

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»
Факультет інформатики та обчислювальної техніки
(повна назва інституту/факультету)

Кафедра інформатики та програмної інженерії
(повна назва кафедри)

«До захисту допущено»

Світлана ПОПЕРЕШНЯК

(підпис)

(ім'я прізвище)

“ ” 2023 р.

Курсова робота

з дисципліни “Компоненти програмної інженерії” за освітньо-професійною
програмою «Інженерія програмного забезпечення інформаційних систем»
спеціальності «121 Інженерія програмного забезпечення»

на тему: Веб-редактор зображень

Виконав

студент III курсу, групи

ІП-11

(шифр групи)

Прищепя Владислав Станіславович

(прізвище, ім'я, по батькові)

(підпис)

Керівник

доцент, к.ф.м.н., доц., Поперешняк С.В.

(посада, науковий ступінь, вчене звання, прізвище та ініціали)

(підпис)

Консультант

ст. вик., Головченко М.М.

(посада, науковий ступінь, вчене звання, прізвище та ініціали)

(підпис)

Засвідчую, що у цій дипломній роботі
немає запозичень з праць інших авторів без
відповідних посилань.

Студент _____

(підпис)

Київ – 2023

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Дисципліна Компоненти програмної інженерії

Спеціальність 121 «Інженерія програмного забезпечення»

Освітньо-професійна програма програмне забезпечення інформаційних управляючих систем та технологій

ЗАТВЕРДЖУЮ

(підпис)

Світлана ПОПЕРЕШНЯК

(ім'я прізвище)

“ ” 2023 р.

ЗАВДАННЯ на курсову роботу студента

Прищепи Владислава Станіславовича

(прізвище, ім'я, по батькові)

- Тема роботи: веб-редактор зображень
керівник роботи: Поперешняк Світлана Володимирівна, к.ф.м.н., доцент
- Строк здачі студентом закінченої роботи: “28” грудня 2023 року
- Вихідні дані до роботи: технічне завдання
- Зміст розрахунково-пояснювальної записки (перелік питань, які підлягають розробці)

1) Аналіз вимог до програмного забезпечення: основні визначення та терміни,

опис предметного середовища, огляд існуючих технічних рішень та відомих

програмних продуктів, розробка функціональних та нефункціональних вимог

2) Моделювання та конструювання програмного забезпечення: опис BPNM-діаграми,

опис MVC архітектури, розробка UML-діаграми компонентів, схема структурна станів

Інтерфейсу, креслення вигляду екранних форм, опис структур даних

3) Тестування програмного забезпечення: опис процесів тестування функціональних і нефункціональних вимог

4) Впровадження та супровід програмного забезпечення: опис процесу білдингу застосунку

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

схема роботи веб-редактору зображень, діаграма використання, схема процесу отримання відгуку веб-редактору зображень, діаграма компонентів веб-редактору зображень, діаграма класів веб-редактору зображень,

6. Дата видачі завдання: “2” листопада 2023 року

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів курсової роботи	Термін виконання етапів роботи	Підписи керівника, студента
1.	Вивчення рекомендованої літератури	до 6.11.2023	
2.	Аналіз існуючих методів розв’язання задачі	6.11.2023 - 13.11.2023	
3.	Постановка та формалізація задачі	13.11.2023 - 13.11.2023	
4.	Аналіз вимог до програмного забезпечення	13.11.2023- 13.11.2023	
5.	Алгоритмізація задачі	13.11.2023- 13.11.2023	
6.	Моделювання програмного забезпечення	14.11.2023- 15.11.2023	
7.	Обґрунтування використовуваних технічних засобів	15.11.2023- 15.11.2023	

8.	<i>Розробка архітектури програмного забезпечення</i>	<i>16.11.2023- 17.11.2023</i>	
9.	<i>Розробка програмного забезпечення</i>	<i>17.11.2023- 24.11.2023</i>	
10.	<i>Налагодження програми</i>	<i>24.11.2023- 25.11.2023</i>	
11.	<i>Виконання графічних документів</i>	<i>25.12.2023- 25.12.2023</i>	
12.	<i>Оформлення пояснювальної записки</i>	<i>25.12.2023- 28.12.2023</i>	
13.	<i>Подання КР на перевірку</i>	<i>28.12.2023</i>	
14.	<i>Захист КР</i>	<i>29.12.2023</i>	

Студент _____

(підпис)

Прищеп Владислав Станіславович _____

(прізвище, ім'я, по батькові)

Керівник _____

(підпис)

Поперешняк Світлана Володимирівна _____

(прізвище, ім'я, по батькові)

АНОТАЦІЯ

Пояснювальна записка дипломного проєкту складається з чотирьох розділів, містить 38 таблиць, 34 рисунків та 13 джерел. Обсяг основної частини складає 70 сторінок.

Курсова робота присвячена розробці веб-редактору зображень для обробки та збереження зображень у веб-середовищі.

Мета: розробка веб-редактору зображень для розширення функціоналу обробки зображень за допомогою створення додаткових функцій.

У розділі аналізу вимог були визначені вимоги до програмного забезпечення.

У розділі моделювання програмного забезпечення була представлена архітектура програмного продукту, а також програмні структури та структура даних, що використовуються в додатку.

У розділі аналізу якості були описані основні тест-кейси та стани системи після проведення тестування.

У розділі впровадження та супроводу було розглянуто процес розгортання веб-редактору зображень у браузері, таких, як Opera.

КЛЮЧОВІ СЛОВА: ВЕБ-РЕДАКТОР ЗОБРАЖЕНЬ, ЗОБРАЖЕННЯ, БРАУЗЕР, ПОЛОТНО, MODULE.

ЗМІСТ

ЗМІСТ.....	1
ВСТУП.....	2
1 АНАЛІЗ ВИМОГ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ.....	3
1.1 Загальні положення.....	3
1.2 Аналіз успішних ІТ-проектів.....	5
1.2.1 Online Image Editor [1].....	6
1.2.2 Canva [2].....	7
1.2.3 Fotor [3].....	7
1.2.4 I Love IMG [4].....	8
1.2.5 BeFunky [5].....	9
1.3 Порівняння існуючих програмних аналогів.....	9
1.4 Актуальність розробки власного програмного засобу.....	12
1.5 Аналіз вимог до програмного забезпечення.....	12
1.5.1 Функціональні вимоги.....	13
1.5.2 Нефункціональні вимоги.....	15
1.5.3 Постановка задачі.....	16
1.6 Висновки до розділу.....	17
2 МОДЕЛЮВАННЯ ТА КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ.....	18
2.1 Моделювання та аналіз програмного забезпечення.....	18
2.2 Архітектура програмного забезпечення.....	20
2.3 Конструювання програмного забезпечення.....	22
2.4 Конструювання структури станів інтерфейсу.....	35
2.5 Креслення вигляду екранних форм.....	36
2.6 Висновки до розділу.....	39
3 ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ.....	40
3.1 Опис процесів тестування.....	40
3.2 Висновки до розділу.....	66
4 ВПРОВАДЖЕННЯ І СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ.....	67
4.1 Розгортання програмного забезпечення.....	67
4.2 Висновки до розділу.....	67
ВИСНОВКИ.....	68
ПЕРЕЛІК ПОСИЛАНЬ.....	70
ДОДАТОК А.....	1
7 ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ.....	11
ДОДАТОК Б.....	1
ДОДАТОК В.....	1
ДОДАТОК Г.....	1

ВСТУП

Редактори зображень — це програми, що дозволяють творчо працювати з фотографіями та графікою. Вони надають можливість редагувати, покращувати та трансформувати зображення за допомогою різноманітних інструментів. Від базових функцій, таких як обрізка та регулювання яскравості, до складних фільтрів, шарів та масок — ці програми стали необхідним інструментом для фотографів, дизайнерів та всіх, хто цікавиться обробкою зображень. Вони відкривають нескінченні можливості для творчості та експериментів, дозволяючи перетворювати звичайні фотографії в шедеври мистецтва.

Завдяки стрімкому розвитку технологій і доступності Інтернету, веб-редактори зображень стали надзвичайно актуальними. Вони пропонують користувачам можливість працювати з фотографіями без необхідності завантаження складних програм на свої пристрої. Це зручно для тих, хто працює на різних пристроях або не має доступу до потужних програмних засобів. Веб-редактори зображень зазвичай мають інтуїтивний інтерфейс, дозволяють зберігати робочі проекти в хмарних сервісах та спрощують спільну роботу над зображеннями, зокрема колаборацію та обмін робочими файлами. Крім того, веб-редактори постійно оновлюються, додаючи нові функції та можливості, які раніше були доступні лише у важких програмах для редагування зображень.

У даній курсовій роботі буде проведений аналіз існуючих рішень та обґрунтована необхідність розробки нового веб редактора зображень, яка відповідає вимогам сучасності та враховує актуальні тенденції в області обробки зображень.

1 АНАЛІЗ ВИМОГ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Аналіз вимог програмного забезпечення є ключовим етапом у розробці програмних продуктів та систем. Цей процес спрямований на збір, аналіз і формалізацію вимог, необхідних для успішного функціонування програми чи системи.

Ключовою метою аналізу вимог є створення документації, яка чітко описує функціональність та характеристики майбутнього програмного забезпечення. Ця документація служить основою для всіх наступних етапів розробки, допомагаючи уникнути непорозумінь та забезпечуючи консистентність у розумінні вимог. Завдяки правильному та повному аналізу вимог вдається забезпечити успішну та ефективну розробку програмного забезпечення, яке відповідає потребам користувачів та бізнес-цілям.

У даній курсовій роботі висвітлено процес аналізу вимог до програмного забезпечення. Розглядаються загальні положення щодо визначення вимог, їхньої класифікації та впливу на подальший процес розробки. Детально розглядається аналіз успішних аналогів, а також порівняння їхніх характеристик та особливостей, що служить основою для формулювання вимог до нового програмного продукту.

1.1 Загальні положення

У даному розділі будуть розглянуті проблеми, які існують у площині створення веб редакторів зображень та існуючі методи їх вирішення.

Веб-редактори зображень є онлайн-інструментами, спеціально розробленими для обробки та редагування фотографій і зображень без необхідності завантаження або встановлення спеціалізованих програм на комп'ютер чи пристрій. Ці редактори надають користувачам доступ до різноманітних інструментів обробки, що дозволяють вносити зміни до зображень, використовуючи лише веб-браузер та підключення до Інтернету.

Принцип роботи веб-редакторів зображень базується на використанні онлайн-інтерфейсу, який надає доступ до різноманітних інструментів для редагування зображень. Користувачі можуть використовувати інструменти для обрізки, корекції кольору, налаштування яскравості та контрастності, застосування фільтрів, а також для ретуші та маніпуляцій з графікою.

Веб-редактори зображень мають кілька переваг, які роблять їх привабливими для користувачів з різних сфер та рівнів досвіду. Ось деякі з найважливіших переваг веб-редакторів зображень:

- доступність через веб-браузер з будь-якого пристрою, де є підключення до Інтернету, що робить їх дуже зручними для користувачів, які шукають швидкий доступ до редагування фотографій без необхідності завантаження або встановлення додаткового програмного забезпечення;
- веб-редактори зазвичай мають простий та легкий у використанні інтерфейс, що дозволяє швидко оволодіти базовими та складнішими функціями обробки зображень навіть для початківців;
- веб-редактори часто оновлюються з додаванням нових функцій та можливостей, тому користувачі мають доступ до постійно розвиваючихся інструментів для редагування;
- немає необхідності завантажувати та встановлювати великі програми на свій комп'ютер чи пристрій, оскільки робота з веб-редакторами відбувається онлайн;
- зображення можуть бути зручно збережені та використані у різних форматах, що полегшує обмін та використання їх у різних програмах та пристроях.

