_WINDOW['image_processing_settings_label'][app_configuration.locale])
self.settings_frame_label.grid(row = 0, column = 0, columnspan=2)

```

```

self.settings_device_label = ttk.Label(self.settings_frame,
                                       text = MAIN_WINDOW["device_label"][app_configuration.locale])
self.settings_device_label.grid(row = 1, column = 0)

self.settings_model_label = ttk.Label(self.settings_frame,
                                       text = MAIN_WINDOW["model_label"][app_configuration.locale])
self.settings_model_label.grid(row = 2, column = 0)

self.settings_model_mode_label = ttk.Label(self.settings_frame,
                                             text = MAIN_WINDOW["mode_label"][app_configuration.locale])
self.settings_model_mode_label.grid(row = 3, column = 0)

self.settings_model_mode_label = ttk.Label(self.settings_frame,
                                             text = MAIN_WINDOW["compute_label"][app_configuration.locale])
self.settings_model_mode_label.grid(row = 4, column = 0)


device_list = [f"CPU: {get_processor_name()}"]
for gpu_id, gpu_name in get_gpus():
    device_list.append(f"{gpu_id}: {gpu_name}")

self.settings_device_list = ttk.Combobox(self.settings_frame, values = device_list,
                                         textvariable=self.settings_device_variable, state='readonly')
self.settings_device_list.set(device_list[0])

```



```

self.settings_device_list.grid(row = 1, column = 1, sticky = 'ew', padx=2)

# Get the current user's home directory
self.model_list = installed_models()

self.settings_model_list = ttk.Combobox(self.settings_frame,
                                       values = self.model_list,
                                       textvariable=self.settings_model_variable,
                                       state='readonly'
                                       )

self.settings_model_list.set(self.model_list[0])
self.settings_model_list.grid(row = 2, column = 1, sticky = 'ew', padx=2)

mode_list = ['fast', 'classic', 'best', 'negative']

self.settings_model_mode_list = ttk.Combobox(self.settings_frame,
                                             values = mode_list,
                                             textvariable=self.settings_model_mode_variable,
                                             state='readonly')

self.settings_model_mode_list.set(mode_list[0])
self.settings_model_mode_list.grid(row = 3, column = 1, sticky = 'ew', padx=2)

compute_list = ['local', 'cloud']

self.settings_compute_list = ttk.Combobox(self.settings_frame,
                                           values = compute_list,
                                           textvariable=self.settings_compute_variable,
                                           state='readonly')

self.settings_compute_list.set(compute_list[0])
self.settings_compute_list.grid(row = 4, column = 1, sticky = 'ew', padx=2)

```

```

self.settings_additional_menu_button = ttk.Button(self.settings_frame,
                                                    text =
MAIN_WINDOW['additional_settings_button'][app_configuration.locale],
                                                    command = lambda: self.AdditionalSettings(self, self.config_file))
self.settings_additional_menu_button.grid(row = 6, column = 0, columnspan = 2)

self.settings_frame.grid(row=0, column=1, sticky='nsew')

self.process_button = ttk.Button(self,
                                  text = MAIN_WINDOW['process_image_button'][app_configuration.locale],
                                  command = lambda:
threading.Thread(target=self.process_image,args=()).start(),
                                  state = 'disabled')
self.process_button.place(relx = 0.5,
                          rely = 0.88,
                          anchor = 'center')

self.bind("<Map>", installed_models)

def recheck(self) -> None:
    self.model_list = installed_models()

```

```

class AdditionalSettings(ttk.Toplevel):
    def __init__(self, master, config_file):
        super().__init__(master)

        self.title(MAIN_WINDOW["additional_settings_title"][app_configuration.locale])
        self.config_file = config_file

        # Blip settings
        ttk.Label(self, text="Blip Image Eval Size:").grid(row=0, column=0, padx=10, pady=5,
sticky="w")
        self.blip_image_eval_size = ttk.Combobox(self, values=[256, 384, 512], state='readonly')
        self.blip_image_eval_size.grid(row=0, column=1, padx=10, pady=5)
        self.blip_image_eval_size.set(384)

        ttk.Label(self, text="Blip Max Length:").grid(row=1, column=0, padx=10, pady=5, sticky="w")
        self.blip_max_length = ttk.Combobox(self, values=[16, 32, 64], state='readonly')
        self.blip_max_length.grid(row=1, column=1, padx=10, pady=5)
        self.blip_max_length.set(config_file.caption_max_length)

        ttk.Label(self, text="Blip Offload:").grid(row=2, column=0, padx=10, pady=5, sticky="w")
        self.blip_offload = ttk.Combobox(self, values=[True, False], state='readonly')
        self.blip_offload.grid(row=2, column=1, padx=10, pady=5)
        self.blip_offload.set(config_file.caption_offload)

        ttk.Label(self, text="Caption Model Name:").grid(row=3, column=0, padx=10, pady=5,
sticky="w")
        self.caption_model_name = ttk.Combobox(self, values=list(CAPTION_MODELS.keys()),
state='readonly')
        self.caption_model_name.grid(row=3, column=1, padx=10, pady=5)
        self.caption_model_name.set(config_file.caption_model_name)

```

```

# Interrogator settings

ttk.Label(self, text="Chunk Size:").grid(row=4, column=0, padx=10, pady=5, sticky="w")
self.chunk_size = ttk.Combobox(self, values=[1024, 2048, 4096], state='readonly')
self.chunk_size.grid(row=4, column=1, padx=10, pady=5)
self.chunk_size.set(config_file.chunk_size)


ttk.Label(self, text="Flavor Intermediate Count:").grid(row=5, column=0, padx=10, pady=5,
sticky="w")
self.flavor_intermediate_count = ttk.Combobox(self, values=[1024, 2048, 4096], state='readonly')
self.flavor_intermediate_count.grid(row=5, column=1, padx=10, pady=5)
self.flavor_intermediate_count.set(config_file.flavor_intermediate_count)


# Buttons

self.save_button = ttk.Button(self,
text=MAIN_WINDOW["save_settings_button"][app_configuration.locale], command=self.save_settings)
self.save_button.grid(row=6, column=0, padx=10, pady=10, sticky="w")


self.close_button = ttk.Button(self, text=CLOSE_BUTTON[app_configuration.locale],
command=self.destroy)
self.close_button.grid(row=6, column=1, padx=10, pady=10, sticky="e")


def save_settings(self):
    settings = {
        "blip_max_length": int(self.blip_max_length.get()),
        "blip_offload": bool(self.blip_offload.get()),
        "caption_model_name": self.caption_model_name.get(),
        "chunk_size": int(self.chunk_size.get()),
        "flavor_intermediate_count": int(self.flavor_intermediate_count.get())
    }

# Implement your saving logic here

```

```

# Update config_file with new settings
self.config_file.caption_max_length = settings["blip_max_length"]
self.config_file.caption_offload = settings["blip_offload"]
self.config_file.caption_model_name = settings["caption_model_name"]
self.config_file.chunk_size = settings["chunk_size"]
self.config_file.flavor_intermediate_count = settings["flavor_intermediate_count"]

self.destroy()

def upload_image(self, *args):
    path = filedialog.askopenfilename()
    if path:
        self.image_file = Image.open(path)
        self.image = ImageTk.PhotoImage(self.image_file)
        self.canvas_for_image.create_image(0, 0, image=self.image, anchor='nw')
        if not self.PICTURE_EXISTS:
            self.upload_button.place_forget()
            self.canvas_for_image.bind('<Button-1>', self.upload_image)
            self.PICTURE_EXISTS = True
            self.canvas_for_image.event_generate("<Configure>",
width=self.canvas_for_image.winfo_width(), height=self.canvas_for_image.winfo_height())
            self.process_button['state'] = 'normal'

def stretch_image(self, event: object) -> None:
    if not self.image:
        return
    global resized_tk

```

[illegible]

```

        anchor = 'center',
        image = resized_tk)

def process_image(self) -> None:
    self.config_file.clip_model_path = os.path.join(os.path.dirname(__file__), 'models')
    self.process_button['state'] = 'disabled'

    window = self.ResultsWindow(self)
    from clip_interrogator import Interrogator

    start_time = time.time()
    if self.settings_compute_variable.get() == 'cloud':
        config_req = ""
        for key, elem in self.config_file.__dict__.items():
            if key == "data_path" or key == "device" or key == 'quiet' or key == 'cache_path':
                continue
            config_req = config_req + f"{key}={elem};"
        request = f"IMG|{os.path.basename(self.image_file.filename)}|{config_req[:-1]}|{self.settings_model_mode_variable.get()}|{USERNAME}"
        client = Client()
        prompts = client.send_image(request, self.image_file.filename)
        if prompts == 'no credit':
            messagebox.ok(MAIN_WINDOW["no_credits_message"][app_configuration.locale], "No
credits!")
        else:
            end_time = time.time()
    else:

```

```

        self.config_file.device = "cuda" if self.settings_device_variable.get().split(sep=":")[0][:3] ==
"GPU" else "cpu"

```

```

        ci = Interrogator(self.config_file)

```

```

        mode = self.settings_model_mode_variable.get()

```

```

        if mode == 'best':

```

```

            prompts = ci.interrogate(self.image_file)

```

```

        elif mode == 'classic':

```

```

            prompts = ci.interrogate_classic(self.image_file)

```

```

        elif mode == 'fast':

```

```

            prompts = ci.interrogate_fast(self.image_file)

```

```

        elif mode == 'negative':

```

```

            prompts = ci.interrogate_negative(self.image_file)

```

```

        end_time = time.time()

```

```

        window.destroy()

```

```

        self.ResultsWindow(self,

```

```

            self.config_file,

```

```

            self.settings_model_mode_variable.get(),

```

```

            self.settings_compute_variable.get(),

```

```

            self.image_file,

```

```

            prompts,

```

```

            round((end_time - start_time),3))

```

```

        self.process_button['state'] = 'enabled'

```

```

class ResultsWindow(ttk.Toplevel):

```

```

    def __init__(self, master, config_file=None, mode=None, compute=None, image_file=None,
result_prompt=None, computation_time=None):

```

```

        super().__init__(master)

```

```

        self.title(MAIN_WINDOW["results_title"][app_configuration.locale])

```



```

self.geometry("800x600")
self.maxsize(900, 600)
self.minsize(600,400)

if not config_file:
    self.loader()
else:
    self.window_pack(config_file, mode, compute, image_file, result_prompt, computation_time)

def window_pack(self, config_file, mode, compute, image_file, result_prompt, computation_time) ->
None:

    self.image_file = image_file
    self.image = ImageTk.PhotoImage(image_file)
    self.result_prompt = result_prompt

    self.columnconfigure(0, weight=1)
    self.columnconfigure(1, weight=1)
    self.rowconfigure(0, weight=1)

    # Left frame for image
    self.left_frame = ttk.Frame(self)
    self.left_frame.grid(row=0, column=0, sticky='nsew')
    self.left_frame.columnconfigure(0, weight=1)
    self.left_frame.rowconfigure(0, weight=1)

    self.canvas_for_image = Canvas(self.left_frame, borderwidth=0, highlightthickness=0,
relief='ridge')
    self.canvas_for_image.grid(row=0, column=0, sticky='nsew')

    # Right frame for settings and prompt

```

```

self.right_frame = ttk.Frame(self)
self.right_frame.grid(row=0, column=1, sticky='nsew')
self.right_frame.columnconfigure(0, weight=1)
self.right_frame.rowconfigure(0, weight=1)
self.right_frame.rowconfigure(1, weight=3)
self.right_frame.rowconfigure(2, weight=1)

# Settings frame
self.settings_frame = ttk.Frame(self.right_frame)
self.settings_frame.grid(row=0, column=0, sticky='nsew')
self.settings_frame.columnconfigure(0, weight=1)

settings_text = (
    f'{MAIN_WINDOW['device_label']}[app_configuration.locale]} {config_file.device}\n'
    f'{MAIN_WINDOW['model_label']}[app_configuration.locale]}
{config_file.clip_model_name}\n'
    f'{MAIN_WINDOW['mode_label']}[app_configuration.locale]} {mode}\n'
    f'{MAIN_WINDOW['compute_label']}[app_configuration.locale]} {compute}\n'
    f'{MAIN_WINDOW['computational_time_label']}[app_configuration.locale]}
{computation_time}s.\n'
)

self.settings_label = ttk.Label(self.settings_frame, text=settings_text, justify='left',
font=("Helvetica", 20))

self.settings_label.grid(row=0, column=0, padx=10, pady=10)

# Prompt frame
self.prompt_frame = ttk.Frame(self.right_frame)
self.prompt_frame.grid(row=1, column=0, sticky='nsew')
self.prompt_frame.columnconfigure(0, weight=1)
self.prompt_frame.rowconfigure(0, weight=1)

```

```

        self.prompt_label = ttk.Label(self.prompt_frame,
text=MAIN_WINDOW["results_prompt_label"][app_configuration.locale], justify='left', font=("Helvetica", 18))
        self.prompt_label.grid(row=0, column=0, padx=10, pady=5)

        self.prompt_text = ttk.Text(self.prompt_frame, height=10, wrap='word')
        self.prompt_text.grid(row=1, column=0, padx=10, pady=5, sticky='nsew')
        self.prompt_text.insert('1.0', self.result_prompt)
        self.prompt_text.configure(state='disabled')

    # Buttons frame
    self.buttons_frame = ttk.Frame(self.right_frame)
    self.buttons_frame.grid(row=2, column=0, pady=10)
    self.buttons_frame.columnconfigure([0, 1], weight=1)

    self.copy_button = ttk.Button(self.buttons_frame,
text=MAIN_WINDOW["copy_button"][app_configuration.locale], command=self.copy_to_clipboard)
    self.copy_button.grid(row=0, column=0, padx=5)

    self.close_button = ttk.Button(self.buttons_frame,
text=CLOSE_BUTTON[app_configuration.locale], command=self.destroy)
    self.close_button.grid(row=0, column=1, padx=5)

    self.resized_tk = None

    self.canvas_for_image.bind('<Configure>', self.stretch_image)
    self.canvas_for_image.bind('<Configure>', self.fill_image)

    self.canvas_for_image.event_generate("<Configure>",
width=self.canvas_for_image.winfo_width(), height=self.canvas_for_image.winfo_height())

    def copy_to_clipboard(self):
        pyperclip.copy(self.result_prompt)

```

```

def stretch_image(self, event: object) -> None:
    if not self.image:
        return

    width = event.width
    height = event.height

    resized_image = self.image_file.resize((width, height))
    self.resized_tk = ImageTk.PhotoImage(resized_image)
    self.canvas_for_image.create_image(0,
                                       0,
                                       image = resized_tk,
                                       anchor = 'nw')

def loader(self) -> None:
    loading = AnimatedGif(self)
    loading.pack(expand=True, fill='both')

def fill_image(self, event: object) -> None:
    if not self.image:
        return

    # current ratio
    canvas_ratio = event.width / event.height
    image_ratio = self.image_file.size[0] / self.image_file.size[1]

    if canvas_ratio > image_ratio: # canvas is wider
        width = int(event.width)
        height = int(width / image_ratio)

```

```

else:
    height = int(event.height)
    width = int(height * image_ratio)

    resized_image = self.image_file.resize((width, height))
    self.resized_tk = ImageTk.PhotoImage(resized_image)
    self.canvas_for_image.create_image(int(event.width / 2),
                                       int(event.height / 2),
                                       anchor = 'center',
                                       image = resized_tk)

class SettingsTab(ttk.Frame):
    def __init__(self, master):
        super().__init__(master)

        # what user can change in settings?
        # Language
        # Theme
        # auto-update option
        #
        # Click check updates
        # Some info about app
        #
        # self.write_settings()

    self.current_theme = self.master.master.current_theme

```

```

# grid
self.columnconfigure((0,3), weight=1, uniform='a')
self.columnconfigure((1,2), weight=4, uniform='a')
self.rowconfigure((0,1,2,3,4,5), weight=1, uniform='a')

# info about app

self.app_name_version = ttk.Label(self,
                                   text = f"{{ APP_INIT['APP_NAME'] }}. Version: {{ APP_INIT['VERSION'] }}")
self.app_name_version.grid(row = 0, column = 1, columnspan=2, sticky='nsew')

# language settings
self.language_setting_frame = ttk.Frame(self)
self.language_setting_label = ttk.Label(self.language_setting_frame,
                                         text = MAIN_WINDOW["language_label"][app_configuration.locale])
self.language_setting_variable = ttk.StringVar()
self.language_list = ["Українська", "English"]
self.language_setting_combobox = ttk.Combobox(self.language_setting_frame,
                                              values=["Українська", "English"],
                                              textvariable=self.language_setting_variable)

self.language_setting_combobox.set(self.language_list[0] if app_configuration.locale == 'UA' else
self.language_list[1])
self.language_setting_label.pack(side='left', fill='both')
self.language_setting_combobox.pack(side='left', fill='x', padx=5, pady=5)
self.language_setting_frame.grid(column=1, row=1, sticky='nsew')

