

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій
(повне найменування факультету)

Кафедра програмних засобів
(повне найменування кафедри)

Пояснювальна записка

до дипломного проекту (роботи)

бакалавр

(ступінь вищої освіти)

на тему ПРОГРАМНА РЕАЛІЗАЦІЯ ЗАСТОСУНКУ
ДЛЯ ЗАПИСУ ЕКРАНУ.
SOFTWARE IMPLEMENTATION OF
A SCREEN RECORDING APPLICATION

Виконав: студент 4 курсу, групи КНТ-121
Спеціальності 121 Інженерія програмного
забезпечення

(код і найменування спеціальності)

Освітня програма (спеціалізація)
Інженерія програмного забезпечення

ПОПОВ К.С.

(ПРИЗВИЩЕ та ініціали)

Керівник ЛЕОЩЕНКО С.Д.

(ПРИЗВИЩЕ та ініціали)

Рецензент БАБЕНКО Н.В.

(ПРИЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет КНТ
Кафедра програмних засобів
Ступінь вищої освіти бакалавр
Спеціальність 121 Інженерія програмного забезпечення
(код і найменування)
Освітня програма (спеціалізація) Інженерія програмного забезпечення
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
Сергій СУББОТІН
« » 2025 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА

ПОПОВА Кирила Сергійовича

(ПРИЗВИЩЕ, ім'я, по батькові)

Тема проєкту (роботи) Програмна реалізація застосунку для запису екрану.
Software Implementation of a Screen Recording Application
керівник проєкту (роботи) доктор філософії, доцент ЛЕОЩЕНКО Сергій
Дмитрович

(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від “28” квітня 2025 року № 209

2. Строк подання студентом проєкту (роботи) 8 травня 2025 року

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

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Матеріали і методи. 3. Опис програми. 4. Експлуатація, тестування та експериментальне дослідження програми. Додаток А Технічне завдання, Додаток Б Опис програми, Додаток В Текст програми, Додаток Г Слайди презентації.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, кількість слайдів, плакатів) Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРИЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	ЛЕОЩЕНКО С.Д., доц.		
Нормоконтроль	ДЕЙНЕГА Л.Ю., ст. викл.		

7. Дата видачі завдання «28» квітня 2025 року.

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.	1 тиждень	Розділ 1
3	Вибір мови програмування та інших технологій розробки.	2 тиждень	Розділ 2
4	Розробка структури програми.	2 тиждень	Розділ 3
5	Розробка програми.	3-4 тижні	Розділи 3, 4
6	Тестування та експериментальне дослідження програмного забезпечення.	5 тиждень	Розділ 4
7	Оформлення пояснювальної записки та документів до неї.	6 тиждень	Додатки
8	Нормоконтроль та рецензування.	7 тиждень	
9	Захист роботи.	8 тиждень	

Студент

(підпис) Кирило ПОПОВ
(Ім'я ПРИЗВИЩЕ)

Керівник проєкту (роботи)

(підпис) Сергій ЛЕОЩЕНКО
(Ім'я ПРИЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра:
95 с., 42 рис., 7 табл., 4 дод., 29 джерел.

ПРОГРАМНИЙ ПРОДУКТ, СКРІНШОТ, ВІДЕО, МАЛЮВАННЯ,
ЗБЕРЕЖЕННЯ ФАЙЛІВ, РЕЖИМИ РОБОТИ.

Об'єкт дослідження – процеси взаємодії програмного забезпечення з операційною системою для доступу до зображення екрана, зокрема принципи обробки графічних даних під час захоплення зображення, відеозапису та нанесення графічних елементів.

Предмет дослідження – методи та інструменти реалізації функціоналу скріншотів, відео- та аудіозапису, а також графічного редагування в середовищі Windows Forms, включно з підходами до оптимізації продуктивності, збереження якості зображення та зручності для користувача.

Метою роботи є створення застосунку з інтуїтивним інтерфейсом, що забезпечує користувачеві можливість здійснювати захоплення зображень з екрана, записувати відео з аудіо супроводом, а також виконувати базове редагування зображень без потреби звертатися до сторонніх редакторів.

Матеріали, методи та технічні засоби: середовище розробки Microsoft Visual Studio, платформа .NET Framework, мова програмування C#, бібліотеки Accord.Video.FFmpeg, NAudio.Wave, GDI+ для обробки графіки та звуку, стандартні засоби Windows.Forms.

Результати. Розроблено програмне забезпечення, яке надає можливість створювати скріншоти, записувати відео зі звуком та малювати на зображеннях, забезпечуючи користувача гнучким та зручним інструментом для роботи з мультимедіа.

Практична цінність роботи полягає у створенні ефективного рішення для запису та редагування вмісту екрана, що може бути використане у сфері створення навчальних матеріалів, інструкцій, презентацій, а також у професійній діяльності, пов'язаній із графічним дизайном та відеомонтажем.

Висновки. Відбулася реалізація програми з широким спектром функцій для зйомки екрана, запису відео та нанесення графічних позначок. Забезпечено зручність у використанні, сумісність із сучасними версіями Windows та підтримку різних форматів збереження файлів.

Галузь використання – програмне забезпечення може застосовуватися у сфері освіти, створення відеоуроків, технічної підтримки, документування, тестування програмного забезпечення та розробки мультимедійного контенту.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 95 pages, 42 figures, 7 tables, 4 appendixes, 29 sources.

SOFTWARE PRODUCT, SCREENSHOT, VIDEO, DRAWING, FILE SAVING, OPERATING MODES.

The object of research is the processes of interaction between software and the operating system for accessing the screen image, particularly the principles of graphical data processing during image capture, video recording, and the application of graphical elements.

The subject of the research is methods and tools for implementing screenshot, video and audio recording functionality, as well as graphical editing in the Windows Forms environment, including approaches to performance optimization, image quality preservation, and user convenience.

The aim of research is to develop an application with an intuitive interface that enables the user to capture screen images, record video with audio, and perform basic image editing without the need for third-party editors.

Materials, methods and technical tools: Microsoft Visual Studio development environment, .NET Framework platform, C# programming language, Accord.Video.FFmpeg and NAudio.Wave libraries for video and audio processing, GDI+ for graphic rendering, and standard Windows Forms tools.

Results. A software solution has been developed that enables users to take screenshots, record video with audio, and draw on images, providing a flexible and user-friendly tool for multimedia tasks.

The practical value of the work is the creation of an efficient solution for screen recording and editing, which can be used for educational content creation, instructional videos, presentations, and professional activities related to graphic design and video editing.

Conclusions. The program has been implemented with a wide range of functions for screen capturing, video recording, and adding graphic annotations. It ensures ease of use, compatibility with modern Windows versions, and support for various file-saving formats.

Scope of use – the software can be applied in education, video tutorials, technical support, documentation, software testing, and multimedia content development.

ЗМІСТ

	С.
Вступ.....	11
1 Аналіз предметної області.....	13
1.1 Огляд предметної області.....	13
1.2 Огляд сучасного стану предметної області	14
1.3 Потреба програмних рішень	14
1.4 Порівняння існуючих аналогів	15
1.4.1 Передмова	15
1.4.2 Snagit.....	15
1.4.3 Lightshot.....	16
1.4.4 Windows Snipping Tool.....	17
1.5 Постановка завдання.....	20
1.5.1 Функціональність системи	20
1.5.2 Вимоги до інтерфейсу	21
1.5.3 Вимоги до безпеки.....	21
1.5.4 Нефункціональні вимоги до продукту	21
1.6 Висновки за розділом.....	22
2 Матеріали і методи.....	23
2.1 Вибір мови програмування	23
2.2 Вибір середовища розробки	27
2.3 Використані бібліотеки.....	29
2.4 Структурна схема розробки	32
2.5 Висновки до розділу	33
3 Основні рішення щодо реалізації компонентів системи.....	34
3.1 Алгоритми функціонування розробленої програми.....	34
3.1.1 Алгоритм виділення ділянки зображення на екрані.....	35
3.1.2 Алгоритм малювання на екрані за допомогою інструментів	36
3.1.3 Алгоритм динамічної роботи з відкритими вікнами	39

3.2	Опис модулів розроблюваної системи	40
3.2.1	Клас frmCapture	41
3.2.2	Клас SettingsMenu.....	42
3.2.3	Клас CatchArea.....	42
3.3	Проектування графічного інтерфейсу.....	44
4	Експлуатація, тестування та експериментальне дослідження програми	47
4.1	Призначення й умови застосування програми	47
4.1.1	Призначення програми	47
4.1.2	Функції програми	47
4.1.3	Умови застосування програми.....	48
4.2	Вимоги до інтерфейсу.....	49
4.3	Умови виконання програми	49
4.4	Методика та результати тестування	50
4.5	Висновки за отриманими результатами.....	62
	Висновки	63
	Перелік джерел посилання	65
	Додаток А Технічне завдання	68
	Додаток Б Опис програми	73
	Додаток В Текст програми	77
	Додаток Д Слайди презентації.....	87

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

API – Application Programming Interface;

AVI – Audio Video Interleave;

FPS – Frames Per Second;

GDI – Graphics Device Interface;

GIF – Graphics Interchange Format;

HTML5 – HyperText Markup Language 5;

IDE – Integrated Development Environment;

JPG – Joint Photographic Experts Group;

JVM – Java Virtual Machine;

PDF – Portable Document Format;

PNG – Portable Network Graphics;

PrtScr – Print Screen;

WPF – Windows Presentation Foundation.

ВСТУП

У сучасному світі, коли технології швидко розвиваються, людям все частіше потрібно мати швидкий і зручний доступ до великої кількості інформації – і зазвичай вона відображається на екрані комп'ютера. Саме тому розробка програм для автоматичного збереження знімків та запису з екрана набуває все більшої актуальності. Такі інструменти значно полегшують фіксацію інформації, допомагають у роботі чи навчанні та дозволяють швидко повертатися до потрібних матеріалів у зручному форматі.

Ще з перших років появи персональних комп'ютерів користувачі намагались зберігати зображення з екрана. Спочатку це були прості застосунки, які робили скріншоти за натисканням певних клавіш. З часом функціональність розширювалась: стало можливим зберігати зображення у різних форматах, обирати ділянку знімку, а пізніше – і редагувати отримані фото. Сучасні операційні системи мають вбудовані інструменти для таких задач, що значно спрощує роботу з візуальною інформацією.

Сучасні програми для захоплення зображення з екрана мають розширені функції, такі як можливість редагування зображень, додавання анотацій, інтеграція з хмарними сервісами та автоматизація процесу. Вони також підтримують відеозапис, що робить їх потужним інструментом для обміну інформацією, документування та вирішення різноманітних завдань. Розвиток таких програм є результатом великого попиту користувачів на зручні та ефективні способи візуалізації та обміну інформацією.

Ідея розробки програми для запису інформації з екрана є актуальною у сьогоdnішній час, оскільки вона відповідає зростаючим вимогам користувачів у багатьох сферах діяльності. Спочатку цей функціонал використовувався для створення знімків екрана для аналізу, навчання, документації або створення візуальних матеріалів для презентацій. З часом потреба в автоматизованому збиранні скріншотів і відео розширилася і охопила різні професійні галузі, включаючи програмістів, викладачів, тестувальників.

Для успішної реалізації програми необхідно вирішити низку завдань, які є невід'ємною частиною процесу розробки програмного забезпечення. Серед основних завдань виділяються:

- аналіз наявних аналогів програмного забезпечення у предметній області з визначенням їхніх переваг та недоліків;
- визначення вимог до програми з урахуванням зручності використання та практичної доцільності на основі особливостей потреб користувачів;
- розробка інтерфейсу користувача, який забезпечить ефективну взаємодію завдяки своїй ергономічності та інтуїтивній зрозумілості;
- проектування та подальша розробка алгоритмів для створення зображень і відео, обробки скріншотів графічними інструментами, можливості збереження отриманих матеріалів із вибором їхнього місця розташування;
- забезпечення стабільної інтеграції та взаємодії з використаними інструментами, системними засобами включно із зовнішніми бібліотеками;
- оформлення механізмів обробки виняткових ситуацій з метою підвищення надійності й безпеки роботи програмного засобу.

Виконання цих завдань дозволить створити ефективний інструмент для автоматизованого захоплення екрана, що відповідатиме сучасним вимогам та є корисним для широкого кола користувачів.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

На сьогоднішній день, коли комп'ютери стали невід'ємною частиною і повсякденного життя, і роботи, і навчання, все більше з'являється потреба в зручних інструментах для збереження того, що користувачі бачать на екрані. Часто треба швидко зафіксувати зображення або записати відео з екрана – наприклад, для створення навчальних матеріалів, технічної документації, презентацій чи під час розробки програмного забезпечення.

Завдяки своєму простому, але потужному механізму, програми для захоплення екрана дозволяють користувачам фіксувати інформацію, яка відображається на екрані, і використовувати її для подальшого аналізу або передачі іншим користувачам. Однак із розвитком технологій та постійним зростанням потреб користувачів виникають нові вимоги до таких інструментів, включаючи необхідність автоматизації процесу захоплення, інтеграції з іншими застосунками, а також розширення можливостей.

1.1 Огляд предметної області

Завдяки сучасним процесам цифровізації та вдосконалення технологій потреба фіксувати все, що відбувається на екрані комп'ютера, лише зростає. Захоплення екрана вже давно стало звичною частиною роботи в багатьох сферах: від програмування й технічної підтримки до освіти, тестування та створення презентацій. Цей процес полягає в отриманні зображення або відео вмісту екрана, яке згодом може бути використано для аналізу, обміну інформацією або архівування [1].

Технології для захоплення екрана мають різноманітні застосування, починаючи від простих знімків екрана і закінчуючи створенням складних відеоуроків або записів для підтримки користувачів. Поява спеціалізованих програм дозволяє значно полегшити процес захоплення, редагування та збереження контенту, що відображається на екрані комп'ютера [2].

1.2 Огляд сучасного стану предметної області

Сучасна індустрія має велику кількість програмних рішень для захоплення екрана, які пропонують різні функціональні можливості: від простих програм для створення скріншотів до комплексних інструментів для запису відео з екрана з можливістю редагування та додавання анотацій. Зазвичай ці програми підтримують збереження в різних форматах (.png, .jpg, .mp4, .avi тощо) і надають вибір конкретної ділянки екрана для захоплення.

Одними з найбільш популярних програм є:

- Snagit – це професійний інструмент для захоплення екрана з можливістю редагування;
- Windows Snipping Tool – вбудований інструмент Windows для захоплення екрана з базовими можливостями редагування;
- Lightshot – легка у використанні програма для швидкого створення скріншотів і їх подальшої обробки користувачем.

1.3 Потреба програмних рішень

Незважаючи на наявність численних програмних продуктів, які дозволяють захоплювати екран, користувачі часто стикаються з функціональними обмеженнями або відсутністю необхідних можливостей для особливих випадків. Однією з основних проблем є те, що більшість програм мають обмежений набір функцій або потребують додаткових налаштувань для виконання певних завдань. Наприклад, захоплення не лише статичних зображень, а й відео з екрана, збереження знімків у кількох форматах одночасно або можливість віддаленого доступу до програм для захоплення [3].

Враховуючи ці проблеми, виникає потреба в більш універсальних та інтуїтивно зрозумілих інструментах для захоплення екрана, які можуть автоматизувати процес збору зображень і відео в реальному часі та забезпечити зручне збереження та обробку отриманого контенту [3].

Крім того, сучасні програмні рішення для захоплення екрана можуть включати інтеграцію з іншими програмами та хмарними сервісами для забезпечення більш ефективного обміну та обробки даних. Тому необхідно розробити програму, яка забезпечить гнучкість налаштувань і дозволить користувачам зручно й швидко виконувати завдання захоплення екрана в різних умовах [3].

1.4 Порівняння існуючих аналогів

Цей застосунок базується на виборі режиму та секунд затримки програми для отримання даних зображення з екрана користувача, які згодом будуть оброблені, щоб відобразити рисунок на головній панелі форми. Мета програми – зробити захоплення екрана у вигляді фото чи відео, обробити фотографію за необхідністю та зберегти у будь-який доступний спосіб зображення.

1.4.1 Передмова

Існують деякі застосунки, у яких подібні алгоритми вже реалізовані, але сама програма має деякі відмінності, у тому числі функціональні. За основу цього застосунку були використані такі матеріали: необхідна технічна документація, програма Windows Snipping Tool, програма Snagit, програма Lightshot.

1.4.2 Snagit

Snagit [4] є популярним програмним забезпеченням для створення знімків та запису відео екрана, розробленим компанією TechSmith. Цей інструмент володіє рядом функцій, для редагування зображень та експорту і спільного доступу, що робить його популярним серед користувачів.

Програма дозволяє створювати анотації, додавати текст, стрілки, рамки та інші графічні елементи, що значно полегшує пояснення матеріалів. Крім того, Snagit підтримує функцію прокручуваного знімка екрана, що дає змогу знімати вміст сторінок або вікон, які виходять за межі екрана [4].

Завдяки інтуїтивно зрозумілому інтерфейсу Snagit підходить як для початківців, так і для досвідчених користувачів. Він підтримує інтеграцію з популярними хмарними сервісами, що дозволяє легко обмінюватися зображеннями та відео. Snagit доступний для Windows і macOS, що робить його універсальним рішенням для професійного та особистого використання. Робота програми продемонстрована на рисунку 1.1.

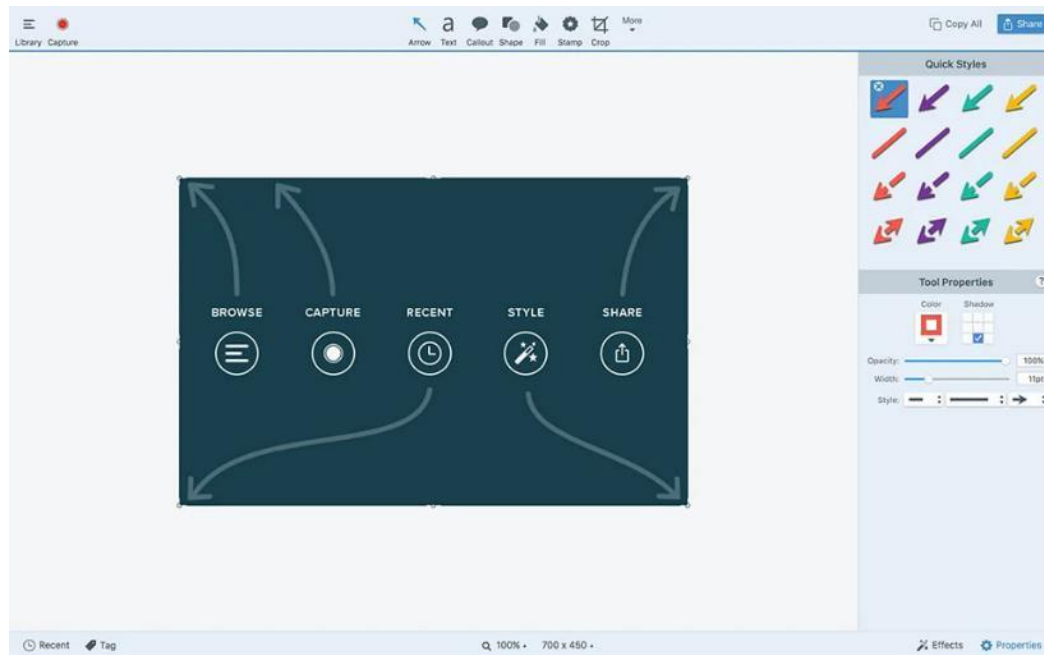


Рисунок 1.1 – Робота програми Snagit [4]

1.4.3 Lightshot

Lightshot [5] – програма, яка здійснює знімки екрана для Windows, macOS і Ubuntu, працює як ножиці з Windows, надаючи додаткові можливості для редагування зображень та публікації їх в Інтернеті. Програма замінює обробник кнопки PrtScr за замовчуванням та перехоплює її для свого запуску, де користувач може обрати потрібну йому ділянку на екрані.

Lightshot має простий та зручний інтерфейс, що дозволяє швидко робити скріншоти, редагувати їх, додаючи текст, стрілки, рамки та інші елементи. Крім того, програма підтримує можливість швидкого завантаження знімків у хмарне сховище та отримання унікального посилання [5].

Однією з корисних функцій Lightshot є пошук схожих зображень в Інтернеті, що може бути корисним для аналізу або порівняння контенту. Програма підтримує гарячі клавіші для ще швидшого доступу до функцій знімка екрана. Завдяки легкості у використанні Lightshot є зручним інструментом як для особистого, так і для професійного застосування. Робота програми продемонстрована на рисунку 1.2.



Рисунок 1.2 – Робота програми Lightshot [5]

1.4.4 Windows Snipping Tool

Windows Snipping Tool [6] є вбудованим у операційну систему Windows інструментом для захоплення зображень з екрана та їх редагування. Це досить потужний та ефективний інструмент, призначений для різних завдань, пов'язаних з роботою з екраном. Простий інтерфейс робить його ідеальним для користувачів, які шукають простий спосіб зробити знімок екрана та додати деякі базові зміни до зображень.