Незважаючи на багато переваг, веб-редактори зображень також мають кілька недоліків, які варто враховувати:

- у порівнянні з потужними програмами для редагування зображень, веб-редактори можуть мати обмежені можливості. Вони не завжди мають

такий же функціонал та продуктивність, що можуть запропонувати локальні програми;

- робота з веб-редакторами вимагає постійного підключення до Інтернету. Це може бути проблематичним у випадках обмеженого або відсутнього доступу до мережі;
- завантаження та виконання деяких операцій зображення можуть займати більше часу через обмежену продуктивність веб-редакторів у порівнянні з локальними програмами;
- обробка великих зображень в веб-редакторах може бути ускладненою через обмежену пропускну здатність Інтернету;
- деякі веб-редактори можуть мати обмежені можливості роботи зі шарами та складними елементами, що робить їх менш зручними для професійного редагування.

Незважаючи на ці обмеження, веб-редактори зображень продовжують розвиватися, надаючи користувачам все більше функцій та зручний інтерфейс для редагування та обробки зображень. Вони стають все більш популярними серед користувачів, які шукають швидкі та зручні способи редагування фотографій та графічних зображень без необхідності встановлення складних програм на свої пристрої.

1.2 Аналіз успішних ІТ-проектів

У цьому розділі ми ретельно аналізуємо успішні ІТ-проекти. Розглядаючи їх переваги та недоліки, ми визначимо ключові аспекти, які можуть визначати ефективні рішення в сучасному ІТ-середовищі. Це вивчення стане важливою основою для подальшого розгляду власного веб редактору зображень.

Успішність веб-редакторів зображень може бути оцінена за декількома критеріями:

- **Функціональність:** Це один із ключових аспектів. Успішний веб-редактор повинен мати широкий спектр функцій і інструментів для редагування,

таких як обрізка, корекція кольору, ретуш, застосування фільтрів, робота з шарами тощо.

- Інтерфейс та користувацький досвід: Інтуїтивний та зрозумілий інтерфейс є важливим аспектом для успішності. Користувальницький досвід має бути зручним і приємним, щоб користувачі могли легко зрозуміти, як використовувати доступні функції.
- Швидкість та продуктивність: Успішність веб-редактора зображень також можна виміряти швидкістю та ефективністю роботи. Важливо, щоб обробка зображень відбувалася швидко, а переходи між різними функціями були плавними.
- Стабільність та підтримка: Успішний веб-редактор повинен мати стабільну роботу та регулярну підтримку та оновлення для виправлення помилок та додавання нових функцій.

Всі ці критерії враховуються для визначення та оцінки успішності веб-редакторів зображень у сучасному ІТ-середовищі.

1.2.1 Online Image Editor [1]

Компанія-виробник: © Marcel.

Основні функціональні можливості:

- Застосунок підтримує два основних формати зображень - .png і .jpg.
- Дозволяє змінювати розмір зображень, що включає збільшення або зменшення їх розміру, вирізати зображення прямокутної форми без втрати якості і повертати їх.
- Надає можливість додавати текстові елементи на зображення.

Переваги програмної системи:

- Простий інтерфейс, який надає приємний досвід користувачу.
- Висока швидкість виконання операційних.
- Стабільна робота (при наявності стабільного інтернет зв'язку у користувача).

Недоліки програмної системи:

- Відсутні певні базові інструменти обробки зображень, як малювання ліній.
- Відсутній набір фільтрів для обробки зображення.
- Наявні інструменти обробки зображень не надають продвинутих можливостей для обробки зображень.

1.2.2 Canva [2]

Компанія-виробник: © Canva.

Основні функціональні можливості:

- Застосунок підтримує два основних формати зображень - .png і .jpg.
- Надає можливість додавати текстові та геометричні елементи на зображення, малювати лінії.

Переваги програмної системи:

- Висока швидкість виконання операційних.
- Стабільна робота (при наявності стабільного інтернет зв'язку у користувача).
- Наявний широкий вибір шаблонів фігур, рамок для тексту.

Недоліки програмної системи:

- Інтерфейс не дуже структурований, через що він не є зручним.
- Відсутні такі базові функції обробки зображень, як обрізання, повертання та масштабування.
- Наявні інструменти обробки зображень не надають продвинутих можливостей для обробки зображень.

1.2.3 Fotor [3]

Компанія-виробник: © 2023 Everimaging.

Основні функціональні можливості:

- Застосунок підтримує два основних формати зображень - .png і .jpg.

- Дозволяє змінювати розмір зображень, що включає збільшення або зменшення їх розміру, вирізати зображення прямокутної форми без втрати якості і повертати їх, застосовувати фільтри.
- Надає можливість додавати текстові та геометричні елементи на зображення, малювати лінії.

Переваги програмної системи:

- Простий інтерфейс, який надає приємний досвід користувачу.
- Стабільна робота (при наявності стабільного інтернет зв'язку у користувача).
- Широкий функціонал для обробки зображень.

Недоліки програмної системи:

- Наявні нарікання на швидкість застосування деяких з наявних інструментів для обробки зображень.

1.2.4 I Love IMG [4]

Компанія-виробник: © iLoveIMG 2023.

Основні функціональні можливості:

- Застосунок підтримує два основних формати зображень - .png і .jpg.
- Дозволяє змінювати розмір зображень, що включає збільшення або зменшення їх розміру, вирізати зображення прямокутної форми без втрати якості і повертати їх, застосовувати фільтри.
- Надає можливість додавати текстові та геометричні елементи на зображення, малювати лінії.

Переваги програмної системи:

- Простий інтерфейс, який надає прийємний досвід користувачу.
- Висока швидкість виконання операційних.
- Стабільна робота (при наявності стабільного інтернет зв'язку у користувача).

Недоліки програмної системи:

- Наявні інструменти обробки зображень не надають продвинутих можливостей для обробки зображень.
- Відсутність вибору, у якому форматі зберігати зображення.

1.2.5 BeFunky [5]

Компанія-виробник: © 2023 BeFunky Inc.

Основні функціональні можливості:

- Застосунок підтримує два основних формати зображень - .png і .jpg.
- Дозволяє змінювати розмір зображень, що включає збільшення або зменшення їх розміру, вирізати зображення прямокутної форми без втрати якості і повертати їх, застосовувати фільтри.
- Надає можливість додавати текстові та геометричні елементи на зображення, малювати лінії.

Переваги програмної системи:

- Простий інтерфейс, який надає прийємний досвід користувачу.
- Висока швидкість виконання операційних.
- Широкий список фільтрів для обробки зображень.

Недоліки програмної системи:

- Наявні недоліки під час використання застосунку, як от не виконання функцій (не стабільна робота).
- Наявні інструменти обробки зображень не надають продвинутих можливостей для обробки зображень.

1.3 Порівняння існуючих програмних аналогів

Розглянувши наявні успішні ІТ-проекти, складемо таблицю порівня.

Таблиця 1.1 Порівняння існуючих програмних аналогів

Назва продукту	Online Image Editor	Canva	Fotor	I Love IMG	BeFunky
Компанія-виробник	© Marcel	© Canva	© 2023 Everimagin	© iLoveIMG	© 2023

			g	2023	BeFunky Inc
Підтримка основних форматів зображення, як от .png і .jpg	+	+	+	+	+
Обрізання зображення	+	-	+	+	+
Вирізання фрагменту прямокутної форми і подальшої вставки у зображення	-	-	-	-	-
Зміна розміру зображення	+	-	+	+	+
Повертання зображення	+	-	+	+	+
Малювання	-	+	+	+	+

я ліній					
Накладання зображень	-	-	+	-	+
Наявність фільтрів	-	-	+	+	+
Функція спотворення зображення	-	-	-	-	-
Додання тексту	+	+	+	+	+
Додання фігур	-	+	+	+	+
Швидкість роботи висока	+	+	-	+	+
Робота стабільна	+	+	+	+	-
Інтерфейс зручний	+	-	+	+	+

Важливо зазначити, що для прикладу відсутності більш продвинутих засобів обробки зображень, таких, як вирізання фрагменту прямокутної форми і подальшої вставки у зображення і функція спотворення зображення.

Загалом видно, що, не враховуючи мої приклади більш продвинутих інструментів, більшість існуючих аналогів або не мають повний базовий

функціонал для обробки зображень, або мають недоліки під час використання, як от нестабільна робота або низька швидкість виконання.

1.4 Актуальність розробки власного програмного засобу

Розробка власного веб редактору зображень може бути актуальною з ряду причин:

- сумісність з різними браузерми: веб редактори зображень, при застосуванні відповідних бібліотек, можуть бути легко переносимі між різними браузерми; це робить розробку більш універсальною та ефективною;
- розвиток функціональності: розробка веб-редактору зображень є відмінною можливістю створити застосунок із розширеними функціональними можливостями, які можуть відповідати конкретним потребам користувачів чи проекту; це особливо корисно у випадках, коли існуючі веб-редактори зображень не влаштовують за вимогами;
- експериментація та навчання: розробка веб-редактору зображень може слугувати відмінною можливістю для навчання та експериментації в області веб-програмування.

1.5 Аналіз вимог до програмного забезпечення

Програмна система, що розробляється, спрямована на створення веб редактору зображень, який надає зручний та гнучкий інтерфейс для обробки зображень. Основною метою цього проекту є створення інструменту, який дозволяє користувачам легко маніпулювати зображеннями, використовуючи графічний інтерфейс, використовуючи для цього Інтернет.

Функціональні завдання та мета програмної системи сфокусовані на створенні веб редактору зображень з метою надання користувачам зручного інтерфейсу для взаємодії з файлами зображень. Головною метою є забезпечення функціональних можливостей для створення та зміни зображень шляхом малювання ліній, накладання фігур, інших зображень, тексту та фільтрів, зміни

розміру, ракурсу та обрізання їх. Це включає в себе можливість вирізання фрагменту прямокутної форми і подальшої вставки у зображення і функція спотворення зображення. Подальшою метою є створення зручного та функціонального інструменту для користувачів, який може бути використаний для навчання або вирішення побутових завдань, забезпечуючи при цьому гнучкість та ефективність взаємодії із файлом зображення через графічний інтерфейс у мережі Інтернет.

1.5.1 Функціональні вимоги

Функціональні вимоги — це конкретні функції чи сервіси, які система чи 14 програмне забезпечення повинні надавати для вирішення конкретних завдань та вимог користувачів. Ці вимоги описують очікувану функціональність та здатність системи виконувати певні операції. Функціональні вимоги зазвичай формулюються як конкретні задачі чи дії, які користувачі системи повинні мати можливість виконати.

Таблиця 1.2 Функціональні вимоги до додатку

Номер	Назва	Опис
R-1	Завантаження зображення	Додаток мусить надавати можливість завантажити зображення з клієнтського сервера у додаток з метою подальшої обробки. Додаток мусить мати можливість завантажити файли зображень декількох різних форматів.
R-2	Створення нового зображення	Додаток мусить надавати можливість створити нове біле зображення з метою подальшої обробки. Додаток мусить надати можливість обрати користувачу розміри порожнього зображення.
R-3	Збереження змін	Додаток мусить зберігати проміжкові стани обробки зображення у вигляді певної послідовності станів зображень. Проміжкові стани обробки

		зображень — стани зображень після виконання кожної дії користувачем, яка змінює зображення і початковий стан після створення чи завантаження.
R-4	Відміна останньої дії	Додаток мусить надавати можливість користувачеві відмінити останню дію користувача шляхом повернення до попереднього стану зображення і видалення стану, з якого було виконане повернення у попередній стан, при цьому за одне застосування зображення мусить повертатися на один стан назад.
R-5	Очищення зображення	Додаток мусить повернути зображення на початковий стан, видаливши усі інші проміжкові стани обробки зображення.
R-6	Завантаження зображення	Додаток мусить надавати можливість завантажувати із додатку поточне зображення у вигляді файлу. При цьому необхідно надати вибір формату файлу для збереження.
R-7	Малювання на зображенні	Додаток мусить надавати можливість малювання ліній на зображенні. При цьому необхідно надати вибір кольору та товщини ліній, а також виду ліній (довільна чи пряма).
R-8	Спотворення зображення	Додаток мусить надавати можливість витягування зображення шляхом стискання зображення.
R-9	Застосування фільтрів	Додаток мусить надавати можливість застосовувати кольорокорекційні фільтри над зображенням з метою зміни стилю зображення. При цьому певні фільтри можуть мати параметри, які регулюють силу ефекту застосованого фільтра.