# cmd setting
self.cmd_display_setting_frame = ttk.Frame(self)
self.cmd_display_setting_label = ttk.Label(self.cmd_display_setting_frame,

```

```

        text = MAIN_WINDOW["display_cmd_label"][app_configuration.locale])
self.cmd_display_setting_variable = ttk.IntVar()
self.cmd_display_setting_variable.set(app_configuration.cmd)
self.cmd_display_setting_checkbutton = ttk.Checkbutton(self.cmd_display_setting_frame,
                                                        variable=self.cmd_display_setting_variable,
                                                        )

self.cmd_display_setting_label.pack(side='left', fill='both')
self.cmd_display_setting_checkbutton.pack(side='left', fill='x', padx=5, pady=5)
self.cmd_display_setting_frame.grid(column=2, row=1, sticky='nsew')

ToolTip(self.cmd_display_setting_frame,
text=MAIN_WINDOW["cmd_tooltip"][app_configuration.locale], bootstyle=(INFO, INVERSE))

# save settings button

self.save_settings_button = ttk.Button(self,
                                       text = MAIN_WINDOW['save_settings_button'][app_configuration.locale],
                                       command = self.write_settings)
self.save_settings_button.grid(row = 5, column = 1, sticky='ew', padx=5)

# check updates button
self.check_update_button = ttk.Button(self,
                                       text =
MAIN_WINDOW["check_updates_button"][app_configuration.locale],
                                       command = self.check_updates)
self.check_update_button.grid(row = 5, column = 2, sticky='ew', padx=5)

# change theme button

```

```

self.change_theme_button = ttk.Button(self,
                                     text = MAIN_WINDOW["theme_button"][app_configuration.locale],
                                     command = self.change_current_theme)
self.change_theme_button.grid(row = 4, column = 2, sticky='ew', padx=5)

# autoupdate checkbox

self.autoupdate_check_frame = ttk.Frame(self)
self.autoupdate_setting_label = ttk.Label(self.autoupdate_check_frame,
                                           text =
MAIN_WINDOW["autoupdate_enable_label"][app_configuration.locale])
self.autoupdate_check_variable = ttk.IntVar()
self.autoupdate_check_variable.set(app_configuration.auto_update)
self.autoupdate_setting_checkbutton = ttk.Checkbutton(self.autoupdate_check_frame,
                                                       variable=self.autoupdate_check_variable,
                                                       )
self.autoupdate_setting_label.pack(side='left', fill='both')
self.autoupdate_setting_checkbutton.pack(side='left', fill='x', padx=5, pady=5)
self.autoupdate_check_frame.grid(column=1, row=2, sticky='nsew')
ToolTip(self.autoupdate_check_frame,
text=MAIN_WINDOW["autoupdate_tooltip"][app_configuration.locale], bootstyle=(INFO, INVERSE))

# save cache model checkbox

self.cache_model_frame = ttk.Frame(self)
self.cache_model_label = ttk.Label(self.cache_model_frame,
                                   text = MAIN_WINDOW["save_cache_label"][app_configuration.locale])
self.cache_model_variable = ttk.IntVar()
self.cache_model_variable.set(app_configuration.cache_save)
self.cache_model_checkbutton = ttk.Checkbutton(self.cache_model_frame,

```



```

        variable=self.cache_model_variable,
    )

    self.cache_model_label.pack(side='left', fill='both')

    self.cache_model_checkbutton.pack(side='left', fill='x', padx=5, pady=5)

    self.cache_model_frame.grid(column=2, row=2, sticky='nsew')

    ToolTip(self.cache_model_frame,
text=MAIN_WINDOW["save_cache_tooltip"][app_configuration.locale], bootstyle=(INFO, INVERSE))

def check_updates(self, silent=True) -> None:
    client = Client()

    result = client.process_request(f"UPD|{APP_INIT['VERSION']}")

    if result == "True" and not silent:
        MessageBox.ok(MAIN_WINDOW["update_message_newest"][app_configuration.locale],
"ImgProPlus Update")
    else:
        mb =
MessageBox.yesno(f"{MAIN_WINDOW['update_message_outdated'][app_configuration.locale]}{result}",
"ImgProPlus Update")

        if mb == "Yes":
            print("Running update")

            # implement logic here

def change_current_theme(self) -> None:
    if self.current_theme == 'darkly':
        self.current_theme = 'journal'

        self.master.master.current_theme = 'journal'

        self.master.master.style.theme_use(self.current_theme)
    else:
        self.current_theme = 'darkly'

```

```

self.master.master.current_theme = 'darkly'

self.master.master.style.theme_use(self.current_theme)


def write_settings(self) -> None:

    settings_dictionary = {"locale" : "EN" if self.language_setting_variable.get() == "English" else
"UA",

                           "cmd" : self.cmd_display_setting_variable.get(),

                           "theme" : self.master.master.current_theme,

                           "auto_update" : self.autoupdate_check_variable.get(),

                           "cache_model" : self.cache_model_variable.get()

                           }


    file_path = os.path.dirname(os.path.realpath(__file__)) + "\\user_settings"

    with open(file_path, "w") as file:

        for key, element in settings_dictionary.items():

            file.write(f"{key}:{element}\n")


class DocumentationTab(ttk.Frame):

    def __init__(self, master):

        super().__init__(master)

        self.docs_uploaded = False

        #self.bind("<Configure>", self.on_frame_pack)

        self.frame = tkinterweb.HtmlFrame(self, messages_enabled=False)

        self.test()


    def on_frame_pack(self, event):

        if self.master.pack_info() is not None and not self.docs_uploaded:

            try:

```

```

        if app_configuration.locale == 'EN':
            self.frame.load_file(os.path.join(os.path.dirname(__file__), 'docs.html'))
        else:
            self.frame.load_file(os.path.join(os.path.dirname(__file__), 'docs_u.html'))
        self.frame.pack(expand=True, fill='both')
        self.docks_uploaded = True
    except FileNotFoundError as e:
        logger.error("Not found documentation file")

def test(self):
    try:
        if app_configuration.locale == 'EN':
            self.frame.load_file(os.path.join(os.path.dirname(__file__), 'docs.html'))
        else:
            self.frame.load_file(os.path.join(os.path.dirname(__file__), 'docs_u.html'))
        self.frame.pack(expand=True, fill='both')
        self.docks_uploaded = True
    except FileNotFoundError as e:
        logger.error("Not found documentation file")

class SendQuestionTab(ttk.Frame):
    def __init__(self, master=None, **kw):
        super().__init__(master, **kw)
        # self.style = Style(theme="yeti") # Change the theme as needed

        self.columnconfigure(1, weight=6, uniform='a')

```

```

self.columnconfigure((0,2), weight=1, uniform='a')
self.rowconfigure(0, weight=5, uniform='a')
self.rowconfigure(1, weight=3, uniform='a')

self.create_widgets()

def create_widgets(self):
    # Create text field
    self.text_field = ScrolledText(self, padding=5, height=10, autohide=True) # Adjust width
    self.text_field.grid(row=0, column=1, padx=10, pady=10, sticky='nsew') # Adjust padx and pady

    # Create send button
    self.send_button = ttk.Button(self,
text=MAIN_WINDOW["send_q_button"][app_configuration.locale], command=self.send_message)
    self.send_button.grid(row=1, column=1, padx=150, pady=(0, 10), sticky="ew")

def send_message(self):
    client = Client()
    message = client.process_request(f'MSG|{self.text_field.get("1.0", ttk.END)}|{USERNAME}')
    if message == "True":
        Messagebox.ok(MAIN_WINDOW["message_sent"][app_configuration.locale],
MAIN_WINDOW["message_sent_title"][app_configuration.locale])
        self.text_field.delete(1.0, ttk.END)
    else:
        Messagebox.show_error(MAIN_WINDOW["message_sent_error"][app_configuration.loclae],
"Something went wrong!")

class ExtensionsTab(ttk.Frame):
    def __init__(self, master: ttk.Frame):

```

```

super().__init__(master=master)

self.create_widgets()

def create_widgets(self):
    # Main frame
    main_frame = ttk.Frame(self)
    main_frame.pack(fill='both', expand=True)

    # Treeview frame on the left
    tree_frame = ttk.Frame(main_frame, width=200)
    tree_frame.pack(side='left', fill='y')

    label = ttk.Label(tree_frame, text=MAIN_WINDOW["models_label"][app_configuration.locale],
font=("Helvetica", 14, "bold"))
    label.pack(side='top', pady=6)

    self.tree = ttk.Treeview(tree_frame)
    self.tree.pack(fill='both', expand=True, padx=3)

    text_frame = ttk.Frame(main_frame)
    text_frame.pack(side='right', fill='both', expand=True)

    self.button_frame = ttk.Frame(text_frame)
    self.button_frame.pack(side='top', fill='x')

    self.button = ttk.Button(self.button_frame,
text=MAIN_WINDOW["download_button"][app_configuration.locale],state='disabled')
    self.button.pack(side='right', padx=5, pady=5)

    # Text frame on the right

```

```

self.text = ttk.Text(text_frame, wrap='word', state='disabled')
self.text.pack(fill='both', expand=True)

# Button frame at the bottom of the text frame

# Adding some sample elements
self.elements = {}
self.get_models()

for element in self.elements:
    self.tree.insert("", 'end', text=element, tags=())

# Bind the treeview selection
self.tree.bind('<<TreeviewSelect>>', self.on_tree_select)

def on_tree_select(self, event):
    self.selected_item = self.tree.selection()[0]
    self.element_name = self.tree.item(self.selected_item, 'text')
    description = self.elements.get(self.element_name,
MAIN_WINDOW["description_unavailable_label"][app_configuration.locale])
    self.text.configure({"state": "normal"})
    self.text.delete(1.0, ttk.END)
    self.text.insert(ttk.END, description[0])
    self.text.configure({"state": "disabled"})

models = installed_models()
filtered_element = self.element_name.replace("/", "_").replace("-", "_").replace('.', "_").lower()
if description[1] == "1" and subscription_status_verification() == "No":

```

```

        self.button.configure(text =
MAIN_WINDOW["buy_premium_button"][app_configuration.locale],

                                command = self.attempt_without_premium,

                                bootstyle = "warning",

                                state='enabled')

elif filtered_element in [model.replace("/", "_").replace("-", "_").replace('.', "_").lower() for model in
models]:

    self.button.configure(text = MAIN_WINDOW["installed_label"][app_configuration.locale],
state='disabled')

else:

    self.button.configure(text = MAIN_WINDOW["download_button"][app_configuration.locale],

                                command = lambda: threading.Thread(target = self.download_model,
args=(self.element_name,)).start(),

                                state = 'enabled',

                                bootstyle='default')

```

```

def attempt_without_premium(self):

    user_answer =
Messagebox.okcancel(MAIN_WINDOW["proposition_message"][app_configuration.locale],
MAIN_WINDOW["proposition_title"][app_configuration.locale])

    if user_answer:

        PaymentWindow(self, serv_conn=True)

def get_models(self):

    client = Client()

    results = client.process_request(f"EXT")

    if results == "Error":

        Messagebox.show_error(MAIN_WINDOW["error_model_download"][app_configuration.locale],
MAIN_WINDOW["error_model_download_title"][app_configuration.locale])

```

```

models = results.split(sep="|")
for model in models:
    temp = model.split(sep=";")
    self.elements[temp[0]] = [temp[1], temp[2]]
#print(self.master.frames[MAIN_WINDOW['notebook_mainpage_tab']][app_configuration.locale]])

# def download_model_with_progress(self, model_name, result):
#     with redirect_stdout():
#         process = subprocess.Popen(command, stdout=subprocess.PIPE, stderr=subprocess.STDOUT,
shell=True, universal_newlines=True)

#         # Initialize the progress bar
#         for line in process.stdout:
#             # Parse the progress from each line of output
#             progress = parse_progress(line)
#             if not progress:
#                 continue
#             self.progress_var.set(int(progress))
#             # Update the progress bar

#         process.wait()

# def parse_progress(line):
#     try:
#         if "%" in line:
#             progress = int(line.split('%')[0].split()[-1]) # Adjust based on your command's output
#             print(progress)
#             return progress

```



```

# except Exception as e:
#     return None
# return None

def download_model(self, model_name):
    from clip_interrogator import download_model_gui
    try:
        self.button.pack_forget()
        self.progress_var = ttk.IntVar()
        self.progress = ttk.Progressbar(
            master = self.button_frame,
            length = 100,
            maximum = 100,
            mode = 'indeterminate',
            variable = self.progress_var,
            bootstyle = 'success-striped'
        )
        self.progress.pack(fill='both', expand=True, padx=20, pady=13)
        self.progress.start()
        result = []
        model_thread = threading.Thread(target = download_model_gui, args=(model_name, result))
        model_thread.start()
        self.button.configure(bootstyle = 'success-outline')
        while model_thread.is_alive():
            for i in range(0,4):
                self.button.configure(text =
f"{MAIN_WINDOW['downloading_model']}[app_configuration.locale]}{i*':.'}")
                time.sleep(1)
            if result[0]:

```

```

        self.progress.pack_forget()

        self.button.pack(side='right', padx=5, pady=5)

        self.button.configure(text = MAIN_WINDOW["installed_label"][app_configuration.locale],
state='disabled')

        self.get_models()

        self.master.update_idletasks()

    except Exception as e:

        logger.error(f"Error with model downloading. {e}")

```

```

class PaymentWindow(ttk.Toplevel):

```

```

    def __init__(self, master, serv_conn=False):

```

```

        super().__init__(master)

```

```

        self.title(MAIN_WINDOW["payment_window_title"][app_configuration.locale])

```

```

        self.geometry('900x700')

```

```

        self.minsize(600, 700)

```

```

        self.maxsize(1000, 700)

```

```

        self.resizable(True, True)

```

```

        self.promotion_frames = []

```

```

        if serv_conn:

```

```

            client = Client()

```

```

            self.data = client.process_request(f"SUB|{app_configuration.locale}")

```

```

            self.display_promotions()

```

```

        else:

```

```

            self.data = None

```

```

        # Add close button at the bottom

        self.close_button = ttk.Button(self, text=CLOSE_BUTTON[app_configuration.locale],
command=self.destroy)

        self.close_button.pack(side="bottom", pady=10)

def display_promotions(self):

    promotions = self.data.split('|')

    for promotion in promotions:

        promotion_data = promotion.split(';')

        description = promotion_data[0]

        price = promotion_data[1]

        details = promotion_data[2]

        # Create a frame for each promotion

        promotion_frame = ttk.Frame(self, padding="10")

        promotion_frame.pack(fill='both', pady=10, padx=10, expand=True)

        self.promotion_frames.append(promotion_frame)

        # Add description label

        description_label = ttk.Label(promotion_frame, text=description, font=("Helvetica", 16, "bold"))

        description_label.pack(anchor='w')

        # Add details label

        details_label = ttk.Label(promotion_frame, text=details, font=("Helvetica", 12))

        details_label.pack(anchor='w')

        # Add price label

```

```

        price_label = ttk.Label(promotion_frame,
text=f"{MAIN_WINDOW['price_label'][app_configuration.locale]} ${price}", font=("Helvetica", 12, "italic"))
        price_label.pack(anchor='w')

    # Add a pay button

    button = ttk.Button(promotion_frame,
text=MAIN_WINDOW["pay_button"][app_configuration.locale], command=lambda d=description, p=price:
self.make_payment(d, p))

    button.pack(anchor='e', pady=5)

def make_payment(self, description, price):

    # Clear all promotion frames

    for frame in self.promotion_frames:

        frame.pack_forget()

    # Form QR code and output it in the window

    url = "http://www.liqpay.ua/checkout?action=pay?currency=USD?version=3?amount=" + price +
"?description=" + description

    self.create_qr_code(url, description, price)

    # Add "Back" button to return to promotions

    self.back_button = ttk.Button(self,
text=MAIN_WINDOW["back_button"][app_configuration.locale], command=self.back_to_promotions)

    self.back_button.pack(pady=10)

    # Hide the close button

    self.close_button.pack_forget()

def back_to_promotions(self):

    # Remove QR code and instruction label

```

```

self.qr_label.pack_forget()
self.instruction_label.pack_forget()
self.payment_info_label.pack_forget()
self.back_button.pack_forget()
self.direct_link_label.pack_forget()

# Show promotion frames again
for frame in self.promotion_frames:
    frame.pack(fill='both', pady=10, padx=10, expand=True)

# Show the close button again
self.close_button.pack(side="bottom", pady=10)

def create_qr_code(self, url, description, price):
    img = qrcode.make(url, image_factory=qrcode.image.pil.PilImage)

    qr_photo = ImageTk.PhotoImage(img.get_image())

    self.qr_label = ttk.Label(self, image=qr_photo)
    self.qr_label.image = qr_photo # keep a reference to prevent garbage collection
    self.qr_label.pack()