Windows Snipping Tool дозволяє робити знімки у кількох режимах, включаючи прямокутне, довільне, віконне та повноекранне захоплення. Також доступна функція затримки, що дає змогу робити скріншоти через відведений проміжок часу, що корисно для фіксації випадних меню або інших динамічних елементів [6].

Інструмент містить базові можливості редагування, такі як виділення, підписування та прості позначки. Після створення знімка його можна зберегти у різних форматах або скопіювати до буфера обміну. Робота програми продемонстрована на рисунку 1.3.

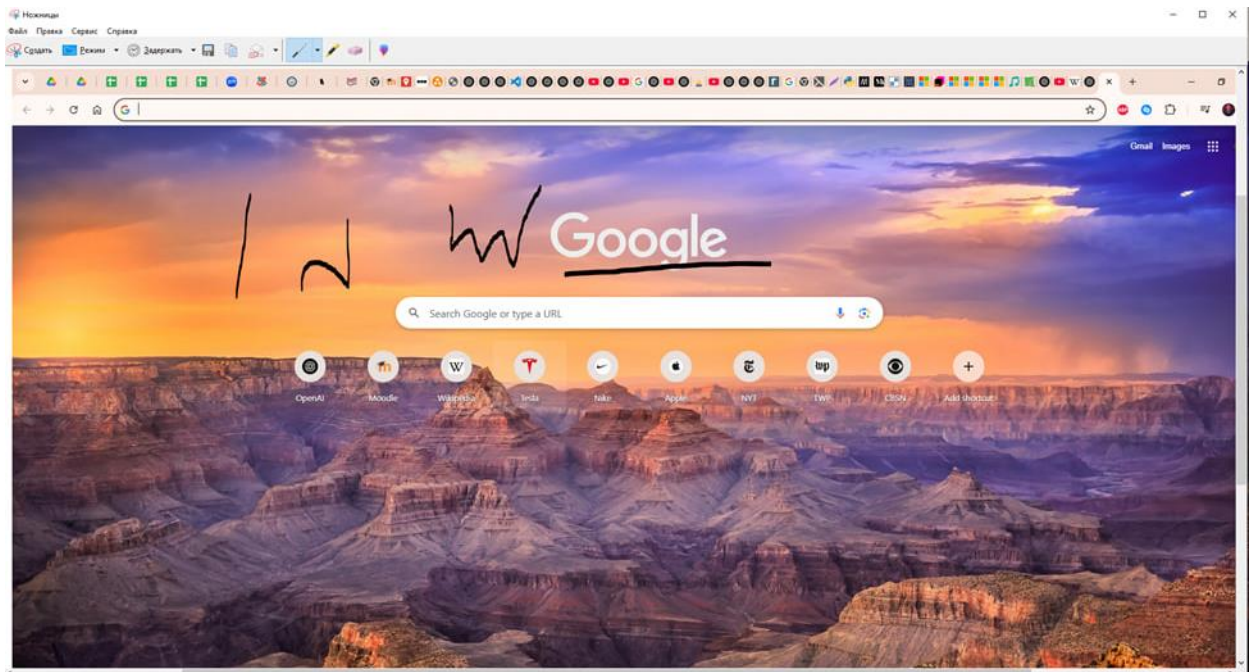


Рисунок 1.3 – Робота програми Windows Snipping Tool [6]

Необхідно виконати порівняння існуючих аналогів за такими критеріями: розробник, робоча платформа, ціна, основні функції, формати збереження, редагування, запис відео, гарячі клавіші, доступність і простота використання. Порівняння існуючих програм для роботи із зображенням екрана представлено у вигляді таблиці 1.1.

Таблиця 1.1 – Порівняння існуючих аналогів

Критерії	Snipping Tool	Snagit	Lightshot
Розробник	Microsoft	TechSmith	Skillbrains
Платформи	Windows	Windows, macOS	Windows, macOS
Ціна	Безкоштовний	Платний (~\$63)	Безкоштовний
Основні функції	Захоплення екрана, просте редагування	Захоплення екрана, запис відео, потужний редактор	Захоплення екрана, базове редагування
Формати збереження	.png, .jpeg, .gif	.png, .jpg, .gif, .mp4, .pdf тощо	.png, .jpg
Редагування	Мінімальні можливості (олівець, маркер)	Розширений редактор з ефектами, текстом	Мінімальні можливості (основні анотації)
Запис відео	Немає	Є	Немає
Гарячі клавіші	Є	Є	Немає
Простота використання	Дуже проста	Складніший, але більш потужний	Легка у використанні

Можна зробити наступні висновки з порівняння. Snipping Tool – це найпростіший варіант, оскільки він вбудований у Windows, безкоштовний і не потребує встановлення додаткових програм. Однак його функціонал є базовим, тому він обмежений: відсутні потужні інструменти редагування, запис відео та хмарне збереження.

Snagit – найпотужніший інструмент серед трьох, оскільки підтримує не лише знімки екрана, а й запис відео, має розширений редактор та додаткові функції, такі як автоматичне розпізнавання вікон. Однак він є платним, що може бути недоліком для деяких користувачів. Найкращий варіант для професійного використання.

Lightshot – легкий, швидкий і безкоштовний інструмент для створення скріншотів. Він має базові можливості редагування та дозволяє швидко завантажувати зображення в хмару. Проте не підтримує запис відео і має менш розширений функціонал, ніж Snagit. Ідеальний для швидкого збереження та обміну скріншотами.

1.5 Постановка завдання

Зважаючи на сучасні можливості у подачі інформації у цифровому вигляді, виникає необхідність розробити програмне забезпечення, яке забезпечить користувачам розширені можливості захоплення зображення з екрана, які виходять за межі функціоналу існуючих рішень. Застосунок дозволить не лише робити знімки вікон і вибраних ділянок, але й додасть інтуїтивні інструменти для редагування знімків, підсвічування, анотацій та збереження історії змін. Система дозволить користувачам взаємодіяти з попередніми знімками, відновлювати попередній стан. Маючи можливість робити запис відео, визначаючи ділянку, наявність звуку чи кількість кадрів у секунду, застосунок отримає весь необхідний спектр для збереження інформації з екрана, а ергономічний, зрозумілий і налаштований користувачем інтерфейс зробить роботу із програмою простою і послідовною.

1.5.1 Функціональність системи

Система має реалізовувати наступні функціональні вимоги:

- робота із захоплення екрана у трьох режимах: повноекранному, виділеної ділянки та відкритого вікна на комп'ютері;
- запис відео у повноекранному та виділеної ділянки режимах з можливістю обрати необхідну кількість FPS (кадрів на секунду);
- збереження захопленого зображення до буфера обміну або на жорсткий диск користувача, обравши відповідний шлях та розширення файлу;
- наявність можливості обробки створеного зображення за допомогою різноманітних фігур (коло, лінія), вибираючи для цього різні кольори та товщину лінії з функціоналом керувати редагуванням;
- редагування зображення за допомогою пензля, який можна оптимізувати за кольором, типом пера та шириною користувачу.

1.5.2 Вимоги до інтерфейсу

Основні вимоги до інтерфейсу включають такі пункти:

- інтуїтивно зрозумілий та зручний інтерфейс, що забезпечує легкий доступ до основних функцій (захоплення екрана, редагування, збереження);
- мінімалістичний дизайн зі зрозумілими кнопками, інструментами;
- підтримка темної та світлої теми для зручності користувача.

1.5.3 Вимоги до безпеки

Для забезпечення безпечної роботи програми передбачені вимоги:

- відсутність несанкціонованого збору, передачі даних користувача;
- захист від несанкціонованого доступу до знімків екрана;
- відсутність прихованих або шкідливих процесів у програмі.

1.5.4 Нефункціональні вимоги до продукту

Нефункціональні аспекти, які повинні бути реалізовані у програмі:

- швидке захоплення та обробка зображень без значного навантаження на систему;
- підтримка різних версій Windows (від Windows 10);
- стабільна робота без зависань і критичних помилок;
- можливість розширення функціоналу;
- підтримка різних форматів файлів для збереження зображення (.png, .jpg, .bmp) і відео (.avi, .mp4).

1.6 Висновки за розділом

У першому розділі відбувся огляд предметної області та ознайомлення з загальноприйнятими основами реалізації програм для захоплення зображення з екрана користувача. Інтернет має деякі сервіси та програмні забезпечення, які в тій чи іншій спосіб намагаються відтворити розширений функціонал програми зі зручним та простим інтерфейсом у використанні. Використовуючи належну документацію роботи з користувацьким екраном, розробки інших програмістів, було виділено ті пункти й категорії, які заклали основу для розробки майбутньої програми в рамках проєкту. Завдяки цьому вдалося реалізувати список функціональних, нефункціональних вимог до програмного забезпечення, яке відповідає потребам користувачів у сфері безпеки та зручності використання.

2 МАТЕРІАЛИ І МЕТОДИ

У цьому розділі будуть представлені основні матеріали та методи, які були використані під час розробки програмного продукту. Важливим аспектом є вибір інструментів, які дозволяють досягти поставлених цілей. Описані матеріали та методи визначають загальний підхід до виконання роботи, а також характеризують ключові етапи використані для збору та обробки даних.

2.1 Вибір мови програмування

Ефективна розробка програмного забезпечення для захоплення зображень з екрана вимагає вибору відповідних мов програмування, що забезпечують високу продуктивність, доступ до графічних ресурсів і зручність для інтеграції з іншими системами. Існує кілька мов програмування, кожна з яких має свої особливості, які дозволяють ефективно реалізовувати функціонал створення скріншотів.

Для швидкого створення скріншотів та обробки графічних зображень одним з найпопулярніших виборів є Python [7]. Завдяки багатьом бібліотекам, таким як Pillow і PyAutoGUI, Python дозволяє швидко захоплювати зображення з екрана, здійснювати їх обробку та зберігання. Це робить його ідеальним для створення простих інструментів для захоплення зображень і автоматизації.

Для розв'язання задач, що потребують високої продуктивності та інтеграції з іншими складними системами, можна звернутися до мов C++ або Java. Ці мови мають потужні графічні бібліотеки та API, такі як WinAPI [8] для C++ чи java.awt [9] для Java, які надають гнучкі інструменти для створення скріншотів з екрана та збереження їх у потрібному форматі.

Якщо важливо створювати інтерактивні та графічно привабливі інтерфейси для захоплення скріншотів, тоді мова JavaScript [10] в поєднанні з HTML5 та бібліотеками, такими як HTML5 Canvas, дозволяє реалізувати можливість захоплення зображень без потреби у сторонніх програмах.

Важливу роль у створенні скріншотів відіграє також наявність мови програмування, що має підтримку для роботи з графічними та медіа-файлами, таких як C# [11],[12] з його класами для роботи з вікнами Windows або Swift для створення скріншотів в середовищі macOS. У поєднанні з відповідними бібліотеками, вони забезпечують доступ до можливостей створення скріншотів з екрана, враховуючи особливості операційних систем.

Таким чином, вибір мови програмування для створення скріншотів залежить від вимог до продуктивності, інтеграції з іншими системами, а також потреби в побудові інтерфейсу користувача. Кожна з популярних мов пропонує свій набір інструментів, що дозволяє ефективно розв'язувати завдання захоплення зображень з екрана .

Вибір мови програмування представлено у вигляді таблиці 2.1, виконавши порівняння за наступними критеріями: продуктивність, підтримка графічних бібліотек, легкість у використанні, гнучкість, кросплатформність, підтримка операційної системи.

Ці критерії були обрані, оскільки вони безпосередньо впливають на ефективність і зручність розробки програми. Продуктивність важлива для швидкого захоплення зображень, підтримка графічних бібліотек дозволяє працювати із зображеннями, а інтеграція з ОС гарантує стабільність роботи програми. Легкість у використанні і гнучкість дозволяють швидко і без зайвих труднощів створювати функціонал, а кросплатформність є плюсом для розширення програми.

Таблиця 2.1 – Порівняння мов програмування

Критерії	C#	Python	Java
1	2	3	4
Продуктивність	Компіляція в рідний код для забезпечення швидкого виконання та високої продуктивності	Інтерпретована мова, але наявні бібліотеки, що оптимізують обчислення (наприклад, NumPy).	Виконання через JVM, що додає додатковий рівень абстракції, знижуючи продуктивність.
Підтримка графічних бібліотек	Є бібліотеки для роботи з графікою (WPF, .NET Graphics), що дають змогу зручно створювати інтерфейси та працювати з екраном.	Наявна множина бібліотек для обробки знімків (Pillow, OpenCV PyAutoGUI).	У наявності є бібліотеки графіки (AWT, Swing, JavaFX), але для складних задач може знадобитись додатковий код.
Підтримка ОС	Інтеграція з Windows API, що дозволяє легко працювати з екраном і системними компонентами.	Підтримка багатьох ОС, але для Windows потребує специфічних бібліотек.	Підтримка різних платформ завдяки JVM, але взаємодія з ОС потребує додаткових бібліотек.
Легкість у використанні	Зрозуміла і чітка синтаксична структура, з потужними інструментами для розробки та налагодження	Проста, зрозуміла синтаксична структура, яка робить мову дуже доступною для новачків	Мова менш гнучка для початківців через більш складний синтаксис та налаштування.
Гнучкість	Легка інтеграція з бібліотеками та системами, що дозволяє створювати адаптивні рішення для захоплення зображень.	Висока гнучкість завдяки великій кількості різних сторонніх бібліотек і модулів для специфічних завдань.	Добре підтримує розробку великих застосунків, але інтеграція з іншими системами може бути більш складною.

Продовження таблиці 2.1

1	2	3	4
Сумісність з платформами	Підтримка через .NET Core для Linux і macOS, але з деякими обмеженнями в порівнянні з іншими мовами	Природна наявність роботи на всіх основних операційних системах.	Java спочатку була розроблена для багатьох платформ і підтримує всі основні операційні системи через JVM.

На основі критеріїв та факторів, зазначених у таблиці, C# є найкращим вибором для розробки програми, орієнтованої на захоплення скріншотів.

По-перше, C# є основною мовою для розробки на платформі Windows Forms та WPF, що забезпечує безпосередній доступ до системних API Windows та дозволяє ефективно працювати з графічними елементами. Завдяки бібліотеці GDI+ [13], яка є частиною .NET, мова забезпечує швидку та зручну візуалізацію графічних зображень, що є важливим для роботи з екраном.

По-друге, використання C# у зв'язці з .NET Framework [14] дозволяє інтегрувати різні інструменти для збереження, обробки та роботи із зображеннями. Вбудовані можливості для роботи з Bitmap, Graphics, Screen забезпечують високу продуктивність для створення і редагування скріншотів.

На відміну від Python, який має простий синтаксис та розвинені бібліотеки для роботи з графікою, C# забезпечує вищу продуктивність у середовищі Windows, оскільки має прямий доступ до WinAPI [15] – набору функцій, що надає ОС Windows, для роботи з графічним інтерфейсом, файлами, процесами, пам'яттю тощо. Це дозволяє виконувати захоплення зображень та їх обробку швидше, ніж у скриптових мовах.

Також важливо зазначити, що, на відміну від Java, яка є кросплатформною мовою, але має менш ефективну інтеграцію з Windows API для графіки, C# надає доступ до системних функцій операційної системи без використання сторонніх бібліотек, складних налаштувань, що дозволяє легко створювати графічний інтерфейс, оптимізований під захоплення зображень.

2.2 Вибір середовища розробки

Вибір середовища розробки є важливим етапом при створенні програмного забезпечення, особливо для таких завдань, як захоплення зображень з екрана. Правильне IDE повинно забезпечувати високу продуктивність, зручність інтеграції з різними мовами програмування, а також ефективний доступ до графічних ресурсів. На ринку існує кілька популярних IDE, наприклад Microsoft Visual Studio [16], JetBrains Rider [17], Visual Studio Code [18]. Кожне з цих середовищ має свої переваги та обмеження, що впливають на вибір середовища для реалізації конкретних функцій програми. Оскільки розробка програм для захоплення зображень з екрана вимагає точності та високої швидкості обробки, вибір необхідного IDE стає ключовим для успішної реалізації проєкту.

Вибір середовища розробки представлено у вигляді таблиці 2.2, виконавши порівняння за наступними критеріями: продуктивність, наявність інструментів для налагодження та тестування, підтримка зовнішніх бібліотек та фреймворків, кросплатформність, зручність у використанні.

Ці критерії були обрані, оскільки вони безпосередньо впливають на ефективність і зручність розробки програми для створення скріншотів. Продуктивність важлива для швидкого захоплення зображень без затримок. Підтримка графічних бібліотек дозволяє працювати з обробкою та збереженням зображень у різних форматах. Інтеграція з операційною системою гарантує стабільність роботи програми, можливість роботи з API Windows або інших платформ. Легкість у використанні спрощує процес розробки, а гнучкість дозволяє легко розширювати функціонал без значних змін у коді. Кросплатформність є додатковою перевагою для розширення можливостей програми на різні операційні системи.

Таблиця 2.2 – Порівняння середовищ

Критерії	Microsoft Visual Studio	JetBrains Rider	VS Code
Продуктивність	Підтримка великих проєктів, швидка компіляція	Трохи повільна підтримка під час роботи з великими проєктами	Легке та швидке IDE, але інколи треба додаткові налаштування
Інструменти для налагодження	Потужний відлагоджувач, інтеграція з профайлерами та іншими інструментами	Вбудовані інструменти для потрібного налагодження, профілювання	Підтримка через розширення не така глибока, як у повноцінних IDE
Підтримка бібліотек та фреймворків	Легка інтеграція .NET бібліотеки, сторонніх пакетів через NuGet	Добре інтегрується з бібліотеками через JetBrains репозиторії	Підтримка .NET із розширенням, але потребує детальне ручне налаштування
Кросплатформність	Підтримує Windows, macOS, Linux	Підтримує Windows, macOS, Linux	Підтримує Windows, macOS, Linux
Зручність користування	Інтуїтивно зрозумілий інтерфейс, наявні потужні, сучасні інструменти для розробки, велика кількість документації	Легкий у використанні, інтерфейс зручний для роботи	Мінімалістичний інтерфейс, але потребує багато додаткових плагінів для роботи

На основі порівняння критеріїв, Microsoft Visual Studio є найкращим вибором для розробки програмного продукту. Завдяки високому рівню продуктивності, потужним інструментам для налагодження, гнучкості в підтримці різноманітних мов програмування та фреймворків, а також великій кількості бібліотек і інтеграцій, це IDE забезпечує оптимальне середовище для розробки. Воно також має великий набір функцій для роботи з великими проєктами, що робить його ефективним для задач, які потребують точності та швидкості, наприклад при створенні програм для захоплення скріншотів.

2.3 Використані бібліотеки

Для реалізації основного функціоналу програми використані наступні бібліотеки:

- `System.Drawing` [19] – бібліотека базових засобів для роботи з графікою (малювання ліній, прямокутників, еліпсів), редагування зображень, роботи з кольорами, кистями та шифрами;
- `System.Drawing.Drawing2D` [20] – бібліотека, що надає додатковий набір інструментів для малювання складних об'єктів, визначаючи стилі обведення, роботу з градієнтами, трансформаціями, створення фігур;
- `System.Threading` [21] – бібліотека, що надає інструменти для роботи з потоками у багатозадачних застосунках, управління їхньої синхронізації, затримок у виконанні;
- `System.Threading.Tasks` [22] – це бібліотека, яке спрощує багатопотокове програмування, зосереджуючись на завданнях. Система автоматично розподіляє завдання між потоками, оптимізуючи продуктивність, дозволяє легко організовувати паралельне виконання;
- `System.Runtime.InteropServices` [23] – бібліотека для взаємодії з операційною системою через виклики до зовнішніх бібліотек (.dll), що мають функціонал для роботи з вікнами, щоб отримати інформацію про запущені застосунки, зміну позиції вікон, координати відкритих програм, отримання заголовка вікон, перелічення усіх відкритих тощо;
- `System.Windows.Forms` [24] – це бібліотека головного простору створення GUI-застосунків для створення віконних форм, роботи з кнопками, текстовими полями та зображеннями, надає можливість оброблювати події;
- `System.Drawing.Imaging` [25] – це бібліотека для роботи із зображеннями на глибшому рівні, включаючи формати файлів та властивості зображень із заданими параметрами, обробка пікселів через `Bitmap`.

Особливої уваги потребують використані у програмі бібліотеки Accord.Video.FFmpeg та NAudio.Wave, що надають можливість для запису файлів екрана та звуку, які згодом об'єднуються у відео з аудіо.

NAudio.Wave [26] – бібліотека, що необхідна для роботи з аудіофайлами, зокрема для запису, обробки та відтворення звуку у різних форматах, таких як .MP3, .WAV тощо. Вона є частиною NAudio – потужного аудіофреймворку для .NET, що забезпечує інтеграцію з широким спектром аудіоінструментів та API.

Однією з основних можливостей бібліотеки є захоплення та обробка звуку в режимі реального часу, а також робота з аудіофайлами.

