

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Інститут інформатики та радіоелектроніки,
Факультет комп'ютерних наук і технологій
(повне найменування інституту, назва факультету)

Кафедра програмних засобів
(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

бакалавр

(ступінь вищої освіти)

на тему РОЗРОБКА ПРОГРАМИ ЗБЕРЕЖЕННЯ ЗНІМКІВ ЕКРАНУ
SCREEN CAPTURE SOFTWARE DEVELOPMENT

Виконав: студент(ка) 4 курсу, групи КНТ-219сп
Спеціальності 122 Комп'ютерні науки
(код і найменування спеціальності)

Освітня програма (спеціалізація)
Комп'ютерні науки

Ганжа Д.І.

(прізвище та ініціали)

Керівник Сердюк С.М.

(прізвище та ініціали)

Рецензент Казурова А.Є.

(прізвище та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
(повне найменування закладу вищої освіти)

Інститут, факультет ПРЕ, ФКНТ
Кафедра програмних засобів
Ступінь вищої освіти бакалавр
Спеціальність 122 Комп'ютерні науки
(код і найменування)
Освітня програма (спеціалізація) Комп'ютерні науки
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
С.О. Субботін
“ ” 2022 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЕКТ (РОБОТУ) СТУДЕНТА(КИ)

Ганжа Дмитро Ігорович

(прізвище, ім'я, по батькові)

1. Тема проекту (роботи) Розробка програми збереження знімків екрану
Screen Capture Software Development

керівник проекту (роботи) Сердюк Сергій Микитович, к.т.н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від “27” квітня 2022 року № 102

2. Строк подання студентом проекту (роботи) 7 червня 2022 року

3. Вихідні дані до проекту (роботи) рекомендована література, технічне
завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Розробка структури програмного забезпечення. 3. Основні рішення щодо реалізації компонентів системи. 4. Керівництво програміста. 5. Керівництво оператора. Додатки.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-5 Основна частина	Сердюк С.М., доцент		
Нормоконтроль	Белова А.В., асистент		

7. Дата видачі завдання 28 березня 2022 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	2–3 тижні	Розділ 1
3	Розробка архітектури програми.	4 тиждень	Розділ 2
4	Розробка програми.	5–6 тижні	Розділ 3
5	Тестування та експериментальне дослідження програми.	7 тиждень	Розділ 4
6	Оформлення пояснювальної записки та документів до неї. Нормоконтроль та рецензування	8–9 тижні	Розділ 5, Додатки
7	Захист роботи.	10 тиждень	

Студент(ка)

(підпис) Ганжа Д.І.
(прізвище та ініціали)

Керівник проєкту (роботи)

(підпис) Сердюк С.М.
(прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 82 с., 4 табл., 38 рис., 2 дод., 18 джерел.

СКРІНШОТ, ЗАХОПЛЕННЯ ЗОБРАЖЕНЬ, ВИДІЛЕННЯ ОБЛАСТІ ЕКРАНУ, РЕДАГУВАННЯ ЗОБРАЖЕНЬ, ЗБЕРЕЖЕННЯ, ЗАВАНТАЖЕННЯ, DESKTOP, ВЕБ-ЗАСТОСУНОК, C#, ОПЕРАЦІЙНА СИСТЕМА, WINDOWS, PYTHON.

Об'єкт дослідження – процеси запису та редагування зображень екрану персонального комп'ютера (ПК).

Предмет дослідження – методи запису та редагування зображень екрану персонального комп'ютера.

Мета роботи – розробка програми, що дозволяє захоплювати зображення з області екрану, редагувати отриманий результат та зберігати вихідне зображення у пам'яті ПК або на сервері.

Матеріали, методи та технічні засоби: об'єктно-орієнтоване програмування; мови програмування C# та Python; програмна платформа .NET; фреймворк Windows Forms для розробки Desktop-застосунків; фреймворк Django для створення веб застосунків та бібліотеки PIL для роботи з зображеннями.

Результати. Розроблено програмне забезпечення, що складається з Desktop застосунку для захоплення та редагування зображень з екрану, а також веб застосунку для збереження зображень у мережі Інтернет.

Галузь використання – інформаційні технології.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 82 pages, 38 figures, 4 tables, 2 appendixes, 18 sources.

SCREENSHOOT, IMAGE CAPTURE, SCREEN VIEW AREAS, IMAGE EDITING, SAVE, IMAGE, DESKTOP, WEB DISPLAY, C#, OPERATING SYSTEM, WINDOWS, PYTHON.

Object of study are a personal computer (PC) screen capture and editing processes.

Subject of study are methods for screen capture and editing personal computer screen images.

The purpose of the work is development of an program that allows to capture images from the screen area, edit the result and save the original image in the PC memory or on the server.

Materials, methods and technical means: object-oriented programming; programming languages C# and Python; .NET software platform; Windows Forms framework for developing desktop applications; Django framework for creating web applications and PIL library for working with images.

Results. Software has been developed, consisting of a Desktop application for capturing and editing images from the screen, as well as a web application for saving images on the Internet.

The field of use is Information Technology.

ЗМІСТ

	С.
Перелік скорочень та умовних позначок.....	8
Вступ.....	9
1 Аналіз предметної області.....	11
1.1 Призначення та способи реалізації скріншотів	11
1.2 Огляд програмних аналогів.....	11
1.3 Розробка технічного завдання	18
1.3.1 Підстава для розробки	18
1.3.2 Мета розробки та її призначення.....	18
1.3.3 Основні вимоги до програмного продукту	18
1.3.4 Вимоги до технічних засобів, операційної системи і мови програмування	19
1.3.5 Вимоги до інформаційної та програмної сумісності.....	20
1.3.6 Вимоги до програмної документації.....	20
1.3.7 Порядок приймання і тестування	20
2 Розробка структури програмного забезпечення	21
2.1 Аналіз технічного завдання	21
2.2 Визначення компонентів системи	22
2.3 Обґрунтування вибору технічних і програмних засобів.....	25
2.4 Розробка сценарію взаємодії користувача з програмою.....	27
3 Основні рішення щодо реалізації компонентів системи.....	30
3.1 Розробка Desktop-застосунку	30
3.1.1 Розробка класів Desktop-застосунку	30
3.1.2 Розробка графічного інтерфейсу користувача.....	33
3.1.3 Особливості реалізації Desktop-застосунку	37
3.2 Розробка веб застосунку.....	37
3.2.1 Розробка класів Desktop-застосунку	37
3.2.2 Розробка графічного інтерфейсу веб застосунку	39
4 Керівництво програміста.....	40

- 4.1 Призначення та умови застосування програми 40
- 4.2 Характеристики програми..... 40
- 4.3 Вхідні та вихідні дані..... 40
- 4.4 Опис атрибутів та методів..... 41
- 4.5 Повідомлення програмісту..... 45
- 5 Керівництво оператора 46
 - 5.1 Призначення та умови виконання програми..... 46
 - 5.2 Виконання програми..... 46
 - 5.3 Повідомлення оператору 50
- Висновки 51
- Перелік джерел посилання 53
- Додаток А. Текст програми..... 55
- Додаток Б. Слайди презентації 74

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

API	– Application Programming Interfaces;
FPT	– File Transport Protocol;
HTTP	– File Transport Protocol;
Гб	– гігабайт;
ГГц	– гігагерц;
ЛКМ	– ліва кнопка миші;
Мб	– мегабайт;
ОС	– операційна система;
ПЗ	– програмне забезпечення;
ПК	– персональний комп'ютер.

ВСТУП

Питання обміну інформацією в мережі Інтернет є дуже актуальним. Звичайною є справа, коли постає проблема показати хід виконання програмного забезпечення, проблеми, що виникають під час його експлуатації і таке інше.

Скріншоти – це те, що більшість користувачів створює на регулярній основі, навіть не замислюючись про це. Вони зручні для збереження зображення, для довідки або доказу. Також це швидкий спосіб поділитися чимось забавним або цікавим. Але якщо користувачу недостатньо функціональності від вбудованої програми знімка екрана свого пристрою, можна перейти на одну з альтернатив програм, що розповсюджуються безкоштовно або платно.

Програми захоплення зображення з екрану дозволяють швидко виділити головне, підкреслити, сформувати точки фокусу, на яких зупиняється зір людини або навпаки, прибрати зайве, те, чого хтось побачити не повинен.

Найкращі програми для створення скріншотів також пропонують безліч можливостей для позначок, редагування та обміну знімками екрана між користувачами. Деякі з них можуть пропонувати захоплення всієї сторінки для довгих веб сайтів або веб сайтів з боковим прокручуванням і навіть запис екрана, наприклад для навчального відео [1].

Подібне програмне забезпечення, найчастіше, надає можливості редагування вхідного зображення, наносячи лінії та прості геометричні фігури, тим самим, подекуди, замінюючи звичні всім редактори графіки, виступаючи простим і швидким рішенням.

Отже, створення програмного забезпечення для захоплення зображення з екрану є актуальним на даний момент.

Однією з корисних функцій є можливість додавати спеціальну затримку перед зйомкою екрана. Така можливість ідеально підходить, якщо

користувачу потрібно зробити щось у програмі в прямому ефірі, щоб зафіксувати кінцевий результат у вигляді знімка. Існують також застосунки, які надають можливість додати аудіо-розповідь, що може зробити, наприклад, навчальні посібники набагато якіснішими та зрозумілішими.

Актуальність даної кваліфікаційної роботи бакалавра полягає у створенні програмного забезпечення, що складається з двох модулів: Desktop застосунку для захоплення та редагування зображень з екрану, а також веб застосунку для збереження зображень у мережі Інтернет.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Призначення та способи реалізації скріншотів

Скріншот – це цифрова фотографія видимого контенту на екрані. Це може бути частина екрана або весь екран, а деякі додаткові інструменти можуть допомогти користувачеві покращити якість та виведення зображення. Скріншоти стали дуже популярними в наші дні, оскільки вони допомагають захоплювати зображення, які можна використовувати як посилання із затримкою початку запису в часі.

Зростання популярності скріншотів широко набуло поширення у сфері освіти і виявилось дуже корисним при створенні навчальних курсів або посібників з різних предметних областей, усунення неполадок або просто при створенні цікавого контенту для читання.

За замовчанням скріншот зберігається у вигляді файлу в буфері обміну або в папці зі скріншотами. Як правило, його також можна відправити безпосередньо на вибраний принтер і роздрукувати.

Якщо не зважати на можливість фотографування екрана реальною камерою, скріншоти існують з середини 1960-х років. У той час користувачі могли надрукувати вміст екрана одним натисканням кнопки [2].

Пізніше, у 1970-х і 80-х роках, розробники зрозуміли, наскільки корисними були знімки екрана, і почали розробляти комп'ютери з вбудованою функцією створення знімків екрана. Наприклад, термінали IBM VDU та операційні системи MS-DOS у 70-х роках мали функцію друку екрану, яка дозволяє роздруковувати текстовий вміст напряму через підключений порт принтера.

Таким чином, ранні версії функції друку екрана могли копіювати лише письмовий контент текстового формату. Однак, потім ця популярна функція була швидко розширена, і була реалізована можливість друку всього вмісту на екрані, або вибраної його частини. Ця можливість значно полегшила багатьом користувачам спілкування один з одним, якщо

необхідно було передати інформацію про те, що відбувається на їхньому екрані.

Сучасні операційні системи зазвичай зберігають зображення у форматі BMP. Після появи смартфонів функція створення знімків екрана швидко стала стандартом для мобільних пристроїв (формат PNG або JPG) і виявилася корисною для збереження та обміну чатами та іншим контентом.

Практично всі сучасні операційні системи мають вбудовану функцію створення знімків екрана, що дозволяє користувачам робити знімки екрана. Скріншоти можна зробити або за допомогою клавіш, або натиснувши відповідну кнопку в інструменті створення знімків екрана.

У Windows та Linux з інтерфейсами GNOME та KDE користувач може зробити скріншот поточного екрана, натиснувши клавішу PrintScreen. При необхідності є також розширена функція скріншота, що дозволяє робити знімки активного вікна на екрані. Починаючи з випуску Windows Vista Microsoft також пропонує Snipping Tool як вбудоване рішення для створення знімків екрана та їх редагування за потреби.

На клавіатурі Mac для macOS немає клавіші PrintScreen для створення скріншотів. Однак у системі Apple є вбудована функція скріншота, доступ до якої можна отримати за допомогою клавіш. Використовуючи клавіші [Command] + [Shift] + [3], можна зробити скріншот всього екрана.

Коли користувач робить знімок екрана, скріншот копіюється в буфер обміну у форматі BMP і може бути вставлений, переданий та редагований в інших програмах або електронних листах. Починаючи з операційної системи Windows 8, знімки екрана також можна знайти в папці «Знімки екрана» одразу після їх створення. На комп'ютерах Mac, попередньо необхідно створити для цієї мети папку, інакше система автоматично збереже зображення на робочий стіл.

Скріншоти можуть бути корисними і навіть необхідні з різних причин. Розглянемо деякі з найбільш поширених причин, з яких користувач може зробити знімок екрана:

- на екрані з'явиться повідомлення про помилку або вікно повідомлення, яке потрібно побачити службі технічної підтримки комп'ютера, операційної системи або вебсайту;
- скріншоти можна використовувати для створення історичних записів про розробку та стан веб сайту, а потім порівнювати їх на різних етапах розробки;
- у покрокових інструкціях підручників використовуються скріншоти, щоб їх було легко зрозуміти читачам і глядачам;
- функція скріншота — дуже популярний інструмент у комп'ютерних іграх, що дозволяє гравцям фіксувати свої успіхи, рекорди, збої та кумедні або незвичні моменти;
- при створенні веб-сайтів та програмного забезпечення скріншоти полегшують спілкування всередині команди розробників при оцінці послідовностей програмування;
- коли справа доходить до знущань або домагань через текстовий або відеочат, скріншоти можна комбінувати з IP-адресами, щоб використовувати їх як докази злочинних дій в Інтернеті в режимі реального часу.

Важливим моментом при редагуванні зображень або написанні онлайн-статей, програмних інструкцій, оглядів або, як у даному випадку дипломних проєктів, необхідно пам'ятати, що онлайн-зображення на екрані захищені авторським правом. Отже, необхідно отримати згоду правовласника, щоб мати можливість публікувати чи розповсюджувати їх.

Хоча новітні операційні системи Windows, Mac та Linux пропонують вбудовані функції створення знімків екрана з базовими функціями, деяким користувачам можуть знадобитися додаткові функції для подальшого редагування зображень. У цих випадках може знадобитися безкоштовний або платний інструмент для створення скріншотів. Ці інструменти не тільки пропонують набагато ширший спектр функцій для виконання детального

редагування знімків екрана, але також дозволяють робити знімки екрана у фільмах і іграх або робити знімки екрана системи при використанні кількох моніторів. Деякі інструменти автоматизують процес завантаження скріншотів у хмарні послуги, такі як Google Drive або Dropbox.

Однією з корисних функцій є також можливість додавання затримки користувача перед створенням знімка екрана. Це дуже актуально для випадку, коли користувачеві необхідно продемонструвати роботу з програмою у прямому ефірі, щоб зафіксувати кінцевий результат як знімок. Крім того, надається можливість додавання аудіо оповіді, що може зробити набагато зрозумілішим різні навчальні посібники [2].

1.2 Огляд програмних аналогів

Перед безпосереднім проектуванням розроблюваної системи було проведено аналіз вже існуючих програмних рішень, виділено їх основні переваги та недоліки, а також сформовані відповідні висновки, щодо результатів аналізу.

Всі розглянуті у даному підрозділі програми-аналоги є безкоштовним програмним забезпеченням.

Одним із аналогів є Snipping Tool – програма для створення знімків екрану, яка розроблена компанією Microsoft та включена до Windows 7, Windows 8 та Windows 10 [3]. Вона має можливість робити знімок вікна або всього екрану, обрати область прямокутної або довільної форми. Після того, як користувач зробить знімок екрана, він автоматично скопіюється у вікно Snipping Tool. Звідти користувач може зробити коментарі, зберегти або ділитися фрагментом. Знімок можна зберегти у вигляді зображення (PNG, GIF або JPEG формату), веб-сторінки, скопіювати у буфер обміну або надіслати знімок електронною поштою.

Інтерфейс програми наведено на рис. 1.1.