# Add direct link label
self.direct_link_label = ttk.Label(self, text=f"Direct Link: {url}")
self.direct_link_label.pack(pady=5)

import webbrowser
self.direct_link_label.bind("<Button-1>", lambda e: webbrowser.open_new(url))

self.payment_info_label = ttk.Label(self, text=f"{description}\n${price}")

```

```
self.payment_info_label.pack(pady=10)
```

```
self.instruction_label = ttk.Label(self,
text=MAIN_WINDOW["scan_code_tip"][app_configuration.locale])
self.instruction_label.pack(pady=10)
```

```
def respond_to_user(self) -> None:
```

```
    #!/bin/bash

    # PUBLIC_KEY='your_public_key'

    # PRIVATE_KEY='your_private_key'

    # API_URL='https://www.liqpay.ua/api/request'

    # JSON="{
    # \"action\" : \"ticket\",
    #   \"version\" : 3,
    #   \"public_key\" : \"${PUBLIC_KEY}\",
    #   \"order_id\" : \"order_id_1\",
    #   \"email\" : \"email@gmail.com\"
    # }"

    # # DATA is base64_encode result from JSON string
    # DATA=$(echo -n ${JSON} | base64)

    # # SIGNATURE is base64 encode result from sha1 binary hash from concatenate string
    # ${PRIVATE_KEY}${DATA}${PRIVATE_KEY}

    # SIGNATURE=$(echo -n "${PRIVATE_KEY}${DATA}${PRIVATE_KEY}" | openssl dgst -
binary -sha1 | base64)

    # # REQ is json response from liqpay

    # REQ=$(curl --silent -XPOST ${API_URL} --data-urlencode data="${DATA}" --data-urlencode
signature="${SIGNATURE}")

    # echo "Result: ${REQ}"

    pass
```

```

class AnimatedGif(ttk.Frame):

    def __init__(self, master):
        super().__init__(master, width=400, height=300)

        # open the GIF and create a cycle iterator
        file_path = Path(__file__).parent.parent / "giphy_2.gif"
        with Image.open(file_path) as im:
            # create a sequence
            sequence = ImageSequence.Iterator(im)
            images = [ImageTk.PhotoImage(s) for s in sequence]
            self.image_cycle = cycle(images)

            # length of each frame
            self.framerate = im.info["duration"]

        self.img_container = ttk.Label(self, image=next(self.image_cycle))
        self.img_container.pack(fill="both", expand="yes")
        self.after(self.framerate, self.next_frame)

    def next_frame(self):
        """Update the image for each frame"""
        self.img_container.configure(image=next(self.image_cycle))
        self.after(self.framerate, self.next_frame)

def installed_models(*args) -> list [str]:
    # Construct the path to the desired directory
    cache_clip_path = os.path.dirname(os.path.realpath(__file__)) + "\\models"
    models = []

```

```

if os.path.exists(cache_clip_path):
    elements = os.listdir(cache_clip_path)
    for element in elements:
        if element[:6] == "models":
            models.append(element.split(sep="CLIP-")[1])
        elif element.split(sep=".")[1] == 'pt':
            models.append(element.split(sep=".")[0])
    return models
else:
    return []

def subscription_status_verification() -> str:
    client = Client()
    result = client.process_request(f"SSV|{USERNAME}")
    if result != "False":
        return result
    else: return "No"

if __name__ == '__main__':
    print("nothing")
    root = ttk.Window()
    PaymentWindow(root)
    root.mainloop()

```

A.7. localization.py

```

CLOSE_BUTTON = {"EN": "Close", "UA": "Закрити"}

LOGIN_PAGE = {
    "login_label": {"EN": "Login (mail)", "UA": "Увійти (пошта)"},

```



```

"auth_label": {"EN": "Authorization", "UA": "Авторизація"},
"password_label": {"EN": "Password", "UA": "Пароль"},
"login_button": {"EN": "Login", "UA": "Увійти"},
"forgot_password_button": {"EN": "Forgot password?", "UA": "Забули пароль?"},
"new_user_label": {"EN": "New user?", "UA": "Новий користувач?"},
"registration_button": {"EN": "Registration", "UA": "Реєстрація"},
"mail_label": {"EN": "Email", "UA": "Електронна пошта"},
"password_repeat_label": {"EN": "Password confirmation", "UA": "Підтвердження паролю"},
"confirmation_code_label": {"EN": "Confirmation code", "UA": "Код підтвердження"},
"return_button": {"EN": "Back", "UA": "Назад"},
"send_confirm_code": {"EN": "Send confirmation code", "UA": "Надіслати код підтвердження"},
"wrong_email_label": {"EN": "Wrong format!", "UA": "Неправильний формат!"},
"password_nomatch_label": {"EN": "Passwords are different!", "UA": "Паролі відрізняються!"},
"wrong_confirmation_code_label": {"EN": "Wrong code!", "UA": "Неправильний код!"},
"password_validation_error": {"EN": "Passwords must have at least 8 characters and contain at least
two of the following: uppercase letters, lowercase letters, numbers, and symbols", "UA": "Пароль повинен
містити принаймні 8 символів та складатися з принаймні двох з наступних: великих літер, малих літер,
цифр та символів."},
"email_validation_error": {"EN": "Wrong format of email. Check it again.", "UA": "Неправильний
формат пошти. Перевірте будь ласка ще раз."},
"message_wrong_email_or_pass": {"EN": "Wrong email or password!", "UA": "Неправильна пошта
або пароль!"},
"registration_title": {"EN": "Registration", "UA": "Реєстрація"},
"email_already_taken_message": {"EN": "Email is already taken. Please use another or try to restore
password.", "UA": "Така пошта вже зайнята. Будь ласка, використайте іншу або спробуйте відновити
пароль."},
"email_already_taken_message_title": {"EN": "Email already taken", "UA": "Пошта вже зайнята"},
"code_sent_message": {"EN": "Your code sent to your malibox. It lasts only 10 minutes.", "UA": "Ваш
код надіслано на вашу пошту. Він діє лише 10 хвилин."},
"code_sent_message_title": {"EN": "Code sent", "UA": "Код надіслано"},
"button_confirm_code": {"EN": "Confirm code", "UA": "Підтвердити код"},
"code_approved_msg": {"EN": "Code approved", "UA": "Код підтверджено"},
"code_approved_msg_title": {"EN": "Code approved", "UA": "Код підтверджено"},

```

"code_not_approved_msg": {"EN": "Wrong code. Please check it again, or contact QA using address

"code_not_approved_msg_title": {"EN": "Wrong code", "UA": "Неправильний код"},

"user_registered_msg": {"EN": "You're registered. Login manually or click ok to restart the app please.", "UA": "Ви зареєстровані. Будь ласка, зв'яжіться з адміністратором або натисніть ок, щоб перезапустити додаток."},

"user_registered_msg_title": {"EN": "Registered", "UA": "Зареєстровано"},

forgot password page

"forgot_window_user_email_label": {"EN": "Input your account email!", "UA": "Введіть пошту від свого аккаунту!"},

"forgot_window_user_email_button": {"EN": "Send!", "UA": "Відправити!"},

"forgot_window_confirmation_code_label": {"EN": "Code sent to your email! Please input code below.", "UA": "Код відправлено на вашу пошту! Будь ласка введіть код нижче."},

"forgot_window_confirmation_code_button": {"EN": "Confirm!", "UA": "Підтвердити!"},

"theme_button": {"EN": "Theme", "UA": "Тема"},

"recover_password_title": {"EN": "Recover password", "UA": "Відновлення пароля"},

"wrong_confirmation_code_msg": {"EN": "Confirmation code error. Please try again", "UA": "Неправильний код підтвердження. Будь ласка, спробуйте ще раз"},

"wrong_confirmation_code_msg_title": {"EN": "Confirmation code error.", "UA": "Неправильний код підтвердження."},

"passwords_dont_match": {"EN": "Passwords don't match", "UA": "Паролі не співпадають"},

"passwords_dont_match_title": {"EN": "Passwords don't match", "UA": "Паролі не співпадають"},

"error_changing_password": {"EN": "Error changing password. Please try again.", "UA": "Помилка при зміні пароля. Будь ласка, спробуйте ще раз."},

"error_changing_password_title": {"EN": "Error changing password.", "UA": "Помилка при зміні пароля."},

"password_changed_successfully": {"EN": "Password changed successfully", "UA": "Пароль змінено"},

"password_changed_successfully_title": {"EN": "Password changed successfully", "UA": "Пароль змінено"},

```
"email_doesnt_exist_msg": {"EN": "Email doesn't exist. Please try again.", "UA": "Пошта не існує.  
Будь ласка, спробуйте ще раз."},
```

```
"email_doesnt_exist_msg_title": {"EN": "Email doesn't exist.", "UA": "Пошта не існує."},
```

```
"send_password_button": {"EN": "Send", "UA": "Відправити"},
```

```
}
```

```
MAIN_WINDOW = {
```

```
  # main page
```

```
  "notebook_mainpage_tab": {"EN": "Main Page", "UA": "Головна сторінка"},
```

```
  "notebook_extensions_tab": {"EN": "Models", "UA": "Моделі"},
```

```
  "notebook_qa_tab": {"EN": "Q&A", "UA": "Питання та відповіді"},
```

```
  "notebook_docs_tab": {"EN": "Documentation", "UA": "Документація"},
```

```
  "notebook_settings_tab": {"EN": "Settings", "UA": "Налаштування"},
```

```
  "notebook_account_tab": {"EN": "Account", "UA": "Обліковий запис"},
```

```
  "image_processing_settings_label": {"EN": "Image Processing Settings", "UA": "Налаштування  
обробки зображень"},
```

```
  "additional_settings_button": {"EN": "Additional Settings", "UA": "Додаткові налаштування"},
```

```
  "process_image_button": {"EN": "Process Image", "UA": "Обробити зображення"},
```

```
  "upload_image_button": {"EN": "Upload Image", "UA": "Завантажити зображення"},
```

```
  "device_label": {"EN": "Device:", "UA": "Пристрій:"},
```

```
  "model_label": {"EN": "Model:", "UA": "Модель:"},
```

```
  "mode_label": {"EN": "Mode:", "UA": "Режим:"},
```

```
  "compute_label": {"EN": "Compute:", "UA": "Виконання:"},
```

```
  "no_credits_message": {"EN": "No credits left!", "UA": "Немає кредитів!"},
```

```
  # account tab
```

```
  "subscription_label": {"EN": "Subscription Status: ", "UA": "Статус підписки:"},
```

```
  "credit_label": {"EN": "Credits:", "UA": "Кредитів:"},
```

```
  "logout_button": {"EN": "Logout", "UA": "Вийти"},
```

```
  "change_password_button": {"EN": "Change Password", "UA": "Змінити пароль"},
```

```
  "purchase_subscription_button": {"EN": "Purchase Subscription", "UA": "Придбати підписку"},
```

"unlimited_credits_label": {"EN": "Unlimited", "UA": "Необмежено"},

"username_label": {"EN": "Username:", "UA": "Ім'я користувача:"},

"sub_status_no": {"EN": "No", "UA": "Hi"},

"sub_status_yes": {"EN": "Yes", "UA": "Так"},

#additional settings tab

"additional_settings_title": {"EN": "Additional Settings", "UA": "Додаткові налаштування"},

"save_settings_button": {"EN": "Save Settings", "UA": "Зберегти налаштування"},

setting tab

"theme_button": {"EN": "Theme", "UA": "Тема"},

"save_settings_button": {"EN": "Save Settings", "UA": "Зберегти налаштування"},

"check_updates_button": {"EN": "Check Updates", "UA": "Перевірити оновлення"},

"language_label": {"EN": "Language:", "UA": "Мова:"},

"display_cmd_label": {"EN": "Display CMD:", "UA": "Показати CMD:"},

"autoupdate_enable_label": {"EN": "Autoupdate enabled:", "UA": "Включено автоматичне оновлення:"},

"save_cache_label": {"EN": "Save cache files:", "UA": "Зберегти файли кешу:"},

"cmd_tooltip": {"EN": "When enabled, shows you CMD with progress", "UA": "Якщо ввімкнено, показує вам CMD з прогресом"},

"autoupdate_tooltip": {"EN": "When enabled, automatically checks for updates", "UA": "Якщо ввімкнено, автоматично перевіряє наявність оновлень"},

"save_cache_tooltip": {"EN": "When enabled, increases performance of model on your machine, by saving safetensor files. Read more about it in Docs.", "UA": "Якщо ввімкнено, збільшує продуктивність моделі на вашій машині, зберігаючи файли safetensor. Докладніше прочитайте у Документації"},

"update_message_outdated": {"EN": "Your application is out dated. New version available - ", "UA": "Ваша програма застаріла. Доступна нова версія - "},

"update_message_newest": {"EN": "Your application is up to date.", "UA": "Ваша програма оновлена."},

results tab

"results_title": {"EN": "Results", "UA": "Результати"},

"computational_time_label": {"EN": "Computational Time:", "UA": "Час обчислення:"},

"results_prompt_label": {"EN": "Resulting Prompt:", "UA": "Отриманий запит:"},

"copy_button": {"EN": "Copy", "UA": "Копіювати"},

sendqt

"send_q_button": {"EN": "Send!", "UA": "Надіслати!"},

"message_sent": {"EN": "Your message was sent!", "UA": "Ваше повідомлення надіслано!"},

"message_sent_title": {"EN": "Message sent", "UA": "Повідомлення надіслано"},

"message_sent_error": {"EN": "Message wasn't delivered.", "UA": "Повідомлення не надіслано."},

extensions tab

"models_label": {"EN": "Models", "UA": "Моделі"},

"download_button": {"EN": "Download", "UA": "Завантажити"},

"description_unavailable_label": {"EN": "Description unavailable", "UA": "Опис недоступний"},

"buy_premium_button": {"EN": "Buy Premium", "UA": "Купити Преміум"},

"installed_label": {"EN": "Installed", "UA": "Встановлено"},

"proposition_message": {"EN": "You don't have premium and can't download this model. \nTo do so you need premium subscription. \nOpen Purchase Window?", "UA": "У вас немає преміум і ви не можете завантажити цю модель. \nДля цього вам потрібно підписку преміум. \nВідкрити вікно оплати?"},

"proposition_title": {"EN": "Buy Premium", "UA": "Купити Преміум"},

"error_model_download": {"EN": "Error while downloading model", "UA": "Помилка при завантаженні моделі"},

"error_model_download_title": {"EN": "Error", "UA": "Помилка"},

"downloading_label": {"EN": "Downloading", "UA": "Завантаження"},

payment window

"payment_window_title": {"EN": "Payment Window", "UA": "Вікно оплати"},

"price_label": {"EN": "Price:", "UA": "Ціна:"},

```

"pay_button": {"EN": "Pay", "UA": "Оплатити"},
"back_button": {"EN": "Back", "UA": "Назад"},
"scan_code_tip": {"EN": "Scan QR code to pay", "UA": "Сканувати QR код для оплати"},

}

```

```

MESSAGES = {

```

```

    "unhandled_exception" : {"EN": "Whoops. Unhandled exception happened. Do you want to send logs
to devs to provide info about it?", "UA": "Ой. Непередбачувана помилка. Ви хочете надіслати логи, щоб
надати інформацію розробникам?"},

```

```

}

```

A.8. settings.py

```

import locale

```

```

import os

```

```

class ApplicationConfiguration():

```

```

    def __init__(self):

```

```

        # values from file

```

```

        self.locale = "UA" if locale.getdefaultlocale()[0] == "uk_UA" else "EN"

```

```

        self.theme = "darkly"

```

```

        self.cache_save = 0

```

```

        self.auto_update = 0

```

```

        self.cmd = 0

```

```

        self.app_name = "ImgProPlus"

```

```

        self.version = "1.0.1"

```

```

self.seconds_to_wait = 1

file_path = os.path.dirname(os.path.realpath(__file__)) + "\\user_settings"
if os.path.exists(file_path):
    with open(file_path, 'r') as file:
        lines = file.readlines()
        for line in lines:
            key, value = line.strip().split(":")
            setattr(self, key.strip(), value.strip())
else:
    settings_dictionary = {
        "locale": "EN",
        "cmd" : 0,
        "theme" : "darkly",
        "auto_update" : 0,
        "cache_save" : 0
    }
    with open(file_path, 'w') as file:
        for key, element in settings_dictionary.items():
            file.write(f"{key}:{element}\n")

def print_config(self):
    for key, value in self.__dict__.items():
        print(f"{key}: {value}")

```

```

STYLE = {
    "REGISTER_FONT": ("Calibri", "20"),
    "AUTH_FONT": ("Calibri", "20")
}