R-10	Додавання об'єктів	Додаток мусить надавати можливість накладати геометричні фігури, текст та інші завантажені зображення на початкове зображення. При цьому параметри кожного об'єкта мусять мати можливість бути зміненими перед вставкою у зображення, а послідовність накладання об'єктів — під час та після накладання об'єктів.
R-11	Обрізання зображення	Додаток мусить надавати можливість обрізати зображення шляхом виділення обраної зони вручну.
R-12	Повертання зображення	Додаток мусить надавати можливість повертати зображення набік. При цьому за один раз можна повернути зображення на будь-який кут.
R-13	Масштабування зображення	Додаток мусить надавати можливість масштабувати зображення. При цьому необхідно надати можливість надати параметри масштабування вручну.
R-14	Копіювання із зображення	Додаток мусить надавати можливість копіювати виділену вручну область зображення і накладати скопійований фрагмент як вставлене зображення.

Таблиця функціональних можливостей включає основні операції редактора зображень.

1.5.2 Нефункціональні вимоги

Нефункціональні можливості вимоги — це характеристики системи чи програмного забезпечення, які не стосуються конкретної функціональності, але визначають якісні аспекти їхньої роботи та характеристики. Ці можливості визначають "якість" системи та включають такі аспекти, як продуктивність, надійність та інші.

Таблиця 1.3 Нефункціональні вимоги

Номер	Назва	Опис
NR-1	Продуктивність	Система повинна забезпечувати відповідний рівень продуктивності для операцій обробки зображень. Час відповіді системи не повинен перевищувати 1 секунду для більшості запитів.
NR-2	Надійність	Система повинна бути стійкою до помилок та забезпечувати надійність при роботі з операціями. При виникненні помилок, система повинна надавати зрозумілі та інформативні повідомлення користувачеві.
NR-3	Зручність інтерфейсу	Користуватський інтерфейс системи повинен бути інтуїтивно зрозумілим та зручним для використання. Результати обробки мають бути відображені коректно.

Наведені в таблиці вимоги допомагають враховувати ключові аспекти, які забезпечують не тільки функціональну повноту, але й задовольняють якість програмного продукту.

1.5.3 Постановка задачі

Розробка може бути застосована під час розробки інших видів програмного забезпечення та у повсякденній взаємодії із зображеннями.

Цільова аудиторія для веб редактора зображень включає здобувачів початкової, середньої та вищої освіти, звичайних користувачів мережі інтернет, які використовують редактори зображень у своїх проектах.

Також програмне забезпечення має виконувати всі поставлені функціональні та нефункціональні вимоги.

1.6 Висновки до розділу

У даному відділі визначено формулювання задачі, розглянуто наявні аналоги редакторів зображень. Представлені альтернативи вважаються достатньо якісними, але вони мають вади, такі як не достаток базових інструментів обробки зображень чи наявні певні нефункціональні недоліки.

Крім того, в першому розділі проведений аналіз вимог, побудовано таблицю для відповідного сценарію використання та кожної функціональної вимоги. Це дозволило отримати більш детальне уявлення про вимоги до мови програмування та технологій, які повинні бути використані.

У якості мови програмування було обрано HTML, CSS та JavaScript, оскільки за допомогою цих мов програмувань розробляються веб-застосунки.

У якості допоміжних засобів були обрані певні користувацькі бібліотеки для JavaScript, які реалізують певні функції взаємодії із зображеннями на базовому рівні, адже вони перевірені та полегшать розробку власного веб-застосунку.

2 МОДЕЛЮВАННЯ ТА КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

У сучасному інформаційному суспільстві програмне забезпечення виступає важливою складовою для вирішення складних завдань та досягнення стратегічних цілей в різних галузях. Процес розробки програмного забезпечення є складним та багатозадачним, вимагаючи глибокого розуміння вимог користувачів, ефективних методів моделювання та конструювання.

Моделювання та конструювання програмного забезпечення становлять ключові етапи у життєвому циклі розробки програм, де визначаються концепції, архітектура та алгоритми, які лягають в основу майбутнього продукту. Ефективне моделювання дозволяє визначити оптимальні рішення та забезпечити гнучкість системи, а конструювання є процесом втілення цих концепцій у функціональний та надійний програмний продукт.

У даній курсовій роботі детально розглядається процес моделювання та конструювання програмного забезпечення. Аналізуються основні підходи до створення моделей, методи та інструменти, які використовуються у цьому процесі. Покладаючи акцент на важливість етапів моделювання та конструювання, робота пропонує висвітлити основні концепції та вирішення, що визначають успішну розробку програмного забезпечення в сучасному інформаційному середовищі.

2.1 Моделювання та аналіз програмного забезпечення

Використаємо BPMN (Business Process Model and Notation) [6] діаграму для опису бізнес-процесу. BPMN — це стандарт для моделювання бізнес-процесів, який надає уніфікований мовний засіб для спільного розуміння, аналізу та оптимізації бізнес-процесів всередині організації. Використовуючи символи та правила, BPMN дозволяє створювати графічні представлення процесів, що полегшує співпрацю між бізнес-аналітиками та розробниками програмного забезпечення.

Використання Бізнес-процесної мережі (BPNM) дозволяє досягти стандартизації в моделюванні бізнес-процесів. Цей міжнародний стандарт надає єдиний набір термінів та концепцій, що сприяє узгодженій роботі різних команд та організацій.

Використання BPNM спрощує керування проектами, надаючи можливість створювати, редагувати та вдосконалювати бізнес-процеси в єдиному середовищі. Це сприяє ефективній співпраці команд та управлінням змінами в організації.

Отримання зворотнього відгуку від редактора зображень є одним цільним бізнес-процесом, тому на рисунку 2.1 опишемо його за допомогою BPMN.

Опис зворотнього відгуку від редактора зображень:

- Запит на операції з зображеннями починається з виклику зовнішньої програми, яка звертається до внутрішніх методів, що зберігаються на сервері.
- Внутрішні методи програми на сервері обробляють отриманий запит.
- Після обробки запиту внутрішні методи програми на сервері рендерять отриманий стан зображення.
- Після рендеру стану зображення внутрішні методи програми на сервері надсилають відповідь зовнішній програмі. Зовнішня програма отримує результат операції, і обробка запиту завершується.

Моделювання веб-редактору зображень використовуючи BPMN дозволяє чітко визначити її етапи та взаємодію компонентів. Це полегшує аналіз та оптимізацію, а також сприяє ефективній співпраці між учасниками розробки. Модель допомагає виявити можливі ризики та вдосконалити дизайн системи, що призводить до створення надійних та продуктивних рішень.

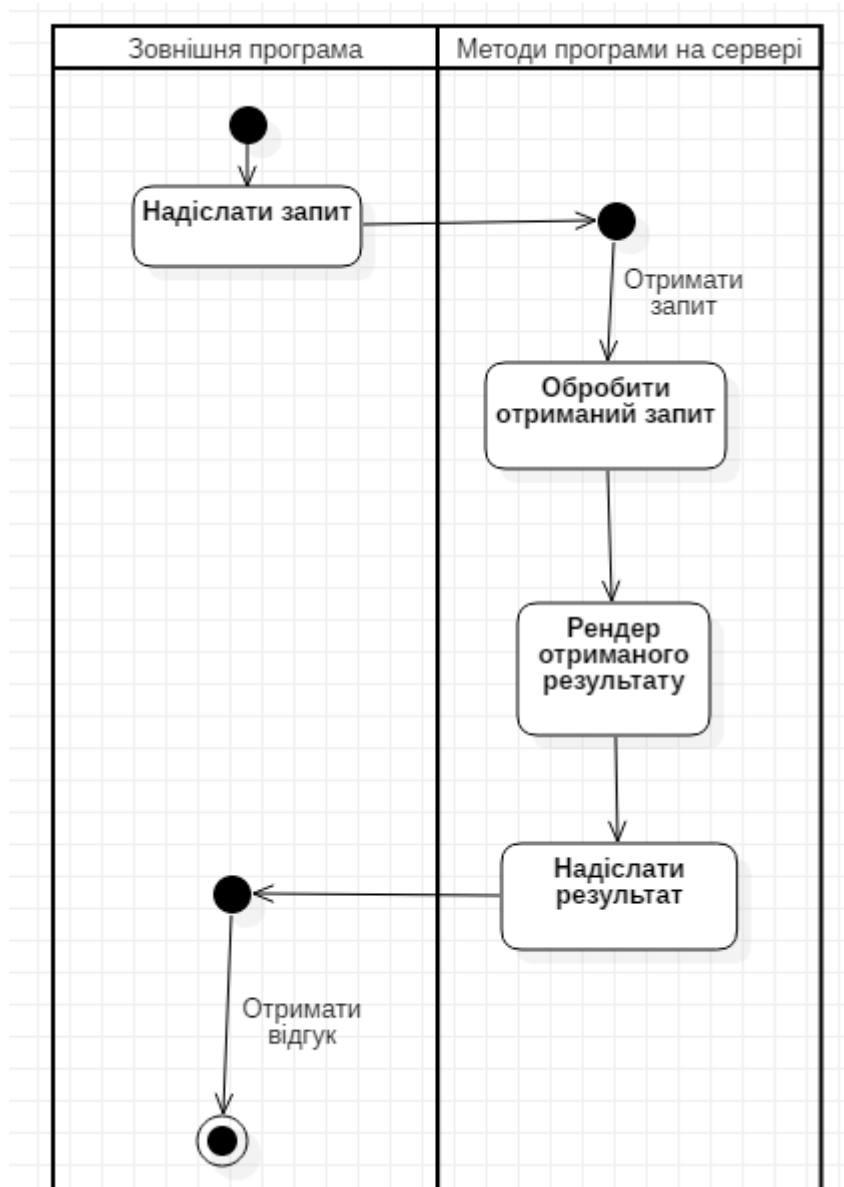


Рисунок 2.1 Схема бізнес-процесу отримання відгуку веб-редактора зображень

2.2 Архітектура програмного забезпечення

В розробці веб-редактора зображень, використання паттернів проектування стає важливим етапом для забезпечення ефективності,

читабельності та легкості управління кодом. Один із таких паттернів - Model-View-Controller (MVC) [7] - дозволяє відокремити представлення, логіку та дані, роблячи архітектуру більш модульною та гнучкою.

Модель у веб-редактора зображень представляє основні концепції, такі як звичайні зображення, JSON файли та методи бібліотеки обробки графічних об'єктів Fabric.js [8]. Кожен з цих елементів відповідає реальній веб-редактора зображень та забезпечує базовий функціонал для взаємодії з користувачем.

Представлення вирішує, як інформація буде відображатися для користувача. У цьому випадку, графічний інтерфейс, реалізований за допомогою мов HTML, CSS та JavaScript [9], служить інструментом взаємодії. Він призначений для зручного введення команд та відображення результатів операцій.

Контролер виконує роль посередника між користувачем та моделлю. Він приймає команди від користувача через графічний інтерфейс та визначає, як ці команди повинні бути оброблені. Після цього він взаємодіє з моделлю для виконання необхідних операцій.

Для кращого розуміння архітектури побудуємо UML-діаграму компонентів. UML (Unified Modeling Language) [10] - це стандартний мовний інструментарій для моделювання об'єктно-орієнтованих систем. UML-діаграма компонентів використовується для візуалізації, спрощення та розуміння взаємодії компонентів програмної системи. Компонент в UML може представляти фізичний або логічний модуль програми.