До переваг даного ПЗ відносять:

- це стандартний застосунок, який не необхідно завантажувати (для користувачів ОС Windows);
- подібний до інших стандартних застосунків ОС Windows інтерфейс;
- можливість «захоплення» екрану з затримкою.

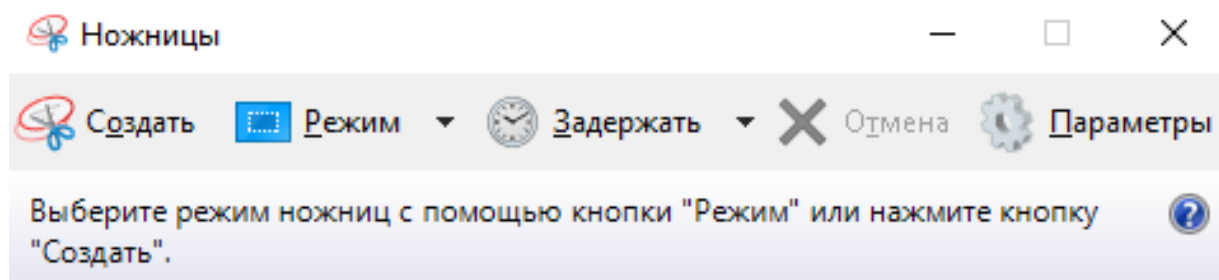


Рисунок 1.1 – Інтерфейс програми Snipping Tool

Серед недоліків програми можна виділити:

- перевантаженість інтерфейсу (високий рівень вкладеності, велика кількість меню, що випадають);
- під час редагування не зберігається початковий розмір «захопленого» зображення.

Наступним аналогом є Lightshot – програма для «захоплення» екрану для ОС Windows та Mac OS, що працює подібно до Snipping Tool, надає додаткові можливості по редагуванню зображень та публікації їх в Internet [4]. Програма розроблена компанією Skillbrains, доступна для завантаження за посиланням: <https://app.prntscr.com/ru/>.

Інтерфейс роботи програми наведено на рис. 1.2.

Серед переваг даного ПЗ можна виділити:

- мінімалістичний зручний інтерфейс з розташованими у одному вікні інструментами;
- можливість завантаження зображення на сервер, публікації його в Internet;

– можливість нанесення простих геометричних фігур під час редагування.

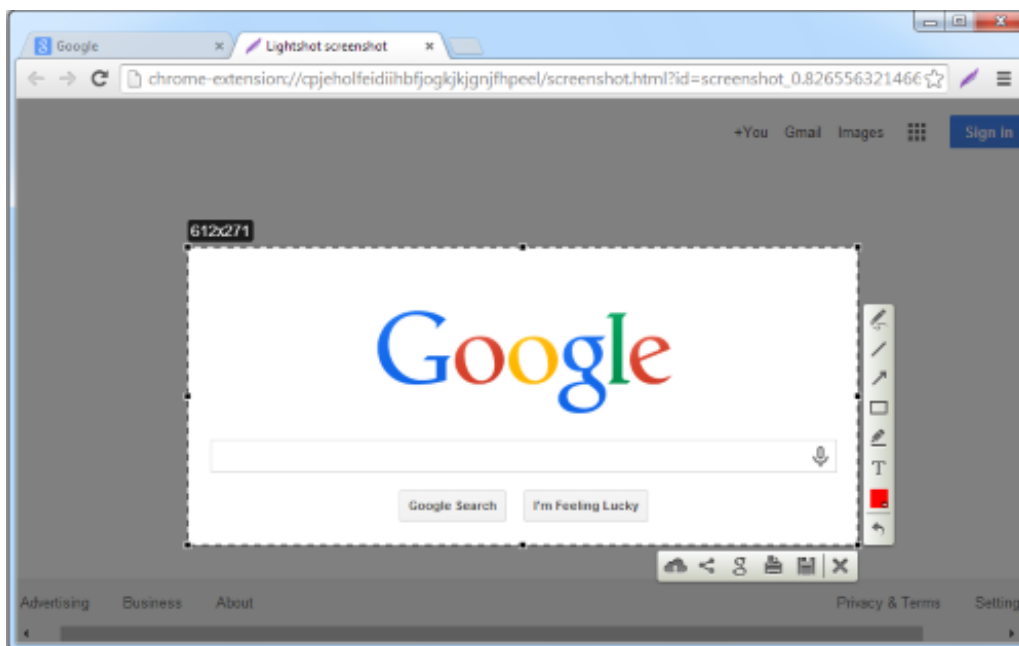


Рисунок 1.2 – Інтерфейс Lightshot

До недоліків даного ПЗ відносять:

- «перевантаження» стандартної клавіші PrintScreen;
- при «захваті» зображення з відкритих у повний екран застосунків, що мають іншу роздільність (відмінну від встановленої у налаштуваннях ОС), програма згортає застосунок.

Програма для захоплення скриншотів Joxi [5], має дещо розширену (відносно інших, розглянутих у цьому підрозділі) панель інструментів. Розроблена вона компанією «Джоксі». Доступна для завантаження за посиланням: <http://joxi.ru/>.

Інтерфейс Joxi наведено на рис.1.3.

Перевагами даного ПЗ можна назвати:

- розвинуті можливості редагування зображення (нанесення малюнків «пензлем», простих геометричних фігур, стрілок, тексту, розмиття,

нумерування, а також зміна положення вже нанесених об'єктів під час редагування);

- можливість захоплення відео;
- можливість збереження та публікації знімку екрану в Internet;
- можливість вибору якості та формату зображення у налаштуваннях;
- можливість вибору швидких клавіш у налаштуваннях.



Рисунок 1.3 – Інтерфейс Joxi

Серед недоліків слід назначити :

- вкладений та перевантажений інтерфейс;
- відсутність можливості відкладеного знімку.

В результаті огляду аналогів було виявлено, що більшість популярних застосунків для «захоплення» зображення з екрану мають наступну функціональність:

- «захоплення» зображення області екрану;
- завантаження зображення на сервер у мережі Інтернет;
- редагування вихідного зображення за допомогою графічних інструментів.

Серед переважаючих недоліків у застосунках-аналогах спостерігаються перевантаженість, а також висока ступінь вкладеності інтерфейсу.

Отже, розроблюваний програмний продукт має реалізувати «вдалу» функціональність, наявну у рішеннях аналогів, а також усувати їх недоліки.

Беручи до уваги усі перераховані вище переваги та недоліки існуючого програмного забезпечення необхідно розробити технічне завдання на застосунок, що розроблюється.

1.3 Розробка технічного завдання

1.3.1 Підстава для розробки

Підставою для розробки служить завдання на дипломну кваліфікаційну роботу бакалавра на тему "Розробка програми збереження знімків екрану", затверджене наказом № 102 від 27 квітня 2022 р. по НУ «Запорізька політехніка».

1.3.2 Мета розробки та її призначення

Метою є розробка програмного забезпечення, що призначене для захоплення зображення з екрану, редагування отриманого зображення та його збереження на сервері або у пам'яті комп'ютера.

1.3.3 Основні вимоги до програмного продукту

Програмне забезпечення, що розроблюється має надавати користувачеві наступні можливості:

- «захоплення» зображення з екрану цілком;
- кадрування («захоплення» області) екрану;

- редагування «захопленого» з екрану зображення (нанесення ліній, простих геометричних фігур (коло, овал, прямокутник, квадрат), стрілок, текстових міток та можливість задавати колір для виконання цих операцій; розмиття області зображення);
- збереження вихідного зображення у пам'яті комп'ютера;
- збереження вихідного зображення на сервері.

Вхідними даними, необхідними для функціонування програмного продукту, є поточне зображення екрану.

Вихідними даними є відредаговане та збережене зображення екрану.

1.3.4 Вимоги до технічних засобів, операційної системи і мови програмування

Програма, що розробляється у вигляді Desktop-застосунку має бути написана на мові C# у середовищі розробки Visual Studio. Програма призначена для роботи на ПК під керуванням ОС Windows, починаючи з версії Windows 7.

Для забезпечення нормальної роботи ОС Windows 7 мають бути виконані наступні системні вимоги [6]:

- 32-розрядний (x86) або 64-розрядний (x64) процесор із тактовою частотою 1 ГГц або швидший;
- 1 гігабайт RAM (для 32-розрядної версії) або 2 ГБ (для 64-розрядної версії);
- 16 ГБ (для 32-розрядної версії) або 20 ГБ (для 64-розрядної версії) вільного місця на жорсткому диску;
- графічний пристрій із підтримкою DirectX 9 і драйвером WDDM 1.0 або новішим.

Для забезпечення нормального функціонування програмного застосунку необхідні також наступні засоби:

- 20 Мб вільного місця на жорсткому диску для встановлення програми;
- маніпулятори вводу (клавіатура, миша);
- підключення до мережі Інтернет.

1.3.5 Вимоги до інформаційної та програмної сумісності

Програмне забезпечення, що розроблюється має підтримувати спільний режим роботи з сімейством ОС Windows. Програма повинна бути підготовлена до можливої модернізації.

1.3.6 Вимоги до програмної документації

Програмна документація має включати: керівництво програміста, керівництво оператора, опис програми та тексти основних програмних модулів.

1.3.7 Порядок приймання і тестування

Тестування даного ПЗ проводиться у вигляді виконання усіх можливих сценаріїв роботи оператора.

Дане ПЗ вважається прийнятним, якщо воно задовольняє усім висунутим вимогам до нього і успішно пройшло етап тестування.

2 РОЗРОБКА СТРУКТУРИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Аналіз технічного завдання

Згідно з викладеними у технічному завданні вимогами до програмного забезпечення, що розроблюється програмний продукт має надавати користувачеві можливість роботи з зображеннями. Переважна кількість сучасних мов програмування загального застосування (такі, наприклад, як C, C++, C#, Java, Python, Ruby і т. д.) мають вже реалізовані бібліотеки обробки графічних файлів, функції та методи редагування графіки і таке інше. Більш детально питання реалізації роботи з зображеннями буде розглянуто на етапі проектування під час вибору технічних і програмних засобів реалізації (зокрема, мови програмування, бібліотек і фреймворків, див. п. 2.3).

Хоча, як вже було зазначено, більшість мов програмування надають стандартне рішення для роботи з зображеннями, найчастіше вони не мають готових функцій для розмиття зображення і їх реалізація покладається на плечі програміста. Проводячи аналіз вже існуючих алгоритмів розмиття зображення, вибір було зроблено на користь простого у реалізації алгоритму, працюючого зі згорткою зображення [7], який можна описати наступним чином за допомогою наступного алгоритму:

```
for image row in input image:
    for pixel in image row:
        for iteration in range blur radius:
            RGB value sum = RGB value sum + pixel's RGB value
            RGB value average = RGB value sum / pixels count
            pixel RGB value = RGB value average
```

Під згорткою зображення мається на увазі робота з матрицями 3x3, де посередині знаходиться поточний піксель, редагування якого проводиться у даний момент, а навколо – пікселі-сусіди. Значення поточного пікселя залежить від значень навколишніх. Як видно з описаного вище алгоритму,

параметри червоного, зеленого і синього кольорів центрального пікселя вираховуються як середнє значення відповідних параметрів навколишніх пікселів.

Як ще також було зазначено у технічному завданні, програмне забезпечення, що розроблюється має надавати можливість користувачу завантажувати зображення на сервері. Рішення поставленої задачі можна реалізувати двома способами:

- скористатись вже готовим програмним рішенням, яке надає програмний інтерфейс (API) для завантаження зображень на сервер [8];
- реалізувати власне API.

Використання вже готового рішення в даному випадку унеможливить можливу модифікацію API, а також може призвести до браку конфіденційності користувацьких даних, приватності завантажених ними зображень.

Завдання передачі зображень у мережі інтернет можливо було би реалізувати з використанням FPT (File Transport Protocol) [9], але з використанням даного протоколу передачі даних неможливо буде відправляти додаткову інформацію про саме зображення (що, наприклад, може стати в нагоді у разі необхідності модифікації програми). Тому доцільним є використання HTTP (Hypertext Transport Protocol) [10], використовуючі різні методи звернень (наприклад, GET, POST, PUT, DELETE та інші). При використанні такого підходу, файли зображень будуть надсилатись у двійковому вигляді на сервер, де вже потім будуть оброблюватись і зберігатись у сховищі.

2.2 Визначення компонентів системи

Як вже було зазначено раніше, програмне забезпечення, що розроблюється складається з двох окремих модулів: Desktop-застосунку та веб-застосунку, які, працюючи у взаємодії, задовольняють висунутим у ТЗ

вимогам. Проектуючи структуру ПЗ, можна розділити виділені модулі на ще менші частки – компоненти. Перелік модулів та компонентів, що проектуються наступний:

а) desktop-застосунок, тобто точка доступу до програми, що надає доступ користувачеві до розроблених інтерфейсів програмного забезпечення. Застосунок, у свою чергу, складається з наступних компонентів:

- 1) програма захоплення зображення, яка реалізує функціональність для захоплення зображення з екрану цілком або виділення певної його області;
- 2) програма-редактор, яка реалізує функціональність для редагування вхідного зображення, надаючи користувачеві інтерфейс для роботи з ним (нанесення ліній, простих геометричних фігур, тексту, розмиття і інші можливості редагування, описані у технічному завданні);

б) веб застосунок, тобто програма, що розміщується на сервері та надає користувачеві інтерфейс для завантаження зображень на сервер. Веб застосунок включає у себе:

- 1) безпосередньо сам застосунок, який являє собою програмний інтерфейс (API) для взаємодії з користувачем;
- 2) сховище, для збереження зображень на сервері;
- 3) база даних, у якій будуть зберігатись дані про завантажені зображення, їх місцезнаходження у сховищі.

Нижче, на рис. 2.1, наведено модель системи, що проектується, подану у вигляді діаграми компонентів.



Рисунок 2.1 – Діаграма компонентів

Подібна структура програмного забезпечення підвищить модульність коду, можливість його незалежного тестування та модифікації (нічого не змінюючи у інших компонентах), а також надасть більше можливостей для повторного використання коду.

Все з тією ж метою підвищення модульності коду було прийняте рішення використання шаблонної архітектури застосунків, зокрема MVC [11] (Model-View-Controller). Програмне забезпечення, що розроблюється умовно розбивається на окремі частини, що розроблюються незалежно одна від одної. Зокрема, цими частинами є:

- модель (Model) – що включає в себе логіку роботи програми, всю послідовність обробки даних, яку вона реалізує;
- представлення (View) – візуалізація, інтерфейс роботи користувача з програмою;
- контролер (Controller - проміжна ланка між моделлю і представленням, яка відповідаючи на запити користувача у інтерфейсній частині програми викликає функції і методи моделі.

Використання подібної архітектури програмного забезпечення задовольняє вимогам, описаним вище, щодо його модульності, можливості повторного використання коду.

Графічне представлення MVC-архітектури наведено на рис. 2.2.

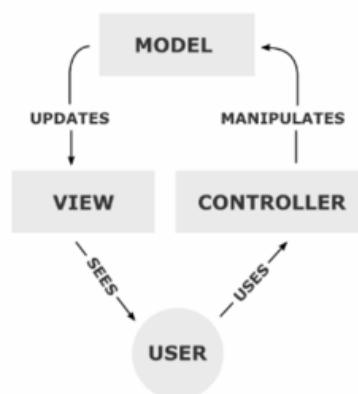


Рисунок 2.2 – MVC-архітектура

2.3 Обґрунтування вибору технічних і програмних засобів

Оскільки програмний продукт, що розроблюється є невеликого розміру та не потребує роботи одразу декількох спеціалістів або обчислювальних машин, основними апаратними засобами реалізації проекту було обрано наявний персональний комп'ютер у повній комплектації.

Переходячи до вибору операційної системи, необхідно звернутись до статистики. Згідно з дослідженнями NetMarketShare [12] (рис. 2.3) ОС Windows 10 та Windows 7 з початку 2022 року займають більш ніж 87 % ринку, а тому програмний продукт, що розроблюється першочергово повинно задовольняти потреби користувачів саме цієї операційної системи.