APP_INIT = {
    "APP_NAME": "ImgProPlus",
    "VERSION": "1.0.1"
}

SERV_IP = {"IP": "127.0.0.1", "PORT": 65432}

# basic text style
TEXT_BASIC_STYLE = 0

# serv address

if __name__ == '__main__':
    conf = ApplicationConfiguration()
    conf.print_config()

```

A.9. client.py

```

import socket
import os
from settings import SERV_IP

class Client():
    def __init__(self):

```



```

self.addr = (SERV_IP["IP"], SERV_IP['PORT'])

self.buffer = 4096

#self.run_client()

def run_client(self):
    # create a socket object
    client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

    server_ip = "127.0.0.1" # replace with the server's IP address connect_client(), send_request(),
close_connection(), update_application(), download_model(), send_payment(), send_logs()

    server_port = 65432 # replace with the server's port number

    # establish connection with server
    client.connect((server_ip, server_port))

    try:
        while True:
            # get input message from user and send it to the server
            msg = input("Enter message: ")
            client.send(msg.encode("utf-8")[:self.buffer])

            # receive message from the server
            response = client.recv(self.buffer)
            response = response.decode("utf-8")

            # if server sent us "closed" in the payload, we break out of
            # the loop and close our socket
            if response.lower() == "closed":

```

```

        break

    print(f"Received: {response}")
except Exception as e:
    print(f"Error: {e}")
finally:
    # close client socket (connection to the server)
    client.close()

def process_request(self, request: str, image_path=None) -> str:
    client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

    client.connect(self.addr)

    try:
        client.send(request.encode("utf-8")[:self.buffer])

        response = client.recv(self.buffer)
        response = response.decode("utf-8")

        return response

    except Exception as e:
        print(f"Error: {e}")
    finally:
        # close client socket (connection to the server)
        client.close()

def send_image(self, request: str, image_path: str) -> str:
    try:

```

```

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect(self.addr)
s.send(request.encode('utf-8'))
response = s.recv(self.buffer).decode('utf-8')
if response == "no credit":
    return response
with open(image_path, 'rb') as f:
    file_data = f.read()
s.sendall(file_data)
s.sendall(b"<END>")
response = s.recv(self.buffer).decode('utf-8')
prompts = s.recv(self.buffer).decode('utf-8')
s.close()
return prompts
except Exception as e:
    print(f"Error: {e}")

if __name__ == '__main__':
    client = Client().run_client()

    # test of image sending

    # client = Client()

    # file = "C:\\Users\\nyokayo\\Desktop\\picture.jpg"

    # request = "IMG|picta.jpg|caption_max_length=32;caption_model_name=blip-
large;caption_offload=True;clip_model_name=ViT-L-
14/openai;clip_model_path=None;clip_offload=False;download_cache=True;chunk_size=2048;flavor_intermedi
ate_count=2048|fast|rapperorwhat@gmail.com"

    # res = client.send_image(request, file)

    # print(res)

```

```
#client.send_image("")
# import os
# print(os.path.getsize(file))
```

A.10. addons.py

```
import os, platform, subprocess, re
import subprocess
import bcrypt
from dataclasses import dataclass
from typing import Optional

def get_processor_name():
    if platform.system() == "Windows":
        var = subprocess.check_output(['wmic', 'cpu', 'get', 'name']).decode('utf-8')
        var = var.replace(" ", "").replace("\n", "").replace("\r", "")
        return var[4:]
    elif platform.system() == "Darwin":
        os.environ['PATH'] = os.environ['PATH'] + os.pathsep + '/usr/sbin'
        command = "sysctl -n machdep.cpu.brand_string"
        return subprocess.check_output(command).decode('utf-8').strip()
    elif platform.system() == "Linux":
        command = "cat /proc/cpuinfo"
        all_info = subprocess.check_output(command, shell=True).decode().strip()
        for line in all_info.split("\n"):
            if "model name" in line:
                return re.sub( ". *model name.*:", "", line, 1)
        return ""

def get_gpus():
```

```

output = subprocess.check_output("nvidia-smi -L", shell=True)
output = output.decode("utf-8").strip().split("\n")
gpus = []
for line in output:
    line, _ = line.split("(", 1)
    gpu_id, gpu_name = line.strip().split(":")
    gpus.append((gpu_id.strip(), gpu_name.strip()))
return gpus

def hash_password(password):
    # Generate a salt
    salt = bcrypt.gensalt()
    # Hash the password
    hashed = bcrypt.hashpw(password.encode('utf-8'), salt)
    return hashed.decode('utf-8')

if __name__ == '__main__':

    gpus = get_gpus()
    for gpu_id, gpu_name in gpus:
        print(f"{gpu_id}: {gpu_name}")

    print(get_processor_name(), end="")

    #print(get_processor_name())

@dataclass
class Config:

```

```

# models can optionally be passed in directly
caption_model = None
caption_processor = None
clip_model = None
clip_preprocess = None

# blip settings
caption_max_length: int = 32
caption_model_name: Optional[str] = 'blip-large' # use a key from CAPTION_MODELS or None
caption_offload: bool = False

# clip settings
clip_model_name: str = 'ViT-L-14/openai'
clip_model_path: Optional[str] = None
clip_offload: bool = False

# interrogator settings
cache_path: str = 'cache' # path to store cached text embeddings
download_cache: bool = True # when true, cached embeds are downloaded from huggingface
chunk_size: int = 2048 # batch size for CLIP, use smaller for lower VRAM
data_path: str = os.path.join(os.path.dirname(__file__), 'data')
device: str = "cuda"
flavor_intermediate_count: int = 2048
quiet: bool = False # when quiet progress bars are not shown

def apply_low_vram_defaults(self):
    self.caption_model_name = 'blip-base'
    self.caption_offload = True
    self.clip_offload = True
    self.chunk_size = 1024

```

```
self.flavor_intermediate_count = 1024
```

A.11. docs.html

```
<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

  <meta name="viewport" content="width=device-width, initial-scale=1.0">

  <title>ImgProPlus Application Documentation</title>

  <style>

    body {

      font-family: Arial, sans-serif;

      margin: 0;

      padding: 20px;

      line-height: 1.6;

    }

    header {

      background: #333;

      color: #fff;

      padding: 10px 0;

      text-align: center;

    }

    nav {

      margin: 20px 0;

    }

    nav a {

      margin: 0 15px;

      text-decoration: none;

      color: #333;

    }

    h1, h2, h3 {
```

```

        color: #333;
    }
    code {
        background: #f4f4f4;
        padding: 2px 5px;
        border-radius: 3px;
    }
    pre {
        background: #f4f4f4;
        padding: 10px;
        border-radius: 5px;
        overflow-x: auto;
    }
    section {
        margin-bottom: 20px;
    }
</style>
</head>
<body>
    <header>
        <h1>ImgProPlus Application Documentation</h1>
    </header>

    <nav>
        <a href="#introduction">Introduction</a>
        <a href="#processimage">Processing Image</a>
        <a href="#install_new_models">Install New Models</a>
        <a href="#buy_premium">Premium Purchase</a>
        <a href="#config-parameters">Config Parameters</a>
    </nav>

```


<section id="introduction">

<h2>Introduction</h2>

<p>

This documentation covers the usage of the ImgProPlus application, detailing how to configure and use the app for PROMPTs generation. The Application leverages CLIP models to generate prompts based on input images, offering different models for various use cases.

</p>

</section>

<section id="processimage">

<h2>Processing image</h2>

<p>

To process an image in the ImgProPlus application, follow these steps:

Click on the "Upload Image" button.

Select the image file you want to process from your computer.

Once the image is uploaded, it will be displayed in the canvas area.

If needed, adjust the image size by stretching or filling it using the provided controls.

In the "Settings" section, choose the desired options for device, model, mode, and compute.

Optionally, click on the "Additional Settings" button to fine-tune advanced settings.

Click on the "Process Image" button to start the processing.

Wait for the processing to complete. A loading animation will indicate that the processing is in progress.

Once the processing is done, the results will be displayed in a new window.

Review the results and analyze the prompts generated based on your selected settings.

You can repeat the process with different images or settings as needed.

</p>

</section>

```
<section id="install_new_models">
```

```
<h2>Installing New Models</h2>
```

```
<p>
```

```
    To install a new model, follow these steps:
```

```
</p>
```

```
<ol>
```

```
<li>Open the "Models" section.</li>
```

```
<li>Select the model you need or prefer.</li>
```

```
<li>Click on "Install". The application will download the specified model for you.</li>
```

```
<li>In case you don't have premium, consider purchasing it to download all the best models.</li>
```

```
</ol>
```

```
</section>
```

```
<section id="buy_premium">
```

```
<h2>Premium Purchase</h2>
```

```
<p>
```

```
    To buy premium for your account, follow these steps:
```

```
</p>
```

```
<ol>
```

```
<li>Open the "Account" section.</li>
```

```
<li>Click on the "Purchase Premium" button.</li>
```

```
<li>Follow the instructions in a new window to complete the purchase.</li>
```

```
<li>After a successful purchase, you'll have information about premium in your account.</li>
```

```
<li>If your models aren't available to download, try restarting the app.</li>
```

```
</ol>
```

```
</section>
```

```
<section id="config-parameters">
```

```
<h2>Config Parameters</h2>
```

```
<p>
```

The Clip Interrogator can be customized using various configuration parameters. Below is a detailed explanation of each parameter:

Parameter	Description
<code>device</code>	Specifies the device to use for computation. Options include <code>cpu</code> and <code>gpu</code> (for CUDA).
<code>clip_model</code>	The CLIP model to use. Examples include <code>ViT-B/32</code> , <code>ViT-L/14</code> , etc.
<code>caption_max_length</code>	Sets the maximum length for generated captions. A higher value produces longer, more detailed captions.
<code>chunk_size</code>	Determines the chunk size for processing. Adjusting this value can affect performance and memory usage.

<code>flavor_intermediate_count</code>	Specifies the number of intermediate flavors to consider. Useful for generating more nuanced prompts.
<code>blip_image_eval_size</code>	Sets the size of the image evaluation in the Blip settings. This parameter determines the dimensions of the image used for evaluation during processing.
<code>blip_max_length</code>	Sets the maximum length for prompts in the Blip settings. A higher value produces longer, more detailed prompts.
<code>blip_offload</code>	Determines whether to enable offloading in the Blip settings. If set to <code>True</code> , certain tasks may be offloaded for improved performance.
<code>caption_model_name</code>	Specifies the name of the caption model to use. This parameter determines the specific model architecture used for generating captions.
<code>mode</code>	Determines the processing mode to use. Available options include <code>fast</code> , <code>negative</code> , <code>classic</code> , and <code>best</code> . Each mode offers different trade-offs between speed and accuracy.
<code>compute</code>	

<td>Specifies the compute platform to use for processing. Options include
 <code>local</code> for local processing and <code>cloud</code> for cloud-based processing.</td>

</tr>

</tbody>

</table>

</section>

</body>

</html>

A.12. slide_widget.py

```
import ttkbootstrap as ttk
```

```
class SlideWidget(ttk.Frame):
```

```
    def __init__(self, parent):
```

```
        super().__init__(master = parent, borderwidth = 2, border = 5)
```

```
        self.start_pos_x = 0.7
```

```
        self.start_pos_y = 0.8
```

```
        self.width = 0.35
```

```
        self.message = None
```

```
    def show_message(self, message):
```

```
        self.message = ttk.Label(self, text = message, wraplength = 150)
```

```
        self.message.pack(expand = True, fill = 'both')
```

```
        self.place(relx = self.start_pos_x, rely = self.start_pos_y, relwidth = self.width, relheight = 0.2)
```

```
    def destroy_message(self):
```

```
        if self.message:
```

```
            self.message.pack_forget()
```

A.13. login_page.py

```
from settings import ApplicationConfiguration, STYLE
```

```

from localization import LOGIN_PAGE, MESSAGES
import ttkbootstrap as ttk
from slide_widget import SlideWidget
from client import Client
import re
from ttkbootstrap.dialogs import Messagebox
from my_logger import setup_logger
import os, sys
from pathlib import Path

SECONDS_TO_WAIT = 1

app_configuration = ApplicationConfiguration()
logger = setup_logger(logger_name="User Application", logger_file="logs.log")

class LoginPage(ttk.Frame):
    def __init__(self, master):
        super().__init__(master = master)

        self.change_theme_button = ttk.Button(self,
                                                text = LOGIN_PAGE["theme_button"][app_configuration.locale],
                                                command = self.change_current_theme)

        self.current_theme = self.master.current_theme
        # self.style.theme_use(self.current_theme)

```

```
self.change_theme_button.place(relx = 0.9, rely = 0.1, anchor = 'center')
self.change_theme_button.lift()
```

```
self.grip = ttk.Sizegrip(self)
self.grip.place(relx=1.0, rely=1.0, anchor="se")
```

```
# self.animated_notification = SlideWidget(self, 1.0, 0.7)
```

```
#grid
self.rowconfigure(0,
                  weight = 2,
                  uniform = 'a')
self.rowconfigure((1,2,3,4,5),
                  weight = 1,
                  uniform = 'a')
self.rowconfigure(6,
                  weight = 1,
                  uniform = 'a')
self.columnconfigure(0,
                     weight = 2,
                     uniform = 'a')
self.columnconfigure(1,
                     weight = 3,
                     uniform = 'a')
self.columnconfigure(2,
                     weight = 2,
                     uniform = 'a')
```



```

self.login_email_entry.grid(column = 1,
                             row = 1,
                             sticky = 'nsew',
                             padx = 10,
                             pady = 10)
self.login_email_entry.focus_force()

self.password_label = ttk.Label(self,
                                 text = LOGIN_PAGE["password_label"][app_configuration.locale],
                                 anchor = 'center')
self.password_label.grid(column = 0,
                          row = 2,
                          sticky = 'nsew',
                          padx = 10,
                          pady = 10)

self.login_password_entry = ttk.Entry(self,
                                       textvariable=self.password_variable,
                                       show="*")
self.login_password_entry.grid(column = 1,
                                row = 2,
                                sticky = 'nsew',
                                padx = 10,
                                pady = 10)

self.login_button = ttk.Button(self,
                                text = LOGIN_PAGE["login_button"][app_configuration.locale],
                                command = self.login_to_app)
self.login_button.grid(column = 1,

```

```

        row = 3,
        sticky = 'nsew',
        padx = 10,
        pady = 10)

self.forgot_password_button = ttk.Button(self,
                                          text = LOGIN_PAGE['forgot_password_button'][app_configuration.locale],
                                          command = lambda: ForgotPasswordWindow(self))

self.forgot_password_button.grid(column = 1,
                                  row = 4,
                                  #sticky = 'nsew',
                                  padx = 10,
                                  pady = 10,)

self.new_user_label = ttk.Label(self,
                                 text = LOGIN_PAGE['new_user_label'][app_configuration.locale],
                                 anchor = 'center')

self.new_user_label.grid(column = 1,
                          row = 5,
                          sticky = 'nsew',
                          padx = 10,
                          pady = 10,
                          )

self.registration_button = ttk.Button(self,
                                       text = LOGIN_PAGE["registration_button"][app_configuration.locale],
                                       command = lambda: RegistrationWindow(self),
                                       )

self.registration_button.place(relx = 0.5,
                               rely = 0.87,

```

```

        anchor = 'center',
        relheight = 0.06,
        relwidth = 0.2)

def change_current_theme(self):
    if self.current_theme == 'darkly':
        self.current_theme = 'journal'
        self.master.style.theme_use(self.current_theme)
    else:
        self.current_theme = 'darkly'
        self.master.style.theme_use(self.current_theme)

def login_to_app(self):
    client = Client()
    verification =
client.process_request(f"UCV|{self.email_variable.get()}|{self.password_variable.get()}")
    if verification == "False":

    messagebox.show_error(LOGIN_PAGE["message_wrong_email_or_pass"][app_configuration.locale], "Error",
parent=self)
    else:
        with open("token", 'w') as f:
            f.write(f"{self.email_variable.get()}|{verification}")
        self.master.login_page.pack_forget()
        self.master.form_main_page()

class ValidationEntry(ttk.Entry):

```

```

def __init__(self, master, entry_type, notification_handler, text_variable, show=None,
second_password_variable=None):
    super().__init__(master=master, textvariable=text_variable, show=show)
    self.entry_type = entry_type
    self.notification = notification_handler

    self.rep = [None, None]

    self.textvariable = text_variable

    self.second_password_variable = second_password_variable if second_password_variable else None

    self.entry_config()

def entry_config(self) -> None:
    if self.entry_type == 'login':
        self.textvariable.trace_add('write', self.typing_email)
    elif self.entry_type == 'uno_password':
        self.textvariable.trace_add('write', self.typing_password)
    elif self.entry_type == 'duo_password':
        self.textvariable.trace_add('write', self.typing_second_password)

def check_password(self, password) -> bool:
    # Check if password length is at least 8 characters
    if len(password) < 8:
        return False

    # Define regular expressions for each character type
    uppercase_regex = r'[A-Z]'

```

```

lowercase_regex = r'[a-z]'
digit_regex = r'\d'
symbol_regex = r'[!@#$%^&*()-_+=~`[\]{}|;:.,<>?]'