Розглянемо схему архітектури:

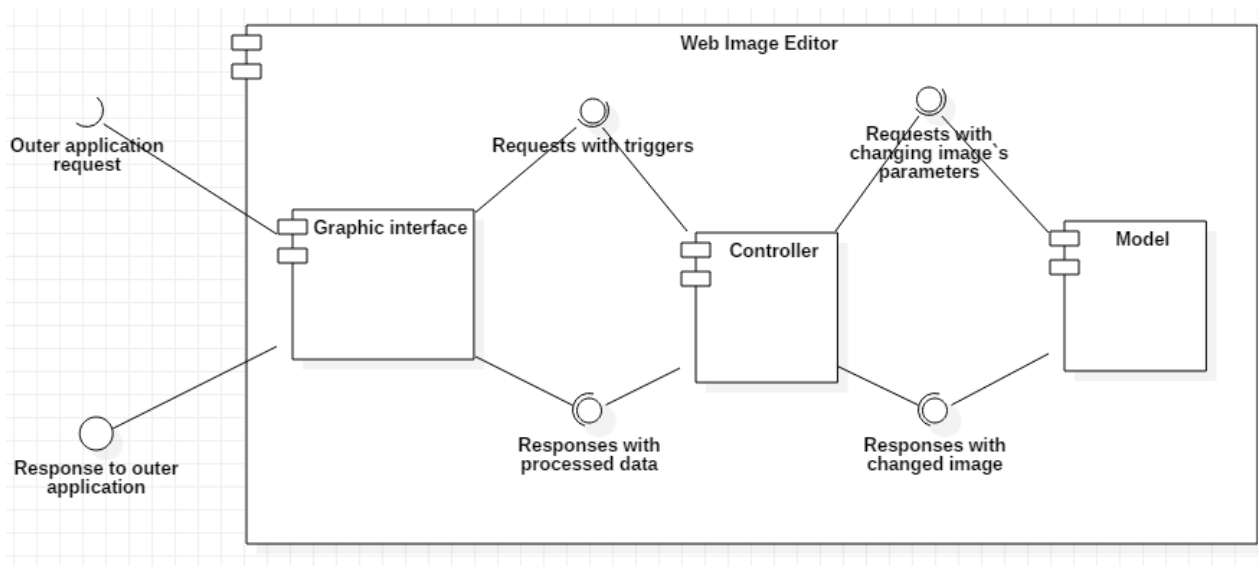


Рисунок 2.2 Діаграма компонентів веб-редактора зображень

Архітектура, побудована за MVC паттерном, дозволяє зберігати код чистим, легко розширювати та модифікувати. Розподіл обов'язків між моделлю, представленням та контролером полегшує роботу над проектом, забезпечуючи оптимальну організацію та легкість управління.

2.3 Конструювання програмного забезпечення

Конструювання програмного забезпечення веб-редактора зображень є критичним етапом у розробці, оскільки від цього залежить ефективність, безпека та функціональність продукту. У даному розділі ми детально розглянемо процес створення веб-редактора зображень та розглянемо ключові аспекти конструювання.

Конструювання веб-редактора зображень має велике значення для досягнення успіху проекту. Цей розділ допоможе розробникам та інженерам краще зрозуміти етапи реалізації віртуальної файлової системи, враховуючи сучасні технології та підходи.

В даному розділі буде проведено детальний аналіз структури даних, яка застосовується у веб-редакторі зображень. У фокусі уваги - проєктовані класи та їх взаємодія, методи, що забезпечують роботу з зображеннями.

Детальний огляд реалізації класів і їхніх методів дозволить виявити сильні та слабкі сторони структури даних, аналізувати оптимальність вибраних рішень та вирішувати проблеми, які можуть виникнути при реалізації веб-редактора зображень.

Паттерн модуль (Module pattern) [11] є шаблоном проектування в програмуванні, який дозволяє створювати простори імен (namespaces) та ізолювати функції, змінні та об'єкти від глобального контексту програми.

Цей паттерн дозволяє створювати приватні змінні та методи, які не доступні ззовні модуля, забезпечуючи контроль доступу до коду і уникнення конфліктів імен у великих програмах. Він сприяє підвищенню безпеки та об'єднанню схожих функцій та змінних в одному місці для зручності управління та розвитку програми.

Переваги використання паттерна Module у веб-редакторі зображень:

- Модуль дозволяє створювати приватні змінні та функції, які не є доступними ззовні. Це сприяє захисту даних та може бути корисним для збереження конфіденційності деяких частин програми.
- Модуль дозволяє групувати схожі функції, змінні та об'єкти в один блок, полегшуючи управління та розуміння коду.
- Модульний підхід сприяє поліпшенню читабельності програми, оскільки код розділяється на логічні блоки, які є легкими для розуміння та підтримки.
- Модульність сприяє створенню відновлюваних, легко масштабованих та перевикористовуваних компонентів, що полегшує розробку програм.

Мова JavaScript дозволяє імплементувати Module паттерн. Для цього потрібно розбити код на логічні модулі, прописавши у головному модулі логіку використання функцій інших модулів. Для цього треба застосовувати тригери та бібліотеку jQuery[12].

jQuery є однією з найпопулярніших JavaScript бібліотек, яка спрощує взаємодію з DOM, анімації, обробку подій, роботу з AJAX і багато іншого. Ось деякі основні переваги та можливості jQuery:

- jQuery забезпечує простий та лаконічний синтаксис, що робить роботу з DOM та JavaScript-функціями більш зрозумілою та менш працездатною.
- jQuery надає широкий набір методів для роботи з DOM-елементами, що полегшує їх вибір, зміну, видалення, створення та маніпулювання ними.
- jQuery пропонує простий спосіб додавання обробників подій для елементів DOM, що дозволяє легко реагувати на дії користувача.

Хоча на початку 2020-х років існувала трендова схильність відмовлятися від jQuery через поліпшення стандартів JavaScript та браузерів, вона залишається корисним інструментом для швидкої розробки та підтримки проектів, особливо для старіших кодових баз або при роботі з багатими DOM-структурами.

Розглянувши методи реалізації Module паттерна, спроектуємо класи для веб-редактора зображень. Для цього опишемо таблиці класів і методів.

Таблиця 2.1 Класи

Назва	Опис
ImageEditor	Головний клас, який реалізовує логіку роботи застосунку, об'єднує об'єкти інших класів.
canvas	Клас, що реалізовує відображення полотна зображення та об'єктів, логіку взаємодії з ними та їх обробки.
canvasSettings	Клас, що реалізує панель налаштувань полотна зображення, як от колір задника чи розмір полотна.
copyPaste	Клас, що реалізує як звичайний функціонал копіювання та вставляння об'єктів, так і функціонал копіювання виділеної області зображення та

	подальшої вставки скопійованого шматка зображення.
lineDrawing	Клас, що реалізовує малювання прямої лінії за допомогою миші на полотні.
textBoxDrawing	Клас, що реалізовує малювання текстового поля за допомогою миші на полотні.
freeDrawSettings	Клас, що реалізовує малювання лінії за допомогою миші на полотні з можливістю вибору кольора та розміру лінії перед її малюванням.
saveInBrowser	Клас, що реалізовує функції збереження, та завантаження стану зображення як початкового.
selectionSettings	Клас, що реалізовує панель налаштувань об'єктів, таких, як фігури, лінії, зображення та текст, після їх додання на полотно шляхом їх вибору за допомогою миші.
shapes	Клас, що реалізовує функцію додання фігур на полотно зображення із переліку існуючих фігур із панелі вибору фігур.
toolbar	Клас, що реалізовує логіку та відображення панелі наявних інструментів для взаємодії із зображенням.
upload	Клас, що реалізовує форму завантаження зображень та її логіку виконання.
utils	Клас, що реалізовує допоміжні утиліти для роботи інших методів класів.
zoom	Клас, що реалізовує функції збільшення та зменшення полотна зображень (zooming).

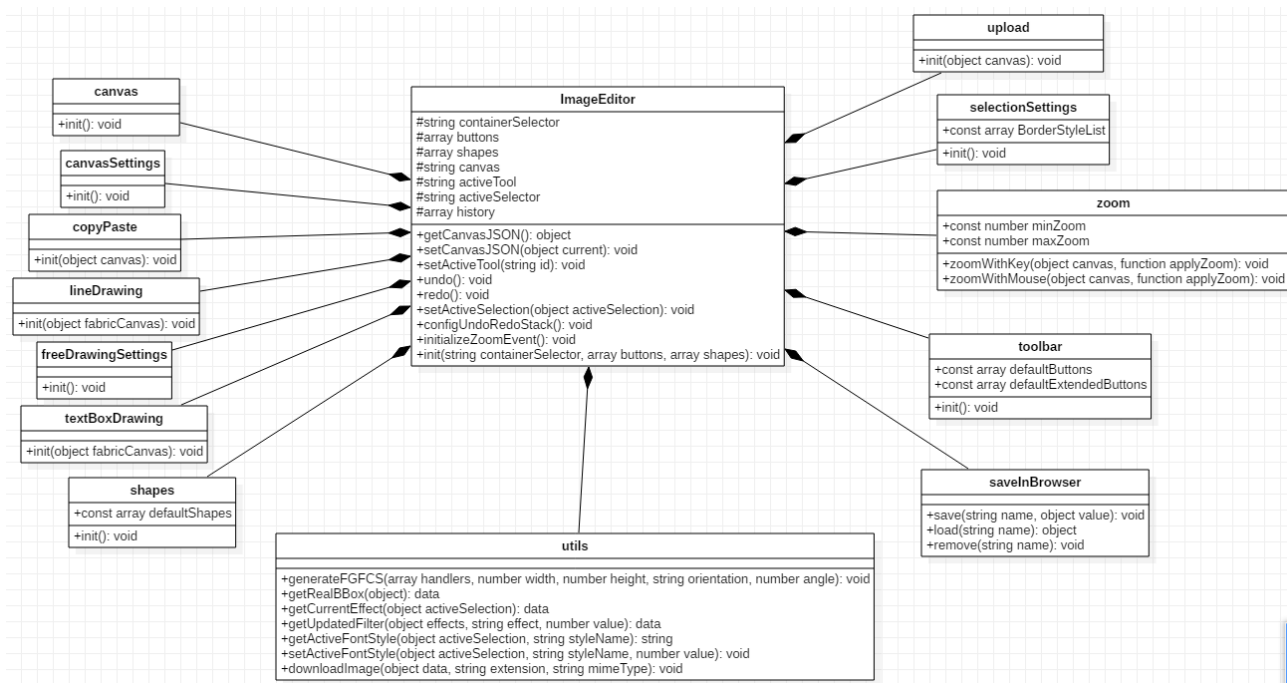


Рисунок 2.3 Діаграма класів

Розглянемо кожен клас детально. Почнемо з класу ImageEditor. Цей клас реалізовує сам редактор зображень. У ньому реалізована логіка взаємодії користувача та функцій, ініціалізуються усі інші об'єкти класів.

Таблиця 2.2 Методи та поля класу ImageEditor

Назва	Тип	Сигнатура	Опис
containerSelector	Захищене поле	containerSelector	Селектор jQuery для контейнеру редактора зображень
buttons	Захищене поле	buttons	Кастомний набір кнопок для редактора зображень
shapes	Захищене поле	shapes	Кастомний набір фігур для редактора зображень
canvas	Захищене поле	canvas	Показчик на полотно зображення

activeTool	Захищене поле	activeTool	Показчик на активний інструмент
activeSelector	Захищене поле	activeSelector	Показчик на активний селектор
history	Захищене поле	history	Історія станів зображення за сесію
getCanvasJSON	Публічний метод	getCanvasJSON()	Перетворює поточний стан полотна у JSON
setCanvasJSON	Публічний метод	setCanvasJSON(current)	Перетворює JSON у поточний стан полотна
setActiveTool	Публічний метод	setActiveTool(id)	Реалізовує логіку перемикання між інструментами за вказаним id
undo	Публічний метод	undo()	Повертає на попередній стан зображення
redo	Публічний метод	redo()	Повертає на наступний стан зображення
setActiveSelection	Публічний метод	setActiveSelection(activeSelection)	Реалізовує логіку вибору селектором об'єкта на полотні
configUndoRedoStack	Публічний метод	configUndoRedoStack()	Реалізовує логіку повертання на стани за допомогою клавіш
initializeZoomEvent	Публічний метод	initializeZoomEvent()	Реалізує логіку приближення чи

			віддалення від полотна зображення
init	Конструктор	init(containerSelector, buttons, shapes)	Приймає кастомні кнопки та форми, селектор контейнеру редактора зображень і ініціалізує усі об'єкти класів, зв'язуючи необхідні функції з відповідними тригерами.