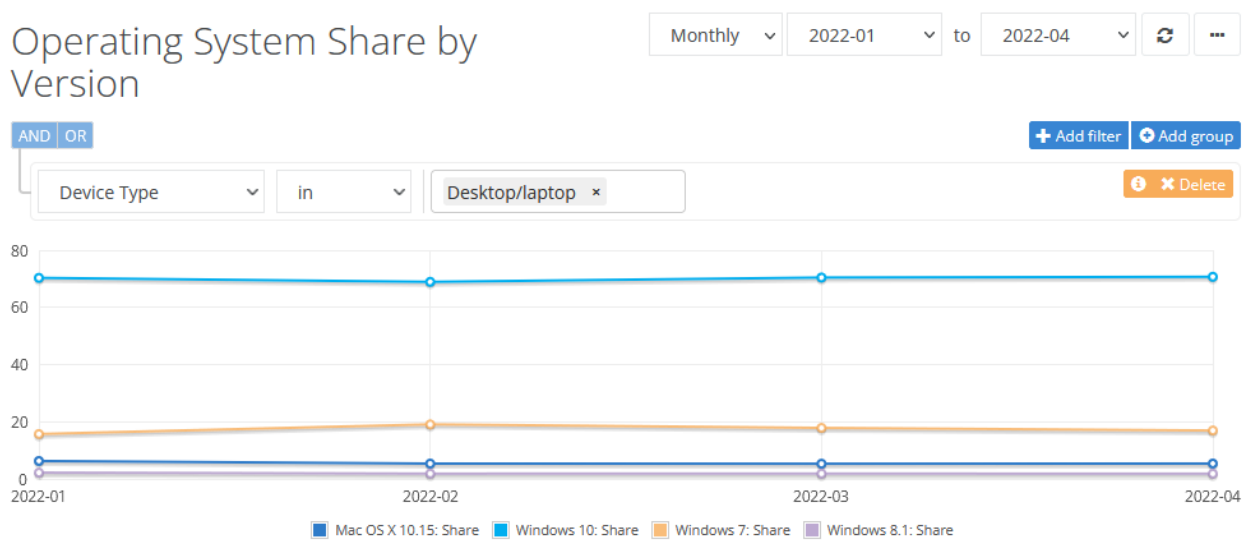


Рисунок 2.3 – Рейтинг операційних систем за популярністю

Щодо вибору мови програмування необхідно враховувати наступні чинники:

- наявність готових програмних рішень (бібліотеки і фреймворки, що дають змогу реалізувати необхідну функціональність);
- надійність (можливість мови програмування уберегти розробника від прийняття невдалих рішень);

– досвід розробника (вивчення нових бібліотек, фреймворків або навіть мов програмування потребує значних затрат у часі і шкідливо впливає на якість розробленого продукту).

Враховуючи усе вищезазначене, у якості мови програмування для реалізації Desktop-застосунку обрано мову C#. Її авторами є компанія Microsoft, яка також розробила ОС Windows, а тому має багато готових рішень для роботи саме з цією операційною системою.

Програми, написані з використанням C#, мають власний «збірник сміття» [13], що забезпечує постійне очищення невикористовуваної оперативної пам'яті. Дана мова програмування має насичений синтаксис, що в певній мірі уберігає програміста від небажаних ефектів (наприклад, невмисного перевантаження функції, тощо).

Графічний інтерфейс застосунку буде виконано з використанням Windows Forms [14], який на відміну від найближчого аналога WPF (Windows Presentation Foundation) є менш перевантаженим та надає достатній функціонал для реалізації поставлених задач.

Робота з зображеннями буде реалізована за допомогою стандартної бібліотеки платформи .NET – Drawing, що надає можливість редагування та збереження зображень у форматі бітової карти (Bitmap). Це особливо важливо для реалізації алгоритму розмиття, визначеного у п. 2.1.

Хоча C# і платформа .NET має можливості написання серверної частини веб застосунків, вони є негнучкими і відносно непопулярними у використанні, тому для реалізації серверу, куди буде завантажуватись готові зображення (знімки екрану) було прийняте рішення використовувати іншу мову програмування. Виходячи з власного досвіду розробника, вибір підпадає на Python, який завдяки своєму високому рівню абстракції та лаконічності синтаксису дає можливість досить швидко вирішувати поставлені задачі.

Серверна частина буде написана з використанням фреймворку Django [15], що має вбудовану функціональність для завантаження і збереження

зображень на сервері (крім того, він є найпопулярнішим серед Python-фреймворків для розробки веб застосунків). Ще однією з перевагою Django є наявність готових рішень для роботи з зображеннями (зокрема, їх зберігання на сервері), що є ключовим аспектом для реалізації поставлених задач. Для роботи із зображеннями Django використовує бібліотеку PIL (Python Imaging Library) [16].

2.4 Розробка сценарію взаємодії користувача з програмою

Визначення послідовності дій користувача має велике значення під час проектування графічного інтерфейсу програмного забезпечення. Це дає змогу визначити моменти інтересу користувача до тієї чи іншої активності та наділити його лише потрібною функціональністю у поточний момент часу, не перевантажуючи інтерфейс. З цією метою було розроблено функціональну блок-схему алгоритму роботи програми, зображену на рис. 2.4.

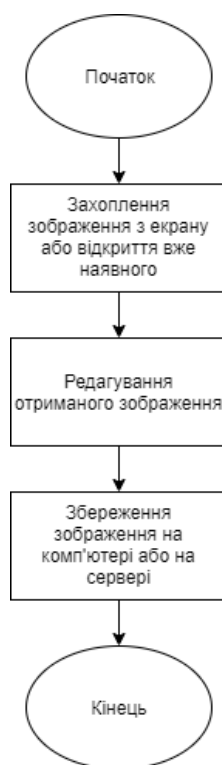


Рисунок 2.4 – Блок-схема алгоритму роботи програми

З блок-схеми алгоритму роботи програми можна побачити, що робота користувача з програмним забезпеченням поділена на три етапи. Кожен з цих етапів повинен бути представлений як найменше однією формою (вікном, сторінкою, тощо).

Для більш детального аналізу активностей користувача була розроблена діаграма прецедентів (рис. 2.5), що описує взаємодію користувача з програмним забезпеченням без прив'язки до часу. З цієї діаграми можна побачити зв'язаність окремих функцій системи, що також становить великий інтерес при проектуванні програмного забезпечення, наприклад, для вдалого вибору і розміщення елементів керування на формах.

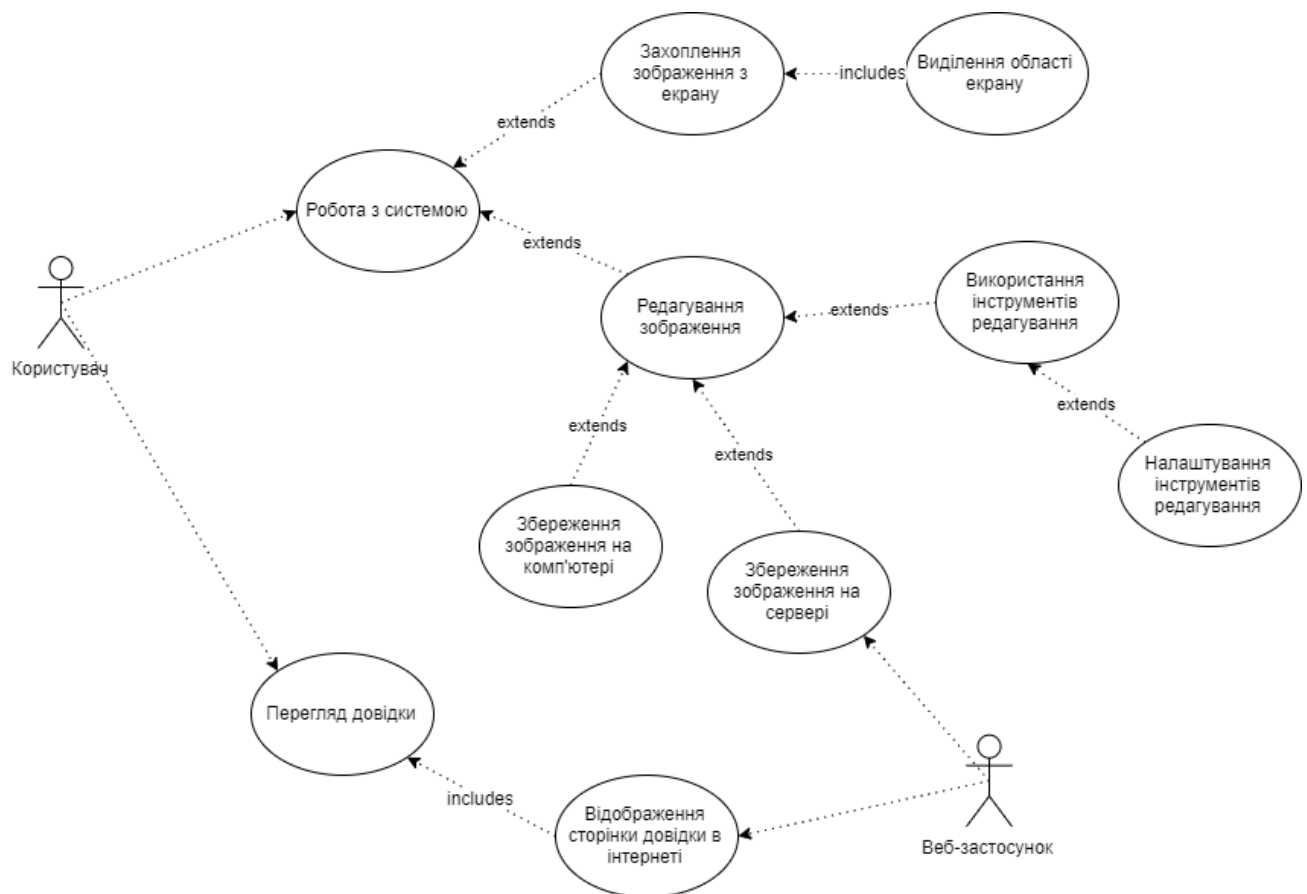


Рисунок 2.5 – Розроблена діаграма прецедентів

З діаграми прецедентів видно, що захоплення зображення з екрану і редагування зображення є рівнозначними діями користувача, не пов'язаними між собою. Це пояснюється тим, що розроблюване програмне забезпечення

може використовуватись користувачами у якості примітивного растрового редактора графіки (як це відбувається з іншими аналогами, розглянутими раніше). Користувач також не має необхідності саме щось змінювати в початковому зображенні під час редагування. Він має змогу без змін завантажити зображення на сервері з метою його поширення в мережі Інтернет.

З рис. 2.5 також видно рішення щодо розміщення довідки на сервері у якості веб сторінки. Це дасть змогу користувачам, які ще не завантажили Desktop-застосунок ознайомитись з його можливостями.

3 ОСНОВНІ РІШЕННЯ ЩОДО РЕАЛІЗАЦІЇ КОМПОНЕНТІВ СИСТЕМИ

3.1 Розробка Desktop-застосунку

3.1.1 Розробка класів Desktop-застосунку

Оскільки обрана мова програмування C# є виключно об'єктно-орієнтованою, то все функціонування програмного забезпечення виражається у вигляді взаємодії окремих класів та сутностей предметної області.

Згідно розробленої архітектури програмного забезпечення, застосунок що розроблюється поділяється умовно на модель, контролер та представлення. Модель, як було зазначено, реалізує виконання бізнес-логіки застосунку. Перелік розроблених класів моделі наступний:

- клас Tool є абстрактним, представляє собою загальний випадок інструменту редагування;
- клас PencilTool реалізує функціональність інструменту «Олівець» (наносить криві лінії на зображення);
- клас LineTool реалізує функціональність інструменту «Пряма лінія»;
- клас RectangleTool реалізує функціональність інструменту «Прямокутник»;
- клас EllipseTool реалізує функціональність інструменту «Еліпс»;
- клас BlurTool реалізує функціональність інструменту «Розмиття» (розмиває виділену область з визначеним);
- клас TextTool реалізує функціональність інструменту «Текст» (наносить визначений текст на зображення);
- клас BitmapCloner призначений для клонування об'єктів класу Bitmap (один із класів для роботи з зображеннями у стандартній бібліотеці Drawing);

– клас `Requester` призначений для відправки POST HTTP-запитів до серверу.

Для визначених класів було розроблено діаграму класів, зображену на рис. 3.1.

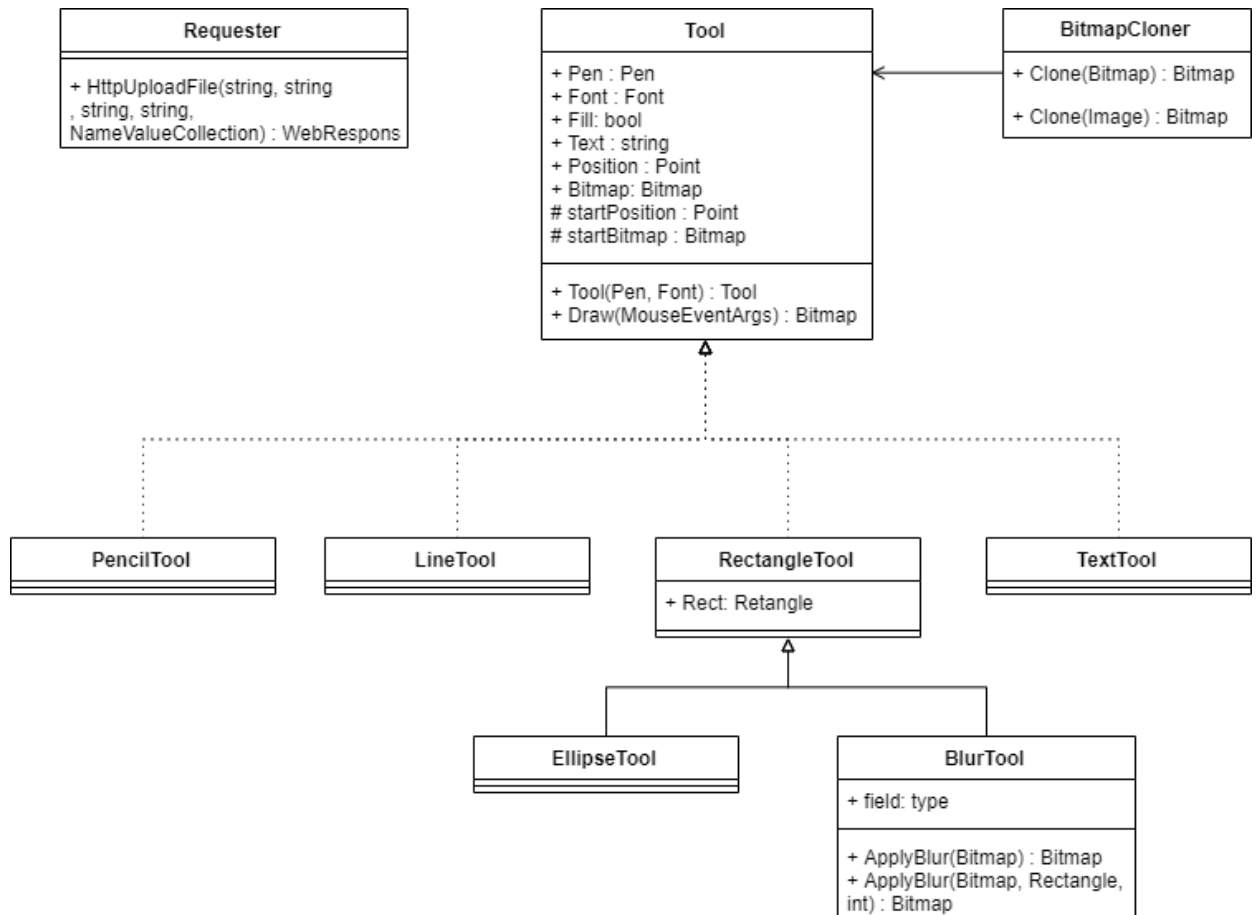


Рисунок 3.1 – Діаграма класів моделі

Розроблені класи контролера, переважно, являють собою класи-нащадки `Form` (стандартний клас `Windows Forms`). Серед класів-контролерів, зокрема, було реалізовано:

- клас `ScreenshoterMainForm`, що обробляє події головної форми застосунку;
- клас `ScreenshoterCaptureForm`, що обробляє події форми захоплення зображення;

- клас `ScreenshoterEditForm`, що обробляє події форми редагування зображення;
- клас `ScreenshoterUploadResultForm`, що обробляє події представлення готового посилання на зображення, завантажене на сервер;
- клас `ScreenshoterTextSetForm`, що обробляє події форми введення тексту для `TextTool`;
- клас `ScreenshotArea`, що реалізує виділення області екрану для захоплення зображення.