# Count the number of character types present in the password
types_count = 0
if re.search(uppercase_regex, password):
    types_count += 1
if re.search(lowercase_regex, password):
    types_count += 1
if re.search(digit_regex, password):
    types_count += 1
if re.search(symbol_regex, password):
    types_count += 1

# Check if at least two character types are present
if types_count >= 2:
    return True
else:
    return False

def validate_password(self, *args) -> None:
    if self.check_password(self.textvariable.get()):
        self.configure(bootstyle = 'success')
        self.rep = [None, True]
    else:
        self.notification.show_message(LOGIN_PAGE['password_validation_error'][app_configuration.locale])
        self.configure(bootstyle = 'danger')
        self.rep = [None, None]

```

```

def typing_password(self, *args) -> None:
    if self.rep[0] is None:
        self.notification.destroy_message()
        self.configure(bootstyle = 'primary')
    else:
        self.after_cancel(self.rep)

    self.rep = self.after(SECONDS_TO_WAIT * 1000, self.validate_password)

# functions for email verification
def validate_email(self) -> None:
    email_regex = r'\b[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Z|a-z]{2,}\b' # regex for email

    if re.fullmatch(email_regex, self.textvariable.get()):
        self.configure(bootstyle = 'success')
        self.rep = [None, True]
    else:

self.notification.show_message(LOGIN_PAGE['email_validation_error'][app_configuration.locale])
        self.configure(bootstyle = 'danger')
        self.rep = [None, None]

def typing_email(self, *args) -> None:
    if self.rep[0] is None:

```

```

        self.notification.destroy_message()

        self.configure(bootstyle = 'primary')
    else:
        self.after_cancel(self.rep)

self.rep = self.after(SECONDS_TO_WAIT * 1000, self.validate_email)

def typing_second_password(self, *args) -> None:
    if self.rep[0] is None:
        self.notification.destroy_message()

        self.configure(bootstyle = 'primary')
    else:
        self.after_cancel(self.rep)

self.rep = self.after(SECONDS_TO_WAIT * 1000, self.validate_second_password)

def validate_second_password(self, *args) -> None:
    if self.textvariable.get() != self.second_password_variable.get():
        self.configure(bootstyle = 'danger')

        self.notification.show_message("Passwords must match!")

        self.rep = [None, None]

    else:
        self.configure(bootstyle = 'success')

        self.rep = [None, True]

```

```

class RegistrationWindow(ttk.Toplevel):
    def __init__(self, master):
        super().__init__(master)

        self.title(LOGIN_PAGE["registration_title"][app_configuration.locale])
        self.geometry = ('900x700')
        self.minsize(600,600)
        self.maxsize(1000,600)
        self.resizable(True, True)

        self.email_variable = ttk.StringVar()
        self.password_variable = ttk.StringVar()
        self.password_repeat_variable = ttk.StringVar()
        self.confirmation_code_variable = ttk.StringVar()

        self.writing_listener = None # var for typing check

        self.rowconfigure(0,
                           weight = 2,
                           uniform = 'a')
        self.rowconfigure((1,2,3,4),
                           weight = 1,
                           uniform = 'a')
        self.rowconfigure(5,
                           weight = 2,
                           uniform = 'a')
        self.rowconfigure(6,
                           weight = 2,
                           uniform = 'a')

```



```

self.password_repeat_label.grid(column = 0,
                                row = 3,
                                sticky = 'nsew',
                                padx = 10,
                                pady = 10)

```

```

self.password_repeat_entry = ValidationEntry(master = self,
                                              entry_type = 'duo_password',
                                              notification_handler = self.notification,
                                              text_variable = self.password_repeat_variable,
                                              second_password_variable = self.password_variable,
                                              show="*")

```

```

self.password_repeat_entry.grid(column = 1,
                                row = 3,
                                sticky = 'nsew',
                                padx = 10,
                                pady = 10)

```

```

self.confirmation_code_label = ttk.Label(self,
                                          text = LOGIN_PAGE["confirmation_code_label"][app_configuration.locale],
                                          anchor = 'center')

```

```

self.confirmation_code_label.grid(column = 0,
                                   row = 4,
                                   sticky = 'nsew',
                                   padx = 10,
                                   pady = 10)

```

```

self.confirmation_code_entry = ttk.Entry(self,

```

```

        textvariable = self.confirmation_code_variable,
        state = 'disabled',
        bootstyle = 'dark'
    )

self.confirmation_code_entry.grid(column = 1,
    row = 4,
    sticky = 'nsew',
    padx = 10,
    pady = 10)

self.registration_button = ttk.Button(self,
    text = LOGIN_PAGE["send_confirm_code"][app_configuration.locale],
    command = lambda: self.send_confirm_code(email = self.email_variable.get()),
    state='disabled')

self.registration_button.grid(column = 1,
    row = 5,
    padx = 10,
    pady = 10)

self.return_button = ttk.Button(self,
    text = LOGIN_PAGE["return_button"][app_configuration.locale],
    command = lambda: self.destroy()
)

self.return_button.place(relx = 0.1,
    rely = 0.9,
    anchor = 'center')

self.transient(master) # set to be on top of the main window
self.grab_set()

```

```

self.after(0, self.validate_entries)

master.wait_window(self)

def validate_entries(self):
    if all([
        self.mail_entry.rep[1] == True,
        self.password_entry.rep[1] == True,
        self.password_repeat_entry.rep[1] == True
    ]):
        self.registration_button.configure(state='enabled')
    else:
        self.registration_button.configure(state='disabled')

self.after(SECONDS_TO_WAIT * 1200, self.validate_entries)

# After a short delay, re-validate the entries

def send_confirm_code(self, email) -> None:
    try:
        client = Client()
        result = client.process_request(f"CLS|{email}")
        if result != "True":

            messagebox.show_error(LOGIN_PAGE["email_already_taken_message"][app_configuration.locale],
            LOGIN_PAGE["email_already_taken_message_title"][app_configuration.locale])

        return

        conf_code = client.process_request(f"SCC|{email}")
        if conf_code != "send":
            messagebox.show_error(conf_code, "Error during registration.")

```

```

        return

    else:

        MessageBox.ok(LOGIN_PAGE["code_sent_message"][app_configuration.locale],
LOGIN_PAGE["code_sent_message_title"][app_configuration.locale])

        self.registration_button.configure(text =
LOGIN_PAGE["button_confirm_code"][app_configuration.locale])

        self.confirmation_code_entry.configure(state = 'enabled')

        self.confirmation_code_entry.configure(bootstyle = 'default')

        self.registration_button.configure(command = self.verify_confirm_code)

    except Exception as e:

        logger.error("Error happened - {e}")

        MessageBox.okcancel(MESSAGES["unhandled_exception"][app_configuration.locale], "Error
during registration")


def verify_confirm_code(self) -> None:

    try:

        client = Client()

        result =
client.process_request(f"VCC|{self.email_variable.get()}|{self.confirmation_code_variable.get()}")

        if result == "True":

            MessageBox.ok(LOGIN_PAGE["code_approved_msg"][app_configuration.locale],
LOGIN_PAGE["code_approved_msg_title"][app_configuration.locale])

        else:

            MessageBox.show_error(LOGIN_PAGE["code_not_approved_msg"][app_configuration.locale],
LOGIN_PAGE["code_not_approved_msg_title"][app_configuration.locale])

        return


    token = client.process_request(f"REG|{self.email_variable.get()}|{self.password_variable.get()}")

    if token == "done":

```

```

        answer =
Messagebox.okcancel(LOGIN_PAGE["user_registered_msg"][app_configuration.locale],
LOGIN_PAGE["user_registered_msg_title"][app_configuration.locale])

        if answer:

            token =
client.process_request(f"UCV|{self.email_variable.get()}|{self.password_variable.get()}")

            with open("token", 'w') as f:

                f.write(f"{self.email_variable.get()}|{token}")

            self.destroy()

            self.master.pack_forget()

            self.master.master.form_main_page()

        else:

            self.destroy()

elif result == 'confirm_wrong':

    Messagebox.show_error("Wrong confirmation password. Please check it again, or contact QA
using address imgproplus@gmail.com.", "Error during registration.")

    else:

        Messagebox.show_error(f"Error during confirmation code. {e}", "Error during registration.")

except Exception as e:

    logger.error(f"Something wrong during confirmation code. Error - {e}")

    Messagebox.show_error(f"Error during confirmation code. {e}", "Error during registration.")

```

```

class ForgotPasswordWindow(ttk.Toplevel):

```

```

    def __init__(self, master):

```

```

super().__init__(master)

self.title(LOGIN_PAGE["recover_password_title"][app_configuration.locale])

self.geometry('500x300')

self.minsize(500, 300)

self.maxsize(1000, 600)


self.notification = SlideWidget(self)


# variable declaration
self.login_variable = ttk.StringVar()
self.new_password_variable = ttk.StringVar()
self.new_password_repeat_variable = ttk.StringVar()
self.confirmation_code_variable = ttk.StringVar()


# callbacks for hints


# email_function = self.register(validate_email)


# confirmation code frame
self.confirmation_code_frame = ttk.Frame(self)

self.confirmation_code_frame.columnconfigure((0,1,2), weight=1)
self.confirmation_code_frame.rowconfigure((0,1,2,3), weight=1)

confirmation_sent_label = ttk.Label(self.confirmation_code_frame,
                                   text =
LOGIN_PAGE["forgot_window_confirmation_code_label"][app_configuration.locale])
confirmation_sent_label.grid(row = 0, column = 1)

```



```

        confirmation_code_entry = ttk.Entry(self.confirmation_code_frame,
textvariable=self.confirmation_code_variable)

        confirmation_code_entry.grid(row = 1, column = 1, sticky='ew')


        confirm_code_button = ttk.Button(self.confirmation_code_frame,
            text =
LOGIN_PAGE["forgot_window_confirmation_code_button"][app_configuration.locale],
            command = self.check_confirmation_code,
            default = 'active') # do it in client i guess


        confirm_code_button.grid(row = 2, column = 1)


        # input login frame


        self.input_login_frame = ttk.Frame(self)


        self.input_login_frame.columnconfigure((0,1,2), weight=1)
        self.input_login_frame.rowconfigure((0,1,2,3), weight=1)


        self.user_input_email_label = ttk.Label(self.input_login_frame,
            text =
LOGIN_PAGE["forgot_window_user_email_label"][app_configuration.locale])
        self.user_input_email_label.grid(row = 0, column = 1)


        self.user_input_email_entry = ValidationEntry(master=self.input_login_frame,
            entry_type = 'login',
            notification_handler = self.notification,
            text_variable = self.login_variable
        )

```

```

self.user_input_email_entry.grid(row = 1, column = 1, sticky='ew')

self.send_code_button = ttk.Button(self.input_login_frame,
                                   text =
LOGIN_PAGE["forgot_window_user_email_button"][app_configuration.locale],
                                   command = lambda : self.send_confirmation_code(self.login_variable.get()),
                                   state = 'disabled',
                                   ) # do it in client i guess

self.send_code_button.grid(row = 2, column = 1)

# create new password frame
self.new_password_frame = ttk.Frame(self)

self.new_password_frame.columnconfigure((0,1,2), weight=1)
self.new_password_frame.rowconfigure((0,1,2,3), weight=1)

new_password_label = ttk.Label(self.new_password_frame,
                               text = LOGIN_PAGE["password_label"][app_configuration.locale])
new_password_label.grid(row = 0, column = 1)

new_password_entry = ValidationEntry(master=self.new_password_frame,
                                     entry_type = 'uno_password',
                                     notification_handler = self.notification,
                                     text_variable = self.new_password_variable,
                                     show="*")

new_password_entry.grid(row = 1, column = 1, sticky='ew')

new_password_repeat_label = ttk.Label(self.new_password_frame,

```

```

        text = LOGIN_PAGE["password_repeat_label"][app_configuration.locale])
new_password_repeat_label.grid(row = 2, column = 1)

new_password_repeat_entry = ValidationEntry(master=self.new_password_frame,
        entry_type = 'duo_password',
        notification_handler = self.notification,
        text_variable = self.new_password_repeat_variable,
        second_password_variable = self.new_password_variable,
        show="*")
)
new_password_repeat_entry.grid(row = 3, column = 1, sticky='ew')

# add here button

self.send_password_button = ttk.Button(self.new_password_frame,
        text = LOGIN_PAGE["send_password_button"][app_configuration.locale],
        command = self.create_new_password)
self.send_password_button.grid(row = 4,
        column = 1,
        sticky='ew',
        pady = 5
)

# binds
self.user_input_email_entry.bind('<Configure>', self.activate_send_email_button)

# return button

```

```

self.return_button = ttk.Button(self,
                                text = LOGIN_PAGE["return_button"][app_configuration.locale],
                                command = lambda: self.destroy()
                                )

self.return_button.place(relx = 0.1,
                        rely = 0.9,
                        anchor = 'center')

self.input_login_frame.pack(expand=True, fill='both')


self.transient(master) # set to be on top of the main window
self.grab_set()

master.wait_window(self)

def send_confirmation_code(self, username: str):
    client = Client()
    result = client.process_request(f"CLS|{username}")
    if result == "True":
        messagebox.show_error(LOGIN_PAGE["email_doesnt_exist_msg"][app_configuration.locale],
                              LOGIN_PAGE["email_doesnt_exist_msg_title"][app_configuration.locale])
        return
    answer = client.process_request(f"SCC|{username}")
    if answer == "send":
        self.input_login_frame.pack_forget()
        self.confirmation_code_frame.pack(expand=True, fill='both')
        self.username = username

```

```
else:
```

```
Messagebox.show_error(LOGIN_PAGE["wrong_confirmation_code_msg"][app_configuration.locale],
LOGIN_PAGE["wrong_confirmation_code_msg_title"][app_configuration.locale], parent=self)
```

```
def activate_send_email_button(self, event: object):
```

```
    self.send_code_button['state'] = 'active'
```

```
def create_new_password(self) -> None:
```

```
    if self.new_password_variable.get() == self.new_password_repeat_variable.get():
```

```
        client = Client()
```

```
        answer = client.process_request(f"RES|{self.username}|{self.new_password_variable.get()}")
```

```
        if answer == "True":
```

```
Messagebox.show_info(LOGIN_PAGE["password_changed_successfully"][app_configuration.locale],
LOGIN_PAGE["password_changed_successfully_title"][app_configuration.locale], parent=self)
```

```
    self.destroy()
```

```
    else:
```

```
Messagebox.show_error(LOGIN_PAGE["error_changing_password"][app_configuration.locale],
LOGIN_PAGE["error_changing_password_title"][app_configuration.locale], parent=self)
```

```
    else:
```

```
        Messagebox.show_error(LOGIN_PAGE["passwords_dont_match"][app_configuration.locale],
LOGIN_PAGE["passwords_dont_match_title"][app_configuration.locale], parent=self)
```

```
def check_confirmation_code(self):
```

```
    client = Client()
```

```
    answer =
```

```
    client.process_request(f"VCC|{self.login_variable.get()}|{self.confirmation_code_variable.get()}")
```

```
    if answer == "True":
```

```

        self.confirmation_code_frame.pack_forget()

        self.new_password_frame.pack(expand=True, fill='both')
    else:

```

```

    messagebox.show_error(LOGIN_PAGE["wrong_confirmation_code_msg"][app_configuration.locale],
    LOGIN_PAGE["wrong_confirmation_code_msg_title"][app_configuration.locale], parent=self)

```

```

def send_logs(error) -> None:

    answer = messagebox.okcancel("You encountered ")

```

```

if __name__ == '__main__':

    master = tkk.Window()

    master.current_theme = 'darkly'

    LoginPage(master).pack(expand=True, fill='both')

    master.mainloop()

```

ДОДАТОК Б – ЗАПИТИ ДО БД

Б.1 Код створення бази даних MySQL

-- RESULTING DOC

-- creating tables

-- Create Users table

```
CREATE TABLE Users (  
    username VARCHAR(50) PRIMARY KEY,  
    password VARCHAR(256)  
);
```

-- Create Payment table

```
CREATE TABLE Payments (  
    paymentID INT PRIMARY KEY,  
    username VARCHAR(50),  
    paymentDate DATE,  
    promotion_id INT,  
    FOREIGN KEY (username) REFERENCES Users(username),  
    FOREIGN KEY (promotion_id) REFERENCES Promotions(promotion_id)  
);
```

-- Create Subscriptions table