Розглянемо клас canvas, який реалізує полотно зображень.

Таблиця 2.3 Методи та поля класу canvas

Назва	Тип	Сигнатура	Опис
init	Конструктор	init()	Ініціалізує полотно редактора зображень, описує логіку взаємодії із об'єктами на полотні.

Розглянемо клас canvasSettings, який реалізує панель налаштування полотна зображень.

Таблиця 2.4 Методи та поля класу canvasSettings

Назва	Тип	Сигнатура	Опис
init	Конструктор	init()	Ініціалізує панель

			налаштування полотна редактора зображень, описує логіку налаштування полотна.
--	--	--	--

Розглянемо клас `copyPaste`, який реалізує функції копіювання та вставки об'єкта(ів) на полотні.

Таблиця 2.5 Методи та поля класу `copyPaste`

Назва	Тип	Сигнатура	Опис
<code>init</code>	Конструктор	<code>init(canvas)</code>	Приймає об'єкт, реалізує логіку копіювання та вставки на полотні.

Розглянемо клас `lineDrawing`, який реалізує функцію малювання прямої лінії на полотні.

Таблиця 2.6 Методи та поля класу `lineDrawing`

Назва	Тип	Сигнатура	Опис
<code>init</code>	Конструктор	<code>init(fabricCanvas)</code>	Приймає полотно редактора зображень, реалізує логіку малювання прямої лінії мишкою.

Розглянемо клас `freeDrawingSettings`, який реалізує функцію вільного малювання на полотні.

Таблиця 2.7 Методи та поля класу `freeDrawingSettings`

Назва	Тип	Сигнатура	Опис
<code>init</code>	Конструктор	<code>init()</code>	Ініціалізує панель початкового налаштування та функцію вільного малювання на полотні.

Розглянемо клас `textBoxDrawing`, який реалізує функцію додавання текстового поля.

Таблиця 2.8 Методи та поля класу `textBoxDrawing`

Назва	Тип	Сигнатура	Опис
<code>init</code>	Конструктор	<code>init(fabricCanvas)</code>	Приймає полотно редактора зображень, реалізує логіку малювання текстового поля мишкою.

Розглянемо клас `shapes`, який реалізує додавання фігур.

Таблиця 2.9 Методи та поля класу `shapes`

Назва	Тип	Сигнатура	Опис
<code>init</code>	Конструктор	<code>init()</code>	Ініціалізує панель

			вибору фігури та функцію додавання її на полотно.
--	--	--	---

Розглянемо клас `upload`, який реалізує завантаження зображення.

Таблиця 2.10 Методи та поля класу `upload`

Назва	Тип	Сигнатура	Опис
<code>init</code>	Конструктор	<code>init(canvas)</code>	Приймає полотно редактора зображень, реалізує форму та логіку завантаження зображень до полотна зображень

Розглянемо клас `selectionSettings`, який реалізує функцію вибору та налаштування обраного об'єкта.

Таблиця 2.11 Методи та поля класу `selectionSettings`

Назва	Тип	Сигнатура	Опис
<code>init</code>	Конструктор	<code>init()</code>	Ініціалізує панель налаштування об'єктів на полотні редактора зображень та логіку вибору

			об'єкта
BorderStyleList	Публічне статичне поле	const BorderStyleList	Перелік стилів границь об'єктів на полотні редактора зображень

Розглянемо клас toolbar, який реалізує панель вибору інструментів для обробки зображень.

Таблиця 2.12 Методи та поля класу toolbar

Назва	Тип	Сигнатура	Опис
init	Конструктор	init()	Ініціалізує панель початкового налаштування та функцію вільного малювання на полотні.
defaultButtons	Публічне статичне поле	const defaultButtons	Стандартний набір кнопок інструментів обробки зображень.
defaultExtendedButtons	Публічне статичне поле	const defaultExtendedButtons	Стандартний набір кнопок допоміжних функцій, наприклад збереження,

			очищення.
--	--	--	-----------

Розглянемо клас `zoom`, який реалізує методи приближення чи віддалення від полотна зображення.

Таблиця 2.13 Методи та поля класу `zoom`

Назва	Тип	Сигнатура	Опис
<code>minZoom</code>	Публічне статичне поле	<code>const minZoom</code>	Мінімальний коефіцієнт приближення
<code>maxZoom</code>	Публічне статичне поле	<code>const maxZoom</code>	Максимальний коефіцієнт приближення
<code>zoomWithKey</code>	Публічний метод	<code>zoomWithKey(canvas, applyZoom)</code>	Метод зміни значення приближення за допомогою комбінацій клавіш
<code>zoomWithMouse</code>	Публічний метод	<code>zoomWithMouse(canvas, applyZoom)</code>	Метод зміни значення приближення за допомогою застосування миші

Розглянемо клас `saveInBrowser`, який реалізує методи збереження стану полотна редактора зображень у браузері.

Таблиця 2.14 Методи та поля класу `saveInBrowser`

Назва	Тип	Сигнатура	Опис
-------	-----	-----------	------

save	Публічний метод	save(name, value)	Приймає назву об'єкта та JSON, який зберігає в вказаному об'єкті.
load	Публічний метод	load(name)	Зчитує JSON з вказаного об'єкта
remove	Публічний метод	remove(name)	Видаляє JSON з вказаного об'єкта

Розглянемо клас `utils`, який реалізує утиліти, необхідні для реалізації додаткових можливостей.

Таблиця 2.15 Методи та поля класу `utils`

Назва	Тип	Сигнатура	Опис
generateFGFCS	Публічний метод	generateFGFCS(handlers, width, height, orientation, angle)	Реалізує логіку визначення обраного кольору із градієнту
getRealBBox	Публічний метод	getRealBBox(obj)	Повертає дані о об'єкті
getCurrentEffect	Публічний метод	getCurrentEffect(activeSelection)	Повертає значення фільтрів об'єкта
getUpdatedFilter	Публічний метод	getUpdatedFilter(effects, effect, value)	Утиліта зміни значень застосованих фільтрів
getActiveFontStyle	Публічний метод	getActiveFontStyle(activeSelection,	Визначення

		styleName)	поточного шрифту
setActiveFontStyle	Публічний метод	setActiveFontStyle(activeSelection, styleName, value)	Встановлення нового шрифту
downloadImage	Публічний метод	downloadImage(data, extension, mimeType)	Утиліта завантаження зображення із редактора зображень

2.4 Конструювання структури станів інтерфейсу

Структурна схема інтерфейсу [13] — це важливий елемент будь-якого проекту, що дозволяє візуалізувати та організувати роботу програми, веб-сайту чи іншого інтерактивного середовища. Використання структурної схеми інтерфейсу у проекті допомагає розуміти його архітектуру, взаємодію компонентів та логіку користувацького досвіду.

Цей інструмент дозволяє візуалізувати взаємозв'язки між різними елементами інтерфейсу, відображаючи їхню ієрархію та спосіб взаємодії. Завдяки структурній схемі, розробники, дизайнери та інші учасники проекту можуть краще розібратися в складності системи, спрямувати зусилля на оптимізацію та вдосконалення інтерфейсу для забезпечення максимального зручного користувацького досвіду.

У цьому контексті структурна схема інтерфейсу є надзвичайно важливим інструментом для розробки та покращення будь-якого проекту, надаючи зрозуміле та систематизоване представлення всієї інформації про його функціональність та взаємодію елементів.

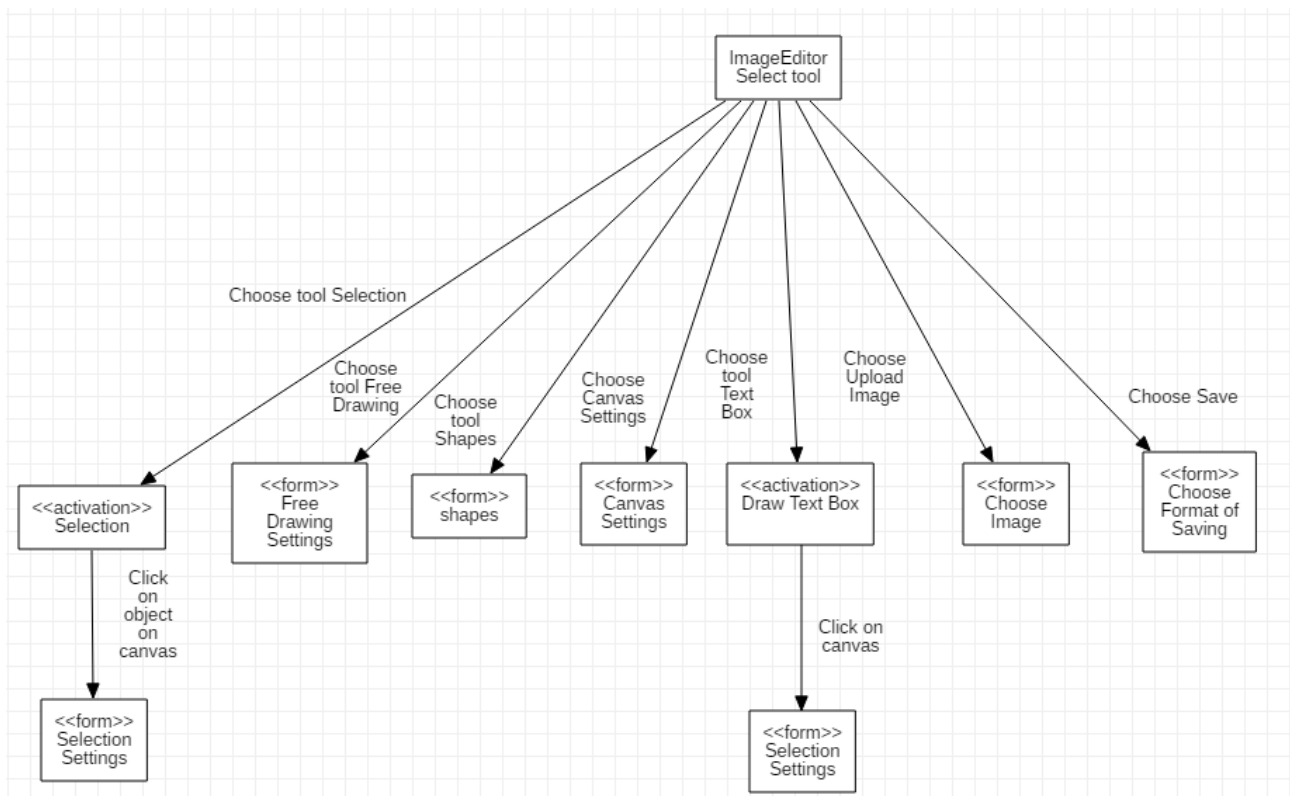


Рисунок 2.4 Структурна схема інтерфейсу веб-редактора зображень

У цій схемі продемонстровано, які зміни у інтерфейсі можуть відбуватися під час застосування певних інструментів редагування зображення. Під <<activation>> мається на увазі активація самих інструментів, інтерфейс не змінюється. <<form>> означає, що у інтерфейсі з'явилася відповідна панель налаштування чи вибору або форма із кнопками для вибору подальшої дії відповідно до раніше описаних функцій та класів у розділі **2.3 конструювання програмного забезпечення**.