Розроблена діаграма вищезазначених класів зображена на рис. 3.2.

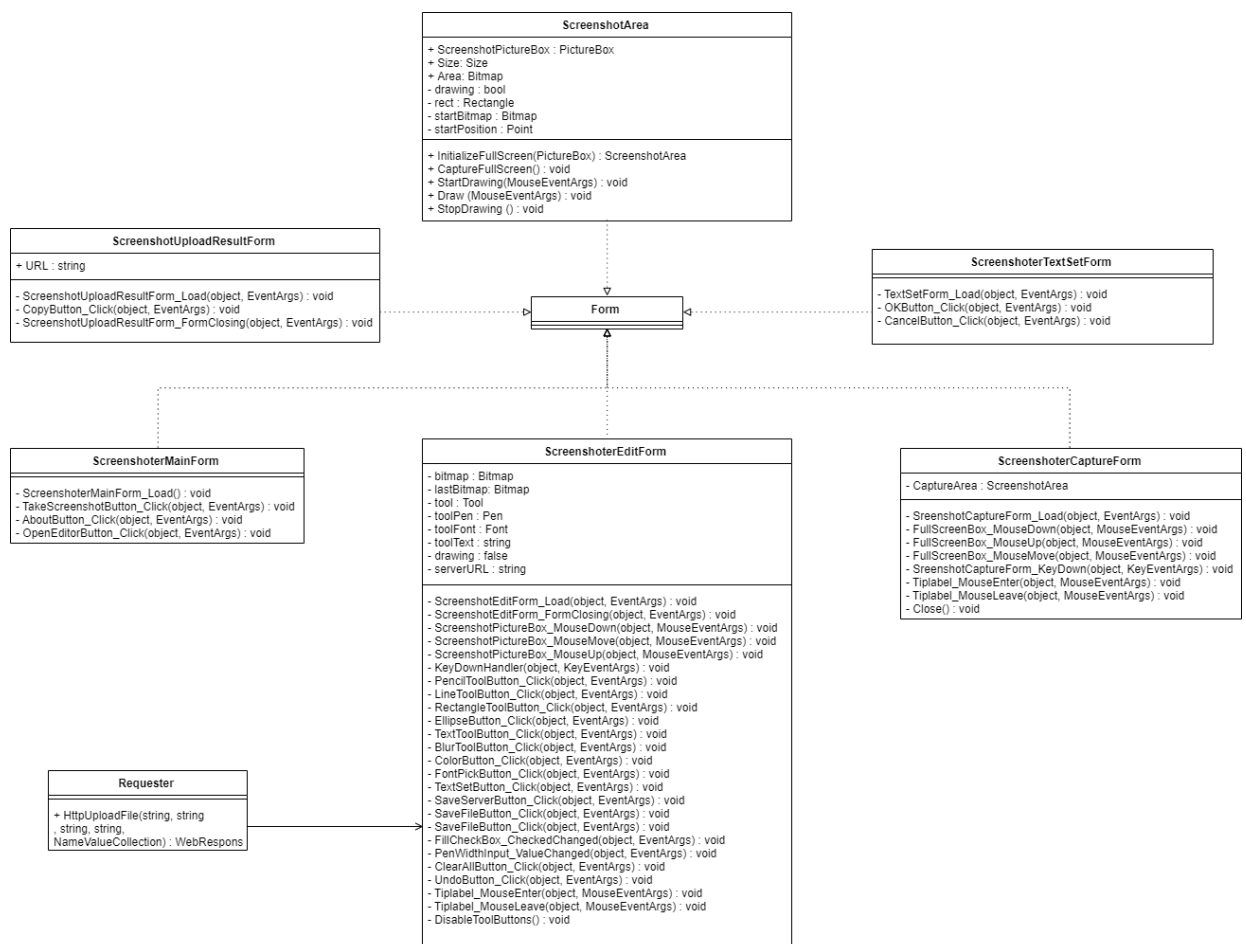


Рисунок 3.2 – Діаграма класів контролеру

3.1.2 Розробка графічного інтерфейсу користувача

Як вже було зазначено на етапі проектування системи, застосунок що розроблюється повинен складатись щонайменше з трьох форм:

- початкової форми, що надає доступ до інших компонентів системи;
- форми захоплення, що надає змогу користувачеві виділити необхідну область екрану для захоплення зображення;
- форми редагування зображення.

Вибір елементів керування проводиться відповідно до активностей користувача на формі. Таким чином, на початковій формі необхідно розмістити три кнопки, натискаючи на котрі користувач зможе здійснювати навігацію по іншим формам. Також, оскільки головна форма є «точкою доступу» до програми, її своєрідним лицем, на неї було додано коротку інформацію про програму, що явно вказує на її призначення. Розроблений дизайн форми представлено на рис. 3.3.

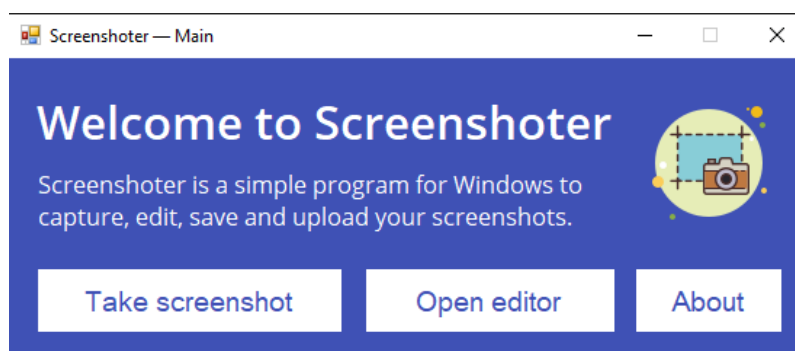


Рисунок 3.3 – Головна форма Desktop-застосунку

Для форми захоплення зображення з екрану було прийняте рішення не використовувати жодні елементи керування. Для навігації користувача та підказки про необхідні дії для досягнення поставленої задачі була залишена текстова підказка зверху екрану (при наведенні підказка зникає для того, щоб не закривати користувачеві зображення під нею. За замовчуванням підказка

має білий колір фону для підвищення контрастності, формуючи таким чином точку фокусу взору користувача). Форма захоплення екрану наведена на рис. 3.4.

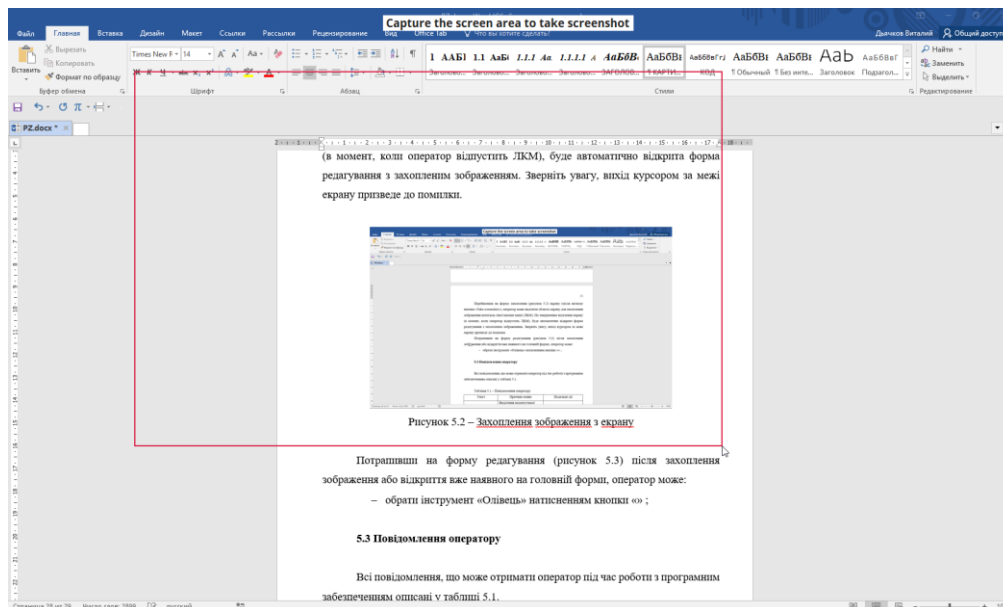


Рисунок 3.4 – Форма захоплення зображення з екрану

При проектуванні форми редагування зображення були використані наступні елементи керування:

- кнопки (для виконання певних операцій, наприклад, збереження зображення, вибору певного інструменту редагування або параметрів, зокрема. кольору, шрифту, тексту);
- поля для вводу (встановлення параметрів інструментів редагування, наприклад, товщини ліній/радіусу розмиття);
- кнопка-флажок (встановлення параметру заповнення фігури, управління логічними змінними).

Відповідно до загальних правил побудови ергономічних [17] інтерфейсів були виконані наступні кроки:

- для позначення поточного значення кольору кнопка вибору зафарбована з його використанням;

- при виборі інструменту «Текст» автоматично відкривається вікно для введення тексту;
- елементи керування зі схожою функціональністю згруповані між собою (наприклад, кнопки вибору інструментів редагування знаходяться поруч);
- панель інструментів знаходиться зліва (оскільки перегляд форми користувачем відбувається зліва-направо, зверху-вниз);
- елементи керування на панелі інструментів зліва розташовані по вертикалі по мірі їх необхідності користувачеві (кнопки збереження знаходяться одразу зверху, оскільки користувач може використовувати програму для збереження зображень області екрану на комп'ютері або на сервері для поширення в мережі інтернет).

Форма редагування наведена на рис. 3.5.

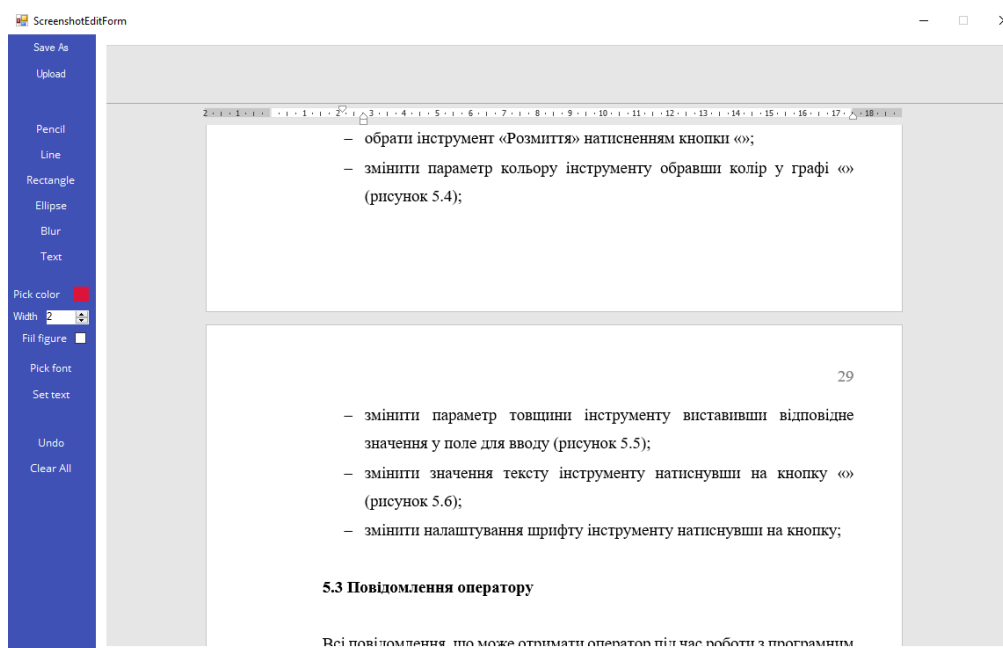


Рисунок 3.5 – Форма редагування

Як вже було зазначено, окрім трьох основних форм були також розроблені дві додаткові, виконані у вигляді діалогових вікон. Це, наприклад, форма введення тексту для інструменту «Текст» (рис. 3.6), а

також форма з результатами завантаження зображення (рис. 3.7), куди виводиться посилання на його місцезнаходження на сервері.

Кнопки, що завершують роботу користувача з вищезазначеними формами розташовані у правому нижньому кутку, куди взір оператора потрапляє в останню чергу після перегляду усієї форми.

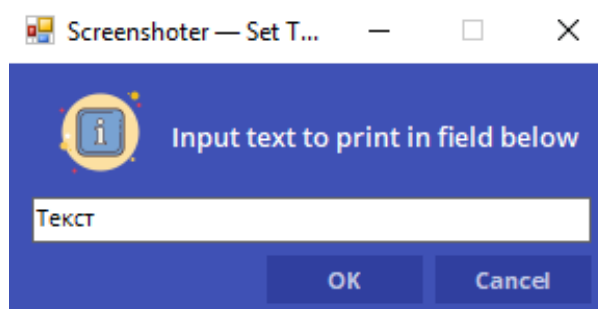


Рисунок 3.6 – Вікно введення тексту

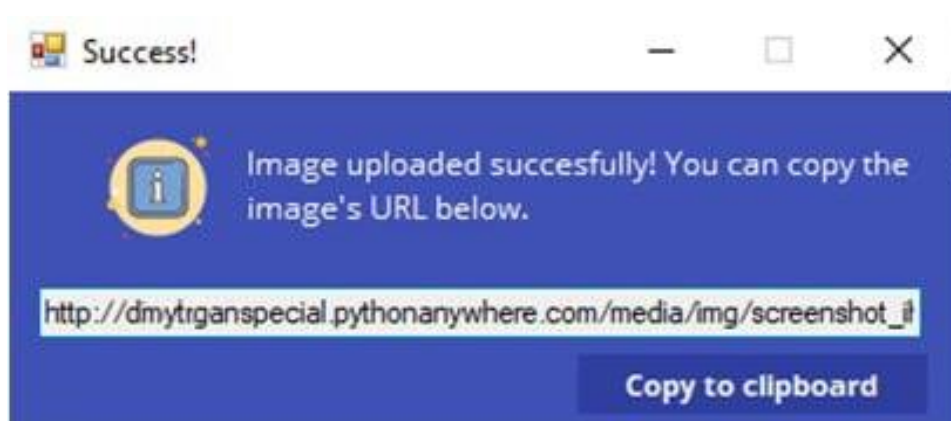


Рисунок 3.7 – Вікно результатів завантаження

Усі форми виконані з використанням одного набору кольорів, подібних шрифтів та повторюваних геометричних форм елементів керування. Дизайн виконано у теплих синіх тонах, що створюють відчуття комфорту та не напруженості інтерфейсу. В якості основного шрифту було обрано Open Sans з закругленими краями, що чудово поєднується з теплими тонами кольорів. Крім того даний шрифт є досить популярним (зокрема, на сервісі Google Fonts, який доволі часто використовується у веб-дизайні).

Зображення, що були використані, підібрані в одному стилі з сервісу icons8 у кольорах, що є контрастними до фонового.

3.1.3 Особливості реалізації Desktop-застосунку

Як вже було зазначено раніше, розроблюваний за допомогою фреймворку Windows Forms застосунок обов'язково складається з представлення (форми) та контролера (оброблювачів подій форми, наприклад, натиснення певної кнопки).

Усередині оброблювачів подій використовуються об'єкти описаних класів моделі, що реалізують логіку роботи програми, її послідовність обробки даних. Таким чином побудована взаємодія незалежних модулів програми, що обмінюються повідомленнями між собою.

Також було розроблено такі класи, як Requester та BitmapCloner, необхідні для дотримання DRY-принципу програмування (Don't Repeat Yourself), що покращує можливості повторного використання коду та ліквідує його надлишковість.

3.2 Розробка веб застосунку

3.2.1 Розробка класів веб застосунку

Розгортання Django-проекту відбувається відповідно до офіційної документації з використанням консольних команд. Django автоматично створює всю необхідну структуру проекту відповідно до MVC-архітектури (в контексті Django дана архітектура має назву MVT (Model-View-Template), залишаючи окремі .ру-файли (файли первинного коду) з контролерами, формами і моделями для доповнення розробником.