```
CREATE TABLE Subscribers(  
    username VARCHAR(50) PRIMARY KEY,  
    end_date DATE,  
    FOREIGN KEY (username) REFERENCES Users(username)  
);
```

-- Create Credits table

```
CREATE TABLE Credits(  
    username VARCHAR(50) PRIMARY KEY,  
    credits INT DEFAULT 5,  
    FOREIGN KEY (username) REFERENCES Users(username)  
);
```

-- Create Requests table

```
CREATE TABLE Requests (  
    requestID INT AUTO_INCREMENT PRIMARY KEY,  
    username VARCHAR(50),  
    requestText TEXT,  
    FOREIGN KEY (username) REFERENCES Users(username)  
);
```

-- Create auth_tokens table

```
CREATE TABLE auth_tokens (  
    id INT AUTO_INCREMENT PRIMARY KEY,  
    token VARCHAR(255) NOT NULL,  
    start_date DATE NOT NULL,  
    end_date DATE NOT NULL,  
    username VARCHAR(50),  
    FOREIGN KEY (username) REFERENCES Users(username)  
);
```

-- Create confirmation_codes table

```
CREATE TABLE confirmation_codes (  
    id INT AUTO_INCREMENT PRIMARY KEY,  
    username VARCHAR(50),
```



```

code varchar(6) NOT NULL,
expiration_date DATETIME NOT NULL,
UNIQUE KEY unique_code (code)
);

```

```
-- Create promotions table
```

```

CREATE TABLE promotions (
    promotion_id INT AUTO_INCREMENT PRIMARY KEY,
    subscription_length ENUM('1 month', '3 months', '6 months', '12 months') NOT NULL,
    cost DECIMAL(10, 2) NOT NULL,
    description TEXT,
    subscription_length_ua ENUM('1 місяць', '3 місяці', '6 місяців', '12 місяців') NOT NULL,
    description_ua TEXT
);

```

```
-- Create models table
```

```

CREATE TABLE models (
    model_name VARCHAR(50),
    description TEXT,
    premium Bool
);

```

```
-- EVENTS
```

```
-- Delete expired hashes event
```

```
DELIMITER $$
```

```
CREATE EVENT delete_expired_hashes
```

```
ON SCHEDULE
```

```
EVERY 1 DAY
```

```
COMMENT 'Delete expired hash records from the auth_tokens table'
```

```
DO
```

```
BEGIN
```

```
    DELETE FROM auth_tokens
```

```
    WHERE CURRENT_DATE() NOT BETWEEN start_date AND end_date;
```

```
END$$
```

```
DELIMITER ;
```

```
-- Delete expired codes event
```

```
DELIMITER $$
```

```
CREATE EVENT delete_expired_codes
```

```
ON SCHEDULE EVERY 1 MINUTE
```

```
DO
```

```
    DELETE FROM confirmation_codes WHERE expiration_date < NOW();
```

```
DELIMITER ;
```

```
INSERT INTO Payments (paymentID, username, paymentDate, promotion_id)
```

```
VALUES
```

```
    (1, 'roflif@gmail.com', '2024-05-01', 5);
```

```
INSERT INTO promotions (subscription_length, cost, description)
```

```
VALUES
```

```
    ('1 month', 10.00, 'Basic subscription for 1 month'),
```

```
    ('3 months', 25.00, 'Standard subscription for 3 months'),
```

```
    ('6 months', 45.00, 'Premium subscription for 6 months with discount'),
```

```
    ('12 months', 80.00, 'Ultimate subscription for 12 months with special offer');
```

```
INSERT INTO promotions (subscription_length, cost, description, subscription_length_ua, description_ua)
```

```
VALUES
```

```
( '1 month', 10.00, 'Basic subscription for 1 month', '1 місяць', 'Базова підписка на 1 місяць'),
( '3 months', 25.00, 'Standard subscription for 3 months', '3 місяці', 'Стандартна підписка на 3 місяці'),
( '6 months', 45.00, 'Premium subscription for 6 months with discount', '6 місяців', 'Преміум підписка на 6
місяців зі знижкою');
```

```
INSERT INTO promotions (subscription_length, cost, description, subscription_length_ua, description_ua)
VALUES ( '12 months', 85.00, 'Premium subscription for 12 months with discount', '12 місяців', 'Преміум
підписка на 12 місяців зі знижкою');
```

```
INSERT INTO models (model_name, description, premium)
```

```
VALUES
```

```
( 'ViT-L-14/openai', 'Vision Transformer (ViT) model trained on the openAI dataset, optimized for various
vision tasks including image classification, object detection, and segmentation.', FALSE),
```

```
( 'ViT-H-14/laion2b_s32b_b79k', 'High-capacity Vision Transformer (ViT) model trained on the laion2b
dataset, designed for complex visual tasks requiring fine-grained feature extraction and understanding.',
FALSE),
```

```
( 'ViT-g-14/laion2b_s34b_b88k', 'General-purpose Vision Transformer (ViT) model trained on the laion2b
dataset, suitable for a wide range of computer vision tasks with moderate computational requirements.', FALSE),
```

```
( 'EVA02-L-14/merged2b_s4b_b131k', 'Large-scale Vision Transformer (ViT) model with merged data from
various sources and high batch size, ideal for handling extensive datasets and achieving state-of-the-art
performance in vision tasks.', TRUE),
```

```
( 'ViT-L-14/datacomp_xl_s13b_b90k', 'Vision Transformer (ViT) model trained on the datacomp_xl dataset
with a large batch size, tailored for handling extremely large-scale datasets and achieving superior performance
in vision applications.', TRUE);
```

ДОДАТОК В – КОД ТЕСТІВ

В.1. Модульні тести

В.1.1. addons_test.py

```
import sys

sys.path.insert(0, "d:\\diploma\\app")

print(sys.path)


import pytest
import bcrypt

from unittest.mock import patch

from addons import Config, get_processor_name, get_gpus, hash_password


def test_config_defaults():

    config = Config()

    assert config.caption_max_length == 32
    assert config.caption_model_name == 'blip-large'
    assert config.caption_offload is False
    assert config.clip_model_name == 'ViT-L-14/openai'
    assert config.clip_model_path is None
    assert config.clip_offload is False
    assert config.cache_path == 'cache'
    assert config.download_cache is True
    assert config.chunk_size == 2048
    assert config.device == 'cuda'
    assert config.flavor_intermediate_count == 2048
```

```

assert config.quiet is False

def test_apply_low_vram_defaults():
    config = Config()
    config.apply_low_vram_defaults()

    assert config.caption_model_name == 'blip-base'
    assert config.caption_offload is True
    assert config.clip_offload is True
    assert config.chunk_size == 1024
    assert config.flavor_intermediate_count == 1024

@patch('platform.system')
@patch('subprocess.check_output')
def test_get_processor_name_windows(mock_subprocess, mock_platform):
    mock_platform.return_value = 'Windows'
    mock_subprocess.return_value = b'Name Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz \r\n\r\n'

    processor_name = get_processor_name()
    assert processor_name == 'Intel(R)Core(TM)i7-9750HCPU@2.60GHz'

@patch('platform.system')
@patch('subprocess.check_output')
def test_get_processor_name_darwin(mock_subprocess, mock_platform):
    mock_platform.return_value = 'Darwin'
    mock_subprocess.return_value = b'Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz\n'

    processor_name = get_processor_name()
    assert processor_name == 'Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz'

```

```

@patch('platform.system')
@patch('subprocess.check_output')
def test_get_processor_name_linux(mock_subprocess, mock_platform):
    mock_platform.return_value = 'Linux'
    mock_subprocess.return_value = b'processor\t: 0\nmodel name\t: Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz\n'

    processor_name = get_processor_name()
    assert processor_name == 'Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz'

@patch('subprocess.check_output')
def test_get_gpus(mock_subprocess):
    mock_subprocess.return_value = b'GPU 0: Tesla K80 (UUID: GPU-12345678-1234-5678-1234-567812345678)\nGPU 1: Tesla T4 (UUID: GPU-87654321-4321-8765-4321-876543218765)'

    gpus = get_gpus()
    assert gpus == [('GPU 0', 'Tesla K80'), ('GPU 1', 'Tesla T4')]

def test_hash_password():
    password = 'securepassword123'
    hashed_password = hash_password(password)

    assert bcrypt.checkpw(password.encode('utf-8'), hashed_password.encode('utf-8'))

# Only include this if you want to run the file directly
if __name__ == '__main__':
    pytest.main()

```

B.1.2. interrogator_tests.py

```

import os

```

```

import tempfile
import pytest
import torch
from PIL import Image
from transformers import BlipForConditionalGeneration
from safetensors.numpy import save_file
import numpy as np
import hashlib

import sys
sys.path.insert(0, "d:\\diploma\\gui")

from clip_interrogator import Config, Interrogator, LabelTable, _truncate_to_fit

# Sample image for testing
def create_sample_image():
    image = Image.new('RGB', (128, 128), color = 'red')
    return image

# Sample config for testing
def create_sample_config():
    return Config(
        caption_model_name='blip-base',
        clip_model_name='ViT-L-14/openai',
        caption_offload=False,
        clip_offload=False,
        cache_path='cache',
        data_path='fakepath',
        quiet=True
    )

```

```
def test_load_caption_model():  
    config = create_sample_config()  
    interrogator = Interrogator(config)  
    interrogator.load_caption_model()  
    assert isinstance(interrogator.caption_model, BlipForConditionalGeneration)
```

```
def test_load_clip_model():  
    config = create_sample_config()  
    interrogator = Interrogator(config)  
    interrogator.load_clip_model()  
    assert interrogator.clip_model is not None
```

```
def test_generate_caption():  
    config = create_sample_config()  
    interrogator = Interrogator(config)  
    interrogator.load_caption_model()  
    sample_image = create_sample_image()  
    caption = interrogator.generate_caption(sample_image)  
    assert isinstance(caption, str)  
    assert len(caption) > 0
```

```
def test_image_to_features():  
    config = create_sample_config()  
    interrogator = Interrogator(config)  
    interrogator.load_clip_model()  
    sample_image = create_sample_image()  
    features = interrogator.image_to_features(sample_image)  
    assert isinstance(features, torch.Tensor)
```



```

def test_label_table():
    config = create_sample_config()
    config.chunk_size = 1 # Force small chunks to test chunking
    interrogator = Interrogator(config)
    labels = ["label1", "label2", "label3"]
    table = LabelTable(labels, "test", interrogator)
    assert len(table.labels) == 3
    assert len(table.embeds) == 3

def test_rank_top():
    config = create_sample_config()
    interrogator = Interrogator(config)
    interrogator.load_clip_model()
    sample_image = create_sample_image()
    image_features = interrogator.image_to_features(sample_image)
    labels = ["hot", "cold", "warm"]
    table = LabelTable(labels, "test", interrogator)
    top_label = table.rank(image_features, top_count=1)[0]
    assert top_label in labels

def test_truncate_to_fit():
    text = "part1, part2, part3, part4"
    tokenize = lambda x: [[1, 2, 3, 4, 5, 0]] # Mock tokenizer
    truncated = _truncate_to_fit(text, tokenize)
    assert truncated == "part1, part2, part3, part4"

# Mock data for caching
def test_label_table_cache():
    config = create_sample_config()
    labels = ["label1", "label2", "label3"]

```

```

sanitized_name = config.clip_model_name.replace('/', '_').replace('@', '_')
desc = "test"
hash = hashlib.sha256(",".join(labels).encode()).hexdigest()
cache_path = os.path.join(config.cache_path, f"{sanitized_name}_{desc}.safetensors")

```

```

embeds = [torch.randn(512).numpy() for _ in labels]
tensors = {
    "embeds": np.stack(embeds),
    "hash": np.array([ord(c) for c in hash], dtype=np.int8)
}
save_file(tensors, cache_path)

```

```

interrogator = Interrogator(config)
table = LabelTable(labels, desc, interrogator)
assert len(table.embeds) == 3

```

```

if __name__ == '__main__':
    pytest.main()

```

B.1.3. client_tests.py

```

import sys
sys.path.insert(0, "d:\\diploma\\gui")

```

```

import unittest
from unittest.mock import patch, MagicMock
import socket
from client import Client # Replace with the actual module name
from settings import SERV_IP

```

```

class TestClient(unittest.TestCase):

    def setUp(self):
        self.client = Client()

    @patch('client.socket.socket')
    def test_process_request(self, mock_socket):
        mock_socket_inst = MagicMock()
        mock_socket.return_value = mock_socket_inst

        # Mocking connect and recv
        mock_socket_inst.recv.return_value = b'success'

        response = self.client.process_request("UCV|username|password")

        mock_socket_inst.connect.assert_called_with((SERV_IP["IP"], SERV_IP['PORT']))
        mock_socket_inst.send.assert_called_with(b'UCV|username|password')
        self.assertEqual(response, "success")

    @patch('client.socket.socket')
    def test_send_image_no_credit(self, mock_socket):
        mock_socket_inst = MagicMock()
        mock_socket.return_value = mock_socket_inst

        # Mocking connect, recv, and sendall
        mock_socket_inst.recv.side_effect = [b'no credit']

        response = self.client.send_image("IMG|picta.jpg|caption_max_length=32;caption_model_name=blip-
large|best|username", "path/to/image.jpg")

```

```

mock_socket_inst.connect.assert_called_with((SERV_IP["IP"], SERV_IP["PORT"]))

mock_socket_inst.send.assert_any_call(b'IMG|picta.jpg|caption_max_length=32;caption_model_name=blip-
large|best|username')

self.assertEqual(response, "no credit")

@patch('client.socket.socket')
def test_send_image_success(self, mock_socket):
    mock_socket_inst = MagicMock()
    mock_socket.return_value = mock_socket_inst

    # Mocking connect, recv, and sendall
    mock_socket_inst.recv.side_effect = [b'ok', b'Image received and saved.', b'Generated prompts']

    with patch('builtins.open', unittest.mock.mock_open(read_data=b'image data')):
        response = self.client.send_image("IMG|picta.jpg|caption_max_length=32;caption_model_name=blip-
large|best|username", "path/to/image.jpg")

    mock_socket_inst.connect.assert_called_with((SERV_IP["IP"], SERV_IP["PORT"]))

    mock_socket_inst.send.assert_any_call(b'IMG|picta.jpg|caption_max_length=32;caption_model_name=blip-
large|best|username')

    mock_socket_inst.sendall.assert_any_call(b'image data')
    mock_socket_inst.sendall.assert_any_call(b'<END>')
    self.assertEqual(response, "Generated prompts")

@patch('client.socket.socket')
@patch('builtins.input', side_effect=["Hello", "closed"])
def test_run_client(self, mock_input, mock_socket):
    mock_socket_inst = MagicMock()
    mock_socket.return_value = mock_socket_inst

```

```
mock_socket_inst.recv.side_effect = [b'Hello, Client', b'closed']
```

```
self.client.run_client()
```

```
mock_socket_inst.connect.assert_called_with((self.client.addr))
```

```
self.assertTrue(mock_socket_inst.send.called)
```

```
self.assertTrue(mock_socket_inst.recv.called)
```

```
if __name__ == '__main__':
```

```
    unittest.main()
```

B.1.4. database_tests.py

```
import sys
```

```
sys.path.insert(0, "d:\\diploma\\server-side")
```

```
print(sys.path)
```

```
import pytest
```

```
from unittest.mock import patch, MagicMock
```

```
from database import Database
```

```
import json
```

```
# Mock config file content
```

```
mock_config = {
```

```
    "host": "localhost",
```

```
    "user": "testuser",
```

```
    "password": "testpassword",
```

```
    "database": "testdb"
```

```
}
```

```

@patch('builtins.open', new_callable=MagicMock)
@patch('json.load', return_value=mock_config)
def test_load_config(mock_json_load, mock_open):
    db = Database("config.json")
    db.load_config("config.json")
    assert db.host == "localhost"
    assert db.user == "testuser"
    assert db.password == "testpassword"
    assert db.database == "testdb"

@patch.object(Database, 'execute_query', return_value=True)
def test_add_new_user_success(mock_execute_query):
    db = Database("config.json")
    db.logger = MagicMock()
    result = db.add_new_user("testuser", "testhash")
    assert result is True

@patch.object(Database, 'execute_query', side_effect=Exception("DB error"))
def test_add_new_user_failure(mock_execute_query):
    db = Database("config.json")
    db.logger = MagicMock()
    with pytest.raises(Exception) as excinfo:
        result = db.add_new_user("testuser", "testhash")
    assert excinfo.value.args[0] == 'DB error'

@patch.object(Database, 'execute_query', return_value=[("hashed_password",)])
def test_check_login_credentials_success(mock_execute_query):
    db = Database("config.json")
    db.logger = MagicMock()

```

```

with patch('bcrypt.checkpw', return_value=True):
    result = db.check_login_credentials("testuser", "testpassword")
    assert result is True

```