2.5 Креслення вигляду екранних форм

Створення креслень екранних форм є ключовим етапом в процесі розробки будь-якого програмного продукту або веб-сайту. Ці креслення дозволяють візуалізувати та концептуалізувати розміщення елементів інтерфейсу на екрані, їх структуру та взаємодію для забезпечення ефективного користувацького досвіду.

Використання креслень екранних форм у проекті є необхідним для уточнення розміщення різних компонентів, таких як кнопки, поля для введення

даних, меню, інформаційні блоки та інші елементи. Це спрощує розуміння логіки взаємодії користувача з програмою чи веб-сайтом, а також допомагає виявити можливі недоліки або пропозиції щодо покращення інтерфейсу на ранніх етапах розробки.

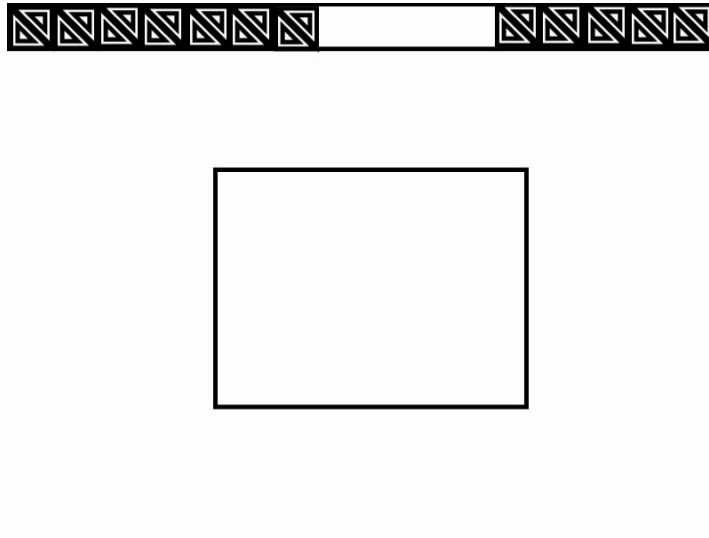


Рисунок 2.5 Креслення інтерфейсу веб-редактора зображень без панелей налаштувань та форм.

На цьому рисунку відображено загальний інтерфейс застосунку. Посередині знаходиться полотно — область обробки зображення. Зверху знаходиться панель з кнопками, які прив'язані до інструментів (ліві) та функціями, як от очищення полотна чи збереження (справа).

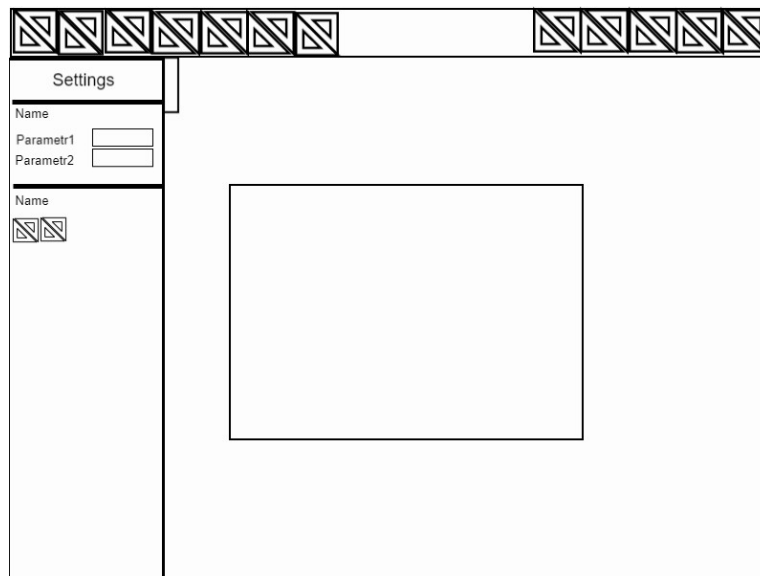


Рисунок 2.6 Креслення інтерфейсу веб-редактора зображень з панеллю налаштування.

На цьому рисунку відображено інтерфейс застосунку разом з панеллю налаштування. Панель може ховатися чи з'являтися при натиску на язичок за умови, що вибраний інструмент, який передбачає панель налаштування. Панель має назву і розділи, які складаються з форм для вибору параметрів налаштування відповідних обраних об'єктів та назв цих розділів. Останній такий розділ має кнопки для стандартних дій, як дублювання та видалення обраних об'єктів.

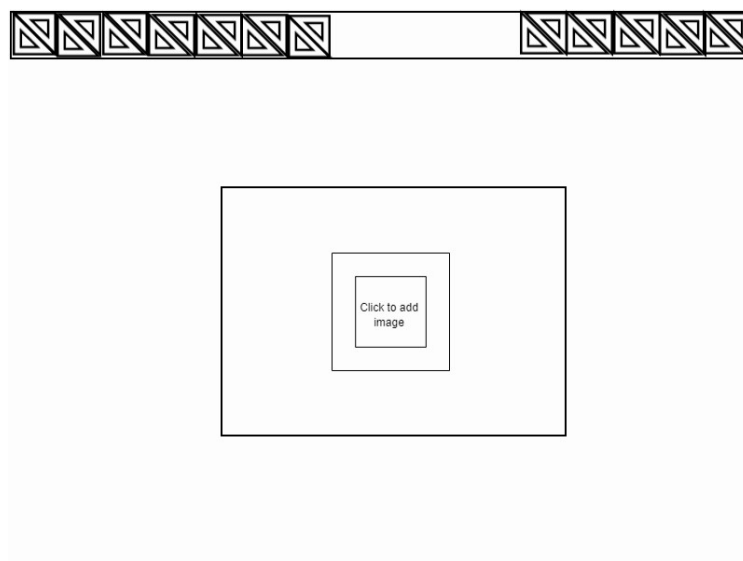


Рисунок 2.7 Креслення інтерфейсу веб-редактора зображень з формою додавання зображення.

На даному рисунку відображено форму вибору зображення. Після натискання на форму відкриється файлова система для пошуку файлів зображень.

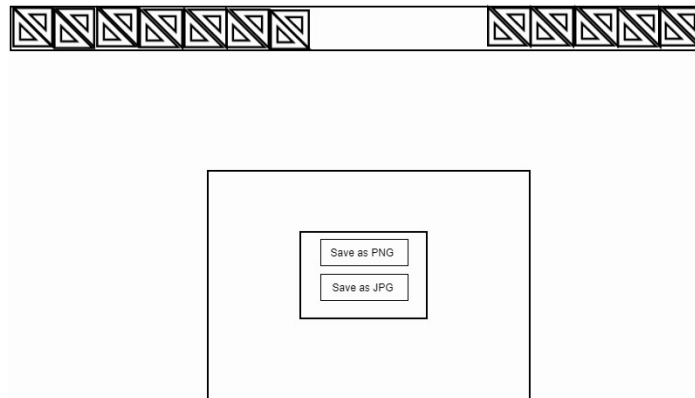


Рисунок 2.8 Креслення інтерфейсу веб-редактора зображень з формою збереження.

На даному рисунку відображено форму вибору формату файлу зображення для збереження.

2.6 Висновки до розділу

В розділі "Конструювання програмного забезпечення" веб-редактора зображень було представлено процес проєктування та реалізації ключових компонентів системи. Основні аспекти включали архітектуру файлової системи, використання шаблонів проєктування, а також побудову класів.

У розділі також висвітлено важливість розробки інтерфейсу користувача для взаємодії застосунком. Для цього був реалізований графічний інтерфейс у вигляді креслень інтерфейсу та структурної схеми інтерфейсу, який дозволяє користувачеві легко використовувати функції веб-редактора зображень.

Отже, результатом цього розділу є створення основної структури та компонентів застосунку, які дозволяють ефективно оброблювати графічні об'єкти та надають зручний інтерфейс для користувача.

3 ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

У розділі ми проведемо процес тестування для впевненості в його надійності та ефективності.

3.1 Опис процесів тестування

У даному пункті будуть проаналізовані усі функціональні вимоги, що були висунуті для веб-редактора зображень, щоб визначити їхню повноту, конкретність, узгодженість з бізнес-потребами та здатність задовольняти очікувані вимоги користувачів.

Перевіримо працездатність функції додавання фігури на полотно. Процес показано в таблиці 3.1.

Таблиця 3.1 Тестування функції додавання фігури

Тест	Додавання фігури на полотно
Номер тесту	1
Початковий стан	Користувач знаходиться в застосунку
Вхідні дані	-
Опис проведення тесту	Натиснути на кнопку «Фігури» на панелі інструментів та натиснути на одну з фігур, наявних у панелі фігур.
Очікуваний результат	Обрана фігура з'явиться у верхньому правому кутку полотна зображення.
Фактичний результат	Обрана фігура з'явилася у верхньому правому кутку полотна зображення.

Як можна побачити на рисунку 3.1, фігура була успішно додана на полотно зображення.

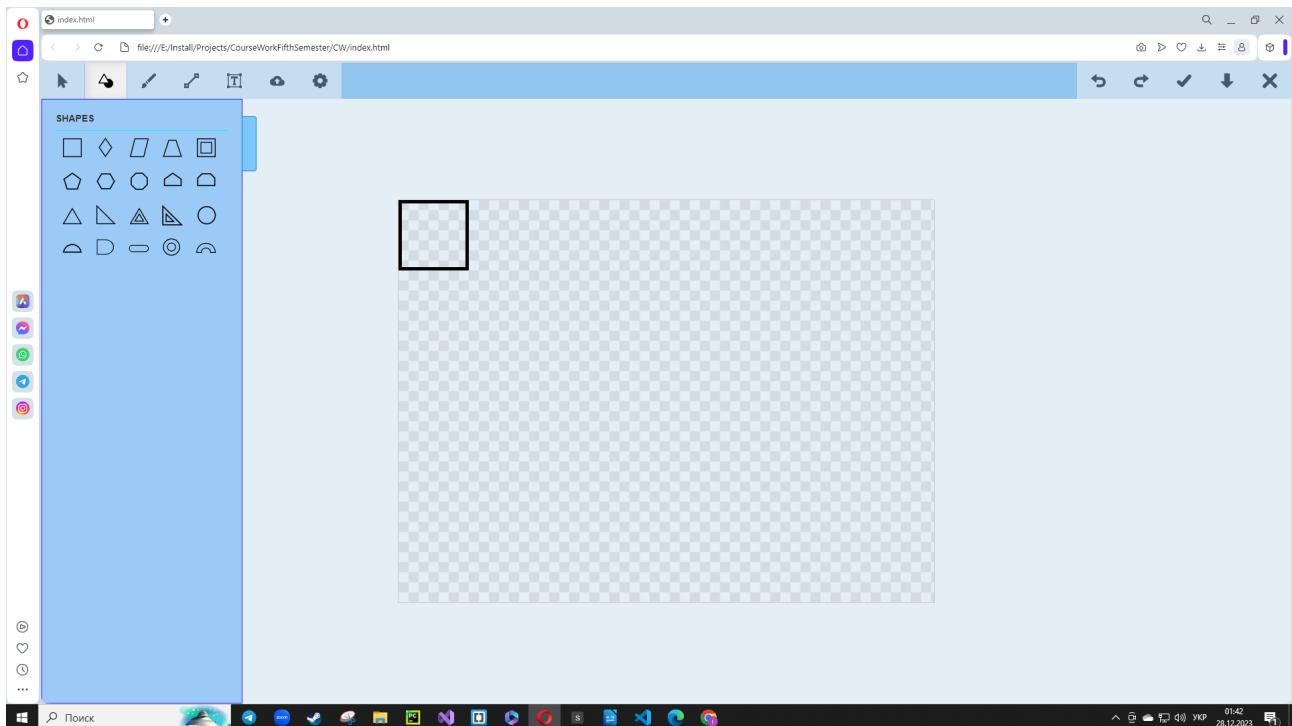


Рисунок 3.1 Ручна перевірка додавання фігури на полотно.