Реалізація моделі, тобто роботи програмного забезпечення з даними, виконується з використанням Django ORM (Object-relation mapping), яке забезпечує взаємодію веб-застосунку з різними сховищами даних (зокрема,

базами даних та популярними на поточний момент системами керування базами даних (СКБД), такими як, наприклад, SQLite, PostgreSQL, MySQL, SQL Server, MongoDB та інші).

При створенні класу-нащадку стандартного класу `Model`, Django автоматично створює нову таблицю у базі даних. Для роботи з зображеннями було створено клас `Image`, що успадковує клас `Model`, що має наступні атрибути:

- `id` – унікальний ідентифікатор зображення у базі даних (згенероване випадкове значення з використанням секретного ключа веб застосунку Django, а також поточних дати та часу);
- `image` – поле, що зберігає інформацію про розташування зображення на сервері (відносний шлях до нього).

Як вже було зазначено раніше, Django ORM надає можливості збереження і ведення обліку зображень у сховищі на сервері, що стало одним із ключових факторів у виборі даного фреймворку.

Для обробки надісланих користувачем зображень було створено нащадок стандартного класу `ModelForm` – `ImageUploadForm`, що проводить валідацію отримуваних даних та автоматично створює об'єкти встановленої в налаштуваннях класу моделі.

Взаємодія користувача і веб застосунку в Django виконується шляхом реалізації відповідних методів у нащадках класу `View`. З цією метою було створено клас `ImageUploadView`, що оброблює POST-запити до серверу за відносним посиланням `/server/upload/`.

Розроблена діаграма класів для веб-застосунку наведена на рис. 3.8.

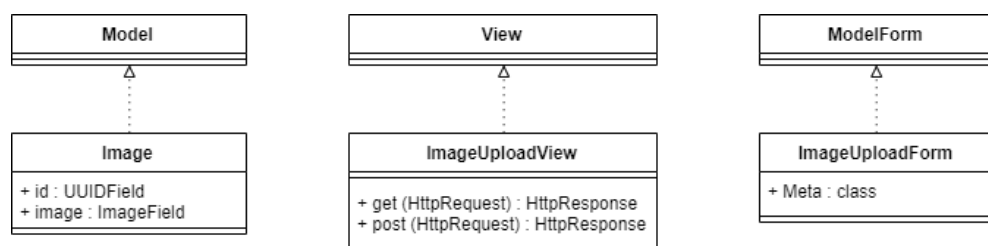


Рисунок 3.8 – Діаграма класів для веб застосунку

3.2.2 Розробка графічного інтерфейсу веб застосунку

Як було визначено на етапі проектування, веб застосунок має реалізовувати статичну сторінку довідки застосунку, доступну для перегляду кожному бажаючому користувачу.

Сторінка доступна за відносним посиланням `/server/about/`. Для обробки GET-запиту до застосунку за цим посиланням була реалізована функція `about`, що приймає у якості параметру запит користувача і повертає у результаті своєї роботи веб сторінку з довідкою.

Враховуючи передбачення користувача від використання довідки, вона має наступну структуру, подібну до довідки у інших застосунках:

- заголовок, з вказанням, що це за веб сторінка;
- навігація по довідці з гіперпосиланнями на окремі розділи;
- безпосередньо самі розділи.

Дизайн сторінки довідки виконаний за тими ж принципами, що і дизайн Dekstop-застосунку, з використанням тих же самих кольорів і шрифтів. Сторінка є адаптивною, тобто пристосованою для перегляду з різних пристроїв (ПК, ноутбуків, планшетів, телефонів тощо).

Розроблена сторінка довідки наведена на рис. 3.9.

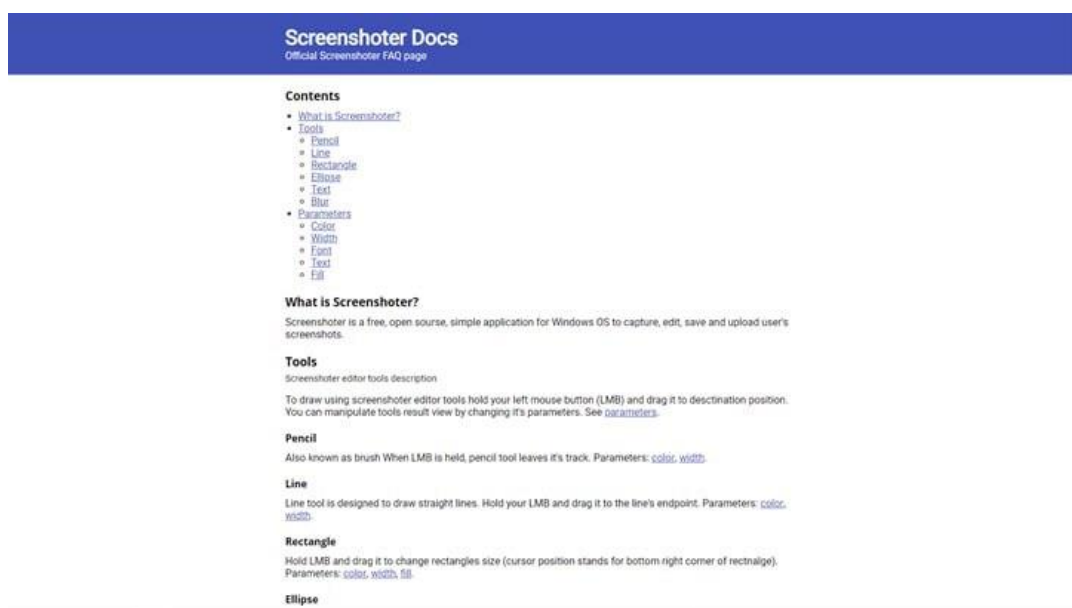


Рисунок 3.9 – Сторінка довідки

4 КЕРІВНИЦТВО ПРОГРАМІСТА

4.1 Призначення та умови застосування програми

Дане програмне забезпечення призначене для захоплення зображення з екрану, редагування отриманого зображення та його збереження на сервері або у пам'яті комп'ютера.

Для коректного функціонування Desktop-застосунку необхідна версія ОС Windows 7 та вище, а також наявне підключення до мережі Інтернет (для відображення сторінки довідки та завантаження зображення на сервер).

4.2 Характеристики програми

Програмне забезпечення відловлює можливі помилки в роботі оператора, запобігаючи від некоректного вводу вхідних даних. Наприклад, при введенні некоректного значення параметру товщини інструменту редагування або під час виділення некоректної області екрану для захоплення зображення (вихід за границі екрану, тощо). Усі повідомлення, що може отримати оператор під час виключних ситуацій описані у пункті 4.4.

Для запуску програмного забезпечення необхідно виконати файл Screenshoter.exe. Програма не потребує та не підтримує виклик з додатковими аргументами з командного рядка.

4.3 Вхідні та вихідні дані

До вхідних даних програми належать область екрану для захоплення зображення або наявне зображення у пам'яті комп'ютера, позиція курсору під час роботи з інструментами редагування, для позначення початкових та кінцевих точок фігур, параметри інструментів редагування (колір, товщина, шрифт, текст, радіус розмиття тощо).

Вихідними параметрами програми виступають збережене у пам'яті комп'ютера зображення або посилання на зображення, що збережене на сервері.

4.4 Опис атрибутів та методів

Перелік усіх класів програми було наведено у пунктах 3.1 та 3.2. Перелік атрибутів та методів кожного з класів наведено у таблиці 4.1.

Таблиця 4.1 – Перелік атрибутів та методів

Клас	Атрибут / Метод	Тип даних	Короткий опис
1	2	3	4
Tool	Pen	Pen	Інформацію про колір та товщину поточного інструменту
	Font	Font	Інформація про шрифт інструменту (для TextTool)
	startPosition	Point	Інформація про початкову точку рисунку
	startBitmap	Bitmap	Інформація про початкове зображення
	Fill	Bool	Логічна змінна, чи потрібно зафарбовувати фігуру всередині
	Text	String	Інформація про текст для рисунку
RectangleTool	Rect	Rectangle	Прямокутник, який необхідно намалювати

Продовження таблиці 4.1

1	2	3	4
Tool	Position	Point	Доступ до startPosition
	Bitmap	Bitmap	Поточний рисунок
	Draw (MouseEventArgs)	Bitmap	Метод, що розміщує відповідну фігуру / рисунок на зображенні
BlurTool	ApplyBlur(Bitmap)	Bitmap	Метод, що виконує розмиття на зображенні
	ApplyBlur(Bitmap, Rectangle, int)	Bitmap	Метод, що розмиває область на зображенні
Requester	HttpUploadFile(String, String, String, String, NameValueCollection)	WebResponse	Відправляє запит до серверу на завантаження зображення
BitmapCloner	Clone(Bitmap)	Bitmap	Метод, що повертає повну копію вхідного зображення
	Clone(Image)		Метод, що повертає повну копію вхідного зображення
ColorPalette	Primary	Color	Основний колір фону
	DarkPrimary		Темніший колір фону
	LightPrimary		Світліший колір фону
	Accedent		Колір для кнопок, означення інших активностей
	Text		Колір для основного тексту на темному фоні

Продовження таблиці 4.1

1	2	3	4
ColorPalette	PrimaryText		Колір для основного тексту на світлому фоні
	SecondaryText		Колір для другорядного тексту на світлому фоні
	Divider		Колір для розмежування
ScreenshotArea	Screenshot PictureBox	PictureBox	«Полотно» для захоплення зображення
	rect	Rectangle	Прямокутник для виділення області
	startBitmap	Bitmap	Зображення до виділення області
	startPosition	Point	Місцезнаходження курсору на початку виділення області
	drawing	bool	Логічна змінна, яка вказує на те, чи захоплюється на даний момент область
	Size	Size	Розмір «полотна»
	Area	Bitmap	Виділена область
	Initialize FullScreen (PictureBox)	Screenshot Area	Створити об'єкт класу для захоплення зображення з екрану цілком
	StartDrawing (MouseEventA rgs)	void	Метод, що викликається на початку виділення області
	Draw(MouseE vent Args)	Void	Метод, що викликається під час виділення області

У таблиці 4.1 наведені лише класи моделі та контролер ScreenshotArea. Усі інші класи, що успадковані від стандартного класу Form є реалізацією його нащадку, що оброблює події представлення (наприклад, відловлює натиснення певної кнопки). Назви методів-оброблювачів подій виглядає наступним чином: Назва об'єкта форми_Подія(об'єкт, аргументи події). Наприклад, ClearAllButton_Click(object sender, EventArgs e). Повертаємий тип – void. Зокрема, програмою обробляються події натискання кнопки (Click), введення нового значення у поле введення цифри (ValueChanged), відображення (Load) та закриття (FormClosing) форми. Перелік унікальних, визначених розробником атрибутів форми наведено у таблиці 4.2.

Таблиця 4.2 – Атрибути форм

Форма	Атрибут	Тип	Опис
1	2	3	4
Screenshoter CaptureForm	CaptureArea	ScreenshotArea	Контролер виділення області екрану
Screenshoter EditForm	tool	Tool	Поточний інструмент редагування
	bitmap	Bitmap	Початкове зображення
	lastBitmap	Bitmap	Останнє зафіксоване зображення після редагування
	toolPen	Pen	Колір та товщина поточного інструменту
	toolFont	Font	Шрифт поточного інструменту (для інструменту «Текст»)
	toolText	String	Текст поточного інструменту (для інструменту «Текст»)
	drawing	Bool	Чи використовує користувач інструмент
	serverURL	String	Посилання на веб-застосунок

Продовження таблиці 4.2

1	2	3	4
Screenshoter UploadResultForm	URL	string	Посилання на завантажене зображення до веб застосунку

Опис атрибутів та методів класів веб застосунку наведено у табл. 4.3.

Таблиця 4.3 – Атрибути та методи класів веб застосунку

Клас	Атрибут / Метод	Тип даних	Опис
Image	id	UUIDField	Унікальний ідентифікатор зображення
	image	ImageField	Дані про зображення
ImageUploadForm	Meta	type	Метадані про форму завантаження зображень
ImageUploadView	get (HttpRequest)	HttpResponse	Оброблювач GET-запиту до адреси завантаження зображення
	post (HttpRequest)	HttpResponse	Оброблювач POST-запиту до адреси завантаження зображення (потребує даних про зображення)

Модифікація веб застосунку відбувається відповідно до офіційної документації Django [18].

4.4 Повідомлення програмісту

Повідомлень програмісту у програмі не передбачено.

5 КЕРІВНИЦТВО ОПЕРАТОРА

5.1 Призначення та умови виконання програми

Дане програмне забезпечення призначене для захоплення зображення з екрану, редагування отриманого зображення та його збереження на сервері або у пам'яті комп'ютера.

Для коректного функціонування Desktop-застосунку необхідна версія ОС Windows 7 та вище, а також наявне підключення до мережі Інтернет (для відображення сторінки довідки та завантаження зображення на сервер).

5.2 Виконання програми

Для запуску програмного забезпечення необхідно виконати файл Screenshooter.exe. Програма не потребує та не підтримує виклик з додатковими аргументами з командного рядка.

Після запуску програми оператор потрапляє на її головну форму (рис. 3.3). Знаходячись на головній формі оператор може:

- почати захоплення екрану, натиснувши на кнопку «Take screenshot»;
- обрати вже існуюче зображення та перейти до редактора, натиснувши кнопку «Open editor»;
- перейти до довідки, натиснувши кнопку «About».

Перейшовши на форму захоплення (рис. 3.4) екрану (після натискання кнопки «Take screenshot»), оператор може виділити область екрану для захоплення зображення натиснувши (ЛКМ). По завершенню виділення екрану (в момент, коли оператор відпустить ЛКМ), буде автоматично відкрита форма редагування з захопленим зображенням. Зверніть увагу, вихід курсором за межі екрану призведе до помилки.

Потрапивши на форму редагування (рис. 3.5) після захоплення зображення або відкриття вже наявного на головній формі, оператор може:

- обрати інструмент «Олівець» шляхом натиснення кнопки «Pencil»;
- обрати інструмент «Пряма лінія» шляхом натиснення кнопки «Line»;
- обрати інструмент «Прямокутник» шляхом натиснення кнопки «Rectangle»;
- обрати інструмент «Еліпс» шляхом натиснення кнопки «Ellipse»;
- обрати інструмент «Розмиття» шляхом натиснення кнопки «Blur»;
- обрати інструмент «Текст» шляхом натиснення кнопки «Text»;
- змінити параметр кольору інструменту обравши колір у графі «Pick color» (рис. 5.1);
- змінити параметр товщини інструменту виставивши відповідне значення у поле для вводу (рис. 5.2);
- змінити значення тексту інструменту натиснувши на кнопку «Set text» (рис. 3.6);
- змінити налаштування шрифту інструменту натиснувши на кнопку «Pick font» (рис. 5.3);
- змінити параметр заповнення інструменту поставивши мітку у «Fill figure»;
- відмінити останні зміни натиснувши кнопку «Undo» або сполучення клавіш клавіатури Ctrl + Z;
- відмінити всі зміни від початку редагування натиснувши кнопку «Clear All» або сполучення клавіш клавіатури Ctrl + Alt + Z;
- зберегти зображення на комп'ютері натиснувши кнопку «Save As» та у відкритому діалоговому вікні обравши директорію для збереження;

- зберегти зображення на сервері, натиснувши кнопку «Upload» (рис. 3.7).