Інша бібліотека, Accord.Video.FFmpeg [27], необхідна для роботи з відеофайлами, зокрема для запису та збереження відео у різних форматах, таких як .mp4, .avi тощо. Вона є частиною Accord.NET Framework і забезпечує інтеграцію з потужним мультимедійним інструментом FFmpeg, що дозволяє працювати з відео та аудіо даними. Однією з основних можливостей бібліотеки є створення відеофайлів на основі послідовності зображень, вказавши для цього кількість захоплень зображення у секунду (FPS). Це дає змогу формувати відеопотік із серії кадрів, що корисно для запису екрана або роботи з анімованими даними.

Процес вибору і розробки графічного користувацького інтерфейсу відіграє важливу роль під час проєктування програмного застосунку, адже це безпосередньо впливає на якість взаємодії користувача із системою, і чим зрозуміліше, привабливіше та привабливіше інтерфейс, тим більше користувач задоволений досвідом роботи з програмою.

C# має два види графічного інтерфейсу користувача (GUI), кожен з яких має власні недоліки та переваги, це WinForms [28] та WPF [29]. Варто розглянути порівняння цих структур за такими параметрами: графічна система, продуктивність, гнучкість UI, простота розробки, робота із зображеннями, сумісність, складність інтеграції зі сторонніми бібліотеками. Порівняння графічних інтерфейсів представлено у вигляді таблиці 2.3.

Таблиця 2.3 – Порівняння графічних інтерфейсів користувача

Параметр	WPF	WinForms
Графічна система	DirectX	GDI+
Продуктивність	Краще для складних UI (прискорене GPU)	Повільніша робота на складних сценах
Простота розробки	Складність освоїти через XAML, MVVM	Примітивні і легкі для засвоєння UI-елементи
Робота із зображеннями	Є кращим варіантом для складної графіки	Ідеальна взаємодія для простих дій з графікою
Сумісність із системами	Працює тільки на нових ОС Windows	Підтримка включає в себе старі версії ОС Windows
Складність інтеграції зі сторонніми бібліотеками	Складність інтеграції з GDI+, FFmpeg, DirectShow	Легкість інтеграції з FFmpeg, NAudio тощо

На основі отриманих даних таблиці можна сформулювати причини, чому саме WinForms є кращим варіантом для розробки проєкту:

Легка інтеграція з GDI+ (Graphics, Drawing):

- WinForms використовує GDI+, що дозволяє легко працювати з графікою (малювання, редагування, збереження скріншотів);
- у WPF реалізація складніша, через необхідність працювати з WriteableBitmap або DrawingVisual, що ускладнює код.

Прямий доступ до Windows API, інтеграція зі сторонніми бібліотеками:

- для запису екрана та звуку часто використовуються API Windows (User32, GDI32, DirectShow), у WinForms легко викликати ці API через P/Invoke;
- у WPF інтеграція є більш ускладненою через орієнтацію на високорівневі UI;
- FFmpeg, NAudio та інші бібліотеки для запису відео та звуку легко інтегруються у WinForms;
- у WPF потрібно створювати обгортки для DirectShow або MediaElement, що складніше у реалізації.

Мінімальні системні вимоги:

- WinForms працює навіть на слабких ПК і старих ОС;
- WPF вимагає сучасну операційну систему (Windows 8+) і потужну відеокарту для продуктивної та плавної роботи користувача.

Легкість розробки та підтримки:

- WinForms має простий та зрозумілий API, не вимагає XAML та складних MVVM-патернів;
- чим менше застосовано коду, тим є швидшою розробка, саме тому WinForms має перевагу над WPF для реалізація нескладних програм;
- зручна робота з класичними Windows-формами та елементами UI.

2.4 Структурна схема розробки

Загальна схема системи за обраною темою на рисунку 2.1.

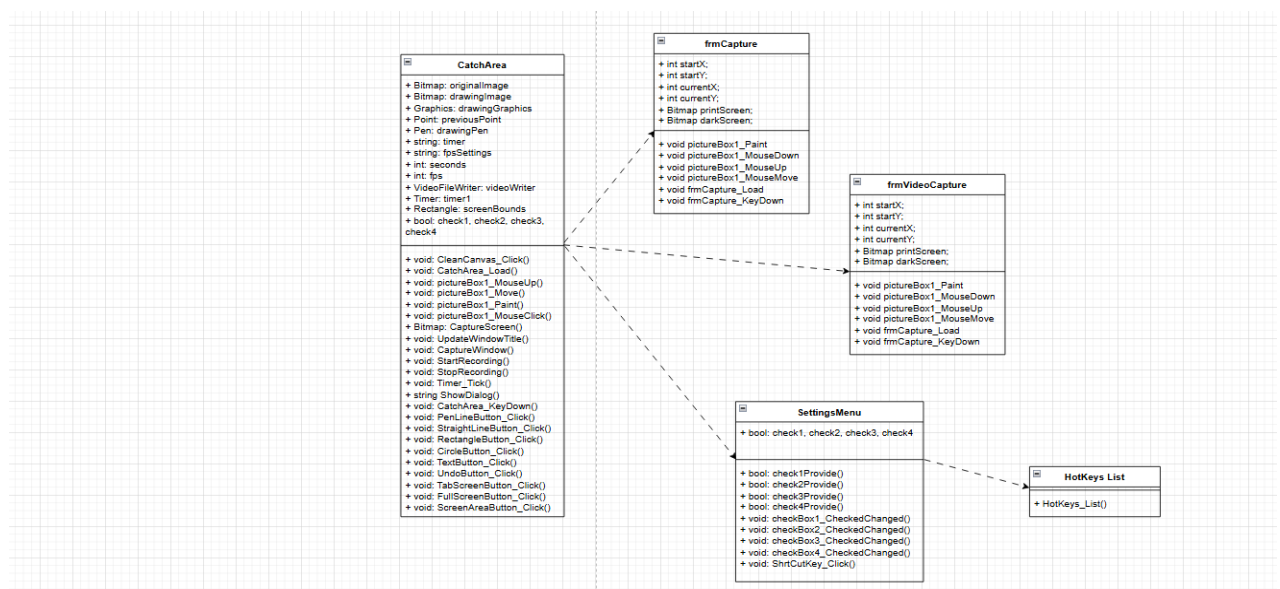


Рисунок 2.1 – Загальна схема системи за обраною темою

2.5 Висновки до розділу

У даному розділі було описано використані матеріали, такі як мова програмування C# та IDE Microsoft Visual Studio, вбудовані та встановлені бібліотеки, вибір графічного інтерфейсу користувача під час розробки програмного продукту, причини їхнього використання, а також наявні класи, застосовані під час роботи із застосунком, і їхні зв'язки у вигляді відповідної структурної схеми.

3 ОСНОВНІ РІШЕННЯ ЩОДО РЕАЛІЗАЦІЇ КОМПОНЕНТІВ СИСТЕМИ

У цьому розділі розглядаються ключові рішення, ухвалені під час розробки системи, які визначають її архітектуру, функціональність та взаємодію між компонентами. Реалізація програмного продукту базується на модульному підході, що забезпечує гнучкість, масштабованість і спрощує подальшу підтримку. Також приділено увагу вибору алгоритмів для забезпечення ефективності роботи системи та розробці графічного інтерфейсу для зручності користувачів.

3.1 Алгоритми функціонування розробленої програми

Схема послідовності дій та діаграма use-case роботи застосунку у розробці зображені на рисунках 3.1, 3.2.

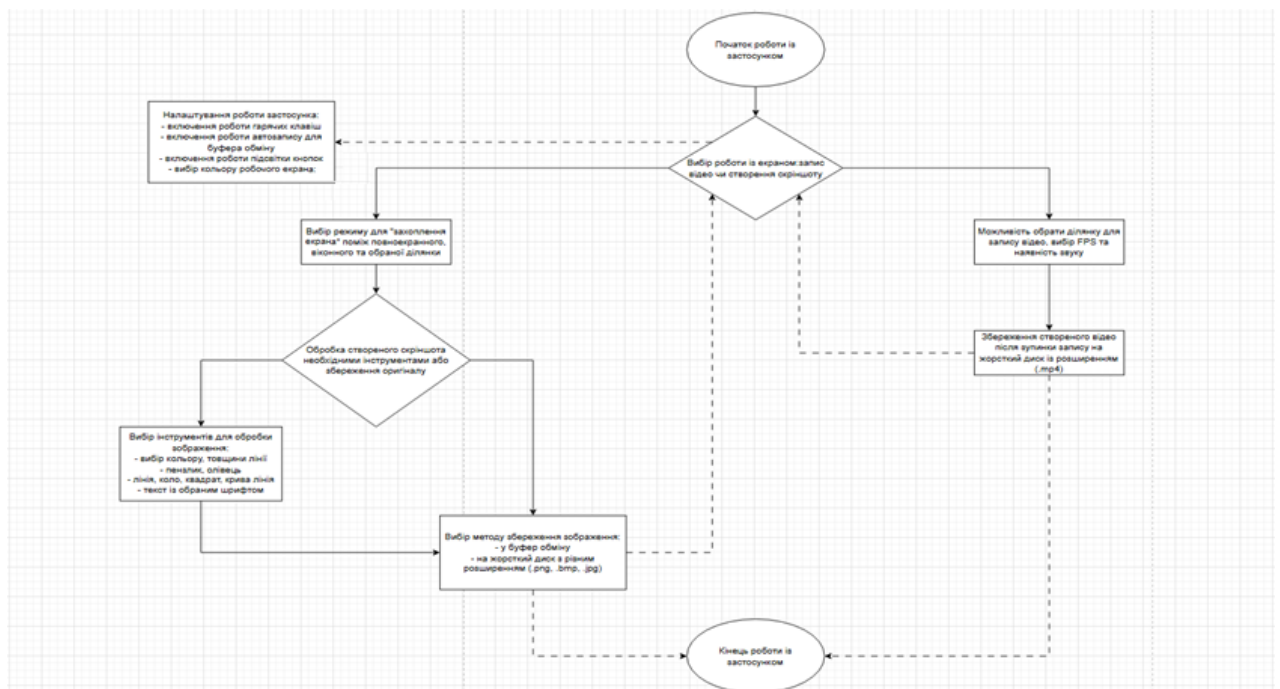


Рисунок 3.1 – Схема послідовності дій розробленої програми

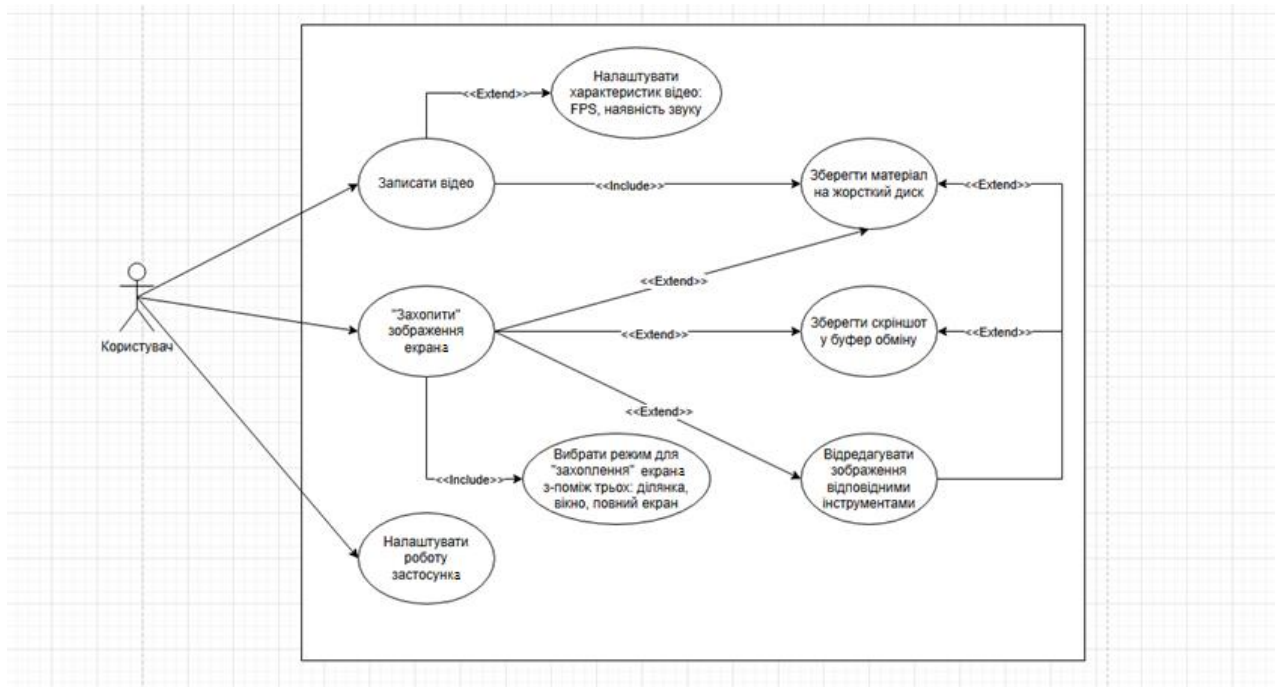


Рисунок 3.2 – Діаграма use-case роботи застосунку

Під час розробки проєкту для захоплення зображення з екрана, необхідно було розробити функціонал, який би відповідав функціональним вимогам, тому це стало запорукою реалізації наступних алгоритмів.

3.1.1 Алгоритм виділення ділянки зображення на екрані

Реалізація у програмі можливості для користувача обрати режим виділення ділянки зображення на екрані для подальшої роботи з ним.

Необхідні кроки представлені далі.

Крок 1. Користувач обирає будь-яку точку на екрані за допомогою курсора та натискає ліву кнопку миші.

Крок 2. Після натискання лівої кнопки миші необхідно вести мишу вздовж тієї ділянки, яку потрібно виділити, утворюючи відповідний прямокутник.

Крок 3. Сформувавши необхідну ділянку, користувач відпускає ліву кнопку миші. Програма запам'ятає координати та розташування ділянки, а також наявну графічну інформацію і передасть її до основного вікна.

Код алгоритму наведений нижче:

```
private void pictureBox1_MouseDown(object sender, MouseEventArgs e){
    if (e.Button == MouseButtons.Left){
        startX = e.X;
        startY = e.Y;
        currentX = e.X;
        currentY = e.Y;
        start = true;
    }
}

private void pictureBox1_MouseUp(object sender, MouseEventArgs e){
    if (e.Button == MouseButtons.Left){
        start = false;
        int width = Math.Abs(currentX - startX);
        int height = Math.Abs(currentY - startY);
        int x = Math.Min(startX, currentX);
        int y = Math.Min(startY, currentY);
        if (width > 0 && height > 0){
            Rectangle selection = new Rectangle(x, y, width, height);
            Bitmap selectedArea = new Bitmap(selection.Width, selection.Height);
            using (Graphics g = Graphics.FromImage(selectedArea)){
                g.DrawImage(printScreen, new Rectangle(0, 0, selectedArea.Width,
                    selectedArea.Height), selection, GraphicsUnit.Pixel);
            }
            Clipboard.SetImage(selectedArea);
        }
        this.Close();
    }
}

private void pictureBox1_MouseMove(object sender, MouseEventArgs e)
{
    if (start)
    {
        currentX = e.X;
        currentY = e.Y;
        pictureBox1.Invalidate(); // Redraw the picture box
    }
}
```

3.1.2 Алгоритм малювання на екрані за допомогою інструментів

Реалізація у програмі можливості обрати необхідний інструмент для обробки і редагування зображення.

Необхідні кроки представлені далі.

Крок 1. Користувач обирає будь-який інструмент для роботи із зображенням (коло, лінія, прямокутник), налаштовує товщину і колір лінії.

Крок 2. Користувач обирає будь-яку точку на екрані за допомогою курсора та натискає ліву кнопку миші.

Крок 3. Після натискання лівої кнопки миші необхідно вести мишу або вздовж тієї ділянки, де необхідно намалювати геометричну фігуру у вигляді кола чи прямокутника, або малювати пряму чи «вільну» лінію за власним бажанням у будь-якому напрямку.

Крок 4. Намалювавши необхідну фігуру чи лінії, користувач відпускає ліву кнопку миші. Програма запам'ятає координати та розташування створених малюнків, відтворить це у головному вікні та передасть до історії редагування зображення.

Код алгоритму наведений нижче:

```
private void pictureBox1_MouseUp(object sender, MouseEventArgs e)
{
    if (e.Button == MouseButton.Left){
        if (drawingGraphics != null && currentRectangle != Rectangle.Empty &&
isDrawingRectangle)
        {
            // Draw the final rectangle on the drawing surface
            drawingGraphics.DrawRectangle(drawingPen, currentRectangle);
            pictureBox1.Image = drawingImage;
            // Reset the rectangle
            currentRectangle = Rectangle.Empty;
        }
        else if (drawingGraphics != null && currentCircle != Rectangle.Empty &&
isDrawingCircle)
        {
            // Draw the final circle on the drawing surface
            drawingGraphics.DrawEllipse(drawingPen, currentCircle);
            pictureBox1.Image = drawingImage;
            currentCircle = Rectangle.Empty;
        }
        else if (drawingGraphics != null && isDrawingLine)
        {
            drawingGraphics.DrawLine(drawingPen, startPoint1, endPoint1);
            pictureBox1.Image = drawingImage;
            startPoint1 = Point.Empty;
            endPoint1 = Point.Empty;
        }

        if (pictureBox1.Image != null)
        {
            history.Push(new Bitmap(pictureBox1.Image));

            if (checkboxbutton)
                Clipboard.SetImage(pictureBox1.Image);
            previousPoint = Point.Empty;
        }
        else{
            MessageBox.Show("No image to work with");
        }
    }
}
```

```

private void pictureBox1_MouseDown(object sender, MouseEventArgs e)
{
    if (e.Button == MouseButtons.Left && isDrawingRectangle)
    {
        previousPoint = e.Location;
        currentRectangle = new Rectangle(e.Location, new Size(0, 0));
    }
    else if (e.Button == MouseButtons.Left && isDrawingCircle)
    {
        previousPoint = e.Location;
        currentCircle = new Rectangle(e.Location, new Size(0, 0));
    }
    else if (e.Button == MouseButtons.Left && isDrawingLine)
    {
        startPoint1 = e.Location;
        endPoint1 = e.Location;
    }
    else if (e.Button == MouseButtons.Left && isTyping)
    {
        previousPoint = e.Location;
    }
    else
        previousPoint = e.Location;
}

Stack<Image> history = new Stack<Image>();

private void pictureBox1_MouseMove(object sender, MouseEventArgs e)
{
    if (e.Button == MouseButtons.Left && drawingGraphics != null)
    {
        if (currentRectangle != null && isDrawingRectangle)
        {
            // Determine the rectangle's top-left corner and its size based on the mouse
            position
            int x = Math.Min(previousPoint.X, e.X);
            int y = Math.Min(previousPoint.Y, e.Y);
            int width = Math.Abs(e.X - previousPoint.X);
            int height = Math.Abs(e.Y - previousPoint.Y);

            // Update the rectangle's location and size
            currentRectangle = new Rectangle(x, y, width, height);

            pictureBox1.Invalidate();
        }
        else if (isDrawingCircle && currentCircle != null)
        {
            previousPoint.Y));
            int diameter = Math.Max(Math.Abs(e.X - previousPoint.X), Math.Abs(e.Y -
            int x = previousPoint.X - diameter / 2;
            int y = previousPoint.Y - diameter / 2;

            currentCircle = new Rectangle(x, y, diameter, diameter);

            pictureBox1.Invalidate();
        }
        else if (isDrawingLine)
        {
            endPoint1 = e.Location;
            pictureBox1.Invalidate(); // Redraw to update the line being drawn
        }
    }
}

```

```

else if(isTyping)
{
    endPoint1 = e.Location;
    pictureBox1.Invalidate(); // Redraw to update the line being drawn

    if (check2){

        TextBox.BackColor = System.Drawing.Color.LightCyan;
        CircleButton.BackColor = System.Drawing.Color.WhiteSmoke;
        RectangleButton.BackColor = System.Drawing.Color.WhiteSmoke;
        StraightLineButton.BackColor = System.Drawing.Color.WhiteSmoke;
        PenLineButton.BackColor = System.Drawing.Color.WhiteSmoke;

    }

}

else
{
    drawingGraphics.DrawLine(drawingPen, previousPoint, e.Location);

    pictureBox1.Image = drawingImage;
    previousPoint = e.Location;
}

}
}

```

3.1.3 Алгоритм динамічної роботи з відкритими вікнами

Реалізація у програмі можливості динамічно оновлювати та додавати або видаляти відкриті вікна для відповідного списку, з якого користувач може обирати відповідну програму для захоплення зображення застосунку.

Необхідні кроки представлені далі.

Крок 1. Створення словника newWindows для збереження відкритих вікон і список openWindowHandles для збереження їхніх дескрипторів.

Крок 2. Використання EnumWindows для отримання всіх відкритих вікон з подальшою перевіркою, чи є вікно видимим, отримавши його заголовок і відфільтрувавши зайві вікна. Якщо вікно нове – треба додати його до списку comboBoxWindows, якщо його назва змінилася – оновити.

Крок 3. Відбувається процес видалення закритих вікон зі списку comboBoxWindows, якщо їх більше немає у списку відкритих вікон.

Крок 4. Оновлення списку existingWindows, зберігаючи актуальні відкриті вікна.