Перевіримо працездатність функції вибору об'єкта на полотні. Процес показано в таблиці 3.2.

Таблиця 3.2 Тестування функції додавання фігури

Тест	Вибір об'єкта на полотні
Номер тесту	2
Початковий стан	Користувач знаходиться в застосунку, наявний об'єкт на полотні
Вхідні дані	-
Опис проведення тесту	Натиснути на кнопку «Вибір» на панелі інструментів та натиснути на наявний об'єкт на полотні.
Очікуваний результат	Обраний об'єкт вибереться, відкриється панель

	налаштувань об'єкта.
Фактичний результат	Обраний об'єкт вибрався, відкрилась панель налаштувань об'єкта.

Як можна побачити на рисунку 3.2, фігура була успішно додана на полотно зображення.

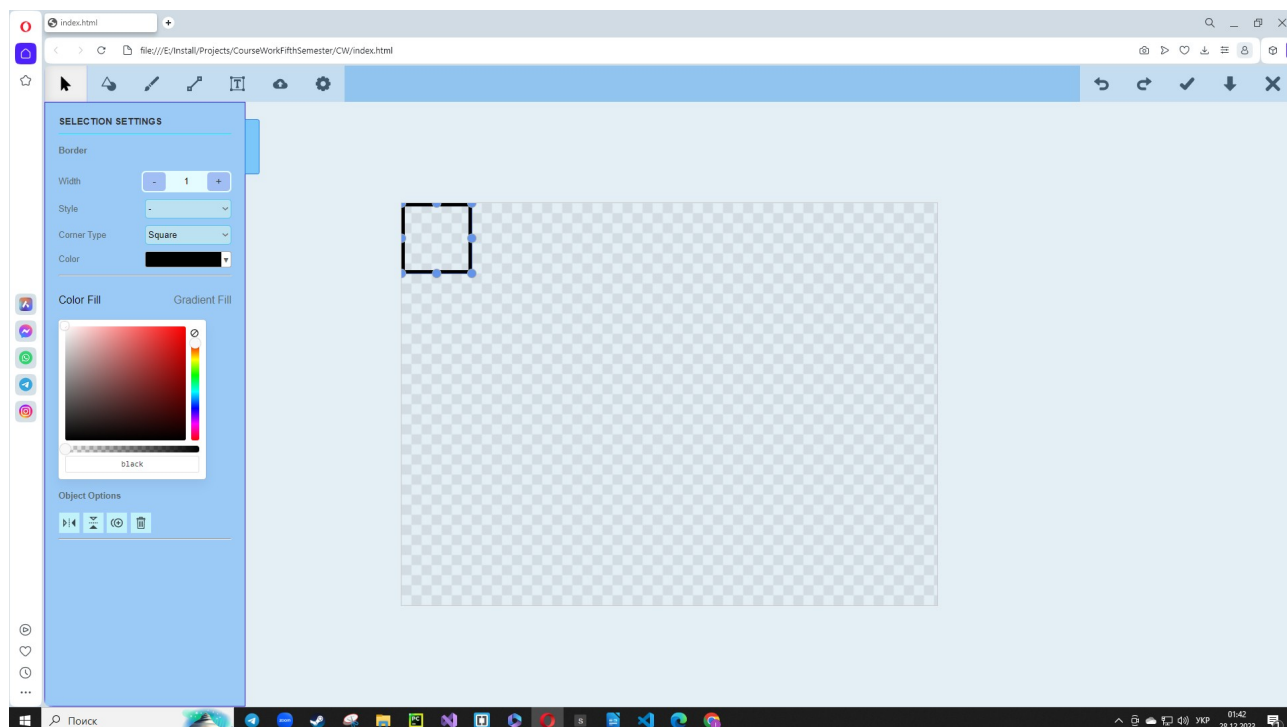


Рисунок 3.2 Ручна перевірка вибору об'єкта на полотні.

Перевіримо працездатність функції зміни об'єкта на полотні. Процес показано в таблиці 3.3.

Таблиця 3.3 Тестування функції зміни об'єкта на полотні

Тест	Зміна об'єкта на полотні
Номер тесту	3
Початковий стан	Користувач знаходиться в застосунку, наявний об'єкт на полотні вибраний
Вхідні дані	-
Опис проведення тесту	Змінити параметри обраного об'єкта в панелі

	налаштувань.
Очікуваний результат	Обраний об'єкт зміниться відповідно до змінених параметрів.
Фактичний результат	Обраний об'єкт змінився відповідно до змінених параметрів.

Як можна побачити на рисунку 3.3, об'єкт був успішно змінений.

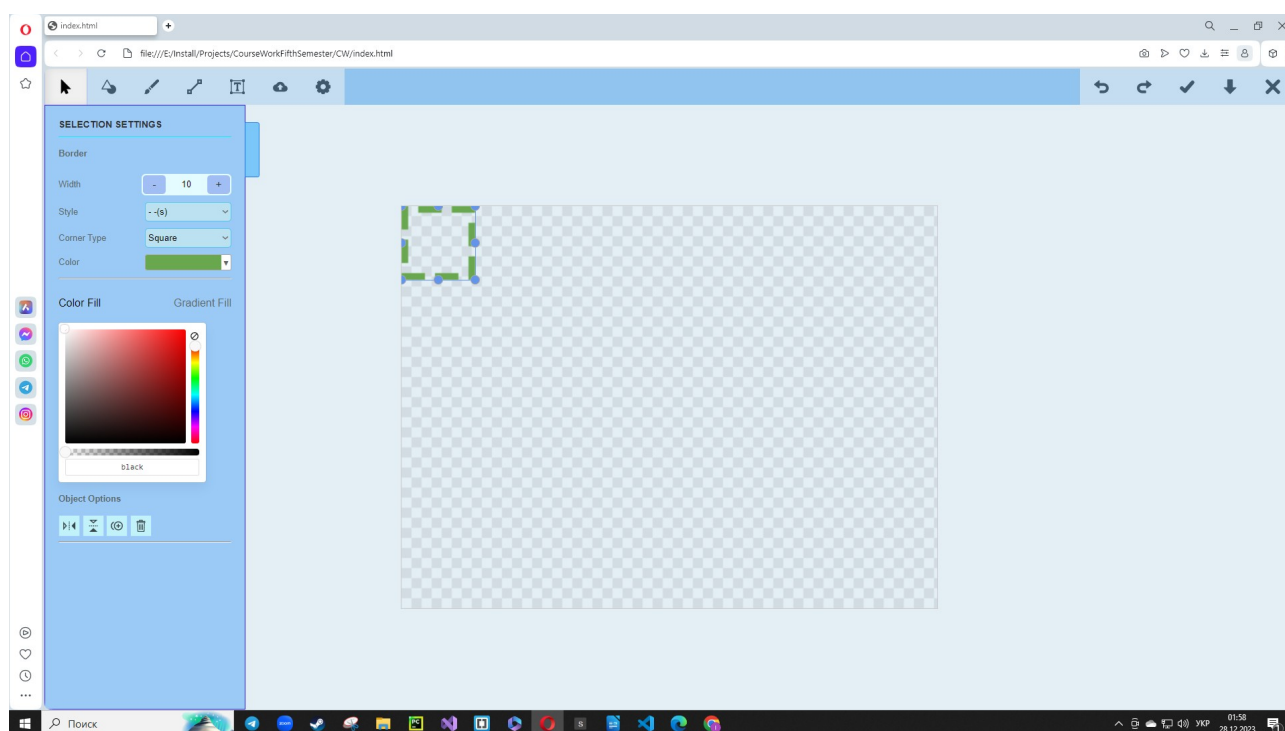


Рисунок 3.3 Ручна перевірка зміни об'єкта на полотні.

Перевіримо працездатність функції дублювання об'єкта на полотні. Процес показано в таблиці 3.4.

Таблиця 3.4 Тестування функції дублювання об'єкта

Тест	Дублювання вибраного об'єкта
Номер тесту	4
Початковий стан	Користувач знаходиться в застосунку, наявний об'єкт на полотні обраний
Вхідні дані	-

Опис проведення тесту	Натиснути на кнопку «Дублювати» на панелі налаштувань.
Очікуваний результат	Обраний об'єкт продублюється.
Фактичний результат	Обраний об'єкт продублювався.

Як можна побачити на рисунку 3.4, об'єкт був успішно продубльований.

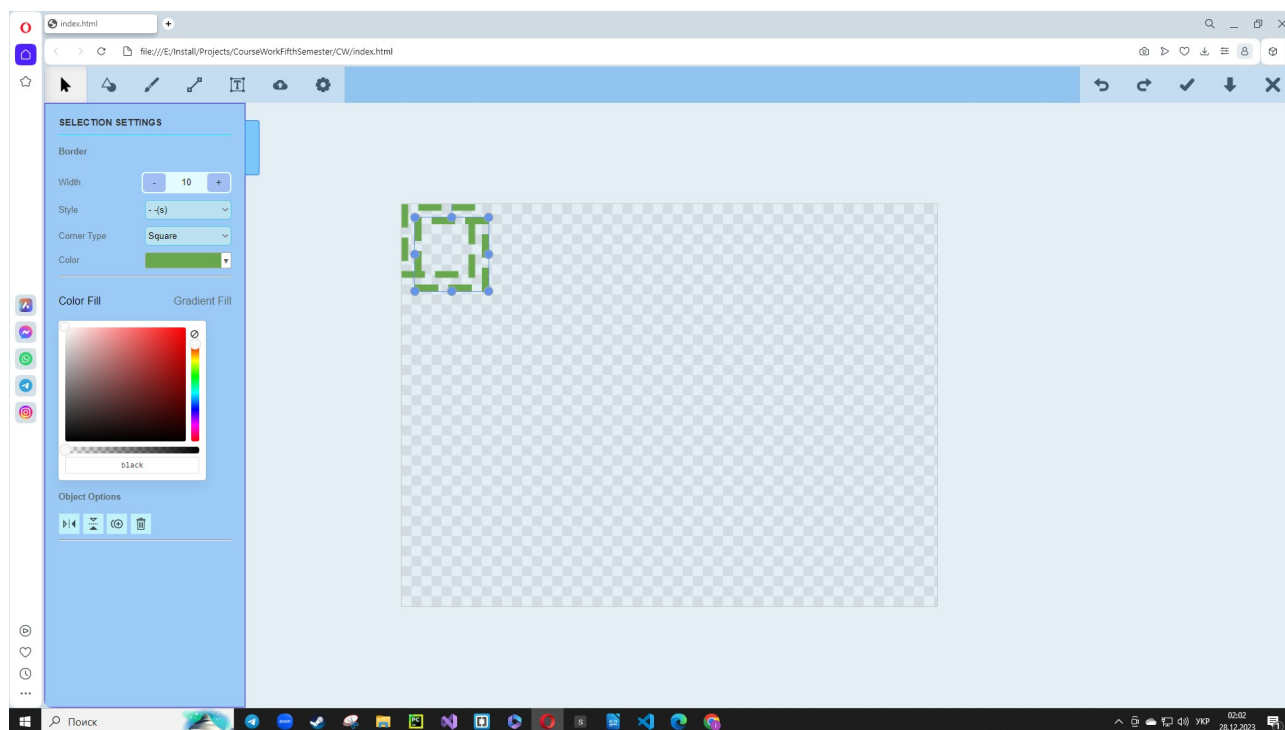


Рисунок 3.4 Ручна перевірка дублювання об'єкта на полотні.

Перевіримо працездатність функції копіювання та вставки виділеної області полотна зображення. Процес показано в таблиці 3.5.

Таблиця 3.5 Тестування функції копіювання та вставки виділеної області полотна зображення

Тест	Копіювання та вставки виділеної області полотна зображення
Номер тесту	5
Початковий стан	Користувач знаходиться в застосунку, наявна група

	об'єктів на полотні
Вхідні дані	-
Опис проведення тесту	Натиснути на кнопку «Вибір» на панелі інструментів, виділити область із групою об'єктів, натиснути кнопку «Дублювати» на панелі налаштувань, за бажанням перемістити скопійований фрагмент полотна.
Очікуваний результат	Обраний фрагмент полотна продублюється.
Фактичний результат	Обраний фрагмент полотна продублювався.