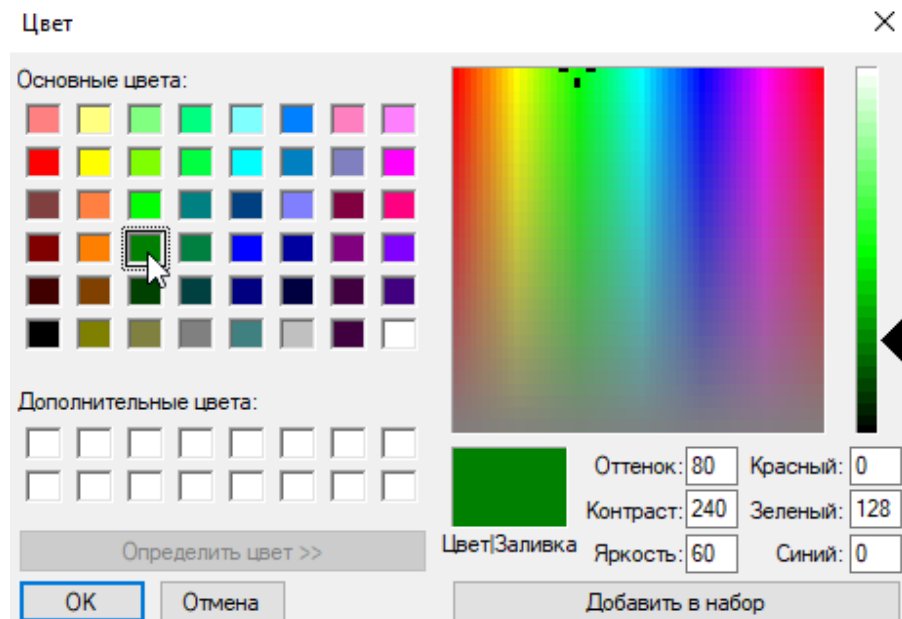


Рисунок 5.1 – Форма вибору кольору

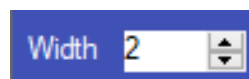


Рисунок 5.2 – Поле для ввода товщини інструменту

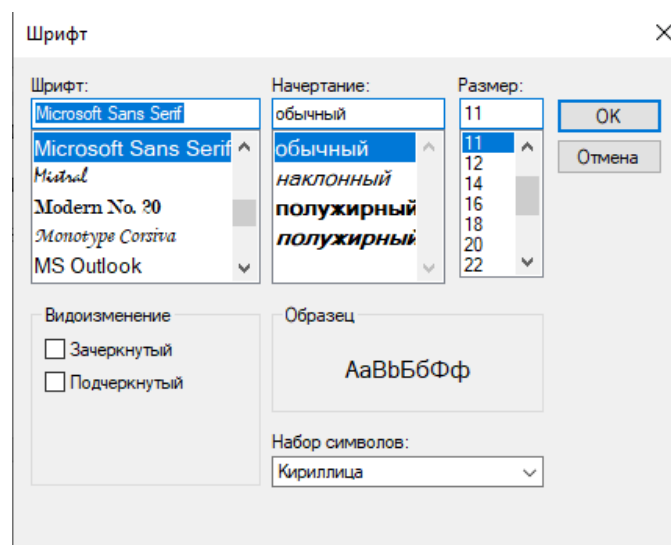


Рисунок 5.3 – Форма зміни шрифту

Параметри інструментів редагування наведені нижче:

- колір: відповідає кольору нанесення фігур/ліній/тексту на зображення;
- товщина: товщина нанесення ліній на зображення, радіус розмиття для відповідного інструменту;
- заповнення: логічна змінна, що відповідає за те, чи зафарбовувати усередині прямокутник та еліпс;
- текст: налаштування тексту для відображення на зображенні для інструменту «Текст»;
- шрифт: налаштування шрифту для відображення тексту на зображенні для інструменту «Текст».

Опинившись на формі результатів зображення (рис. 3.7), оператор може натиснути кнопку «Copy to clipboard» для того, щоб зберегти посилання на зображення у буфері обміну свого пристрою. Також можна вручну скопіювати посилання з поля для введення.

Опинившись на сторінці довідки (рис. 3.9) оператор може скористатись навігацією (рис. 5.4) для швидкого переходу по довідці шляхом гіперпосилань.

Contents

- [What is Screenshoter?](#)
- [Tools](#)
 - [Pencil](#)
 - [Line](#)
 - [Rectangle](#)
 - [Ellipse](#)
 - [Text](#)
 - [Blur](#)
- [Parameters](#)
 - [Color](#)
 - [Width](#)
 - [Font](#)
 - [Text](#)
 - [Fill](#)

Рисунок 5.4 – Навігація по довідці

5.3 Повідомлення оператору

Всі повідомлення, що може отримати оператор під час роботи з програмним забезпеченням описані у табл. 5.1.

Таблиця 5.1 – Повідомлення оператору

Текст	Причина появи	Подальші дії
Unacceptable area selection, please retry!	Виділення недопустимої області екрану (наприклад, вихід курсором за його межі)	Повторно виділити область екрану
Image successfully saved	Зображення успішно збережене у пам'яті комп'ютера	
File upload error	Невдала спроба завантажити зображення на сервер	Перевірити підключення до мережі інтернет, переконатися у коректності даних, що передаються для завантаження
Image uploaded successfully	Зображення успішно завантажене на сервер	
Invalid blur zone!	Виділення недопустимої області зображення для застосування розмиття	Повторно виділити область зображення

ВИСНОВКИ

За результатами виконання дипломної кваліфікаційної роботи бакалавра вирішено актуальну задачу збереження та редагування знімків екрану за допомогою методів стандартних бібліотек платформи .NET та розробки власного API.

Для вирішення поставлених задач, в процесі виконання роботи було проведено детальний аналіз предметної області, а також огляд вже існуючих програмних аналогів. Визначено, що у більшості аналогів спостерігаються функціональна перенасиченість, а також висока ступінь вкладеності інтерфейсу.

Відповідно до поставленої мети роботи було виконано розробку технічного завдання на застосунок, що розроблявся. Аналіз технічного завдання показав, що система, що проектується має бути реалізована у вигляді двох окремих модулів – Desktop-застосунку та веб застосунку. Desktop-застосунок реалізує функціональність для захоплення зображення з екрану та редагування вхідного зображення. Веб застосунок реалізовано у вигляді програми, що розміщується на сервері та надає користувачеві інтерфейс для завантаження зображень знімків екрану на сервер.

У відповідності з функціональними вимогами, зазначеними у технічному завданні, розроблено діаграму прецедентів, діаграму компонентів та блок-схему алгоритму роботи програми. Реалізовано основну структуру програмного забезпечення та розроблено необхідні програмні класи для Desktop-застосунку та веб застосунку. Для обох застосунків було спроектовано та розроблено графічний інтерфейс користувача. Виконано тестування на наборі тест-кейсів.

Практична цінність розробленого програмного забезпечення полягає в тому, що воно надає можливості покращити інформаційний обмін між користувачами. Програма також може використовуватися як примітивний

графічний редактор зображень. Знімки зображень екрану або відредаговані зображення зберігаються на сервері.

Програмна документація включає: керівництво програміста; керівництво оператора; опис програми та тексти основних програмних модулів.

Галузь використання розробленої системи – інформаційні технології.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Humphries, S. The Best Screenshot Software for Capturing Images and Video [Electronic resource]. – Access mode: <https://www.reviewgeek.com/58981/the-best-screenshot-software-for-capturing-images-and-video/>.
2. What is a screenshot? [Electronic resource]. – Access mode: <https://www.ionos.com/digitalguide/websites/web-design/what-is-a-screenshot/>.
3. Snipping Tool – Вікіпедія [Електронний ресурс] – Режим доступу: https://en.wikipedia.org/wiki/Snipping_Tool.
4. Lightshot [Electronic resource]. – Access mode: <https://app.prntscr.com>.
5. Joxi [Electronic resource]. – Access mode: <https://www.crunchbase.com/organization/joxi>.
6. Вимоги до системи для Windows 7 [Electronic resource]. – Access mode: <https://support.microsoft.com/uk-ua/windows/вимоги-до-системи-для-windows-7-df0900f2-35>.
7. Kernel (image processing) – Вікіпедія [Електронний ресурс] – Режим доступу: [https://en.wikipedia.org/wiki/Kernel_\(image_processing\)](https://en.wikipedia.org/wiki/Kernel_(image_processing)).
8. Application programming interface – Вікіпедія [Електронний ресурс] – Режим доступу: https://en.wikipedia.org/wiki/Application_programming_interface.
9. File Transport Protocol – Вікіпедія [Електронний ресурс] – Режим доступу: https://en.wikipedia.org/wiki/File_Transfer_Protocol.
10. Hypertext Transfer Protocol – Вікіпедія [Електронний ресурс] – Режим доступу: https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol.
11. Model-view-controller – Вікіпедія [Електронний ресурс] – Режим доступу: <https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller>.

12. Operating systems market share – NetMarketShare [Електронний ресурс] – Режим доступу: <https://netmarketshare.com/operating-system-market-share.aspx?options=%7B%22filter%22%3A%7B%22%24and%22%3A%5B%7B%22deviceType%22%3A%7B%22%24in%22%3A%5B%22Desktop%22Flaptop%22%5D%7D%7D%5D%7D%2C%22dateLabel%22%3A%22Custom%22%2C%22attributes%22%3A%22share%22%2C%22group%22%3A%22platformVersion%22%2C%22sort%22%3A%7B%22share%22%3A-1%7D%2C%22id%22%3A%22platformsDesktopVersions%22%2C%22dateInterval%22%3A%22Monthly%22%2C%22dateStart%22%3A%222022-%22%2C%22dateEnd%22%3A%222022-04%22%2C%22segments%22%3A%22-1000%22%7D>.

13. Garbage collector (computer science) – Вікіпедія [Електронний ресурс] – Режим доступу: [https://en.wikipedia.org/wiki/Garbage_collection_\(computer_science\)](https://en.wikipedia.org/wiki/Garbage_collection_(computer_science)).

14. Windows Forms – Вікіпедія [Електронний ресурс] – Режим доступу: https://en.wikipedia.org/wiki/Windows_Forms.

15. Django (Web framework) – Вікіпедія [Електронний ресурс] – Режим доступу: [https://en.wikipedia.org/wiki/Django_\(web_framework\)](https://en.wikipedia.org/wiki/Django_(web_framework)).

16. Python Imaging Library – Вікіпедія [Електронний ресурс] – Режим доступу: https://en.wikipedia.org/wiki/Python_Imaging_Library.

17. Human factors and ergonomics – Вікіпедія [Електронний ресурс] – Режим доступу: https://en.wikipedia.org/wiki/Human_factors_and_ergonomics.

18. Blur an image with c# [Electronic resource]. – Access mode: <https://gist.github.com/superic/8165746>.

ДОДАТОК А
Текст програми

A.1 Код Desktop-застосунку

A.1.1 Файл ScreenshoterMainForm.cs

```

using System.Drawing;
using System.Windows.Forms;

namespace Screenshoter
{
    public partial class ScreenshoterMainForm : Form
    {
        public ScreenshoterMainForm()
        {
            InitializeComponent();

            private void ScreenshoterMainForm_Load(object sender,
EventArgs e)
            {
                /* Top panel */
                this.TopPanel.BackColor = ColorPalette.Primary;
                this.Header.ForeColor = ColorPalette.Text;
                this.Header.Text = "Welcome to Screenshoter";
                this.Description.Text = "Screenshoter is a simple program
for Windows to\n" +
                    "capture, edit, save and upload your screenshots.";
                this.Description.ForeColor = ColorPalette.Text;

                /* Buttons */
                this.TakeScreenshotButton.BackColor = ColorPalette.Text;
                this.TakeScreenshotButton.ForeColor =
ColorPalette.Primary;
                this.OpenEditorButton.BackColor = ColorPalette.Text;
                this.OpenEditorButton.ForeColor = ColorPalette.Primary;
                this.AboutButton.BackColor = ColorPalette.Text;
                this.AboutButton.ForeColor = ColorPalette.Primary;
            }

            private void TakeScreenshotButton_Click(object sender,
EventArgs e)
            {
                var captureForm = new ScreenshotCaptureForm();
                this.Hide();
                System.Threading.Thread.Sleep(300);
                DialogResult result = captureForm.ShowDialog();
                if (result == DialogResult.OK)
                {
                    this.Show();
                }
                else

```

```

        {
            Application.Exit();
        }
    }
    private void AboutButton_Click(object sender, EventArgs e)
    {
        System.Diagnostics.Process.Start("http://
kovspecial.pythonanywhere.com/server/about/");
    }
    private void OpenEditorButton_Click(object sender, EventArgs
e)
    {
        OpenFileDialog dialog = new OpenFileDialog();
        dialog.Filter = "JPEG images(*.jpg)|*.jpg|Bitmap
images(*.bmp)|*.bmp|PNG images(*.png)|*.png|All files(*.*)|*.*";

        if (dialog.ShowDialog() == DialogResult.Cancel)
            return;
        Image image = Image.FromFile(dialog.FileName);
        ScreenshotEditForm form = new ScreenshotEditForm(image as
Bitmap);
        this.Hide();
        if (form.ShowDialog() != DialogResult.Cancel)
        {
            this.Show();
        }
    }
}

```

A.1.2 Файл ScreenshoterCaptureForm.cs

```

using System;
using System.Drawing;
using System.Windows.Forms;

namespace Screenshoter
{
    public partial class ScreenshotCaptureForm : Form
    {
        ScreenshotArea CaptureArea;

        public ScreenshotCaptureForm()
        {
            InitializeComponent();
        }
    }
}

```

```

        private void ScreenshotCaptureForm_Load(object sender,
EventArgs e)
        {
            this.WindowState = FormWindowState.Maximized;
            this.FormBorderStyle = FormBorderStyle.None;
            this.Bounds = Screen.GetBounds(this);
            this.CaptureArea =
ScreenshotArea.InitializeFullScreen(this.FullScreenBox);
            int width = (Screen.PrimaryScreen.Bounds.Width / 2) -
(this.Tiplabel.Size.Width / 2);
            int height = 20;
            this.Tiplabel.Location = new Point(width, height);
            this.Tiplabel.BackColor = ColorPalette.Text;
            this.Tiplabel.ForeColor = ColorPalette.PrimaryText;
        }

        private void FullScreenBox_MouseDown(object sender,
MouseEventArgs e)
        {
            this.CaptureArea.StartDrawing(e);
        }

        private void FullScreenBox_MouseUp(object sender,
MouseEventArgs e)
        {
            this.CaptureArea.StopDrawing();
            ScreenshotEditForm editForm;

            try
            {
                editForm = new
ScreenshotEditForm(this.CaptureArea.Area);
            }
            catch (OutOfMemoryException ex)
            {
                this.CaptureArea.Refresh();

                MessageBox.Show("Unacceptable area selection! Please,
retry", "Area selection error",
                    MessageBoxButtons.OK, MessageBoxIcon.Error);

                return;
            }

            this.Close(DialogResult.Cancel);
            this.Hide();
            DialogResult result = editForm.ShowDialog();
            this.Close(result);
        }

```

```

        private void FullScreenBox_MouseMove(object sender,
MouseEventArgs e)
        {
            this.CaptureArea.Draw(e);
        }

        private void ScreenshotCaptureForm_KeyDown(object sender,
KeyEventArgs e)
        {
            if (e.KeyCode == Keys.Escape)
            {
                this.Close(DialogResult.OK);
            }
        }

        public void Close(DialogResult result)
        {
            this.DialogResult = result;
            this.Close();
        }

        private void Tiplabel_MouseEnter(object sender, EventArgs e)
        {
            this.Tiplabel.Visible = false;
        }

        private void Tiplabel_MouseLeave(object sender, EventArgs e)
        {
            this.Tiplabel.Visible = true;
        }
    }
}

```

A.1.3 Файл ScreenshoterEditForm.cs

```

using System.Collections.Specialized;
using System.Drawing;
using System.IO;
using System.Net;
using System.Windows.Forms;

namespace Screenshoter
{
    // Edit form, provides user with methods, that
    // allows to draw on captured screenshot
    public partial class ScreenshotEditForm : Form
    {
        // Start state of bitmap, got from ScreenshotCaptureForm
        private Bitmap bitmap;
    }
}

```

```

// Last state of bitmap. Needed to perform undo operation
private Bitmap lastBitmap;

// Current tool, used by user
private Tool tool;

// Current tool's pen options
private Pen toolPen;

// Current tool's font (used by TextTool)
private Font toolFont;

private string toolText = "Text";

// Boolean variable. True state means that user
// currently draws a figure on edit area
private bool drawing = false;

private string serverURL =
"http://diachkovspecial.pythonanywhere.com/server/upload/";

// Main initializer. Requires an screenshot bitmap
public ScreenshotEditForm(Bitmap bm)
{
    InitializeComponent();

    this.bitmap = bm;

    // KeyDownHandler provides quickbinds to perform
    // routine operations, such as undo, etc
    this.KeyDown += this.KeyDownHandler;
}

// Onload functions, sets all default values
private void ScreenshotEditForm_Load(object sender, EventArgs
e)
{
    // Pencil (brush) tool is used by default
    this.tool = new PencilTool();

    this.BackColor = ColorPalette.Text;

    // When activated, buttons are highlighted by contrast
color
    this.PencilToolButton.BackColor = ColorPalette.Text;
    this.PencilToolButton.ForeColor = ColorPalette.Primary;

    // Margin between widgets
    int marginSize = 12;