Код алгоритму наведений нижче:

```
private void LoadOpenWindows()
{
    Dictionary<IntPtr, string> newWindows = new Dictionary<IntPtr, string>();
    List<IntPtr> openWindowHandles = new List<IntPtr>();

    EnumWindows(new EnumWindowsProc((hWnd, lParam) =>
    {
        if (IsWindowVisible(hWnd)){
            int length = GetWindowTextLength(hWnd);
            if (length > 0){
                var windowTitle = new System.Text.StringBuilder(length + 1);
                GetWindowText(hWnd, windowTitle, windowTitle.Capacity);

                string title = windowTitle.ToString();

                // Фільтрація непотрібних вікон
                if (title != "Program Manager" && title != "Параметры" &&
                    title != "Microsoft Text Input Application" && title != "Application")
                {
                    openWindowHandles.Add(hWnd);
                    newWindows[hWnd] = title;

                    if (!existingWindows.ContainsKey(hWnd)){
                        // Додаємо нове вікно, якщо його не було
                        comboBoxWindows.Items.Add(new WindowItem { Handle = hWnd, Title =
title });
                    }
                    else if (existingWindows[hWnd] != title){
                        // Якщо назва вікна змінилася, оновлюємо її в списку
                        UpdateWindowTitle(hWnd, title);
                    }
                }
            }
        }
        return true;
    }), IntPtr.Zero);
    // Видаляємо закриті вікна
    for (int i = comboBoxWindows.Items.Count - 1; i >= 0; i--){
        WindowItem item = (WindowItem)comboBoxWindows.Items[i];
        if (!openWindowHandles.Contains(item.Handle)){
            comboBoxWindows.Items.RemoveAt(i);
            existingWindows.Remove(item.Handle);
        }
    }
    existingWindows = newWindows;
    comboBoxWindows.DisplayMember = "Title";
}
```

3.2 Опис модулів розроблюваної системи

Для реалізації програми, було використано класи та форми для побудови алгоритмів, методів та зазначення необхідних змінних.

3.2.1 Клас frmCapture

Клас, який відповідає за виділення ділянки на екрані для того, щоб створити відповідний знімок на необхідній ділянці (за розміром і координатами). Він містить у собі методи, які оброблюють події миші, це необхідно для того, щоб зрозуміти, звідки користувач починає свою роботу, яку ділянку захоплює, де він зупинився. Дані класу наведені в таблиці 3.1.

Таблиця 3.1 – Опис полів і методів класу frmCapture

Поля і методи класу	Опис
Перший стовпець	Другий стовпець
1	2
Private:	
int startX	Початкова точка на осі X для виділення ділянки прямокутної форми
int startY	Початкова точка на осі Y для виділення ділянки прямокутної форми
int currentX	Поточна точка на осі X для виділення ділянки прямокутної форми
int currentY	Поточна точка на осі Y для виділення ділянки прямокутної форми
Bitmap printScreen	Змінна для запису виділеного зображення
Bitmap darkScreen	Змінна для затемнення зображення екрана
void pictureBox1_MouseDown	Метод, що визначає початкові значення по осі X та Y для початку виділення ділянки
void pictureBox1_Paint	Метод, що визначає значення ширини та довжини виділеного прямокутника, налаштовує колір та ширину динамічної лінії виділення ділянки зображення на екрані, висвітлює зображення всередині обраної ділянки
void pictureBox1_MouseUp	Метод, що завершує процес виділення ділянки, формує відповідне зображення за отриманими координатами та зберігає для подальшої роботи
void pictureBox1_MouseMove	Метод, що динамічно оновлює значення координати поточної точки за осями X та Y й змінює розташування виділеної ділянки
void frmCapture_KeyDown	Метод, що реагує на натискання клавіші Esc, щоб покинути режим захоплення екрана

3.2.2 Клас SettingsMenu

Клас, який відповідає за роботу форми з налаштувань, яка за допомогою відповідних «прапорців» оптимізує можливості та зовнішній вигляд застосунку. Дані класу наведені в таблиці 3.2.

Таблиця 3.2 – Опис полів і методів класу SettingsMenu

Поля і методи класу	Опис
Перший стовпець	Другий стовпець
1	2
Private:	
bool check1	Логічна змінна, що відповідає за наявність повідомлення перед закриттям застосунку
bool check2	Логічна змінна, що відповідає за наявність виділення активних кнопок інструментів
bool check3	Логічна змінна, що відповідає за підключення роботи комбінації гарячих клавіш
bool check4	Логічна змінна, що оптимізує робочий простір, вмикаючи або світлий, або темний режим
bool check5	Логічна змінна, що відповідає за наявність автоматичного копіювання останньої версії зображення до буфера обміну
bool check1Provide()	Метод, що передає значення змінної check1
bool check2Provide()	Метод, що передає значення змінної check2
bool check3Provide()	Метод, що передає значення змінної check3
bool check4Provide()	Метод, що передає значення змінної check4
bool check5Provide()	Метод, що передає значення змінної check5

3.2.3 Клас CatchArea

Головний клас і компонент всієї програми, який працює з головною формою застосунку, містить відповідні змінні та методи, з'єднаний з другорядними формами для забезпечення відповідного функціонування.

Наприклад, клас CatchArea містить логіку обробки і збереження зображення за допомогою відповідних інструментів, вбудовані методи з принципами запису відео, створення скріншотів у різних режимах.

Локальні змінні допомагають працювати з іншим функціоналом, використовуючи отримані дані для функціонування системи, як FPS чи секунди затримки роботи програми перед захопленням зображення. Дані класу наведені в таблиці 3.3.

Таблиця 3.3 – Опис полів і методів класу CatchArea

Поля і методи класу	Опис
Перший стовпець	Другий стовпець
1	2
Private:	
Bitmap originalImage	Початкове зображення створеного скріншоту
Bitmap drawingImage	Версія створеного скріншоту для подальшого його редагування і опрацювання користувачем
Pen drawingPen	Олівець для редагування зображення, з наявною можливістю зміни його кольору, товщини лінії
String timer	Рядок для конвертації обраних секунд користувачем для затримки у числове значення
String fpsSettings	Рядок для конвертації обраних кадрів у секунду користувачем для запису відео у значення числа
Int width	Встановлене значення числа для ширини лінії олівця або пензля для обробки зображення
VideoFileWriter videoWriter	Змінна бібліотеки Accord.Video.FFmpeg для запису відео з екрана користувача
WaveInEvent waveIn	Змінна бібліотеки NAudio.Wave для запису звуку користувача і подальшим об'єднанням з відео
void CleanCanvas_Click	Метод для повного очищення отриманого зображення після обробки користувачем
void CaptureWindow	Метод для захоплення зображення обраного користувачем відкритого вікна
void StartRecording	Метод початку запису відео
void StopRecording	Метод закінчення запису відео
void MergeAudioVideo	Метод об'єднання файлів записаного звуку та відео у єдине ціле
string ShowDialog	Метод для налаштування стилю та розміру шрифту користувачем
void CatchAreaKeyDown	Метод для обробки натиснутих комбінацій гарячих клавіш

Продовження таблиці 3.3

1	2
void AreaRecord_Click	Метод для переходу на іншу форму, щоб зробити «контур ділянки екрана» для запису відео
void PenLine_Click	Метод вибору інструмента олівця/пензля
void StraightLine_Click	Метод вибору інструмента прямої лінії
void Rectangle_Click	Метод вибору інструмента прямокутника
void Circle_Click	Метод вибору інструмента кола
void Text_Click	Метод вибору інструмента для запису тексту
void TxtFnt_Click	Метод налаштування розмір та стиль тексту
void Undo_Click	Метод для відміни останньої зміни під час обробки зображення
void TabScreen_Click	Метод для створення скріншоту відкритого вікна
void FullScreen_Click	Метод для створення скріншоту повного екрана
void ScreenArea_Click	Метод для створення скріншоту виділеної ділянки екрана користувачем

3.3 Проєктування графічного інтерфейсу

У сучасному програмному забезпеченні графічний інтерфейс користувача (GUI) відіграє ключову роль у забезпеченні ефективної та інтуїтивно зрозумілої взаємодії між користувачем і системою. Одним із важливих аспектів проєктування інтерфейсу є ергономіка – адаптація технічних засобів до фізичних та психологічних можливостей людини. Якісно і ергономічно спроектований інтерфейс сприяє зменшенню навантаження на користувача, підвищенню продуктивності та зниженню ймовірності помилок.

Під час виконання роботи для реалізації графічного інтерфейсу було використано технологію Windows.Forms – одну з найпоширеніших платформ для створення десктопних застосунків у середовищі .NET. WinForms забезпечує засоби швидкої побудови візуальних елементів (форм, кнопок, полів введення тощо) та дозволяє налаштовувати їхню поведінку і зовнішній вигляд. Серед переваг цієї технології варто відзначити простоту у використанні, підтримку подійно-орієнтованої моделі програмування, роботу із зображеннями, а саме: обробка, редагування за допомогою різноманітних інструментів, збереження тощо.

На початковому етапі було створено низку прототипів інтерфейсу, які дозволили оцінити загальну структуру застосунку, розташування основних елементів та логіку переходів між формами. Прототипи, зображені на рисунках 3.3, 3.4, стали основою для фінальної версії інтерфейсу, яка була вдосконалена з урахуванням ергономічних вимог та зворотного зв'язку.

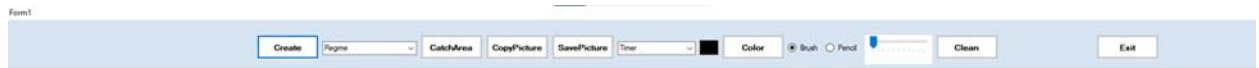


Рисунок 3.3 – Прототип початкової версії інтерфейсу програми з можливістю зробити знімок зображення екрана з подальшою мінімальною обробкою



Рисунок 3.4 – Прототип початкової версії інтерфейсу програми з покращеною обробкою зображення, режимами роботи зі скріншотами

Розроблений інтерфейс застосунку, зображений на рисунку 3.5, є зручним і інтуїтивно зрозумілим, забезпечуючи користувачеві швидкий доступ до основних функцій. Усі елементи керування мають чіткі підписи та згруповані за своїм призначенням у три основні секції: «Скріншот», «Запис» і «Малювання».

Схема кольорів інтерфейсу виконана в контрастних тонах: сіра панель, білі кнопки та чорні елементи для покращеної видимості. Крім того, наявність повзунка дозволяє регулювати товщину ліній у режимі малювання. Загалом, дизайн є функціональним, мінімалістичним і адаптованим для використання.



Рисунок 3.5 – Головна форма застосунку

4 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

4.1 Призначення й умови застосування програми

4.1.1 Призначення програми

Програма призначена для створення скріншоту різними режимами, налаштування та запису відео, редагування знімка за допомогою різних інструментів та його подальше збереження на пристрій або до буфера обміну для використання користувачем.

4.1.2 Функції програми

Програма виконує наступні функції:

- захоплення зображення з екрана (вікно програми, виділена ділянка, повноекранний режим);
- редагування зображення за допомогою олівця чи пензля, геометричних фігур (пряма та крива лінія, коло, квадрат, текст тощо);
- налаштування роботи програми за допомогою користувацького меню, де користувач має можливість підключити або відключити спеціальні додаткові можливості (автокопіювання зображення, зміна кольору теми, наявність комбінації гарячих клавіш тощо);
- редагування ширини стержня й кольору олівця чи пензля;
- поетапне чи повне очищення фотографії від елементів редагування;
- визначення за допомогою таймера часу очікування, після якого застосунок повинен зробити захоплення зображення;
- запис відео без та зі звуком, налаштування FPS, вибір ділянки на екрані для подальшого запису;
- використання комбінацій гарячих клавіш для роботи у застосунку;
- збереження зображення на комп'ютер або до буфера обміну.

4.1.3 Умови застосування програми

Призначення програми:

- запис відео з екрана, зняття скріншотів та малювання поверх створених зображень;
- особисте або професійне використання користувачем (створення навчальних матеріалів, презентацій, технічної документації тощо).

Дозволене використання:

- запис і редагування екранних матеріалів у рамках особистих і професійних потреб;
- використання для створення навчальних матеріалів, відеоуроків, демонстрацій.

Заборонене використання:

- незаконний запис відео чи створення фото, без згоди учасників;
- обхід систем захисту авторських прав;
- створення шкідливого контенту (фальсифікації або маніпуляції).

Обмеження функціональності:

- запис може бути обмежений залежно від апаратного забезпечення та налаштувань операційної системи;
- якість знімків і відеозапису може залежати від обраних параметрів (FPS, ділянка запису тощо);
- деякі функції можуть вимагати додаткових дозволів (наприклад, запис аудіо).

4.2 Вимоги до інтерфейсу

Основні вимоги до інтерфейсу включають такі аспекти:

- інтуїтивно зрозумілий та зручний інтерфейс, з чітко розташованими елементами, що забезпечує легкий доступ до основних функцій;
- мінімалістичний дизайн для уникнення перевантаження зайвими інструментами та елементами;
- візуальне зміна зображення кнопок активних інструментів;
- підтримка темної та світлої теми для зручності користувача;
- використання групування елементів за функціональністю для покращення навігації;
- наявність спливаючих підказок при наведенні на елементи.

4.3 Умови виконання програми

Для коректного запуску та функціонування програми необхідно дотримуватись таких умов:

Апаратура:

- процесор: мінімум 2-ядерний, рекомендується 4+ ядра для кращої продуктивності;
- оперативна пам'ять: мінімум 4 ГБ, рекомендується 8 ГБ+;
- відеокарта: інтегрована або дискретна (рекомендується підтримка OpenGL/DirectX для запису екрана).

Програмне забезпечення:

- операційна система: Windows 10/11 (можлива підтримка старіших версій або Linux/Mac, якщо реалізовано);
- драйвери: актуальні драйвери відеокарти та аудіосистеми;

- додаткові компоненти: може знадобитися .NET Framework, DirectX, бібліотеки для обробки відео.

Налаштування середовища:

- доступ до екрана для знімків і запису (може вимагати адміністраторських прав);
- доступ до мікрофона та системного звуку для запису аудіо;
- вільний простір на диску (мінімум 500 МБ для роботи, більше – для збереження відеофайлів).

Обмеження роботи:

- обмежена коректність в режимах енергозбереження;
- обмежена продуктивність на слабких ПК;
- під час запису великого відео програма може використовувати багато ресурсів системи.

4.4 Методика та результати тестування

Тестування програми проводилося вручну, без застосування модульних або автоматизованих тестів. Основна увага була зосереджена на перевірці функціональних, користувацьких та нефункціональних можливостей застосунку.

Метою тестування було оцінити коректність роботи основних функцій програми, її зручність для користувача, стабільність та продуктивність у різних сценаріях використання.

Під час тестування були виконані такі перевірки:

- функціональне тестування: оцінка коректності виконання знімків екрана, запису відео, збереження та обробки отриманих матеріалів, а також функцій малювання;

- користувацьке тестування: аналіз зручності інтерфейсу, інтуїтивності керування, відповідності очікуванням користувачів;
- нефункціональне тестування: перевірка продуктивності, навантаження на систему, стабільності роботи та сумісності з різними конфігураціями обладнання.

Головною задачею програми є захоплення зображення з екрана користувача для подальшої роботи з ним з можливістю зберегти на жорсткий диск чи до буфера обміну. Однією з можливостей користувача є вибір ділянки для захоплення з подальшою обробкою отриманого скріншоту. Процес створення, результат роботи та подальше збереження на жорсткому диску зображено на рисунках 4.1 – 4.3.

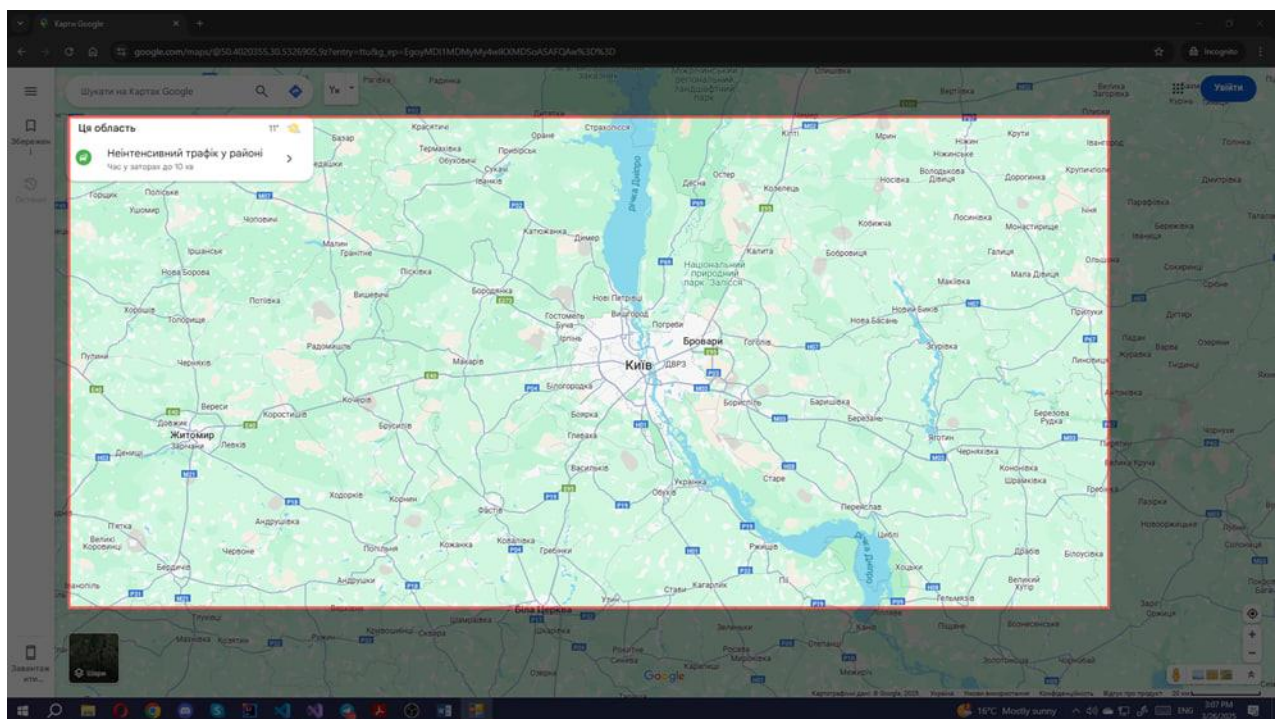


Рисунок 4.1 – Процес вибору ділянки для захоплення зображення з екрана

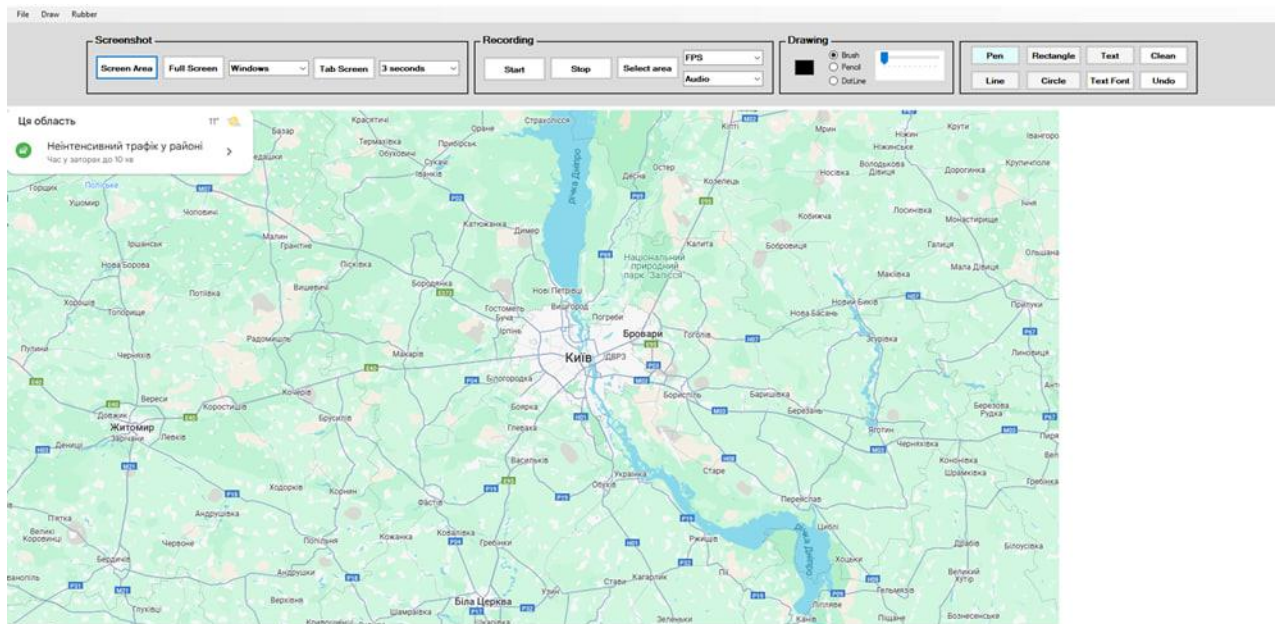


Рисунок 4.2 – Результат роботи застосунку: зображення необхідної форми

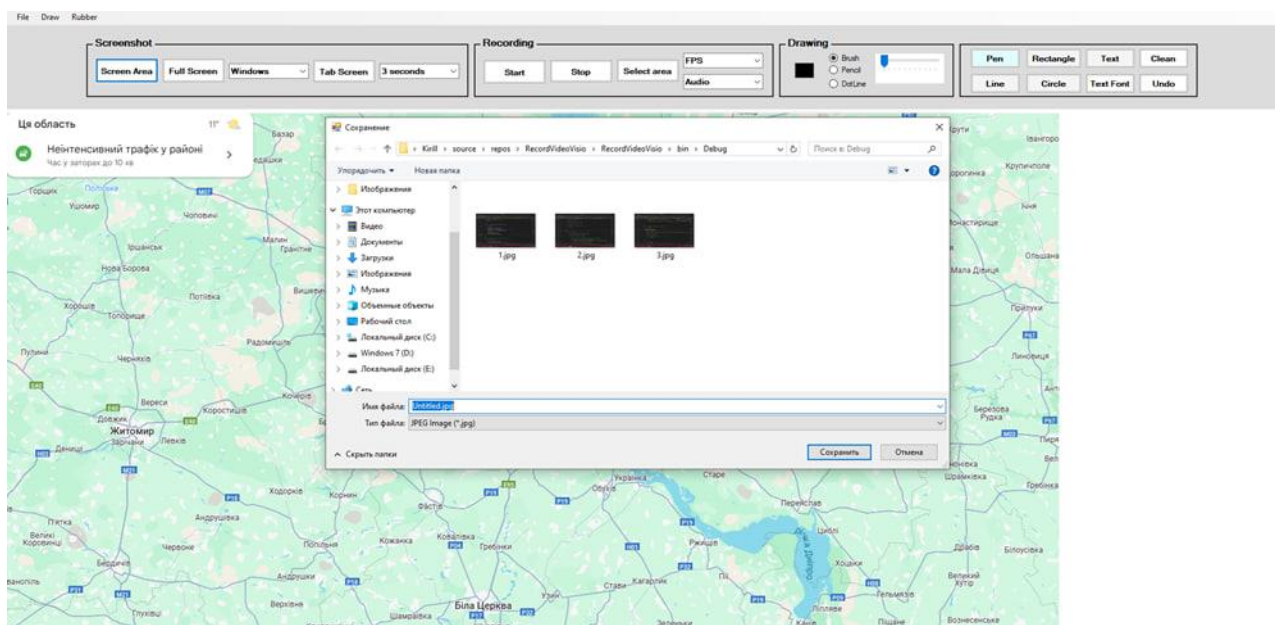


Рисунок 4.3 – Процес збереження зображення на жорсткий диск

Наступними можливостями програми є створення скріншотів за допомогою віконного та повноекранного режиму. Віконний режим динамічно аналізує всі відкриті вікна програми користувача, оновлюючи, додаючи та видаляючи перелік у списку тих програм, з форми яких можна створити зображення. Результат роботи двох режимів зображено на рисунках 4.4, 4.5.