```

@patch.object(Database, 'execute_query', return_value=[("hashed_password",)])
def test_check_login_credentials_failure(mock_execute_query):
    db = Database("config.json")
    db.logger = MagicMock()
    with patch('bcrypt.checkpw', return_value=False):
        result = db.check_login_credentials("testuser", "testpassword")
        assert result is False

```

```

@patch.object(Database, 'execute_query', return_value=False)
def test_check_login_credentials_no_user(mock_execute_query):
    db = Database("config.json")
    db.logger = MagicMock()
    result = db.check_login_credentials("testuser", "testpassword")
    assert result is False

```

```

@patch.object(Database, 'execute_query', return_value=True)
def test_remove_auth_token_success(mock_execute_query):
    db = Database("config.json")
    db.logger = MagicMock()
    result = db.remove_auth_token("testtoken")
    assert result is True

```

```

@patch.object(Database, 'execute_query', return_value=False)
def test_remove_auth_token_failure(mock_execute_query):
    db = Database("config.json")
    db.logger = MagicMock()

```

```
result = db.remove_auth_token("testtoken")
```

```
assert result is False
```

```
@patch.object(Database, 'execute_query', return_value=True)
```

```
def test_add_auth_token(mock_execute_query):
```

```
    db = Database("config.json")
```

```
    db.logger = MagicMock()
```

```
    result = db.add_auth_token("testuser", "testpassword")
```

```
    assert result is not None
```

```
@patch.object(Database, 'execute_query', return_value=[(1,)])
```

```
def test_verify_auth_token_success(mock_execute_query):
```

```
    db = Database("config.json")
```

```
    db.logger = MagicMock()
```

```
    result = db.verify_auth_token("testtoken")
```

```
    assert result is True
```

```
@patch.object(Database, 'execute_query', return_value=[(0,)])
```

```
def test_verify_auth_token_failure(mock_execute_query):
```

```
    db = Database("config.json")
```

```
    db.logger = MagicMock()
```

```
    result = db.verify_auth_token("testtoken")
```

```
    assert result is False
```

```
@patch.object(Database, 'execute_query', return_value=[])
```

```
def test_check_login_exists(mock_execute_query):
```

```
    db = Database("config.json")
```

```
    db.logger = MagicMock()
```

```
    result = db.check_login_exists("testuser")
```

```
    assert result is True
```



```

@patch.object(Database, 'execute_query', return_value=True)
def test_add_user_premium_status_success(mock_execute_query):
    db = Database("config.json")
    db.logger = MagicMock()
    result = db.add_user_premium_status("testuser", "2024-12-31")
    assert result is True

```

```

@patch.object(Database, 'execute_query', return_value=False)
def test_add_user_premium_status_failure(mock_execute_query):
    db = Database("config.json")
    db.logger = MagicMock()
    result = db.add_user_premium_status("testuser", "2024-12-31")
    assert result is False

```

```

@patch.object(Database, 'execute_query', return_value=True)
def test_add_payment_success(mock_execute_query):
    db = Database("config.json")
    db.logger = MagicMock()
    result = db.add_payment("testuser", "2024-12-31", "promo1")
    assert result is True

```

```

@patch.object(Database, 'execute_query', return_value=False)
def test_add_payment_failure(mock_execute_query):
    db = Database("config.json")
    db.logger = MagicMock()
    result = db.add_payment("testuser", "2024-12-31", "promo1")
    assert result is False

```

```

@patch.object(Database, 'execute_query', return_value=True)

```

```
def test_delete_promotion_success(mock_execute_query):
```

```
    db = Database("config.json")
```

```
    db.logger = MagicMock()
```

```
    result = db.delete_promotion("promo1")
```

```
    assert result is True
```

```
@patch.object(Database, 'execute_query', return_value=False)
```

```
def test_delete_promotion_failure(mock_execute_query):
```

```
    db = Database("config.json")
```

```
    db.logger = MagicMock()
```

```
    result = db.delete_promotion("promo1")
```

```
    assert result is False
```

```
@patch.object(Database, 'execute_query', return_value=True)
```

```
def test_insert_promotion_success(mock_execute_query):
```

```
    db = Database("config.json")
```

```
    db.logger = MagicMock()
```

```
    result = db.insert_promotion("6 months", "49.99", "Half year subscription")
```

```
    assert result is True
```

```
@patch.object(Database, 'execute_query', return_value=False)
```

```
def test_insert_promotion_failure(mock_execute_query):
```

```
    db = Database("config.json")
```

```
    db.logger = MagicMock()
```

```
    result = db.insert_promotion("6 months", "49.99", "Half year subscription")
```

```
    assert result is False
```

```
@patch.object(Database, 'execute_query', return_value=True)
```

```
def test_delete_user_success(mock_execute_query):
```

```
    db = Database("config.json")
```

```

db.logger = MagicMock()
result = db.delete_user("testuser")
assert result is True

```

```

@patch.object(Database, 'execute_query', return_value=False)
def test_delete_user_failure(mock_execute_query):
    db = Database("config.json")
    db.logger = MagicMock()
    result = db.delete_user("testuser")
    assert result is False

```

```

@patch.object(Database, 'execute_query', return_value=True)
def test_delete_payment_success(mock_execute_query):
    db = Database("config.json")
    db.logger = MagicMock()
    result = db.delete_payment("payment1")
    assert result is True

```

```

@patch.object(Database, 'execute_query', return_value=False)
def test_delete_payment_failure(mock_execute_query):
    db = Database("config.json")
    db.logger = MagicMock()
    result = db.delete_payment("payment1")
    assert result is False

```

```

@patch.object(Database, 'execute_query', return_value=True)
def test_add_request_success(mock_execute_query):
    db = Database("config.json")
    db.logger = MagicMock()
    result = db.add_request("testuser", "This is a test request")

```

```
assert result is True
```

```
@patch.object(Database, 'execute_query', return_value=False)
```

```
def test_add_request_failure(mock_execute_query):
```

```
    db = Database("config.json")
```

```
    db.logger = MagicMock()
```

```
    result = db.add_request("testuser", "This is a test request")
```

```
    assert result is False
```

```
@patch.object(Database, 'execute_query', return_value=[("2024-12-31",)])
```

```
def test_verify_premium_status_success(mock_execute_query):
```

```
    db = Database("config.json")
```

```
    db.logger = MagicMock()
```

```
    result = db.verify_premium_status("testuser")
```

```
    assert result == "2024-12-31"
```

```
@patch.object(Database, 'execute_query', return_value=[])
```

```
def test_verify_premium_status_failure(mock_execute_query):
```

```
    db = Database("config.json")
```

```
    db.logger = MagicMock()
```

```
    result = db.verify_premium_status("testuser")
```

```
    assert result is False
```

```
@patch.object(Database, 'execute_query', return_value=[("6 months", "49.99", "Half year subscription")])
```

```
def test_get_subscriptions(mock_execute_query):
```

```
    db = Database("config.json")
```

```
    db.logger = MagicMock()
```

```
    result = db.get_subscriptions("EN")
```

```
    assert result == [("6 months", "49.99", "Half year subscription")]
```

```
@patch.object(Database, 'execute_query', return_value=True)
```

```
def test_reset_password_success(mock_execute_query):
```

```
    db = Database("config.json")
```

```
    db.logger = MagicMock()
```

```
    result = db.reset_password("testuser", "newpassword")
```

```
    assert result is True
```

```
@patch.object(Database, 'execute_query', return_value=False)
```

```
def test_reset_password_failure(mock_execute_query):
```

```
    db = Database("config.json")
```

```
    db.logger = MagicMock()
```

```
    result = db.reset_password("testuser", "newpassword")
```

```
    assert result is False
```

```
@patch.object(Database, 'execute_query', return_value=[("user1",), ("user2",)])
```

```
def test_get_all_users(mock_execute_query):
```

```
    db = Database("config.json")
```

```
    db.logger = MagicMock()
```

```
    result = db.execute_query("SELECT * FROM users")
```

```
    assert result == [("user1",), ("user2",)]
```

B.1.5. server_tests.py

```
import sys
```

```
sys.path.insert(0, "d:\\diploma\\server-side")
```

```
print(sys.path)
```

```
import pytest
```

```
from unittest.mock import patch, MagicMock
```

```

from server import Server # Replace with the actual module name
from database import Database
import os
import socket

@pytest.fixture
def server():
    server_instance = Server()
    yield server_instance
    # Teardown: Close database connection after each test

def test_get_models(server):
    # Mock the database response
    server.server_database.get_models = MagicMock(return_value=[("model1", "description1", "version1"),
                                                                ("model2", "description2", "version2")])

    # Call the method under test
    models = server.get_models()

    # Check if the method returns the expected data
    assert models == b'model1;description1;version1|model2;description2;version2'

def test_reset_password(server):
    # Mock the database response
    server.server_database.reset_password = MagicMock(return_value=True)

```

```

# Call the method under test
result = server.reset_password("username", "new_password")

# Check if the method returns the expected result
assert result == b'True'

def test_get_subscriptions(server):
    # Mock the database response
    server.server_database.get_subscriptions = MagicMock(return_value=[("subscription1", "description1",
"price1"),
                                                                    ("subscription2", "description2", "price2")])

    # Call the method under test
    subscriptions = server.get_subscriptions("locale")

    # Check if the method returns the expected data
    assert subscriptions == b'subscription1;description1;price1|subscription2;description2;price2'

def test_register_new_user(server):
    # Mock the database response
    server.server_database.add_new_user = MagicMock(return_value=True)

    # Call the method under test
    result = server.register_new_user("username", "password")

    # Check if the method returns the expected result
    assert result == b'done'

```

```
def test_send_confirmation_code(server):  
    # Mock the database response  
    server.server_database.generate_confirmation_code = MagicMock(return_value="123456")  
  
    # Call the method under test  
    result = server.send_confirmation_code("username")  
  
    # Check if the method returns the expected result  
    assert result == b'send'  
  
def test_verify_confirmation_code(server):  
    # Mock the database response  
    server.server_database.verify_confirmation_code = MagicMock(return_value=True)  
  
    # Call the method under test  
    result = server.verify_confirmation_code("username", "123456")  
  
    # Check if the method returns the expected result  
    assert result == b'True'  
  
def test_verify_user_premium(server):  
    # Mock the database response  
    server.server_database.verify_premium_status = MagicMock(return_value=True)  
  
    # Call the method under test  
    result = server.verify_user_premium("username")
```



```
# Check if the method returns the expected result
assert result == b'True'
```

```
def test_check_updates(server):
```

```
    # Call the method under test
    result = server.check_updates("1.0.2")
```

```
    # Check if the method returns the expected result
    assert result == b'True'
```

```
def test_verify_credits(server):
```

```
    # Mock the database response
    server.server_database.verify_user_credits = MagicMock(return_value=10)
```

```
    # Call the method under test
    result = server.verify_credits("username")
```

```
    # Check if the method returns the expected result
    assert result == b'10'
```

```
def test_verify_token(server):
```

```
    # Mock the database response
    server.server_database.verify_auth_token = MagicMock(return_value=True)
```

```
    # Call the method under test
    result = server.verify_token("token")
```

```

# Check if the method returns the expected result
assert result == b'True'

def test_user_exists(server):
    # Mock the database response
    server.server_database.check_login_exists = MagicMock(return_value=True)

    # Call the method under test
    result = server.user_exists("username")

    # Check if the method returns the expected result
    assert result == b'True'

def test_user_verification(server):
    # Mock the database response
    server.server_database.check_login_credentials = MagicMock(return_value=True)
    server.server_database.add_auth_token = MagicMock(return_value="token")

    # Call the method under test
    result = server.user_verification("username", "password")

    # Check if the method returns the expected result
    assert result == b'token'

# Add more test cases for other methods as needed

```

В.2. Інтеграційні тести

B.2.1. integrated_db_serv.py

```
import sys

sys.path.insert(0, "d:\\diploma\\server-side")

print(sys.path)


import pytest
import socket
import threading
import os
import bcrypt
from server import Server
from database import Database


def create_test_image():
    from PIL import Image

    # Define the size of the square image
    size = 500

    # Create a new square image
    img = Image.new('RGB', (size, size), 'red')

    # Save the image as test_image.jpg
    img.save('uploaded_images\\test_image.jpg')
```

```

# Fixture to set up the test database
@pytest.fixture(scope="module")
def test_db():
    # Setup the test database
    db = Database('config.json') # Ensure you have a test config file
    db.connect()

    # Populate with some test data if needed
    # db.add_new_user('test_user', bcrypt.hashpw('password'.encode('utf-8'), bcrypt.gensalt()))

    yield db

    # Teardown the test database
    db.close()

# Fixture to set up the server
@pytest.fixture(scope="module")
def server(test_db):
    server_instance = Server()
    server_instance.server_database = test_db # Use the test database
    server_thread = threading.Thread(target=server_instance.run_server, daemon=True)
    server_thread.start()

    yield server_instance

    # No explicit teardown needed since daemon=True ensures it exits with the main program

def test_register_new_user(server):
    # Use a socket to communicate with the server

```

```

client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client_socket.connect((server.addr_ip, server.addr_port))

request = "REG|new_user|password"
client_socket.send(request.encode("utf-8"))
response = client_socket.recv(server.buffer).decode("utf-8")

assert response == "done"

# Cleanup: Remove the user from the database after the test
server.server_database.delete_user("new_user")
client_socket.close()

def test_user_verification(server):
    # Pre-register a user in the test database
    username = "test_user"
    password = "password"
    hashed_password = bcrypt.hashpw(password.encode('utf-8'), bcrypt.gensalt())
    server.server_database.add_new_user(username, hashed_password)

    # Use a socket to communicate with the server
    client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    client_socket.connect((server.addr_ip, server.addr_port))

    request = f"UCV|{username}|{password}"
    client_socket.send(request.encode("utf-8"))
    response = client_socket.recv(server.buffer).decode("utf-8")

    assert response == server.server_database.add_auth_token(username, password)

```

```

# Cleanup: Remove the user from the database after the test
server.server_database.delete_user(username)

client_socket.close()

def test_process_image(server):

    # Pre-register a user in the test database
    username = "test_user"
    password = "password"
    hashed_password = bcrypt.hashpw(password.encode('utf-8'), bcrypt.gensalt())
    server.server_database.add_new_user(username, hashed_password) # Ensure the user has credits

    # Use a socket to communicate with the server
    client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    client_socket.connect((server.addr_ip, server.addr_port))

    image_name = "test_image.jpg"
    image_path = os.path.join(server.images_dir, image_name)

    config = "caption_max_length=32;caption_model_name=blip-large"
    request = f"IMG|{image_name}|{config}|best|{username}"
    client_socket.send(request.encode("utf-8"))
    response = client_socket.recv(server.buffer).decode("utf-8")

    # Assuming the server should return some prompt or "no credit"
    print(f"Server response: {response}") # Print the server's response (prompt or response)
    assert response != "no credit"

    # Cleanup: Remove the user and the image from the database and filesystem after the test

```

```
server.server_database.delete_user(username)
```

```
if __name__ == '__main__':
```

```
    #create_test_image()
```

```
    pass
```

B.2.2. login_page_tests.py

```
import sys
```

```
sys.path.insert(0, "d:\\diploma\\app")
```

```
print(sys.path)
```

```
import unittest
```

```
from unittest.mock import patch, MagicMock
```

```
import ttkbootstrap as ttk
```

```
from login_page import LoginPage, ApplicationConfiguration, LOGIN_PAGE
```

```
class TestLoginPage(unittest.TestCase):
```

```
    def setUp(self):
```

```
        # Setup a root window for the test
```

```
        self.root = ttk.Window()
```

```
        self.root.current_theme = 'journal'
```

```
        self.app_configuration = ApplicationConfiguration()
```

```
        self.login_page = LoginPage(master=self.root)
```

```
        self.root.login_page = self.login_page
```

```
        self.root.form_main_page = self.form_main_page
```

```
    def form_main_page(self):
```

```

print("Imitation of main_page_packing")

def tearDown(self):
    # Clean up the root window
    self.root.destroy()

def test_initialization(self):
    # Test if LoginPage initializes with all components
    self.assertIsInstance(self.login_page, ttk.Frame)
    self.assertIsNotNone(self.login_page.change_theme_button)
    self.assertIsNotNone(self.login_page.login_email_entry)
    self.assertIsNotNone(self.login_page.login_password_entry)
    self.assertIsNotNone(self.login_page.login_button)

@patch('client.Client.process_request', return_value="False")
@patch('tkbootstrap.dialogs.Messagebox.show_error')
def test_login_failure(self, mock_messagebox, mock_process_request):
    # Test login failure case
    self.login_page.email_variable.set("test@example.com")
    self.login_page.password_variable.set("wrongpassword")
    self.login_page.login_to_app()

    mock_process_request.assert_called_with("UCV|test@example.com|wrongpassword")

mock_messagebox.assert_called_with(LOGIN_PAGE["message_wrong_email_or_pass"][self.app_configuration.locale], "Error", parent=self.login_page)