```

```

way:        // The actual form height value is calculated the next
            // Form Height + Top Panel (contains minimize, maximize,
            // close buttons), etc.

            // When this.Height 0 is setted the actual form height
won't       // be a zero value. It gets the Top Panel value, that is
            // stored in this.Height variable at this moment
            this.Height = 0;

            this.ScreenshotPictureBox.Size = bitmap.Size;
            this.ScreenshotPictureBox.Image = bitmap;
            this.ScreenshotPictureBox.Location      =          new
Point(this.ToolPanel.Width + marginSize, marginSize);

            // Form height is calculated the next way:
            // TopPanel height + ToolPanel height + Margin between
screenshot // and ToolPanel + screenshot height + margin afret
screenshot // and ToolPanel + screenshot height + margin afret
            this.Height      =      Math.Max(this.Height      +
this.ToolPanel.Height,
            this.Height      +      (marginSize      *      2)      +
this.ScreenshotPictureBox.Height);
            this.ToolPanel.Height = this.Height;

            this.Width = this.ToolPanel.Width + (marginSize * 3) +
this.ScreenshotPictureBox.Width;

            this.lastBitmap = BitmapCloner.Clone(this.bitmap);

            // defatuls
            this.toolPen = new Pen(Color.Crimson, 2);

            this.toolFont = new Font(FontFamily.GenericSansSerif, 11);

            this.ColorButton.BackColor = this.toolPen.Color;

            // Customizing
            // Panel
            this.ToolPanel.BackColor = ColorPalette.Primary;

            // Buttons

            foreach(Control control in this.ToolPanel.Controls)
            {
                if (typeof(Button).IsInstanceOfType(control))
                {
                    control.BackColor = ColorPalette.Primary;
                    control.ForeColor = ColorPalette.Text;
                }
            }

```

```

        }
    }

    this.ColorButton.BackColor = this.toolPen.Color;
    this.ColorPickTipLabel.ForeColor = ColorPalette.Text;

    this.FillCheckBox.ForeColor = ColorPalette.Text;
}

private void ScreenshotEditForm_FormClosing(object sender,
FormClosingEventArgs e)
{
    this.DialogResult = DialogResult.OK;
}

private void ScreenshotPictureBox_MouseDown(object sender,
MouseEventArgs e)
{
    this.tool.Bitmap =
BitmapCloner.Clone(this.ScreenshotPictureBox.Image);
    this.lastBitmap =
BitmapCloner.Clone(this.ScreenshotPictureBox.Image);
    this.drawing = true;
    this.tool.Position = e.Location;
    this.tool.Text = this.toolText;

    if (!typeof(BlurTool).IsInstanceOfType(this.tool))
        this.tool.Fill = this.FillCheckBox.Checked;
}

private void ScreenshotPictureBox_MouseMove(object sender,
MouseEventArgs e)
{
    if (this.drawing)
    {
        this.ScreenshotPictureBox.Image = this.tool.Draw(e);
    }
}

private void ScreenshotPictureBox_MouseUp(object sender,
MouseEventArgs e)
{
    this.drawing = false;

    if (typeof(BlurTool).IsInstanceOfType(this.tool))
    {
        Bitmap blurred = (this.tool as
BlurTool).ApplyBlur(this.lastBitmap);

        this.ScreenshotPictureBox.Image = blurred as Image;
    }
}

```

```

    }

    private void KeyDownHandler(object sender, KeyEventArgs e)
    {
        if (e.Control && e.Alt && e.KeyCode == Keys.Z)
        {
            this.ClearAllButton_Click(sender, e as EventArgs);
        }
        else if (e.Control && e.KeyCode == Keys.Z)
        {
            this.ScreenshotPictureBox.Image = this.lastBitmap as
Image;
            this.tool.Bitmap =
BitmapCloner.Clone(this.lastBitmap);
        }
    }

    private void PencilToolButton_Click(object sender, EventArgs
e)
    {
        this.DisableToolButtons();
        this.tool = new PencilTool(this.toolPen);
        this.PencilToolButton.BackColor = ColorPalette.Text;
        this.PencilToolButton.ForeColor = ColorPalette.Primary;
    }

    private void RectangleToolButton_Click(object sender,
EventArgs e)
    {
        this.DisableToolButtons();
        this.tool = new RectangleTool(this.toolPen);
        this.RectangleToolButton.BackColor = ColorPalette.Text;
        this.RectangleToolButton.ForeColor = ColorPalette.Primary;
    }

    private void EllipseToolButton_Click(object sender, EventArgs
e)
    {
        this.DisableToolButtons();
        this.tool = new EllipseTool(this.toolPen);
        this.EllipseToolButton.BackColor = ColorPalette.Text;
        this.EllipseToolButton.ForeColor = ColorPalette.Primary;
    }

    private void LineToolButton_Click(object sender, EventArgs e)
    {
        this.DisableToolButtons();
        this.tool = new LineTool(this.toolPen);
        this.LineToolButton.BackColor = ColorPalette.Text;
        this.LineToolButton.ForeColor = ColorPalette.Primary;
    }

```

```

private void ColorButton_Click(object sender, EventArgs e)
{
    ColorDialog dialog = new ColorDialog();
    dialog.FullOpen = true;
    dialog.Color = this.tool.Pen.Color;

    if (dialog.ShowDialog() == DialogResult.Cancel)
    {
        return;
    }

    this.ColorButton.BackColor = this.tool.Pen.Color =
        this.toolPen.Color = dialog.Color;
}

private void TextToolButton_Click(object sender, EventArgs e)
{
    this.SetTextButton_Click(sender, e);

    this.DisableToolButtons();
    this.tool = new TextTool(this.toolPen, this.toolFont);
    this.TextToolButton.BackColor = ColorPalette.Text;
    this.TextToolButton.ForeColor = ColorPalette.Primary;
}

private void BlurToolButton_Click(object sender, EventArgs e)
{
    this.DisableToolButtons();
    this.tool = new BlurTool();
    this.tool.Fill = false;
    this.BlurToolButton.BackColor = ColorPalette.Text;
    this.BlurToolButton.ForeColor = ColorPalette.Primary;
}

// Resets the tool panel buttons color
private void DisableToolButtons()
{
    foreach (Control control in this.ToolPanel.Controls)
    {
        if (typeof(Button).IsInstanceOfType(control) &&
            control.Name.Contains("Tool"))
        {
            control.BackColor = ColorPalette.Primary;
            control.ForeColor = ColorPalette.Text;
        }
    }
}

private void PenWidthInput_ValueChanged(object sender,
    EventArgs e)

```

```

        {
            this.tool.Pen.Width = this.toolPen.Width =
(float)this.PenWidthInput.Value;
        }

        private void PenWidthApplyButton_Click(object sender,
EventArgs e)
        {
            this.PenWidthInput_ValueChanged(sender, e);
        }

        private void ClearAllButton_Click(object sender, EventArgs e)
        {
            this.ScreenshotPictureBox.Image =
BitmapCloner.Clone(this.bitmap) as Image;
            this.tool.Bitmap = BitmapCloner.Clone(this.bitmap);
        }

        private void FontPickButton_Click(object sender, EventArgs e)
        {
            FontDialog dialog = new FontDialog();
            dialog.FontMustExist = true;
            dialog.Font = this.toolFont;

            if (dialog.ShowDialog() == DialogResult.Cancel)
                return;

            this.toolFont = this.tool.Font = dialog.Font;
        }

        private void SaveFileButton_Click(object sender, EventArgs e)
        {
            SaveFileDialog dialog = new SaveFileDialog();
            dialog.Filter = "JPEG images(*.jpg)|*.jpg|Bitmap
images(*.bmp)|*.bmp|PNG images(*.png)|*.png|All files(*.*)|*.*";

            if (dialog.ShowDialog() == DialogResult.Cancel)
                return;

            this.ScreenshotPictureBox.Image.Save(System.IO.Path.GetFullPath(dialog
.FileName));
            MessageBox.Show("Image succesfully saved!", "Success",
MessageBoxButtons.OK, MessageBoxIcon.Information);
        }

        private void SaveServerButton_Click(object sender, EventArgs
e)
        {
            this.ScreenshotPictureBox.Image.Save("screenshot.jpg");
            NameValueCollection nvc = new NameValueCollection();

```

```

        WebResponse response =
Requester.HttpUploadFile(this.serverURL, @"screenshot.jpg", "data",
        "image/jpeg", nvc);

        if (response != null)
        {
            using (StreamReader reader = new
StreamReader(response.GetResponseStream()))
            {
                var result = reader.ReadToEnd();

                if (!result.EndsWith(".jpg"))
                {
                    MessageBox.Show(String.Format("{0}: {1}",
result,
                    (response as HttpWebResponse).StatusCode),
                    "File upload error",
                    MessageBoxButtons.OK,
                    MessageBoxIcon.Error);
                }
                else
                {
                    var form = new
ScreenshotUploadResultForm(result);
                    form.ShowDialog();
                }
            }

            File.Delete("screenshot.jpg");
        }

        private void SetTextButton_Click(object sender, EventArgs e)
        {
            ScreenshoterTextSetForm form = new
ScreenshoterTextSetForm();

            if (form.ShowDialog() == DialogResult.OK)
            {
                this.toolText = form.TextInput.Text;
            }
        }

        private void FillCheckBox_CheckedChanged(object sender,
EventArgs e)
        {
            this.tool.Fill = this.FillCheckBox.Checked;
        }
    }
}

```

A.1.4 Файл Tools.cs

```

using System;
using System.Drawing;
using System.Windows.Forms;

namespace Screenshooter
{
    public abstract class Tool
    {
        // Default Pen Value; Crimson 2px width
        public Pen Pen = new Pen(Color.Crimson, 2);

        // Default Font Value; 11pt Sans Serif
        public Font Font = new Font(FontFamily.GenericSansSerif, 11);

        // Curson position at the start of drawing
        protected Point startPosition;

        // Bitmap state at the start of drawing
        protected Bitmap startBitmap;

        // Property, that shows if the figure should be filled
        // Used by RectangleTool class, etc.
        public bool Fill;

        // A vartable stroing text to draw on screenshot
        // The default value is "Text"
        public string Text = "Text";

        // Standard initilizer; replaces the default Pen & Font
        public Tool(Pen pen = null, Font font = null)
        {
            if (pen != null) this.Pen = pen;
            if (font != null) this.Font = font;
        }

        // Property to access private startPosition
        public Point Position
        {
            get => this.startPosition;
            set => this.startPosition = value;
        }

        // Property to reset startBitmap
        public Bitmap Bitmap
        {
            set => this.startBitmap = value;
        }
    }
}

```

```

        // Main tool method; draws a figure using current cursos
position
        // from MouseEventArgs
        public abstract Bitmap Draw(MouseEventArgs e);
    }

    // A pencil (brush) tool
    public class PencilTool : Tool
    {
        public PencilTool(Pen pen = null) : base(pen) { }

        // Drawing a curve line, following the cursor
        public override Bitmap Draw(MouseEventArgs e)
        {
            var g = Graphics.FromImage(this.startBitmap as Image);
            g.DrawLine(this.Pen, this.startPosition, e.Location);

            this.startPosition = e.Location;

            return this.startBitmap;
        }
    }

    // A straight line tool
    public class LineTool : Tool
    {
        public LineTool(Pen pen = null) : base(pen) { }

        // Draws a line from startPosition to current cursor position
        public override Bitmap Draw(MouseEventArgs e)
        {
            Bitmap bm = BitmapCloner.Clone(this.startBitmap);

            var g = Graphics.FromImage(bm as Image);
            g.DrawLine(this.Pen, this.startPosition, e.Location);

            return bm;
        }
    }

    public class TextTool : Tool
    {
        public TextTool(Pen pen = null, Font font = null) : base(pen,
font) { }

        public override Bitmap Draw(MouseEventArgs e)
        {
            Bitmap bm = BitmapCloner.Clone(this.startBitmap);

```

```

        var g = Graphics.FromImage(bm as Image);
        g.DrawString(this.Text, this.Font, this.Pen.Brush,
this.Position);

        return bm;
    }
}

// Rectangle tool
public class RectangleTool : Tool
{
    // Default rectangle value
    // 0, 0 - location (X, Y)
    // 2,2 - size (width, height)
    public Rectangle Rectangle = new Rectangle(0, 0, 2, 2);

    public RectangleTool(Pen pen = null, Font font = null) :
base(pen, font)
    {
        if (pen == null) this.Rectangle = new Rectangle(0, 0,
(int)this.Pen.Width, (int)this.Pen.Width);
    }

    // Draws a rectangle. If Fill property is true,
    // fills the figure with current Pen's color
    public override Bitmap Draw(MouseEventArgs e)
    {
        this.Rectangle.Location = new Point(Math.Min(e.X,
this.startPosition.X),
        Math.Min(e.Y, this.startPosition.Y));

        this.Rectangle.Width = Math.Abs(e.X -
this.startPosition.X);
        this.Rectangle.Height = Math.Abs(e.Y -
this.startPosition.Y);

        Bitmap bm = BitmapCloner.Clone(this.startBitmap);

        var g = Graphics.FromImage(bm as Image);

        if (this.Fill)
            g.FillRectangle(this.Pen.Brush, this.Rectangle);
        else
            g.DrawRectangle(this.Pen, this.Rectangle);

        return bm;
    }
}

// RectangleTool is inherited by EllipseTool, because of
// DrawEllipse method uses Rectangle instance in it

```

```

public class EllipseTool : RectangleTool
{
    public EllipseTool(Pen pen = null, Font font = null) :
base(pen, font) { }

    // Draws an ellipse. If Fill property is true,
    // fills the figure with current Pen's color
    public override Bitmap Draw(MouseEventArgs e)
    {
        this.Rectangle.Location = new Point(Math.Min(e.X,
this.startPosition.X),
        Math.Min(e.Y, this.startPosition.Y));

        this.Rectangle.Width = Math.Abs(e.X -
this.startPosition.X);
        this.Rectangle.Height = Math.Abs(e.Y -
this.startPosition.Y);

        Bitmap bm = BitmapCloner.Clone(this.startBitmap);

        var g = Graphics.FromImage(bm as Image);

        if (this.Fill)
            g.FillEllipse(this.Pen.Brush, this.Rectangle);
        else
            g.DrawEllipse(this.Pen, this.Rectangle);

        return bm;
    }
}

public class BlurTool : RectangleTool
{
    public BlurTool(Pen pen = null, Font font = null) : base(pen,
font) { }

    public Bitmap ApplyBlur(Bitmap image)
    {
        try
        {
            image = ApplyBlur(image, this.Rectangle,
(int)this.Pen.Width);
        }
        catch (ArgumentOutOfRangeException ex)
        {
            MessageBox.Show("Invalid blur zone! Please, select
another one.", "Selecting area error",
                MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
        catch (DivideByZeroException ex)
        {

```

```

        MessageBox.Show("Invalid blur zone! Please, select
another one.", "Selecting area error",
        MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
    return image;
}
// Розмиття зображення. Використовується підхід [18]
public static unsafe Bitmap ApplyBlur(Bitmap image, Rectangle
rectangle, int blurSize)
{
    Bitmap blurred = new Bitmap(image.Width, image.Height);

    // Робимо точну копію наданого растрового зображення
    using (Graphics graphics = Graphics.FromImage(blurred))
        graphics.DrawImage(image, new Rectangle(0, 0,
image.Width, image.Height),
        new Rectangle(0, 0, image.Width, image.Height),
GraphicsUnit.Pixel);

    // Обробка кожного пікселю у прямокутнику розмиття
    for (int xx = rectangle.X; xx < rectangle.X +
rectangle.Width; xx++)
    {
        for (int yy = rectangle.Y; yy < rectangle.Y +
rectangle.Height; yy++)
        {
            int avgR = 0, avgG = 0, avgB = 0;
            int blurPixelCount = 0;

            // середній колір червоного, зеленого та синього
            для кожного пікселя в
            // розмір розмиття, перевіривши, що немає виходу
            за межі зображення
            for (int x = xx; (x < xx + blurSize && x <
image.Width); x++)
            {
                for (int y = yy; (y < yy + blurSize && y <
image.Height); y++)
                {
                    Color pixel = blurred.GetPixel(x, y);

                    avgR += pixel.R;
                    avgG += pixel.G;
                    avgB += pixel.B;

                    blurPixelCount++;
                }
            }

            avgR = avgR / blurPixelCount;
            avgG = avgG / blurPixelCount;