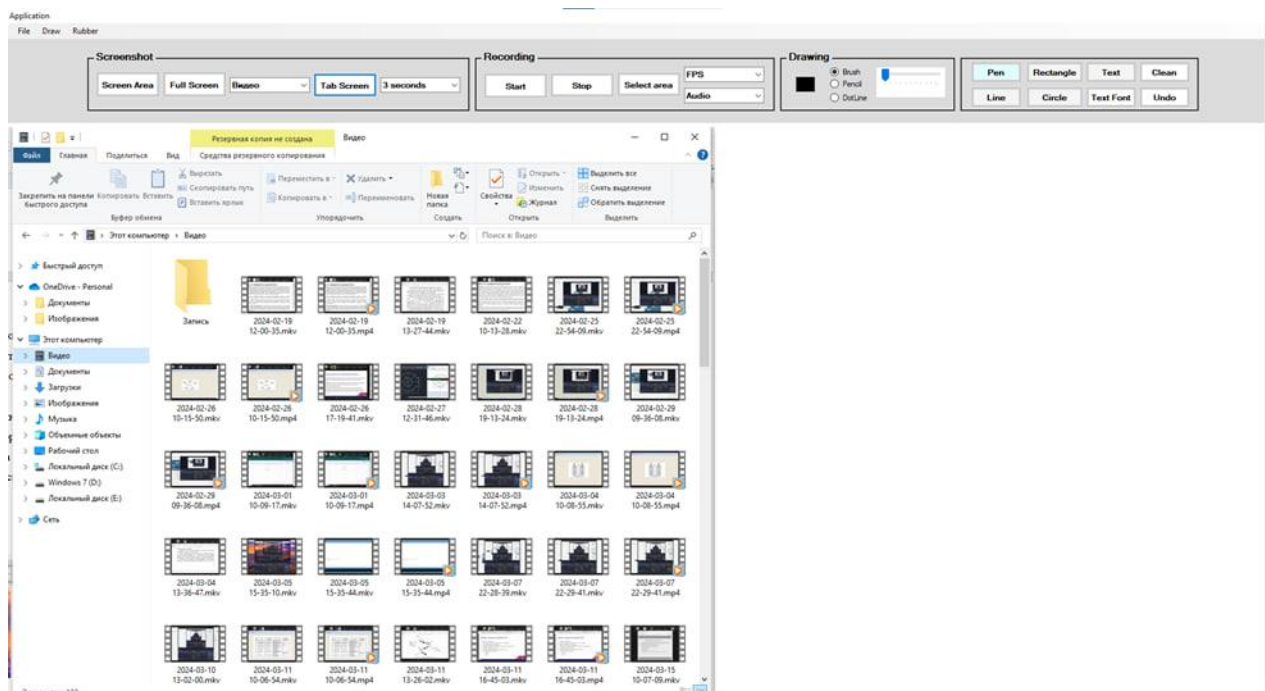


Рисунок 4.4 – Результат роботи віконного режиму застосунку: зображення програми у відповідній формі

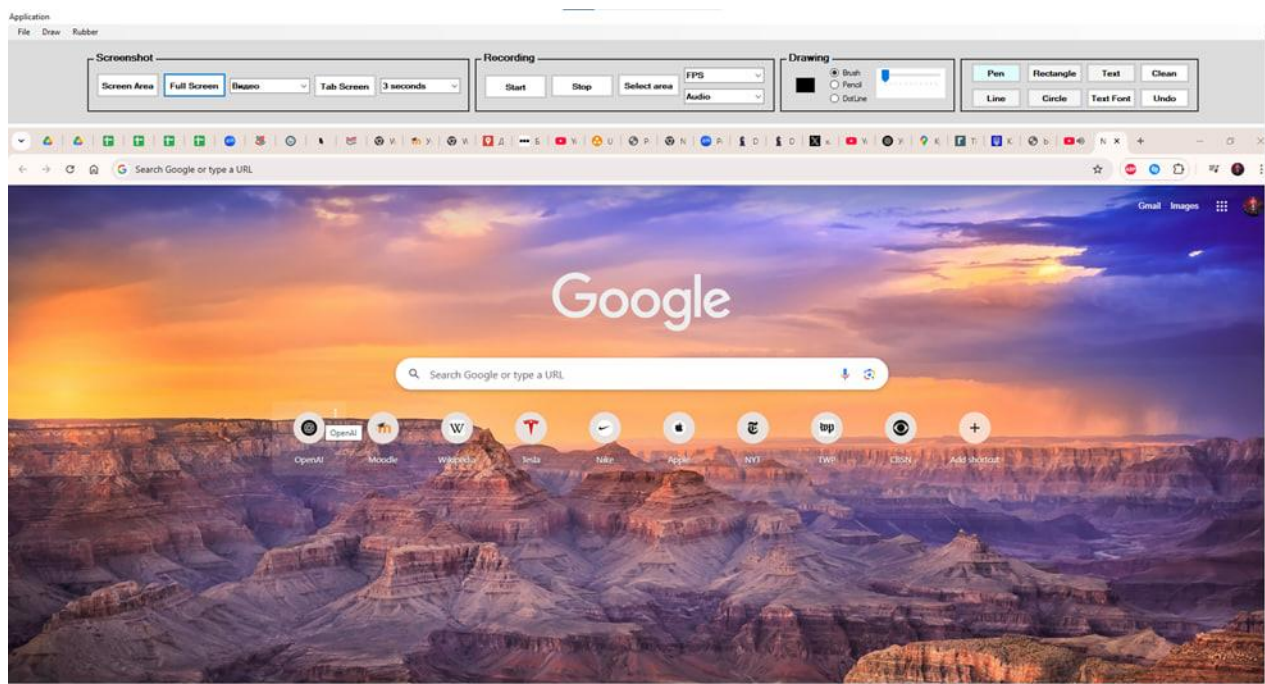


Рисунок 4.5 – Результат роботи повноекранного режиму застосунку: зображення у формі повного екрана користувача

Зробивши скріншот, користувач може обробити зображення за допомогою різних вбудованих інструментів застосунку, як олівець, пензель, пряма та крива лінії, текст тощо, налаштувавши відповідну ширину, колір та тип лінії. Програма також надає можливість очистити зображення від користувацької діяльності. Відредагований скріншот користувачем зображено на рисунках 4.6, 4.7.

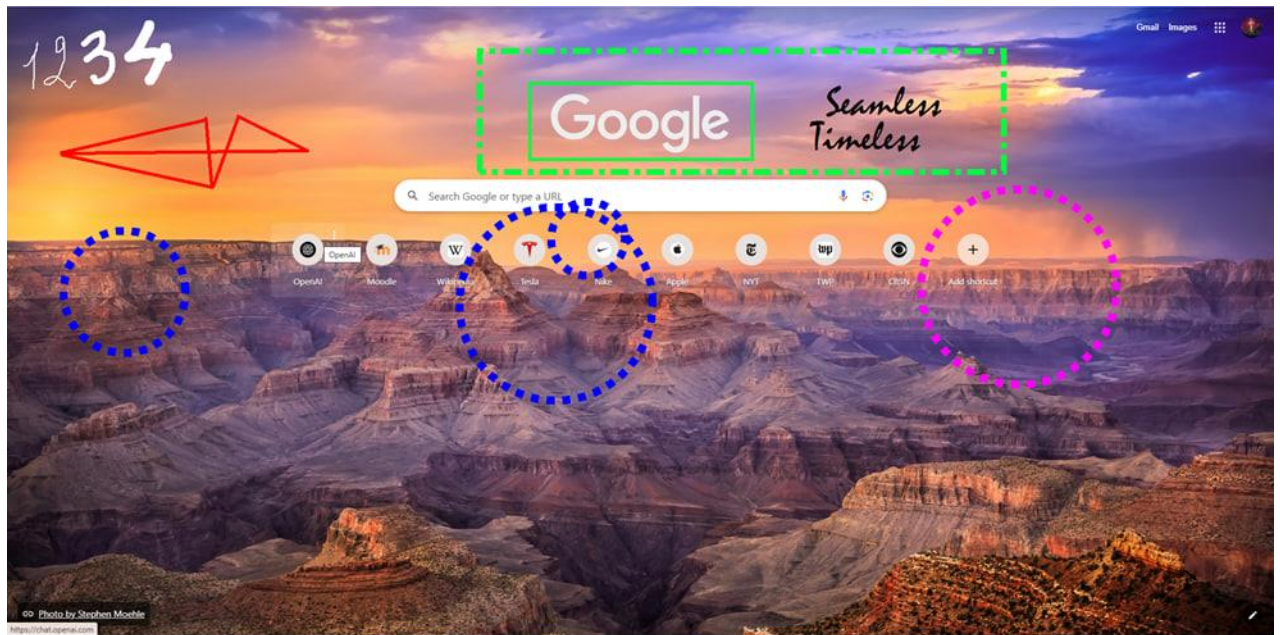


Рисунок 4.6 – Оброблений знімок екрана інструментами

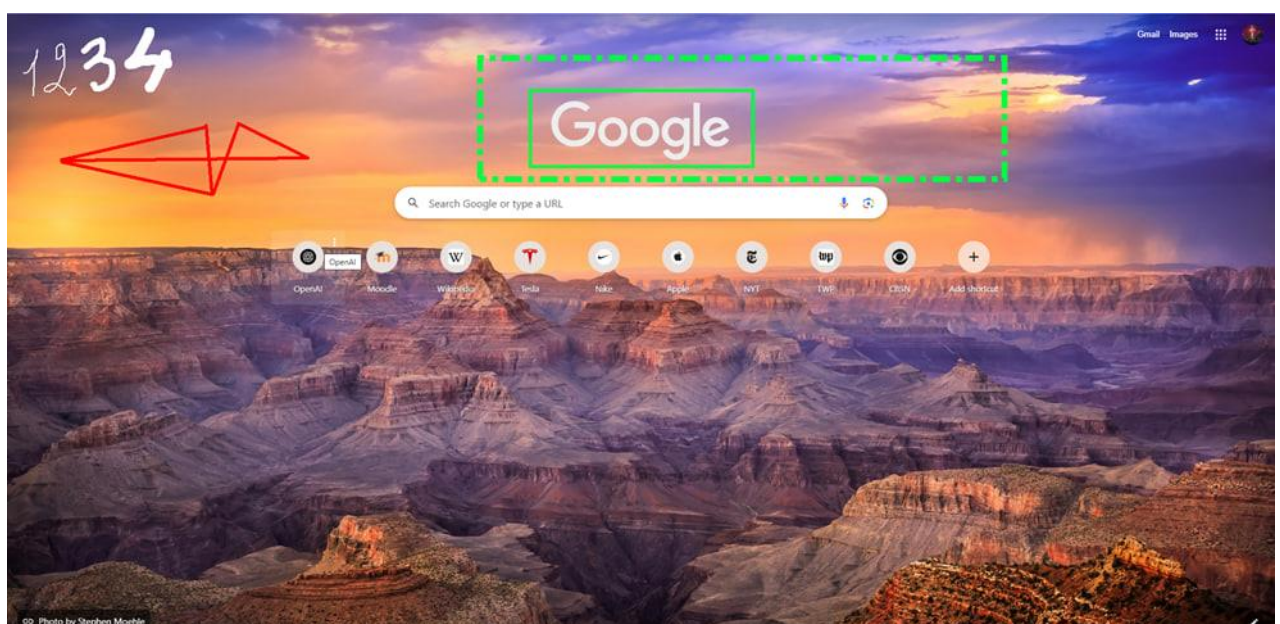


Рисунок 4.7 – Результат поступового видалення рисунків на зображенні

Однією з вимог до функціоналу є наявність гарячих клавіш, комбінації яких дозволяють виконувати дії у застосунку, не натискаючи мишею на відповідні кнопки на панелі. Для відтворення цієї особливості, було створено меню з налаштуванням функцій та форму з переліком комбінацій для зручності у роботі користувачам. Це меню також допомагає змінити колір фону, включити або виключити автоматичне збереження зображення до буфера обміну, включили або виключити оздоблення ввімкнених кнопок. Демонстрацію роботи меню взаємодії з користувачем зображено на рисунках 4.8, 4.9.

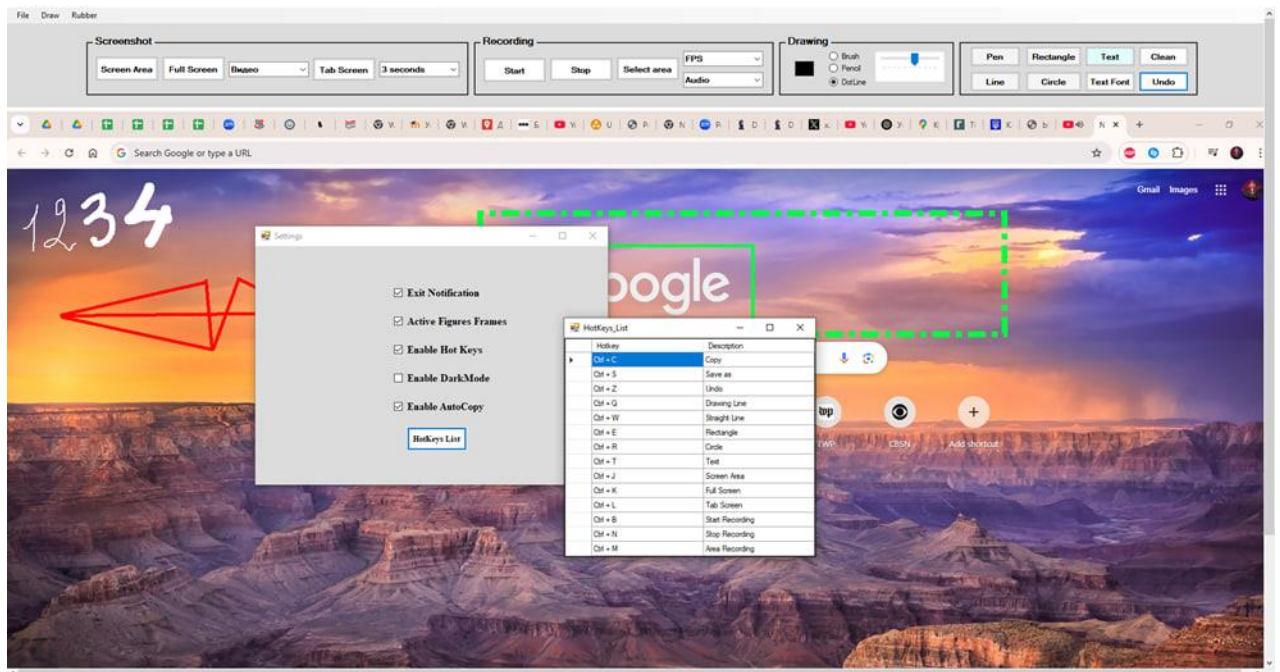


Рисунок 4.8 – Відкрите меню взаємодії з користувачем та форма з переліком наявних комбінацій гарячих клавіш для використання

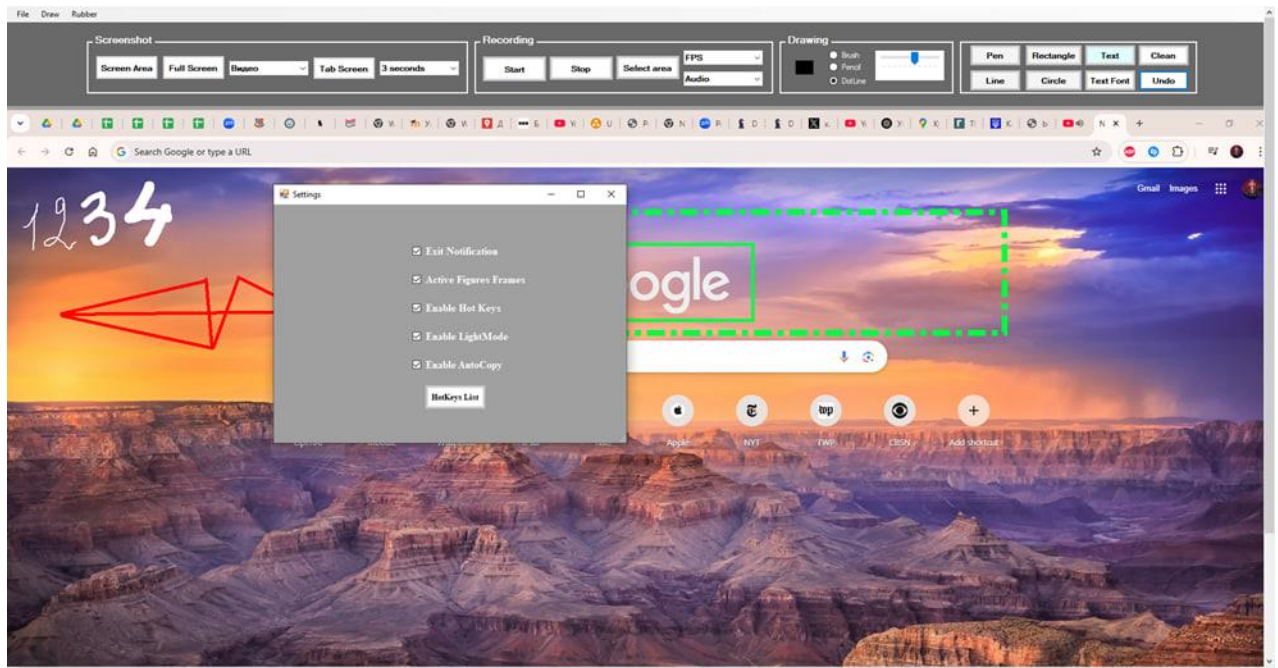


Рисунок 4.9 – Змінений колір інтерфейсу користувача на темний режим у меню взаємодії з користувачем

Оскільки тема роботи програми є робота із зображеннями екрана, то функціонал застосунку, що направлений на запис та збереження відео є одним із найбільш важливим у всьому проєкті. Користувач власноруч може обрати кількість кадрів на секунду, наявність звуку, режим запису відео (повноекранний чи обраної ділянки) перед тим, як файл буде збережений на жорсткому диску комп'ютера

У процесі запису відео було обрано відеокліп у мережі YouTube під назвою «West Coast» виконавиці Lana del Ray, налаштування FPS було забезпечено на рівні 11 кадрів на секунду, запис відбувся із записом звуку. Файл для запису був збережений на жорсткому диску C: із розширенням .mp4. Процес запису, збереження та відтворення відео зображено на рисунках 4.10 – 4.12.