@patch('client.Client.process_request', return_value="sometoken")
@patch('builtins.open', new_callable=unittest.mock.mock_open)
def test_login_success(self, mock_open, mock_process_request):
    # Test login success case

```



```

self.login_page.email_variable.set("test@example.com")
self.login_page.password_variable.set("correctpassword")
self.login_page.login_to_app()

mock_process_request.assert_called_with("UCV|test@example.com|correctpassword")
mock_open.assert_called_with("token", 'w')
mock_open().write.assert_called_with("test@example.com|sometoken")

```

```

def test_change_theme(self):
    # Test theme change
    initial_theme = self.login_page.current_theme
    print(initial_theme)
    self.login_page.change_current_theme()
    new_theme = self.login_page.current_theme

    self.assertNotEqual(initial_theme, new_theme)

```

```

@patch('login_page.RegistrationWindow')

```

```

def test_registration_button(self, mock_registration_window):
    # Test registration button triggers RegistrationWindow
    self.login_page.registration_button.invoke()
    mock_registration_window.assert_called_with(self.login_page)

```

```

@patch('login_page.ForgotPasswordWindow')

```

```

def test_forgot_password_button(self, mock_forgot_password_window):
    # Test forgot password button triggers ForgotPasswordWindow
    self.login_page.forgot_password_button.invoke()
    mock_forgot_password_window.assert_called_with(self.login_page)

```

```

if __name__ == '__main__':

```

```
unittest.main()
```

B.2.3. admin_app_tests.py

```
import sys
```

```
sys.path.insert(0, "d:\\diploma\\server-side")
```

```
print(sys.path)
```

```
import pytest
```

```
import tkinter as tk
```

```
from admin_app import UsersTab, PaymentsTab, QuestionsTab, ServerTab
```

```
@pytest.fixture
```

```
def root():
```

```
    return tk.Tk()
```

```
def test_payments_tab_creation(root):
```

```
    payments_tab = PaymentsTab(root)
```

```
    payments_tab.pack()
```

```
    assert payments_tab.winfo_exists() == 1
```

```
    assert isinstance(payments_tab.tree, tk.ttk.Treeview)
```

```
    assert len(payments_tab.tree.get_children()) == 1 # Assuming the table is empty initially
```

```
    payments_tab.master.destroy()
```

```
def test_users_tab_creation(root):
```

```
    users_tab = UsersTab(root)
```

```
    users_tab.pack()
```

```
    assert users_tab.winfo_exists() == 1
```

```

assert isinstance(users_tab.tree, tk.ttk.Treeview)
assert len(users_tab.tree.get_children()) == 5 # Assuming the table is empty initially

users_tab.master.destroy()

def test_questions_tab_creation(root):
    questions_tab = QuestionsTab(root)
    questions_tab.pack()

    assert questions_tab.winfo_exists() == 1
    assert isinstance(questions_tab.tree, tk.ttk.Treeview)
    assert len(questions_tab.tree.get_children()) == 0 # Assuming the table is empty initially

    questions_tab.master.destroy()

def test_request_tip_window_creation(root):
    requests = ["Request 1", "Request 2", "Request 3"]
    request_tip_window = ServerTab.RequestTipWindow(root, requests)

    assert request_tip_window.winfo_exists() == 1
    assert isinstance(request_tip_window.requests_text, tk.Text)
    request_tip_window.requests_text.configure(state="normal") # Enable state to check text
    text_content = request_tip_window.requests_text.get("1.0", tk.END)

    for request in requests:
        assert request in text_content

    request_tip_window.master.destroy()

```

В.3. Функціональні тести

В.3.1. payme_tests.py

```

import sys

sys.path.insert(0, "d:\\diploma\\server-side")
sys.path.insert(0, "d:\\diploma\\app")


import pytest

from app import PaymentWindow

import ttkbootstrap as ttk


@pytest.fixture
def app(qtbot):
    """Fixture to create an instance of the app for testing."""
    app = PaymentWindow(ttk.Window())
    qtbot.addWidget(app)
    return app


def test_TP01(app, qtbot):
    """Тестування можливості придбання підписки."""
    qtbot.mouseClick(app.subscription_page_button, qtbot.LeftButton)
    subscriptions = app.get_available_subscriptions()
    app.select_subscription(subscriptions[0])
    qtbot.mouseClick(app.purchase_button, qtbot.LeftButton)
    assert app.subscription_activated.isVisible()


def test_TP02(app, qtbot):
    """Тестування можливості перегляду наявних підписок."""
    qtbot.mouseClick(app.view_subscriptions_button, qtbot.LeftButton)
    assert app.subscription_list.isVisible()
    for subscription in app.subscription_list:

```

```
assert subscription.has_price_and_discount_info()
```

```
def test_TP03(app, qtbob):
    """Тестування можливості керування підпискою."""
    qtbob.mouseClick(app.manage_subscription_button, qtbob.LeftButton)
    app.select_subscription_to_manage(app.active_subscription)
    qtbob.mouseClick(app.change_settings_button, qtbob.LeftButton)
    app.update_subscription_settings({"renewal": False, "notification": True})
    qtbob.mouseClick(app.save_settings_button, qtbob.LeftButton)
    assert app.settings_updated.isVisible()
```

B.3.2. image_tests.py

```
import sys
sys.path.insert(0, "d:\\diploma\\server-side")
sys.path.insert(0, "d:\\diploma\\app")

import pytest
from app import MainTab
import ttkbootstrap as ttk

@pytest.fixture
def app(qtbob):
    """Fixture to create an instance of the app for testing."""
    app = MainTab()
    qtbob.addWidget(app)
    return app

def test_TO01(app, qtbob):
    """Тестування можливості системи перетворювати різні формати зображень."""
    formats = ["image.jpg", "image.png", "image.bmp"]
```

```

for format in formats:
    app.load_image(format)
    qtbot.mouseClick(app.convert_button, qtbot.LeftButton)
    assert app.image_converted.isVisible()

```

```

def test_TO02(app, qtbot):
    """Тестування функціоналу масштабування та обрізки зображень."""
    app.load_image("image.jpg")
    qtbot.mouseClick(app.resize_button, qtbot.LeftButton)
    qtbot.mouseClick(app.crop_button, qtbot.LeftButton)
    assert app.image_resized.isVisible()
    assert app.image_cropped.isVisible()

```

```

def test_TO05(app, qtbot):
    """Тестування можливості збереження результатів обробки зображень."""
    app.load_image("image.jpg")
    qtbot.mouseClick(app.process_button, qtbot.LeftButton)
    qtbot.mouseClick(app.save_button, qtbot.LeftButton)
    assert app.save_dialog.isVisible()

```

```

def test_TO06(app, qtbot):
    """Тестування опрацювання зображень за допомогою завантажених моделей обробки."""
    app.load_image("image.jpg")
    app.select_model("model_1")
    qtbot.mouseClick(app.process_button, qtbot.LeftButton)
    assert app.processed_image.isVisible()

```

```

def test_TO07(app, qtbot):
    """Тестування можливості вибору типу опрацювання зображень."""
    qtbot.mouseClick(app.processing_type_button, qtbot.LeftButton)

```

```
app.select_processing_type("Type_1")
assert app.processing_type_selected.isVisible()
```

```
def test_TO08(app, qtbot):
```

```
    """Тестування відображення повідомлення про недостатню кількість кредитів чи відсутність підписки."""
```

```
    app.select_processing_type("Server")
    qtbot.mouseClick(app.process_button, qtbot.LeftButton)
    assert app.insufficient_credits_message.isVisible()
```

```
def test_TO09(app, qtbot):
```

```
    """Тестування відображення повідомлення про перевірку потужності ПК."""
```

```
    app.select_processing_type("Local")
    qtbot.mouseClick(app.process_button, qtbot.LeftButton)
    assert app.checking_pc_power_message.isVisible()
```

```
def test_TO10(app, qtbot):
```

```
    """Тестування можливості налаштування опрацювання зображень."""
```

```
    qtbot.mouseClick(app.settings_button, qtbot.LeftButton)
    app.adjust_settings({"brightness": 50, "contrast": 30})
    qtbot.mouseClick(app.apply_settings_button, qtbot.LeftButton)
    assert app.settings_applied.isVisible()
```

```
def test_TO11(app, qtbot):
```

```
    """Тестування можливості збереження результатів опрацювання зображень за заданими PROMPTs."""
```

```
    app.load_image("image.jpg")
    qtbot.mouseClick(app.process_button, qtbot.LeftButton)
    qtbot.mouseClick(app.save_prompt_button, qtbot.LeftButton)
    app.set_prompt("example_prompt")
    assert app.prompt_saved.isVisible()
```

```
def test_TO12(app, qtbot):
    """Тестування можливості завантаження нових моделей обробки зображень."""
    qtbot.mouseClick(app.load_model_button, qtbot.LeftButton)
    app.upload_model("new_model")
    assert app.model_uploaded.isVisible()
```

B.3.3. admin_tests.py

```
import sys
sys.path.insert(0, "d:\\diploma\\server-side")
sys.path.insert(0, "d:\\diploma\\app")
```

```
import pytest
from app import MenuApp
import login_page as AuthWindow
import admin_app as AdminPanel
import ttkbootstrap as ttk
```

```
@pytest.fixture
def app(qtbot):
    """Fixture to create an instance of the app for testing."""
    app = MenuApp(ttk.Window())
    qtbot.addWidget(app)
    return app
```

```
def test_TA01(app, qtbot):
    """Тестування змоги системи авторизувати користувачів."""
    assert app.auth_window.isVisible()
    assert "Login" in app.auth_window.title()
```



```
def test_TA02(app, qtb):
```

```
    """Тестування авторизації користувача за допомогою логіну і паролю."""
    app.auth_window.login_input.setText("user_login")
    app.auth_window.password_input.setText("user_password")
    qtb.mouseClick(app.auth_window.login_button, qtb.LeftButton)
    assert app.main_interface.isVisible()
```

```
def test_TA03(app, qtb):
```

```
    """Тестування авторизації користувача за допомогою пошти і паролю."""
    app.auth_window.email_input.setText("user@example.com")
    app.auth_window.password_input.setText("user_password")
    qtb.mouseClick(app.auth_window.login_button, qtb.LeftButton)
    assert app.main_interface.isVisible()
```

```
def test_TA04(app, qtb):
```

```
    """Тестування реєстрації користувача за допомогою пошти і паролю."""
    app.auth_window.register_button.click()
    app.auth_window.email_input.setText("newuser@example.com")
    app.auth_window.password_input.setText("new_password")
    qtb.mouseClick(app.auth_window.register_button, qtb.LeftButton)
    assert "Confirmation Code" in app.auth_window.title()
```

```
def test_TA05(app, qtb):
```

```
    """Тестування функціоналу для керування аккаунтом користувача."""
    app.auth_window.login_input.setText("user_login")
    app.auth_window.password_input.setText("user_password")
    qtb.mouseClick(app.auth_window.login_button, qtb.LeftButton)
    assert app.account_page.isVisible()
```

```

def test_TA06(app, qtb):
    """Тестування можливості користувачів налаштовувати параметри облікового запису."""
    app.auth_window.login_input.setText("user_login")
    app.auth_window.password_input.setText("user_password")
    qtb.mouseClick(app.auth_window.login_button, qtb.LeftButton)
    app.account_page.settings_button.click()
    assert app.settings_page.isVisible()

def test_TA07(app, qtb):
    """Тестування функціоналу керування користувачами та їх обліковими записами."""
    app.auth_window.login_input.setText("admin")
    app.auth_window.password_input.setText("admin_password")
    qtb.mouseClick(app.auth_window.login_button, qtb.LeftButton)
    assert app.admin_panel.isVisible()
    app.admin_panel.manage_users_button.click()
    assert app.admin_panel.user_management_page.isVisible()

def test_TA08(app, qtb):
    """Тестування створення тестового користувача та перегляду списку існуючих користувачів."""
    app.auth_window.login_input.setText("admin")
    app.auth_window.password_input.setText("admin_password")
    qtb.mouseClick(app.auth_window.login_button, qtb.LeftButton)
    app.admin_panel.manage_users_button.click()
    app.admin_panel.user_management_page.create_user("testuser", "testuser@example.com")
    assert "testuser" in app.admin_panel.user_management_page.user_list

def test_TA09(app, qtb):
    """Тестування відображення підтверджувального вікна при видаленні користувача."""
    app.auth_window.login_input.setText("admin")
    app.auth_window.password_input.setText("admin_password")

```

```

qtboto.mouseClick(app.auth_window.login_button, qtboto.LeftButton)
app.admin_panel.manage_users_button.click()
app.admin_panel.user_management_page.select_user("testuser")
qtboto.mouseClick(app.admin_panel.user_management_page.delete_button, qtboto.LeftButton)
assert app.admin_panel.user_management_page.confirmation_dialog.isVisible()

```

```
def test_TA10(app, qtboto):
```

```

    """Тестування можливості адміністраторів керувати запитаннями та відповідями."""
    app.auth_window.login_input.setText("admin")
    app.auth_window.password_input.setText("admin_password")
    qtboto.mouseClick(app.auth_window.login_button, qtboto.LeftButton)
    app.admin_panel.manage_questions_button.click()
    assert app.admin_panel.question_management_page.isVisible()

```

```
def test_TA11(app, qtboto):
```

```

    """Тестування можливості адміністраторів керувати платежами."""
    app.auth_window.login_input.setText("admin")
    app.auth_window.password_input.setText("admin_password")
    qtboto.mouseClick(app.auth_window.login_button, qtboto.LeftButton)
    app.admin_panel.manage_payments_button.click()
    assert app.admin_panel.payment_management_page.isVisible()

```

```
def test_TA12(app, qtboto):
```

```

    """Тестування сторінки управління платежами."""
    app.auth_window.login_input.setText("admin")
    app.auth_window.password_input.setText("admin_password")
    qtboto.mouseClick(app.auth_window.login_button, qtboto.LeftButton)
    app.admin_panel.manage_payments_button.click()
    assert app.admin_panel.payment_management_page.isVisible()

```

```
def test_TA13(app, qtbot):
    """Тестування сторінки деталізації платежів."""
    app.auth_window.login_input.setText("admin")
    app.auth_window.password_input.setText("admin_password")
    qtbot.mouseClick(app.auth_window.login_button, qtbot.LeftButton)
    app.admin_panel.manage_payments_button.click()
    app.admin_panel.payment_management_page.select_payment("payment_1")
    assert app.admin_panel.payment_detail_page.isVisible()
```

```
def test_TA14(app, qtbot):
    """Тестування можливості адміністраторів переглядати логи серверу."""
    app.auth_window.login_input.setText("admin")
    app.auth_window.password_input.setText("admin_password")
    qtbot.mouseClick(app.auth_window.login_button, qtbot.LeftButton)
    app.admin_panel.view_logs_button.click()
    assert app.admin_panel.logs_page.isVisible()
```

B.3.4. infor_tests.py

```
import sys

sys.path.insert(0, "d:\\diploma\\server-side")
sys.path.insert(0, "d:\\diploma\\app")

import pytest

from app import MenuApp, DocumentationTab

import ttkbootstrap as ttk

@pytest.fixture
def menu_app(qtbot):
    """Fixture to create an instance of MenuApp for testing."""
    root = ttk.Window()
```

```

root.current_theme = 'journal'

window = MenuApp(root)

qtbol.addWidget(window)

return window

```

```

def test_ti1(menu_app, qtbol):

    """Test the possibility for users to read the application documentation."""

    # Simulate the transition to the application documentation page
    menu_app.show_documentation()

    # Check if the documentation page is displayed correctly
    assert menu_app.documentation_page.isVisible()

    assert "Documentation" in menu_app.documentation_page.title()

    # Add more assertions to check the contents of the documentation page

```

```

def test_ti2(menu_app, qtbol):

    """Test the functionality of the image scaling and cropping page."""

    # Simulate the transition to the image scaling and cropping page
    menu_app.show_scaling_and_cropping()

    # Check if the scaling and cropping page is displayed correctly
    assert menu_app.scaling_and_cropping_page.isVisible()

    # Set the parameters for scaling and cropping
    menu_app.scaling_and_cropping_page.set_dimensions(800, 600)
    menu_app.scaling_and_cropping_page.set_scaling_mode('Fit')
    menu_app.scaling_and_cropping_page.set_cropping_mode('Center')

    # Simulate pressing the "Apply" button

```

```
qtboto.mouseClick(menu_app.scaling_and_cropping_page.apply_button, qtboto.LeftButton)
```

```
# Verify that the changes take effect
```

```
assert menu_app.scaling_and_cropping_page.apply_changes()
```