```

```

        avgB = avgB / blurPixelCount;

        // тепер, коли відомо середнє значення розміру
        розмиття, встановлюємо для кожного пікселя цей колір
        for (int x = xx; x < xx + blurSize && x <
image.Width && x < rectangle.Width; x++)
            for (int y = yy; y < yy + blurSize && y <
image.Height && y < rectangle.Height; y++)
                blurred.SetPixel(x, y,
Color.FromArgb(avgR, avgG, avgB));
            }
        }

    return blurred;
}
}
}

```

A1.5 Код веб застосунку

A1.5.1 Файл server.views

```

import json

from django.shortcuts import render
from django.http import HttpResponse
from django.views.generic import View

from server.models import Image
from server.forms import ImageUploadForm

def about(request):
    return render(request, 'server/about.html')

class ImageUploadView(View):

    def get(self, request):
        return render(request, 'server/image_upload.html')

    def post(self, request):

        if len(request.FILES) < 1:

            return HttpResponse(
                'No files attached', status=401,
                content_type='text/plain'
            )

```

```

form = ImageUploadForm(request.POST, request.FILES)

if not form.is_valid():

    return HttpResponse(
        'Invalid data', status=401,
        content_type='text/plain'
    )

image = form.save()

filepath = ''.join((
    str(request.scheme), '://', str(request.get_host()),
    str(image.data.url)
))

return HttpResponse(
    filepath, status=200, content_type='text/plain'
)

```

A1.5.2 Файл server.models

```

import uuid
from django.db import models

class Image(models.Model):
    id = models.UUIDField(primary_key=True, default=uuid.uuid4,
                          editable=False)
    data = models.ImageField(upload_to='img/')
    uploaded = models.DateTimeField(auto_now_add=True)

```

A1.5.3 Файл server.forms

```

from django import forms

from server.models import Image

class ImageUploadForm(forms.ModelForm):

    class Meta:
        model = Image
        fields = ('data',)

```

ДОДАТОК Б
Слайди презентації

ДИПЛОМНА РОБОТА

РОЗРОБКА ПРОГРАМИ ЗБЕРЕЖЕННЯ ЗНІМКІВ ЕКРАНУ

Розробив: ст. групи КНТ-219сп
Керівник: доцент, к.т.н.

Ганжа Д.І.
Сердюк С.М.

Рисунок Б.1 – Слайд №1

Мета роботи

розробка програми, що дозволяє захоплювати зображення з області екрану, редагувати отриманий результат та зберігати вихідне зображення у пам'яті ПК або на сервері

Об'єкт дослідження – процеси запису та редагування зображень екрану ПК

Предмет дослідження – методи запису та редагування зображень екрану ПК



Задачі, що вирішуються

- провести аналіз предметної області;
- провести огляд програмних аналогів;
- розробка технічного завдання;
- розробка структури програмного забезпечення;
- розробка моделей програмного забезпечення;
- програмна реалізація застосунку;
- розробка необхідної документації.

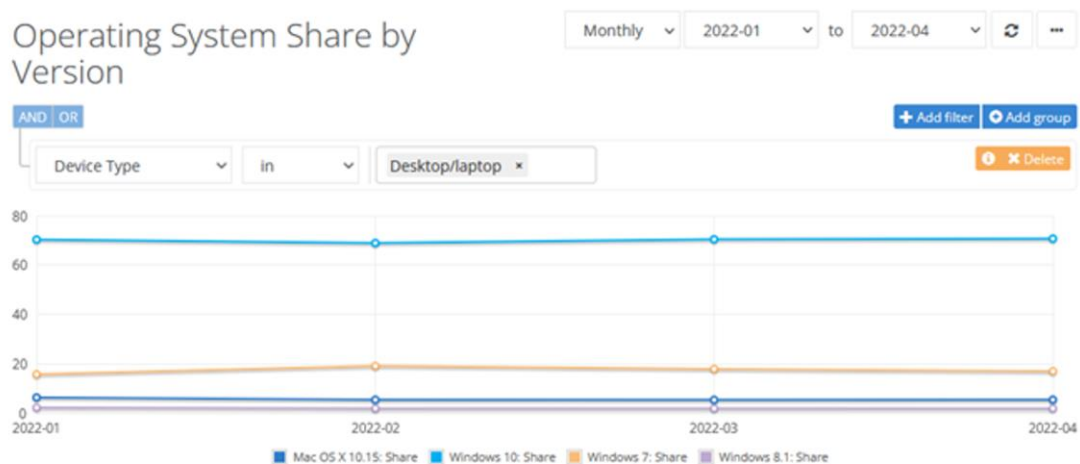


НУ "ЗП", Кафедра програмних засобів

Рисунок Б.3 – Слайд №3

3

Рейтинг операційних систем 2022

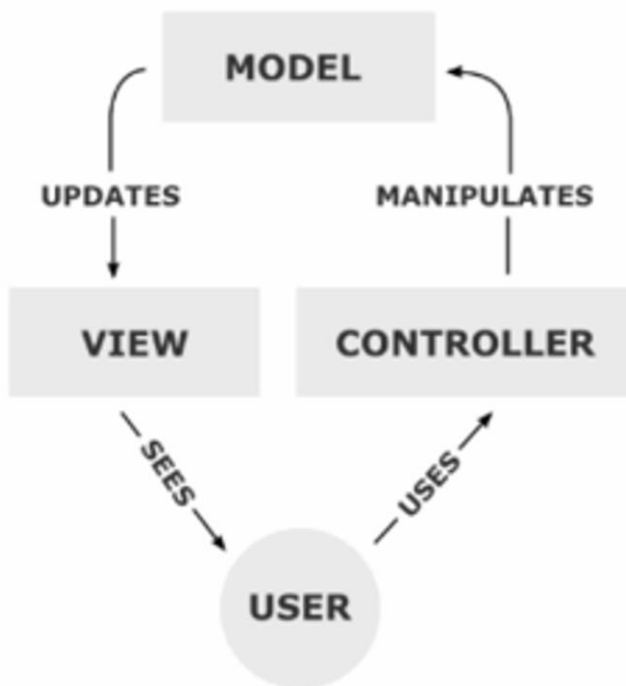


НУ "ЗП", Кафедра програмних засобів

Рисунок Б.4 – Слайд №4

4

MVC-архітектура



НУ "ЗП", Кафедра програмних засобів

Рисунок Б.5 – Слайд №5

5

Діаграма компонентів

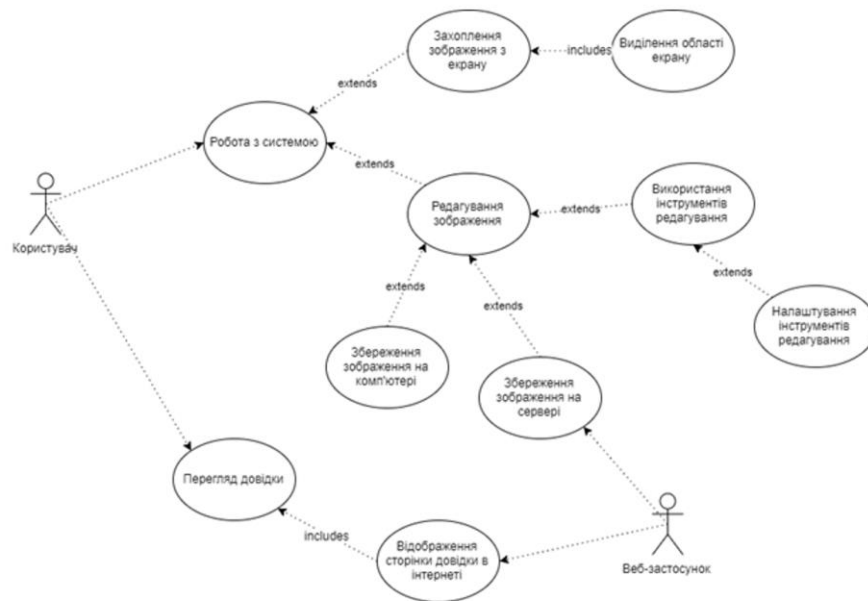


НУ "ЗП", Кафедра програмних засобів

Рисунок Б.6 – Слайд №6

6

Діаграма прецедентів



НУ "ЗП", Кафедра програмних засобів

7

Рисунок Б.7 – Слайд №7

Блок-схема алгоритму роботи програми



НУ "ЗП", Кафедра програмних засобів

8

Рисунок Б.8 – Слайд №8



9

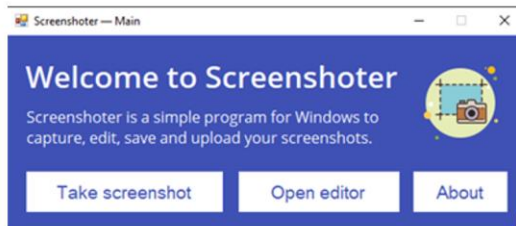
Діаграма класів контролеру



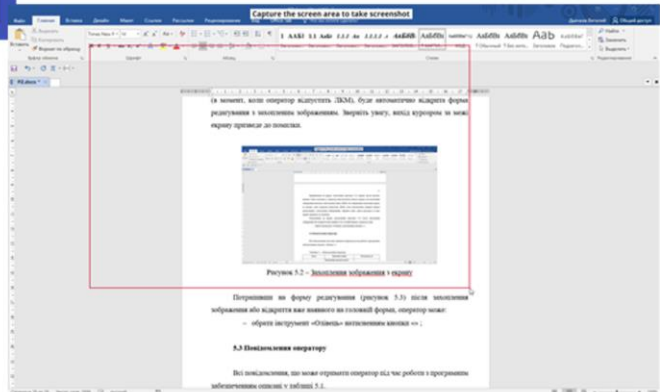
10

Рисунок Б.10 – Слайд №10

Форми Desktop-застосунку



Головна форма



Форма захоплення зображення з екрану

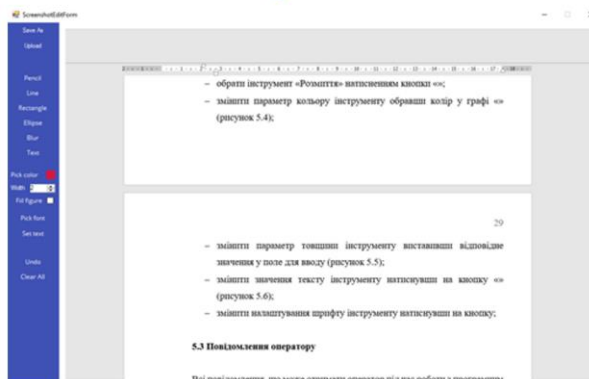


НУ "ЗП", Кафедра програмних засобів

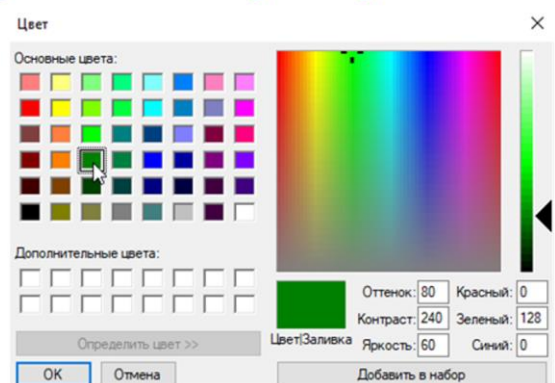
Рисунок Б.11 – Слайд №11

11

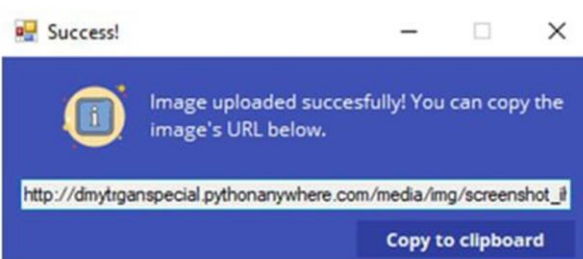
Форми Desktop-застосунку



Головна форма



Форма вибору кольору



Вікно результатів завантаження

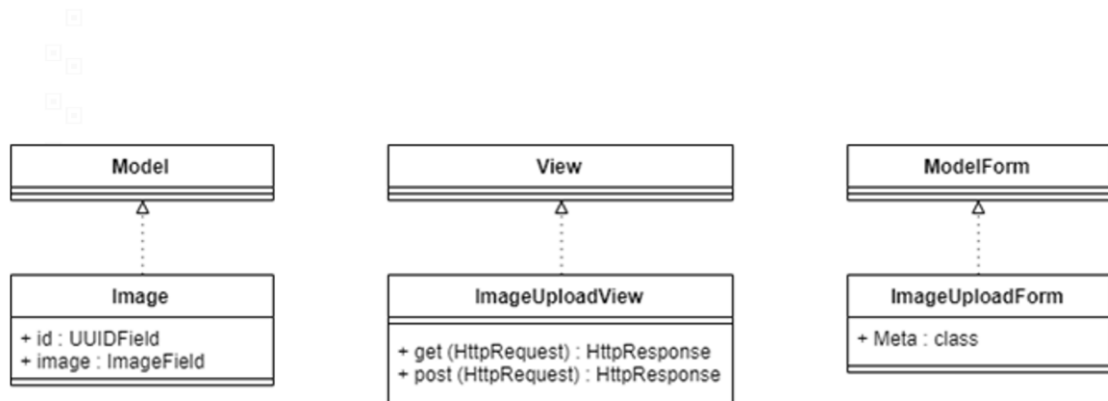


НУ "ЗП", Кафедра програмних засобів

Рисунок Б.12 – Слайд №12

12

Діаграма класів веб застосунку

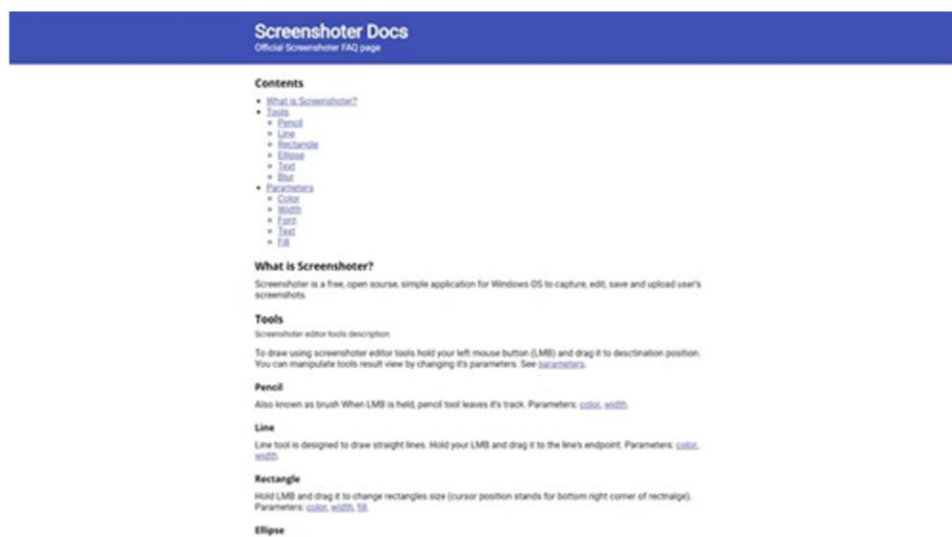


НУ "ЗП", Кафедра програмних засобів

13

Рисунок Б.13 – Слайд №13

Сторінка довідки веб застосунку



НУ "ЗП", Кафедра програмних засобів

14

Рисунок Б.14 – Слайд №14

Висновки

- проведено аналіз предметної області;
- проведено огляд програмних аналогів;
- розроблено технічне завдання;
- розроблено структуру програмного забезпечення;
- розроблено моделі програмного забезпечення;
- програмно реалізовано застосунок;
- розроблено необхідну документацію.



НУ "ЗП", Кафедра програмних засобів

15

Рисунок Б.15 – Слайд №15

**Доклад закінчено,
дякую за увагу!**



НУ "ЗП", Кафедра програмних засобів

16

Рисунок Б.16 – Слайд №16