Рисунок 4.11 – Результат роботи застосунку: створення тематичного відеофайлу розширення .mp4

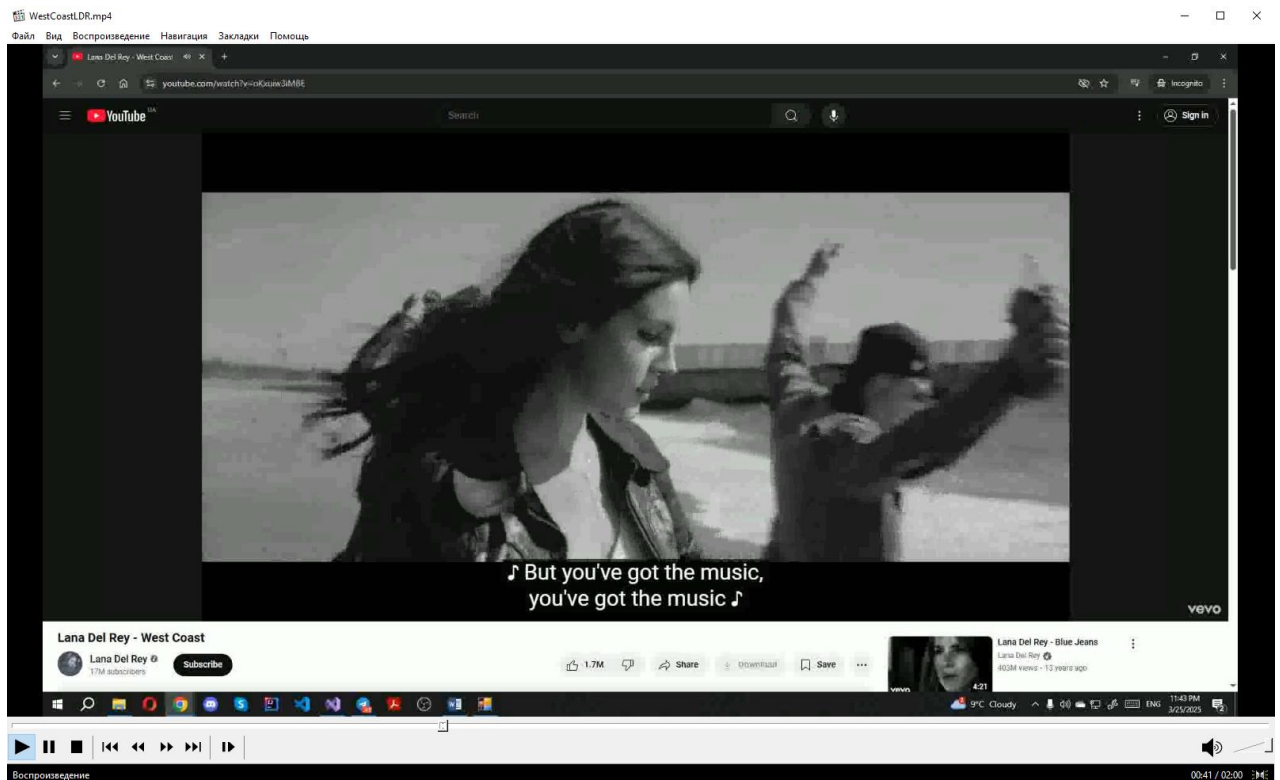


Рисунок 4.12 – Перегляд записаного відеофайлу у середовищі MPC-НС (x64)

Окрім запису відео у повноекранному режимі, у програмі можна обрати для цього відповідну ділянку. Принцип роботи такий самий, як і для запису зображення з екрана, де користувач визначає початкову і кінцеву точки, формуючи прямокутник відповідної довжини та ширини – саме це програма буде записувати з іншими додатковими параметрами на вибір користувача.

У цьому процесі запису відео було обрано відеокліп у мережі YouTube під назвою «Fortnight» виконавиці Taylor Swift, налаштування FPS було забезпечено на рівні 12 кадрів на секунду, запис відбувся із записом звуку. Файл для запису був збережений на жорсткому диску C: із розширенням .mp4. Процес виділення ділянки для запису та відтворення відео зображено відповідно на рисунках 4.13, 4.14.

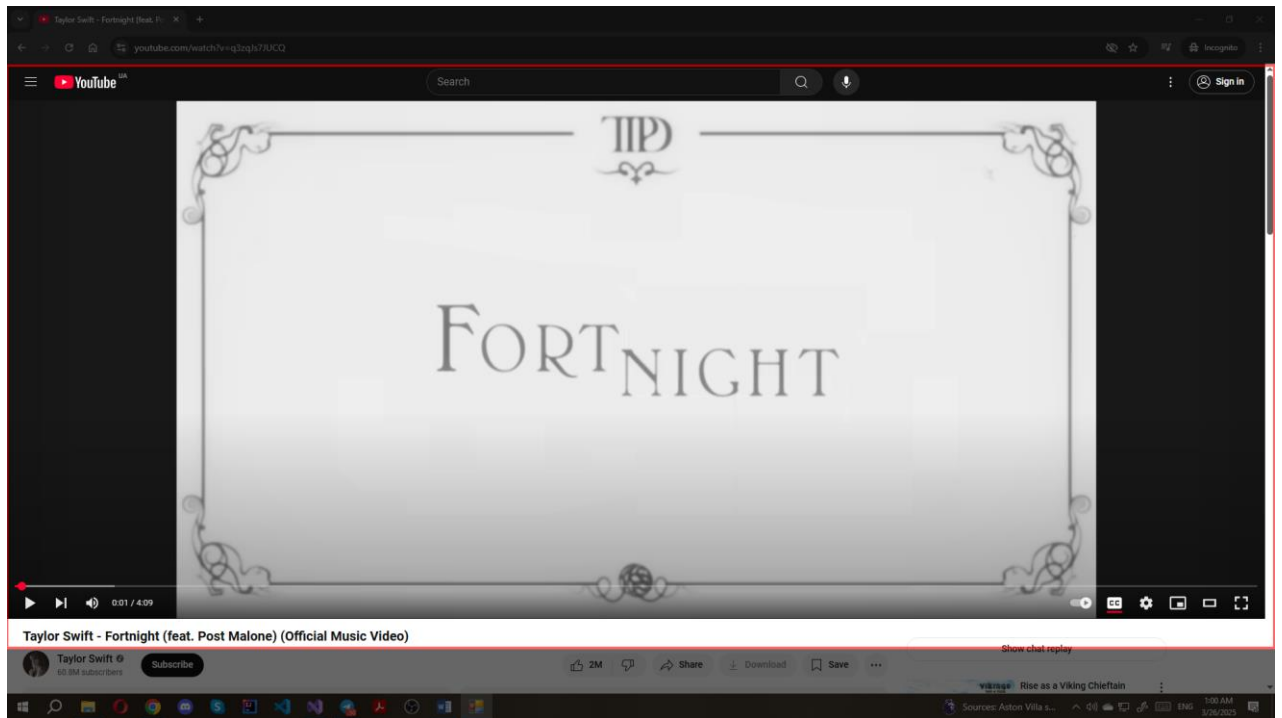


Рисунок 4.13 – Процес виділення ділянки на екрані для запису відео



Рисунок 4.14 – Перегляд записаного відеофайлу у середовищі MPC-НС (x64)

Повідомлення системи під час роботи відіграють важливу роль для взаємодії з користувачем, вони можуть позначати успішний та невдалий перебіг подій, помилки та попередження. Це допомагає зрозуміти причини помилки, чого треба уникати та дає змогу обрати варіант відповіді при попередженні.

Цей застосунок не є винятком, у ньому реалізовані алгоритми обробки помилкових ситуацій, попередження користувача перед виходом із системи та повідомлення про успішну операцію зі збереження файлу чи запису або зупинці відео. Приклади різних за характером повідомлень у застосунку зображені на рисунках 4.15 – 4.17.

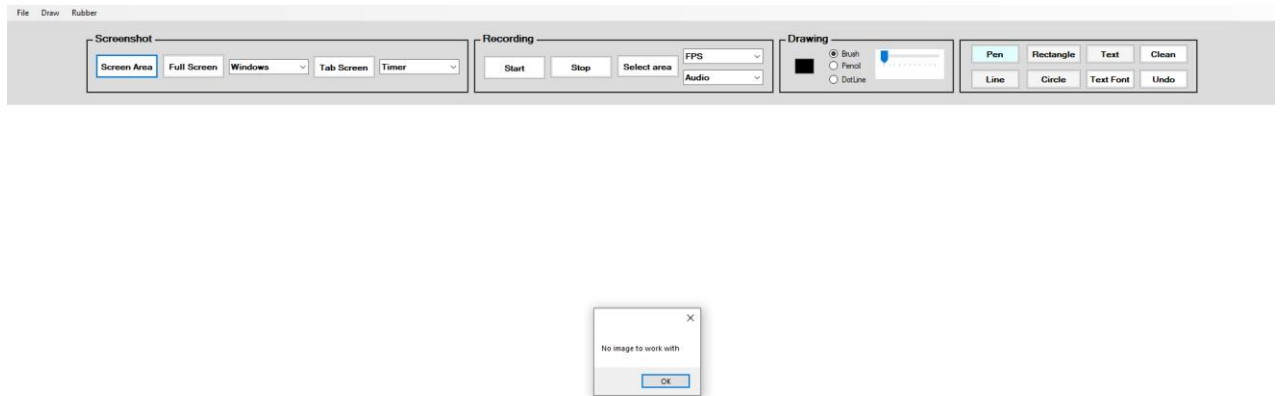


Рисунок 4.15 – Обробка виняткової ситуації під час відсутності зображення

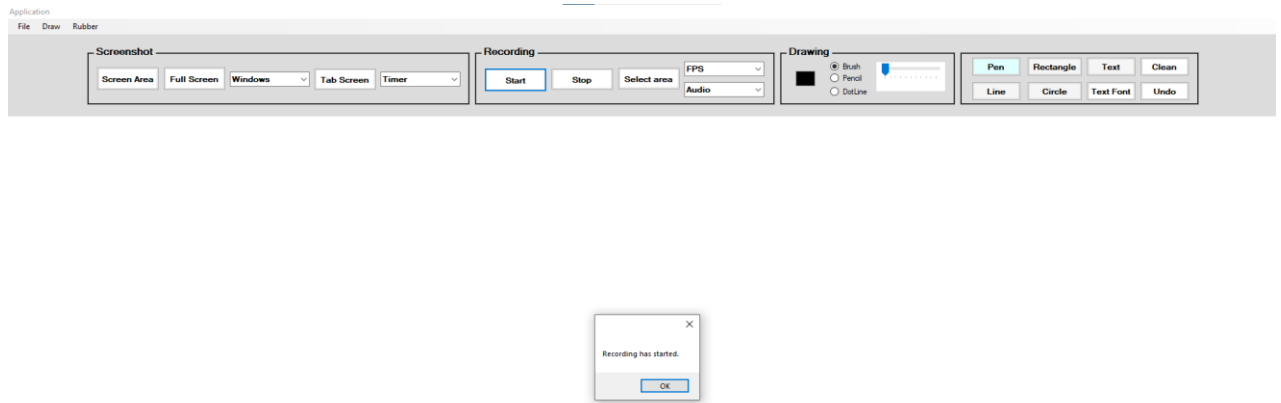


Рисунок 4.16 – Повідомлення про успішний початок запису відео

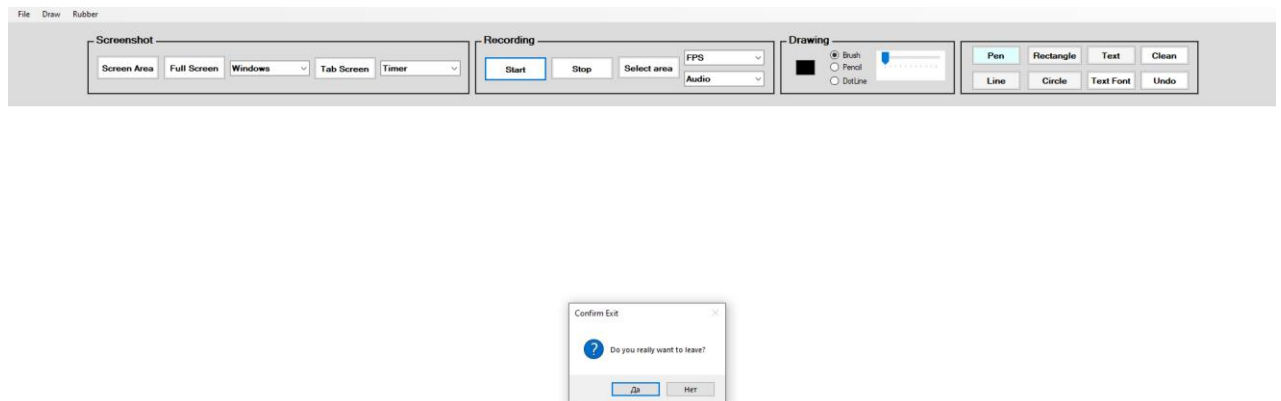


Рисунок 4.17 – Попередження користувачу стосовно виходу із програми

4.5 Висновки за отриманими результатами

У результаті експлуатаційного дослідження програмного забезпечення було підтверджено відповідність функціональних і нефункціональних можливостей заявленим вимогам. Проведене тестування, що включало ручну перевірку основних функцій, обробку виняткових ситуацій та аналіз інтерфейсу користувача, показало стабільну та коректну роботу програми.

Програма успішно виконує поставлені завдання, забезпечує належний рівень взаємодії з користувачем і відповідає визначеним умовам застосування. Всі функції, передбачені технічним завданням, були реалізовані та пройшли перевірку без виявлення критичних помилок.

У ході роботи було формалізовано призначення програми, визначено її основні функції, уточнено умови застосування, сформульовано вимоги до інтерфейсу та встановлено умови її виконання. Це дозволило забезпечити узгодженість між вимогами, реалізацією та тестуванням, що гарантує відповідність програмного забезпечення поставленим завданням.

ВИСНОВКИ

В ході виконання дипломної кваліфікаційної роботи бакалавра було розроблено програмне забезпечення для захоплення зображень з екрана, запису відео та нанесення графічних позначок. Для цього було виконано ряд завдань:

- проведено аналіз існуючих методів захоплення зображень та відео;
- обрано оптимальну мову програмування та технології для реалізації застосунку;
- розроблено архітектуру програмного забезпечення;
- реалізовано інтерфейс користувача та основні функціональні можливості програми.

Розроблений застосунок був реалізований на платформі .NET Framework з використанням C# та Windows Forms. Для роботи з графічними зображеннями застосовувалася бібліотека GDI+, для запису відео – Accord.Video.FFmpeg, а для обробки звуку – NAudio.Wave.

Застосунок було протестовано у середовищі Windows 10, що підтвердило його стабільну роботу, коректне захоплення зображень та відео, а також швидкодію при виконанні графічних операцій.

Застосунок реалізує такі функціональні вимоги:

- створення скріншотів екрана, активного вікна або вибраного місця;
- запис відео з можливістю вибору ділянки екрана та частоти кадрів;
- обробка аудіо під час відеозапису;
- малювання поверх зображень (лінії, кола, прямокутники, текст);
- збереження створених файлів у різних графічних форматах.

Розроблений програмний продукт успішно виконує всі поставлені задачі, відповідаючи як функціональним, так і інтерфейсним вимогам. Впровадження інтеграції з існуючими бібліотеками та системними компонентами забезпечило високу продуктивність, стабільність роботи та відповідність необхідним стандартам безпеки й надійності.

Розробка відповідної програми дозволила створити універсальний застосунок для роботи із графічною інформацією, що об'єднує в собі різні можливості захоплення, обробки та збереження зображень і відео з екрана. Завдяки підтримці кількох режимів захоплення, включаючи вибір певного вікна чи ділянки екрана, та гнучким налаштуванням через користувацьке меню, було значно покращено зручність і комфорт використання застосунку для кінцевих користувачів.

Створення цього застосунку є важливим кроком у розвитку технологій захоплення і обробки графічної інформації. Програма сприяє підвищенню якості роботи з візуальними даними, збільшує ефективність процесів обробки, збереження та управління зображеннями та відео, що відкриває нові можливості для подальшого розширення її функціональності в майбутньому.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. 71 Media Consumption Statistics for 2025 [Electronic resource]. – Access mode: <https://www.keevee.com/digital-media-consumption-statistics>.
2. Top 5 Uses for an Online Screen Recorder in Education and Business. [Electronic resource]. – Access mode: <https://cyberpanel.net/blog/top-5-uses-for-an-online-screen-recorder-in-education-and-business>.
3. The Best Screen Recording Software of 2025 (Free & Paid Apps) [Electronic resource]. – Access mode: <https://myelearningworld.com/best-screen-recording-software/>.
4. Snagit Features [Electronic resource]. – Access mode: <https://www.techsmith.com/snagit/features/>.
5. Lightshot Features [Electronic resource]. – Access mode: <https://app.prntscr.com/en/index.html>.
6. Windows Snipping Tool Features [Electronic resource]. – Access mode: <https://support.microsoft.com/en-us/windows/use-snipping-tool-to-capture-screenshots-00246869-1843-655f-f220-97299b865f6b>.
7. How to record your screen on Python [Electronic resource]. – Access mode: <https://thepythoncode.com/article/make-screen-recorder-python>.
8. Using Video Capture – Win32 apps [Electronic resource]. – Access mode: <https://learn.microsoft.com/en-us/windows/win32/multimedia/using-video-capture>.
9. Java Abstract Window Kit [Electronic resource]. – Access mode: https://en.wikipedia.org/wiki/Abstract_Window_Toolkit.
10. JavaScript. Using Screen Capture API [Electronic resource]. – Access mode: https://developer.mozilla.org/enUS/docs/Web/API/Screen_Capture_API/Using_Screen_Capture.
11. C# Screenshot Utility to Capture a Portion of the Screen [Electronic resource]. – Access mode: <https://nishanc.medium.com/c-screenshot-utility-to-capture-a-portion-of-the-screen-489ddceeee49>.

12. Capturing screenshots using C# and p/invoke [Electronic resource]. – Access mode: <https://www.cyotek.com/blog/capturing-screenshots-using-csharp-and-p-invoke>.
13. GDI+ Tutorial for Beginners [Electronic resource]. – Access mode: <https://www.csharp.com/article/gdi-tutorial-for-beginners/>.
14. Features of .NET Framework [Electronic resource]. – Access mode: <https://www.tatvasoft.com/outsourcing/2023/02/features-of-net.html>.
15. Windows API Index [Electronic resource]. – Access mode: <https://learn.microsoft.com/en-us/windows/win32/apiindex/windows-api-list>.
16. Microsoft Visual Studio Features [Electronic resource]. – Access mode: <https://visualstudio.microsoft.com/vs/features/>.
17. JetBrains Rider Features [Electronic resource]. – Access mode: <https://www.jetbrains.com/rider/features/>.
18. Visual Studio Code Features [Electronic resource]. – Access mode: <https://code.visualstudio.com/>.
19. System.Drawing Namespace [Electronic resource]. – Access mode: <https://learn.microsoft.com/en-us/dotnet/api/system.drawing?view=windowsdesktop-9.0>.
20. System.Drawing2D Namespace [Electronic resource]. – Access mode: <https://learn.microsoft.com/en-us/dotnet/api/system.drawing.imaging?view=windowsdesktop-9.0>.
21. System.Threading Namespace [Electronic resource]. – Access mode: <https://learn.microsoft.com/en-us/dotnet/api/system.threading?view=net-9.0>.
22. System.Threading.Tasks Namespace [Electronic resource]. – Access mode: <https://learn.microsoft.com/en-us/dotnet/api/system.threading.tasks?view=net-9.0>.
23. System.Runtime.InteropServices Namespace [Electronic resource]. – Access mode: <https://learn.microsoft.com/en-us/dotnet/api/system.runtime.interopservices?view=net-9.0>.

24. System.Windows.Forms Namespace [Electronic resource]. – Access mode: <https://learn.microsoft.com/en-us/dotnet/api/system.windows.forms?view=windowsdesktop-9.0>.

25. System.Drawing.Imaging Namespace [Electronic resource]. – Access mode: <https://learn.microsoft.com/en-us/dotnet/api/system.drawing.imaging?view=windowsdesktop-9.0>.

26. 30 Days of NAudio Documentation [Electronic resource]. – Access mode: <https://markheath.net/post/30-days-naudio-docs>.

27. FFmpeg Documentation [Electronic resource]. – Access mode: <https://ffmpeg.org/ffmpeg.html>.

28. Overview – Windows Forms .NET Framework [Electronic resource]. – Access mode: <https://learn.microsoft.com/en-us/dotnet/desktop/winforms/windows-forms-overview?view=netframeworkdesktop-4.8>.

29. What is Windows Presentation Foundation – WPF .NET [Electronic resource]. – Access mode: <https://learn.microsoft.com/en-us/dotnet/desktop/wpf/overview/?view=netdesktop-9.0>.

ДОДАТОК А
Технічне завдання

Вступ

Розроблюваний проєкт за темою «Програмна реалізація застосунку для запису екрану» є продуктом для роботи з графічною інформацією, яка поєднує в собі функції створення скріншотів, запису відео з екрана та редагування зображень. Основною метою розробки є забезпечення користувачів зручним інструментом для швидкого та ефективного захоплення екрана, запису відеоінструкцій і створення графічних позначок без необхідності використовувати сторонні програми.

Актуальність такого застосунку пояснюється зростаючою потребою у створенні навчальних матеріалів, демонстрації проблем технічної підтримки, підготовці відеоуроків та інших візуальних матеріалів. Об'єднання в одному застосунку можливостей для запису, редагування та малювання робить процес значно швидшим і зручнішим для користувача.

Очікуваним результатом є створення повнофункціональної програми, яка дозволить користувачам без додаткового програмного забезпечення виконувати весь спектр завдань, пов'язаних із захопленням екрана, записом відео та графічними анотаціями, що значно спростить їхню роботу та підвищить продуктивність.

А.1 Підстава для розробки

Підставою для розробки є завдання на дипломну кваліфікаційну роботу на тему «Програмна реалізація застосунку для запису екрану», затверджене наказом Національного університету «Запорізька політехніка» № 209 від 28 квітня 2025 року.

А.2 Основні вимоги до програми, що розробляється

А.2.1 Вимоги до функціональних характеристик

Функціональні вимоги відіграють ключову роль у розробці програмного забезпечення, адже саме вони визначають, які можливості має надавати програма користувачу. Від чіткості та повноти цих вимог залежить ефективність роботи застосунку, а також зручність його використання. У процесі проектування, реалізації та тестування програмного продукту функціональні характеристики залишаються основою, на яку спирається вся структура програми.

Функціональні можливості програми повинні забезпечувати реалізацію таких дій користувача:

- захоплення зображення з екрана (вікно програми, виділена ділянка, повноекранний режим);
- редагування зображення за допомогою олівця чи пензля, геометричних фігур (пряма та крива лінія, коло, квадрат, текст з обраним стилем та розміром);
- налаштування роботи програми за допомогою користувацького меню, де користувач має можливість підключити або відключити спеціальні додаткові можливості (автокопіювання зображення, зміна кольору теми, наявність комбінації гарячих клавіш тощо);
- редагування ширини стержня й кольору олівця чи пензля;
- поетапне чи повне очищення фотографії від елементів редагування;
- визначення за допомогою таймера часу очікування, після якого застосунок повинен зробити захоплення зображення;
- запис відео без та зі звуком, налаштування FPS, вибір ділянки на екрані для подальшого запису;
- використання комбінацій гарячих клавіш для роботи у застосунку;
- збереження зображення на комп'ютер або до буфера обміну.

A.2.2 Вимоги до інтерфейсу програми

Інтерфейс програми повинен забезпечувати такі можливості:

- інтуїтивно зрозумілий та зручний інтерфейс, з чітко розташованими елементами, що забезпечує легкий доступ до основних функцій;
- мінімалістичний дизайн для уникнення перевантаження зайвими інструментами та елементами;
- наявність гарячих клавіш для швидкого виклику функцій;
- підтримка темної та світлої теми для зручності користувача;
- використання групування елементів за функціональністю для покращення навігації;
- наявність спливаючих підказок при наведенні на елементи.

A.2.3 Вимоги до надійності

Програмне забезпечення має забезпечувати стабільну та безперебійну роботу в умовах штатного використання. У випадку некоректного введення даних або помилкових дій користувача програма повинна обробляти виняткові ситуації без аварійного завершення роботи.

З метою підвищення надійності необхідно врахувати:

- захист від втрати даних при створенні скріншотів та запису відео;
- використання механізмів скасування останніх дій при малюванні;
- запобігання зависанням при обробці великих графічних файлів;
- коректне вивільнення ресурсів після завершення роботи з відео- або графічними файлами;
- мінімізацію споживання оперативної пам'яті та процесорних ресурсів для забезпечення швидкої роботи на середніх та слабких системах;
- програма повинна мати механізм автоматичного виявлення та обробки можливих помилок без необхідності втручання користувача.

A.2.4 Вимоги до складу та параметрів технічних засобів

Для коректної роботи програми рекомендується наступна конфігурація апаратного забезпечення:

Мінімальні вимоги:

- процесор: 2-ядерний, 2.0 ГГц або аналогічний;
- оперативна пам'ять: 4 ГБ;
- відеокарта: вбудована з підтримкою DirectX 10;
- місце на диску: 500 МБ вільного простору;
- операційна система: Windows 10 (x64).

Рекомендовані вимоги:

- процесор: 4-ядерний, 3.0 ГГц або вище;
- оперативна пам'ять: 8 ГБ або більше;
- відеокарта: NVIDIA/AMD;
- місце на диску: 1 ГБ вільного простору;
- операційна система: Windows 10/11 (x64).

ДОДАТОК Б
Опис програми

Б.1 Загальні відомості

Розроблена програма за темою «Програмна реалізація застосунку для запису екрану» призначена для захоплення зображень екрана, запису відео та нанесення графічних позначок. Вона поєднує можливості створення скріншотів, запису відео, а також редагування та коментування графічного контенту.

Програма розроблена для використання в середовищі Windows (від версії 10) та орієнтована на користувачів, яким необхідно швидко та зручно працювати із зображеннями та відео. Вона містить інтуїтивно зрозумілий графічний інтерфейс і підтримує базові операції редагування, що робить її зручною навіть для користувачів без спеціальних знань у графічному дизайні.

Б.2 Функціональне призначення

Перелік розробленого функціоналу, який містить програма:

Редагування зображень:

- малювання ліній, кіл, прямокутників;
- додавання тексту, можливість налаштувати шрифт;
- вибір кольору та товщини ліній;
- видалення або скасування останньої дії;
- вибір стилю пера чи лінії.

Захоплення екрана:

- запис фото обраної ділянки, всього екрана, активного вікна;
- захоплення зображення із таймером.

Запис відео:

- вибір місця на екрані для запису;
- налаштування частоти кадрів (FPS);
- додавання фонові аудіодоріжки.

Редагування зображень:

- малювання ліній, кіл, прямокутників;
- додавання тексту, можливість налаштувати шрифт;
- вибір кольору та товщини ліній;
- видалення або скасування останньої дії.

Збереження даних:

- збереження зображення до буфера обміну, на жорсткий диск у форматах розширення .png, .jpg;
- збереження відеозапису на жорсткий диск у форматі розширення або .mp4 або .avi.

Б.3 Опис логічної структури

Програма складається з кількох класів, один з яких є головним, і кожен з яких відповідає за відповідну функціональність та обов'язки.

Клас frmCapture відповідає за виділення ділянки на екрані для того, щоб зробити скріншот на необхідній ділянці (за розміром і координатами). Він містить у собі методи, які оброблюють події миші, це необхідно для того, щоб зрозуміти, звідки користувач починає свою роботу, яку ділянку захоплює, де він зупинився.

Клас SettingsMenu, який відповідає за роботу форми з налаштувань, яка за допомогою відповідних «прапорців» оптимізує можливості та зовнішній вигляд застосунку.

Головний клас `CatchArea` і компонент всієї програми, який працює з головною формою застосунку, містить відповідні змінні та методи, з'єднаний з другорядними формами для забезпечення відповідного функціонування. Клас містить логіку обробки і збереження зображення за допомогою відповідних інструментів, вбудовані методи з принципами запису відео, створення скріншотів у повноекранному, віконному режимах, а також у ділянці на екрані, обраною користувачем. Локальні змінні допомагають працювати з іншим функціоналом, використовуючи отримані дані для функціонування системи, як-от FPS чи секунди затримки роботи програми перед захопленням зображення.

Б.4 Використані технічні засоби

Для роботи програми використовується стандартне середовище `Windows Forms` на платформі `.NET Framework`, що забезпечує зручне створення графічного інтерфейсу користувача та взаємодію з системними ресурсами. Розробка здійснювалася в `Microsoft Visual Studio` із використанням мови програмування `C#`. Графічний інтерфейс реалізовано на базі `WinForms`, а для обробки зображень, малювання та взаємодії з графічними об'єктами використовується бібліотека `GDI+`. Запис відео реалізовано за допомогою бібліотеки `Accord.Video.FFmpeg`, яка дозволяє здійснювати кодування та збереження відеопотоків у популярних форматах. Для роботи із записом та обробкою аудіо використовується бібліотека `NAudio.Wave`, що забезпечує взаємодію з різними аудіоформатами та джерелами звуку. Збереження створених файлів, таких як зображення та відео, здійснюється за допомогою стандартних засобів `Windows`, зокрема діалогового вікна `SaveFileDialog`, яке надає користувачеві можливість вибору місця збереження та формату файлу.

ДОДАТОК В
Текст програми

В.1 Файл CatchArea.cs

```

private void CleanCanvas_Click(object sender, EventArgs e)
{
    if (originalImage != null){
        originalImage = new Bitmap(originalImage);
        pictureBox1.Image = originalImage;
        drawingImage = new Bitmap(originalImage); // Знімок для малювання
        drawingGraphics = Graphics.FromImage(drawingImage);
        history.Clear(); // Очищення стеку зображень
        history.Push(pictureBox1.Image);
        if(checkbutton) // Автозбереження у разі активності функції
            Clipboard.SetImage(pictureBox1.Image);
    }
}

private void pictureBox1_MouseUp(object sender, MouseEventArgs e)
{
    if (e.Button == MouseButton.Left){
        if (drawingGraphics != null && currentRectangle != Rectangle.Empty &&
isDrawingRectangle){
            // Використання проекції для зображення прямокутника на знімку
            drawingGraphics.DrawRectangle(drawingPen, currentRectangle);
            pictureBox1.Image = drawingImage; // Оновлення скріншоту
            currentRectangle = Rectangle.Empty; // Видалення даних про прямокутник
        }
        else if (drawingGraphics != null && currentCircle != Rectangle.Empty && isDrawingCircle)
        {
            // Використання проекції для зображення кола на знімку
            drawingGraphics.DrawEllipse(drawingPen, currentCircle);
            pictureBox1.Image = drawingImage;
            // Видалення даних про коло
            currentCircle = Rectangle.Empty;
        }
        else if (drawingGraphics != null && isDrawingLine){
            drawingGraphics.DrawLine(drawingPen, startPoint1, endPoint1);
            pictureBox1.Image = drawingImage;
            startPoint1 = Point.Empty; // Видалення даних про початкову точку прямої лінії
            endPoint1 = Point.Empty; // Видалення даних про кінцеву точку прямої лінії
        }
        if (pictureBox1.Image != null){ // Робота з кривою прямою
            history.Push(new Bitmap(pictureBox1.Image)); // Оновлення стеку
            if (checkbutton)
                Clipboard.SetImage(pictureBox1.Image);
            previousPoint = Point.Empty; // Видалення попередніх даних про поточну точку
        }
        else{ // Обробка виняткових ситуацій – відсутність знімку в полі для роботи
            MessageBox.Show("No image to work with");
        }
    }
}

```

```

bool startFromArea = false;
private void pictureBox1_MouseDown(object sender, MouseEventArgs e)
{
    if (e.Button == MouseButton.Left) // Обробка винятків для запобігання випадкових ліній
    {
        startFromArea = true;
    }
    //
    if (e.Button == MouseButton.Left && isDrawingRectangle)
    {
        previousPoint = e.Location; // Поточне розташування курсора
        currentRectangle = new Rectangle(e.Location, new Size(0, 0)); // Початок роботи з фігурою
    }
    else if (e.Button == MouseButton.Left && isDrawingCircle)
    {
        previousPoint = e.Location;
        currentCircle = new Rectangle(e.Location, new Size(0, 0));
    }
    else if (e.Button == MouseButton.Left && isDrawingLine)
    {
        startPoint1 = e.Location; // Координати початкової точки прямої лінії
        endPoint1 = e.Location; // Координати кінцевої точки прямої лінії
    }
    else if (e.Button == MouseButton.Left && isTyping)
    {
        previousPoint = e.Location; // Координати поточної точки для написання тексту
    }
    else
        previousPoint = e.Location; // Координати поточної точки кривої лінії
}

Stack<Image> history = new Stack<Image>();

private void pictureBox1_MouseMove(object sender, MouseEventArgs e)
{
    if (e.Button == MouseButton.Left && drawingGraphics != null && startFromArea == true)
    {
        if (currentRectangle != null && isDrawingRectangle){ // Робота з прямокутником

            int x = Math.Min(previousPoint.X, e.X); // Визначення початкової точки за X
            int y = Math.Min(previousPoint.Y, e.Y); // Визначення початкової точки за Y
            int width = Math.Abs(e.X - previousPoint.X); // Визначення ширини фігури
            int height = Math.Abs(e.Y - previousPoint.Y); // Визначення висоти фігури

            currentRectangle = new Rectangle(x, y, width, height); // Оновлення даних фігури

            pictureBox1.Invalidate(); // Відтворення проекції на основі даних прямокутника
        }
    }
}

```

```

else if (isDrawingCircle && currentCircle != null){

    int diameter = Math.Max(Math.Abs(e.X - previousPoint.X), Math.Abs(e.Y -
previousPoint.Y)); // Визначення діаметра для запобігання розробки овалу
    int x = previousPoint.X - diameter / 2; // Визначення центра кола за X
    int y = previousPoint.Y - diameter / 2; // Визначення центра кола за Y

    currentCircle = new Rectangle(x, y, diameter, diameter); // Оновлення даних фігури

    pictureBox1.Invalidate(); // Відтворення проекції на основі даних кола
}
else if (isDrawingLine){
    endPoint1 = e.Location; // Оновлення даних про кінцеву точку прямої лінії
    pictureBox1.Invalidate(); // Відтворення проекції на основі даних прямої лінії
}
else if(isTyping){
    endPoint1 = e.Location;
    pictureBox1.Invalidate(); // Відтворення тексту
    if (check2){
        TextButton.BackColor = System.Drawing.Color.LightCyan;
        CircleButton.BackColor = System.Drawing.Color.WhiteSmoke;
        RectangleButton.BackColor = System.Drawing.Color.WhiteSmoke;
        StraightLineButton.BackColor = System.Drawing.Color.WhiteSmoke;
        PenLineButton.BackColor = System.Drawing.Color.WhiteSmoke;
    }
}
else{
    drawingGraphics.DrawLine(drawingPen, previousPoint, e.Location);
    pictureBox1.Image = drawingImage; // Динамічне оновлення зображення без Paint
    previousPoint = e.Location; // Оновлення позиції курсора
}
}
}
EnumWindowsProc(IntPtr hWnd, IntPtr lParam); // функція, яку викликає WinAPI для кожного
вікна для подальших дій з ним (ID, розташування, назва тощо)

[StructLayout(LayoutKind.Sequential)]
private struct RECT // Структура опису розташування вікна за його координатами
{
    public int Left;
    public int Top;
    public int Right;
    public int Bottom;
}

```

```

private void LoadOpenWindows()
{
    Dictionary<IntPtr, string> newWindows = new Dictionary<IntPtr, string>(); // словник
    відкритих вікон
    // словник для роботи з comboBoxWindows
    List<IntPtr> openWindowHandles = new List<IntPtr>(); // список для перевірки чи відкриті
    вікна
    EnumWindows(new EnumWindowsProc((hWnd, lParam) => // делегат кожного відкритого
    вікна
    {
        if (IsWindowVisible(hWnd)){
            int length = GetWindowTextLength(hWnd);
            if (length > 0){
                var windowTitle = new System.Text.StringBuilder(length + 1);
                GetWindowText(hWnd, windowTitle, windowTitle.Capacity);

                string title = windowTitle.ToString();
                if (title != "Program Manager" && title != "Параметры" &&
                    title != "Microsoft Text Input Application" && title != "Application"){
                    openWindowHandles.Add(hWnd);
                    newWindows[hWnd] = title;

                    if (!existingWindows.ContainsKey(hWnd))
                    {
                        // Додавання нового вікна, якщо його не було у випадному списку
                        comboBoxWindows.Items.Add(new WindowItem { Handle = hWnd, Title = title });
                    }
                    else if (existingWindows[hWnd] != title)
                    {
                        // Якщо назва вікна змінилася, оновлення її в списку
                        UpdateWindowTitle(hWnd, title);
                    }
                }
            }
        }
    }
    return true;
}), IntPtr.Zero);

// Видалення закритих вікон
for (int i = comboBoxWindows.Items.Count - 1; i >= 0; i--) // цикл зворотній, щоб не збити
елементи вгору
{
    WindowItem item = (WindowItem)comboBoxWindows.Items[i];
    if (!openWindowHandles.Contains(item.Handle)){
        comboBoxWindows.Items.RemoveAt(i);
        existingWindows.Remove(item.Handle);
    }
}
existingWindows = newWindows;
comboBoxWindows.DisplayMember = "Title";
}
private Dictionary<IntPtr, string> existingWindows = new Dictionary<IntPtr, string>();

```

```

private void UpdateWindowTitle(IntPtr hWnd, string newTitle)
{
    for (int i = 0; i < comboBoxWindows.Items.Count; i++){
        WindowItem item = (WindowItem)comboBoxWindows.Items[i];
        if (item.Handle == hWnd)
        {
            comboBoxWindows.Items[i] = new WindowItem { Handle = hWnd, Title = newTitle };
            existingWindows[hWnd] = newTitle;
            break;
        }
    }
}

private void CaptureWindow(IntPtr hWnd) // Метод пошуку точок розташування вікон
{
    GetWindowRect(hWnd, out RECT rect); // На основі унікального номера вікна пошук
    координат розташування
    int width = rect.Right - rect.Left; // Ширина вікна
    int height = rect.Bottom - rect.Top; // Висота вікна

    Bitmap bmp = new Bitmap(width, height); // Створення об'єкта растрового зображення
    using (Graphics g = Graphics.FromImage(bmp)){ // Копіювання зображення екрана до
    створеного растрового зображення
        g.CopyFromScreen(new Point(rect.Left, rect.Top), Point.Empty, new Size(width, height));
    }

    originalImage = new Bitmap(bmp); // Оригінальне зображення на основі скріншоту

    pictureBox1.Image = originalImage; // Заповнення поля даними
    history.Clear();
    history.Push(new Bitmap(pictureBox1.Image));

    drawingImage = new Bitmap(originalImage); // Створення об'єкта для малювання на основі
    оригінального знімка
    drawingGraphics = Graphics.FromImage(drawingImage);
    previousPoint = Point.Empty;

    pictureBox1.Width = originalImage.Width;
    pictureBox1.Height = originalImage.Height;
}

```

В.2 Файл frmCapture.cs

```
private void pictureBox1_Paint(object sender, PaintEventArgs e)
{
    if (start)
    {
        int width = Math.Abs(currentX - startX); // Ширина фігури
        int height = Math.Abs(currentY - startY); // Висота фігури
        int x = Math.Min(startX, currentX); // Початкова точка фігури по X
        int y = Math.Min(startY, currentY); // Початкова точка фігури по Y

        Rectangle selection = new Rectangle(x, y, width, height);

        e.Graphics.DrawImage(printScreen, selection, selection, GraphicsUnit.Pixel);
        // висвітлювання виділеної ділянки прямокутника шаром printScreen, де selection є
        // джерелом і місцем для розташування нової ділянки (за координатами)
        using (Pen dashedPen = new Pen(Color.Red, 4)) // створення рамки виділеної ділянки
        червоним кольором
        {
            dashedPen.DashStyle = DashStyle.Solid;
            e.Graphics.DrawRectangle(dashedPen, selection); // Проекція прямокутної ділянки
            червоного кольору для виділення «рамки» під час захоплення зображення
        }
    }
}

private void pictureBox1_MouseDown(object sender, MouseEventArgs e)
{
    if (e.Button == MouseButtons.Left) // ініціалізація даних на основі курсора
    {
        startX = e.X;
        startY = e.Y;
        currentX = e.X;
        currentY = e.Y;
        start = true;
    }
}

private void pictureBox1_MouseMove(object sender, MouseEventArgs e)
{
    if (start) // Зміна точок координат, відтворення проекції на зображенні
    {
        currentX = e.X;
        currentY = e.Y;
        pictureBox1.Invalidate();
    }
}
```

```

private void pictureBox1_MouseUp(object sender, MouseEventArgs e)
{
    if (e.Button == MouseButtons.Left)
    {
        start = false;
        int width = Math.Abs(currentX - startX);
        int height = Math.Abs(currentY - startY);
        int x = Math.Min(startX, currentX);
        int y = Math.Min(startY, currentY);

        if (width > 0 && height > 0)
        {
            Rectangle selection = new Rectangle(x, y, width, height); // створення ділянки на основі
отриманих даних про початкову точку та ширину/висоту вікна
            Bitmap selectedArea = new Bitmap(selection.Width, selection.Height); // об'єкт для запису
знімку в створене растрове зображення

            using (Graphics g = Graphics.FromImage(selectedArea))
            {
                g.DrawImage(printScreen, new Rectangle(0, 0, selectedArea.Width,
selectedArea.Height), selection, GraphicsUnit.Pixel); // DrawImage, а не CopyFromScreen, тому
що це є копіювання лише ділянки екрана, а не всього поля, є процесом перенесення контенту
виділеної ділянки до піксельного об'єкта
            }

            Clipboard.SetImage(selectedArea);
        }

        this.Close();
    }
}

private void frmCapture_Load(object sender, EventArgs e) // Налаштування екрана під час
запуску відповідного режиму
{
    this.Hide();

    int screenWidth = Screen.PrimaryScreen.Bounds.Width;
    int screenHeight = Screen.PrimaryScreen.Bounds.Height;
    // На основі розширення екрана створюються 2 зображення, одне є світлим, інше - темним
    printScreen = new Bitmap(screenWidth, screenHeight);
    darkScreen = new Bitmap(screenWidth, screenHeight);

    using (Graphics graphics = Graphics.FromImage(printScreen)){
        graphics.CopyFromScreen(0, 0, 0, 0, printScreen.Size);
        // printScreen запис повноекранного зображення з екрана
        // створивши до того graphics на основі екрана для подальшого малювання
    }
}

```

```

// Створюємо затемнену копію екрана
using (Graphics g = Graphics.FromImage(darkScreen)) // створення інструмента для
малювання на darkScreen
{
    g.DrawImage(printScreen, 0, 0); // копія з printScreen до darkScreen
    using (SolidBrush semiTransparentBrush = new SolidBrush(Color.FromArgb(100, 0, 0, 0)))
    {
        g.FillRectangle(semiTransparentBrush, 0, 0, screenWidth, screenHeight);
        // створивши за допомогою semiTransparentBrush напівпрозору темну заливку
        // відбувається затемнення зображення darkScreen
    }
}
this.Show(); // Виведення на екран затемнене зображення
}

```

В.3 Файл SettingsMenu.cs

```

public SettingsMenu(bool chckbtn1, bool chckbtn2, bool chckbtn3, bool chckbtn4, bool
chckbtn5)
{
    InitializeComponent();
    checkBox4.Text = "Enable DarkMode";
    checkBox4.Font = new Font("Times New Roman", 12, FontStyle.Bold);
    if (chckbtn1) // Перевірка включення функції повідомлення перед виходом із застосунку
        checkBox1.Checked = true;
    else if (!chckbtn1)
        checkBox1.Checked = false;

    if (chckbtn2) // Перевірка включення функції підсвічування активних кнопок інструментів
        checkBox2.Checked = true;
    else if (!chckbtn2)
        checkBox2.Checked = false;

    if (chckbtn3) // Перевірка включення функції активації гарячих клавіш
        checkBox3.Checked = true;
    else if (!chckbtn3)
        checkBox3.Checked = false;

    if (chckbtn4) // Перевірка включеної чи ввімкненої темної чи світлої теми
        checkBox4.Checked = true;
    else if (!chckbtn4)
        checkBox4.Checked = false;

    if (chckbtn5) // Перевірка включення функції автоматичного збереження до буфера
        checkBox5.Checked = true;
    else if (!chckbtn5)
        checkBox5.Checked = false;
}

```

```

if (!chckbbtn4) // зміна кольору на темну тему інтерфейсу
{
    this.BackColor = Color.Gainsboro;
    checkBox1.ForeColor = Color.Black;
    checkBox2.ForeColor = Color.Black;
    checkBox3.ForeColor = Color.Black;
    checkBox4.ForeColor = Color.Black;
    checkBox5.ForeColor = Color.Black;

    checkBox1.BackColor = Color.Gainsboro;
    checkBox2.BackColor = Color.Gainsboro;
    checkBox3.BackColor = Color.Gainsboro;
    checkBox4.BackColor = Color.Gainsboro;
    checkBox5.BackColor = Color.Gainsboro;
}
else // зміна кольору на світлу тему інтерфейсу
{
    this.BackColor = SystemColors.ControlDark;

    checkBox1.ForeColor = Color.White;
    checkBox2.ForeColor = Color.White;
    checkBox3.ForeColor = Color.White;
    checkBox4.ForeColor = Color.White;
    checkBox5.ForeColor = Color.White;

    checkBox1.BackColor = SystemColors.ControlDark;
    checkBox2.BackColor = SystemColors.ControlDark;
    checkBox3.BackColor = SystemColors.ControlDark;
    checkBox4.BackColor = SystemColors.ControlDark;
    checkBox5.BackColor = SystemColors.ControlDark;
}
}

```

ДОДАТОК Д
Слайди презентації

Програмна реалізація застосунку для запису екрану

Кафедра програмних засобів НУ «Запорізька Політехніка»
Автор - студент групи КНТ-121 Попов Кирило Сергійович
Керівник – д-р. філософії, доцент Леошенко Сергій Дмитрович

Рисунок Д.1 – Слайд 1

Об'єкт, предмет та мета роботи

Об'єкт дослідження – процеси взаємодії програмного забезпечення з операційною системою для доступу до зображення екрана, зокрема принципи обробки графічних даних під час захоплення зображення, відеозапису та нанесення графічних елементів на зроблені знімки.

Предмет дослідження – методи та інструменти реалізації функціоналу скріншотів, відео- та аудіозапису, а також графічного редагування в середовищі Windows Forms, включно з підходами до оптимізації продуктивності, збереження якості зображення та зручності для користувача.

Метою роботи є створення застосунку з інтуїтивним інтерфейсом, що забезпечує користувачеві можливість здійснювати захоплення зображень з екрана, записувати відео з аудіо супроводом, а також виконувати базове редагування зображень без потреби звертатися до сторонніх редакторів.

Рисунок Д.2 – Слайд 2

Постановка завдання

- аналіз наявних аналогів програмного забезпечення у предметній області з визначенням їхніх переваг та недоліків;
- визначення вимог до програми з урахуванням зручності користування та практичної доцільності та особливостей потреб користувачів;
- розробка інтерфейсу користувача, який забезпечить ефективну взаємодію завдяки своїй ергономічності та інтуїтивній зрозумілості;
- проектування та подальша розробка алгоритмів для створення зображень і відео, обробки скріншотів графічними інструментами, можливості збереження отриманих матеріалів із вибором їхнього місця розташування;
- забезпечення стабільної інтеграції та взаємодії з використаними інструментами, системними засобами включно із зовнішніми бібліотеками;
- оформлення механізмів обробки виняткових ситуацій з метою підвищення надійності й безпеки роботи програмного засобу.

2

Рисунок Д.3 – Слайд 3

Порівняння існуючих аналогів

Критерії	Snipping Tool	Snagit	Lightshot
Розробник	Microsoft	TechSmith	Skillbrains
Платформа	Windows	Windows, macOs	Windows, macOS
Ціна	Безкоштовний	Платний (\$63)	Безкоштовний
Основні функції	Захоплення екрана, просте редагування знімків	Захоплення екрана, запис відео, потужний редактор	Захоплення екрана, базове редагування знімків
Формати збереження	.png , .jpg , .gif	.png , .jpg , .gif , .mp4	.png , .jpg
Редагування	Мінімальні можливості (олівці, маркер)	Розширений редактор з ефектами, текстом	Мінімальні можливості (основні анотації)
Запис відео	Відсутній	У наявності	Відсутній
Гарячі клавіші	Є	Є	Немає
Простота використання	Дуже проста у використанні	Середня у використанні	Легка у використанні

3

Рисунок Д.4 – Слайд 4

Вимоги до програмного забезпечення



4

Рисунок Д.5 – Слайд 5

Вибір мови програмування

Критерії	C#	Python	Java
Продуктивність	Компіляція забезпечує швидке виконання та високу продуктивність	Інтерпретована мова, наявні бібліотеки, що оптимізують роботу з кодом	Виконання через JVM, що додає абстракцію і знижує продуктивність
Підтримка графічних бібліотек	Наявні бібліотеки з графікою для роботи з екраном, створення графічних інтерфейсів	Множина бібліотек для обробки зображень (Pillow, OpenCV, PyAutoGUI)	Бібліотеки графіки (AWT, Swing), але для складних задач необхідний додатковий код та модулі
Підтримка ОС	Інтеграція з Windows API для роботи з екраном і іншими системними компонентами	Підтримка роботи з компонентами, але потребує для цього специфічні бібліотеки	Підтримка багатьох платформ завдяки JVM, але взаємодія з ОС потребує додаткових бібліотек
Легкість у використанні	Зрозуміла і чітка синтаксична структура для роботи	Проста і доступна синтаксична структура	Мова є менш гнучкою через більш складний синтаксис, структуру і налаштування
Гнучкість	Легка інтеграція з бібліотеками, системами для адаптивних рішень	Висока гнучкість завдяки великій кількості сторонніх бібліотек та модулів	Добре підтримує розробку великих застосунків, але інтеграція з іншими системами є більш складною

5

Рисунок Д.6 – Слайд 6

Вибір середовища розробки

Критерії	Microsoft Visual Studio	JetBrains Rider	VS Code
Продуктивність	Підтримка великих проєктів, швидка компіляція коду	Трохи зменшена продуктивність роботи з великими проєктами	Легке та швидке IDE, але інколи потребує додаткові налаштування
Інструменти для налагодження	Потужний відлагоджувач, інтеграція з інструментами	Вбудовані інструменти надають можливість для налагоджування та профілювання проєктів	Підтримка через розширення, але не така глибока, як у інших повноцінних IDE
Підтримка бібліотека та фреймворків	Легка інтеграція з .NET бібліотеками, пакетами NuGet	Легка інтеграція з бібліотеками через JetBrains репозиторії	Підтримка .NET із розширенням, але потребує детальне налаштування
Кросплатформність	Підтримка Linux, macOS, Windows	Підтримка Linux, macOS, Windows	Підтримка Linux, macOS, Windows
Зручність використання	Інтуїтивно зрозумілий інтерфейс, велика документація, сучасні інструменти для роботи	Легкий у використанні, зручний інтерфейс для розробки програм	Мінімалістичний інтерфейс, наявна потреба у налаштуванні багатьох плагінів для роботи

6

Рисунок Д.7 – Слайд 7

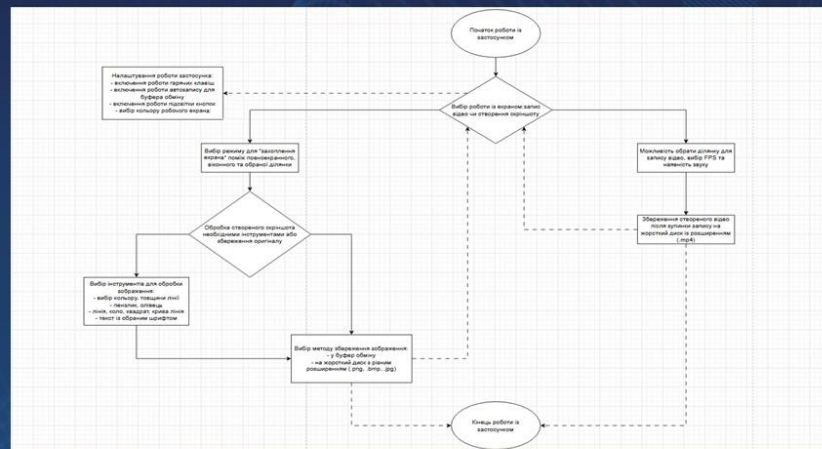
Вибір GUI

Критерії	WPF	WinForms
Графічна система	DirectX	GDI+
Продуктивність	Продуктивність є кращою для складних UI (прискорене GPU)	Повільніша робота на складних сценах
Простота розробки	Складність освоїти через архітектуру (XAML, MVVM)	Примітивні і легкозасвоювані для використання UI-елементи
Робота із зображеннями	Є кращим варіантом для роботи зі складною графікою	Ідеальна взаємодія для простих дій з нескладною графікою
Сумісність із системами	Працює тільки на нових версіях ОС Windows	Підтримка включає в себе старі версії ОС Windows
Складність інтеграції зі сторонніми бібліотеками	Складність інтеграції з GDI+, FFmpeg, DirectShow	Легкість інтеграції з FFmpeg, DirectShow

7

Рисунок Д.8 – Слайд 8

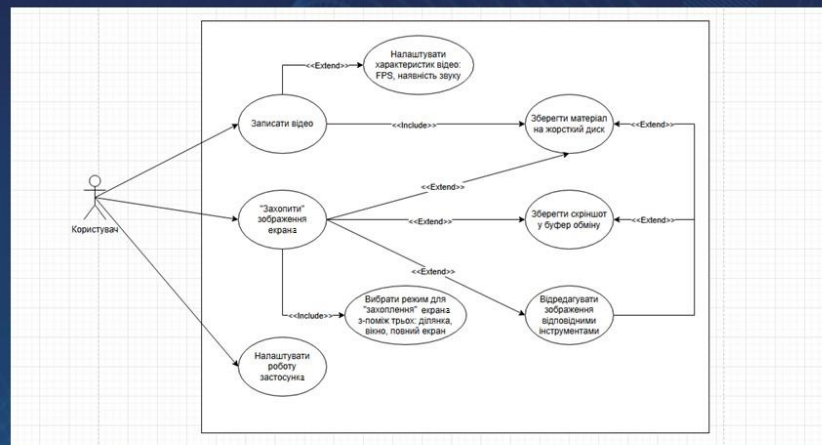
Схема послідовності роботи програми



8

Рисунок Д.9 – Слайд 9

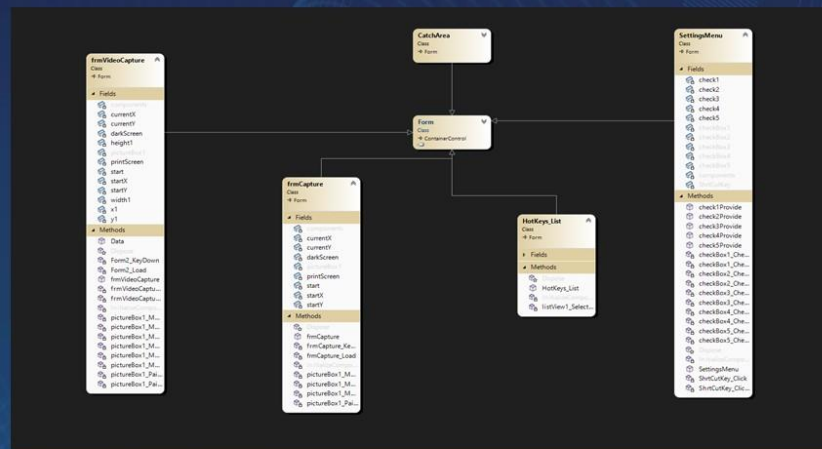
Діаграма use-case роботи застосунка



9

Рисунок Д.10 – Слайд 10

Розроблені програмні модулі



10

Рисунок Д.11 – Слайд 11

Розроблені програмні алгоритми

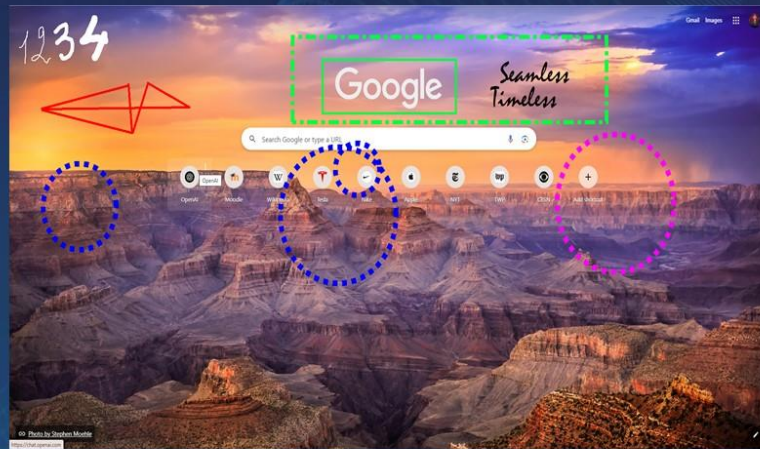
Для функціонування програмного забезпечення було розроблено такі основні алгоритми:

- алгоритм роботи з трьома режимами захоплення зображення (вікно програми, виділена ділянка, повноекранний знімок) з можливістю використання таймера;
- алгоритм для редагування зображення за допомогою олівця, пензля, геометричних фігур (пряма та крива лінії, коло, квадрат тощо) з можливістю налаштування ширини, кольору лінії;
- алгоритм налаштування роботи програми за допомогою користувацького меню;
- алгоритм запису відео без та зі звуком з налаштуванням FPS та можливістю вибору ділянки чи повного екрану;
- алгоритм використання комбінацій гарячих клавіш для роботи у застосунку;
- алгоритм збереження зображення та відео на комп'ютер з вибором відповідних розширень, наявна можливість зберегти знімок екрана до буферу обміну.

11

Рисунок Д.12 – Слайд 12

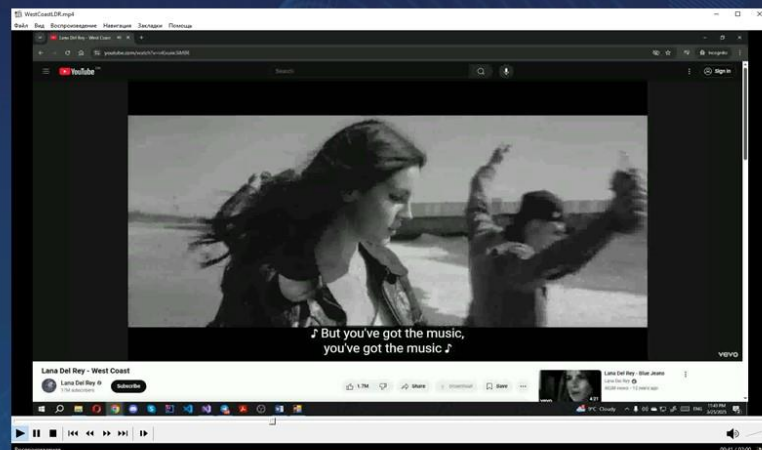
Результати роботи програми



12

Рисунок Д.13 – Слайд 13

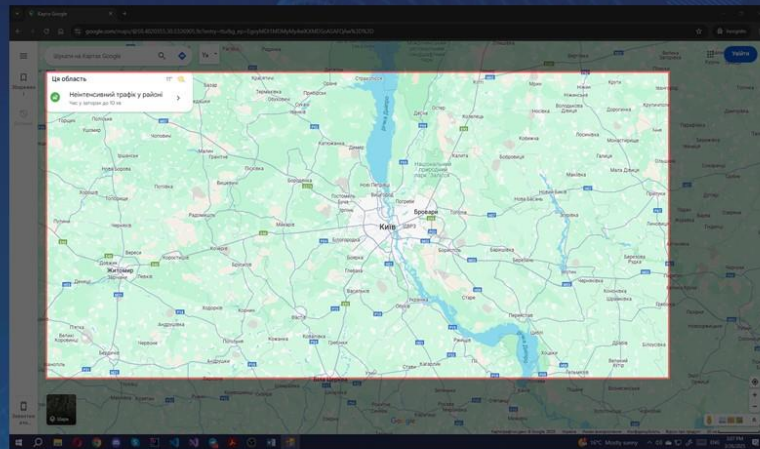
Результати роботи програми



13

Рисунок Д.14 – Слайд 14

Результати роботи програми



14

Рисунок Д.15 – Слайд 15

Висновки

В ході виконання дипломної кваліфікаційної роботи бакалавра було розроблено програмне забезпечення для захоплення зображень з екрана, запису відео та нанесення графічних позначок. Для цього було виконано ряд завдань: аналіз існуючих методів захоплення зображень та відео, вибір оптимальної мови програмування та технології для реалізації застосунку, розроблено архітектуру програмного забезпечення, реалізовано інтерфейс користувача та основні функціональні можливості програми.

Розроблений застосунок був реалізований на платформі .NET Framework з використанням C# та Windows Forms. Для роботи з графічними зображеннями застосовувалася бібліотека GDI+, для обробки звуку – NAudio.Wave, для запису відео – Accord.Video.FFmpeg.

Застосунок надає користувачам можливість створити скріншоти екрана у різних режимах, записувати відео з можливістю вибору ділянки та частоти кадрів у секунду, наявності чи відсутності аудіо, оброблювати зроблені знімки екрана (лінії, кола, прямокутники, текст), зберігати створені файли у різних форматах.

Розроблений програмний продукт успішно виконує всі поставлені задачі, відповідаючи як функціональним, так і інтерфейсним вимогам. Впровадження інтеграції з існуючими бібліотеками та системними компонентами забезпечило високу продуктивність, стабільність роботи та відповідність необхідним стандартам безпеки й надійності.

Створення цього застосунку є важливим кроком у розвитку технологій захоплення і обробки графічної інформації. Програма сприяє підвищенню якості роботи з візуальними даними, збільшує ефективність процесів обробки, збереження та управління зображеннями та відео, що відкриває нові можливості для подальшого розширення її функціональності в майбутньому.

Рисунок Д.16 – Слайд 16