Як можна побачити на рисунку 3.5, фігура була успішно додана на полотно зображення.

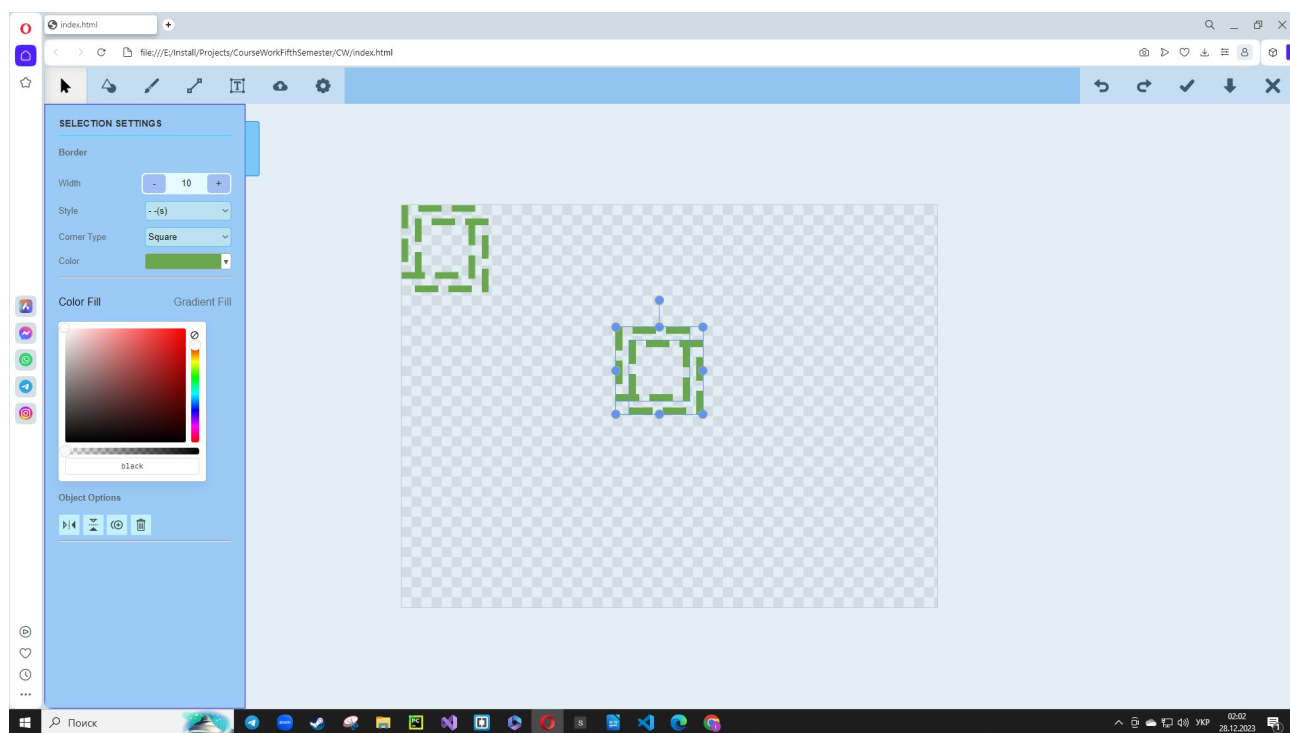


Рисунок 3.5 Ручна перевірка дублювання фрагменту полотна.

Перевіримо працездатність функції видалення об'єкта на полотні. Процес показано в таблиці 3.6.

Таблиця 3.6 Тестування функції видалення фігури

Тест	Видалення об'єкта на полотні
Номер тесту	6
Початковий стан	Користувач знаходиться в застосунку, наявний об'єкт на полотні вибраний
Вхідні дані	-
Опис проведення тесту	Натиснути на кнопку «Видалити» на панелі інструментів.
Очікуваний результат	Обраний об'єкт видалиться, закриється панель налаштувань об'єкта.
Фактичний результат	Обраний об'єкт видалився, закрилась панель налаштувань об'єкта.

Як можна побачити на рисунку 3.6, об'єкт був успішно видалений з полотна зображення.

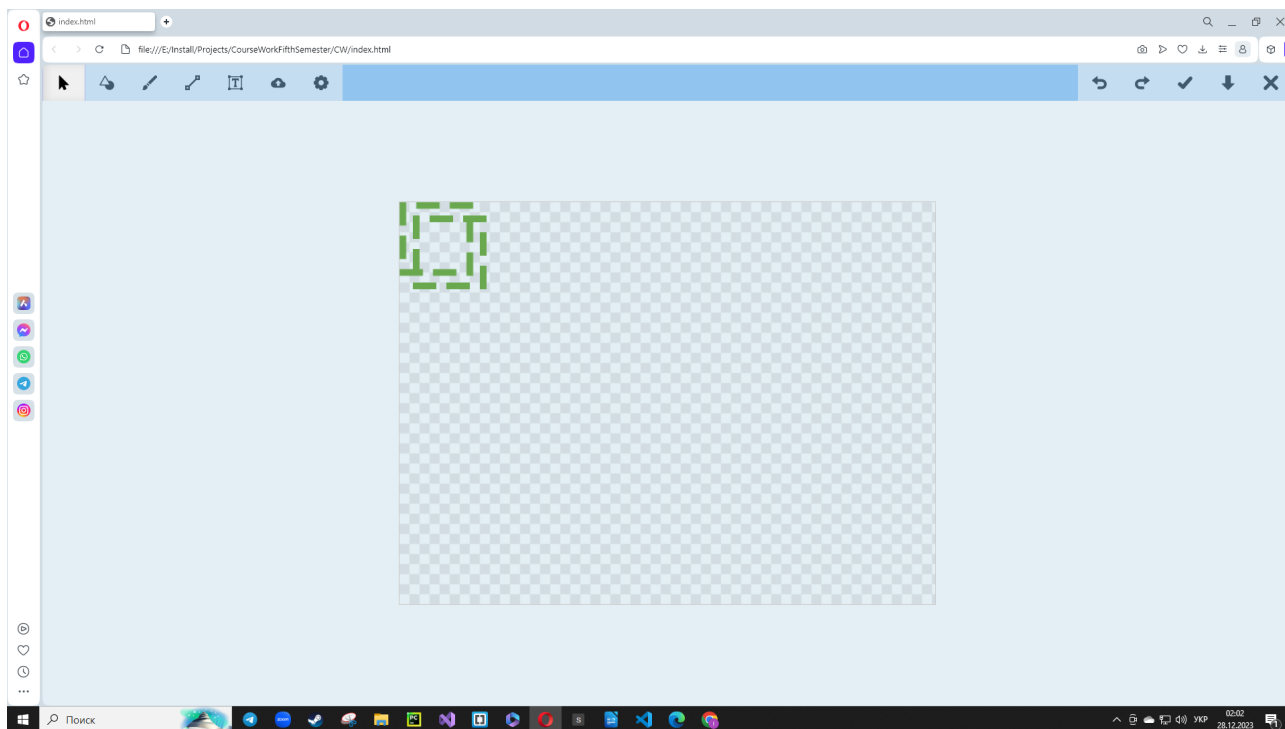


Рисунок 3.6 Ручна перевірка видалення об'єкта з полотна.

Перевіримо працездатність функції малювання довільної лінії на полотні.

Процес показано в таблиці 3.7.

Таблиця 3.7 Тестування функції малювання довільної лінії

Тест	Малювання довільної лінії
Номер тесту	7
Початковий стан	Користувач знаходиться в застосунку
Вхідні дані	-
Опис проведення тесту	Натиснути на кнопку «Малювання довільної лінії», за бажанням змінити параметри малювання лінії в панелі налаштувань, намалювати лінію на полотні.
Очікуваний результат	Намалюється лінія на полотні.
Фактичний результат	Намалювалася лінія на полотні.

Як можна побачити на рисунку 3.7, лінія була успішно додана на полотно зображення.

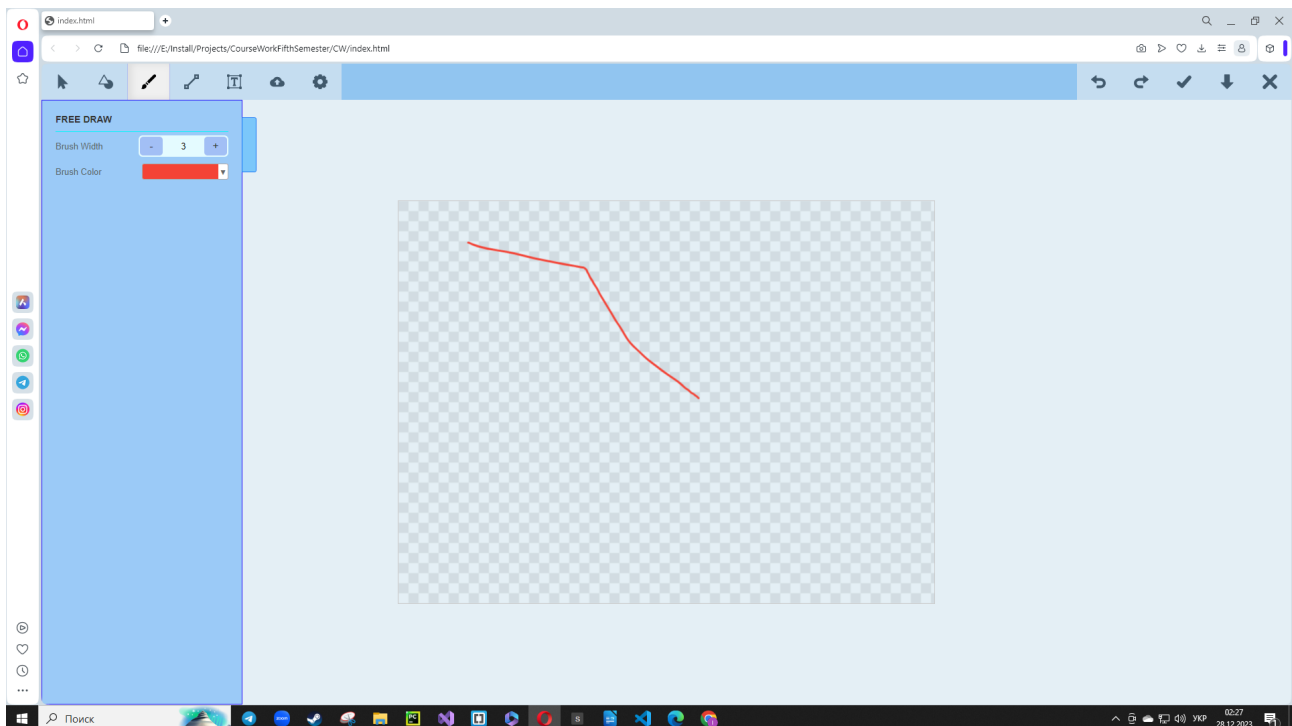


Рисунок 3.7 Ручна перевірка малювання довільної лінії на полотні.

Перевіримо працездатність функції малювання прямої лінії на полотні.
Процес показано в таблиці 3.8.

Таблиця 3.8 Тестування функції малювання прямої лінії

Тест	Малювання прямої лінії на полотні
Номер тесту	8
Початковий стан	Користувач знаходиться в застосунку
Вхідні дані	-
Опис проведення тесту	Натиснути на кнопку «Малювання прямої лінії» на панелі інструментів та протягнути мишкою по полотну.
Очікуваний результат	На полотні намалюється пряма лінія.
Фактичний результат	На полотні намалювалася пряма лінія.

Як можна побачити на рисунку 3.8, пряма лінія була успішно додана на полотно зображення.

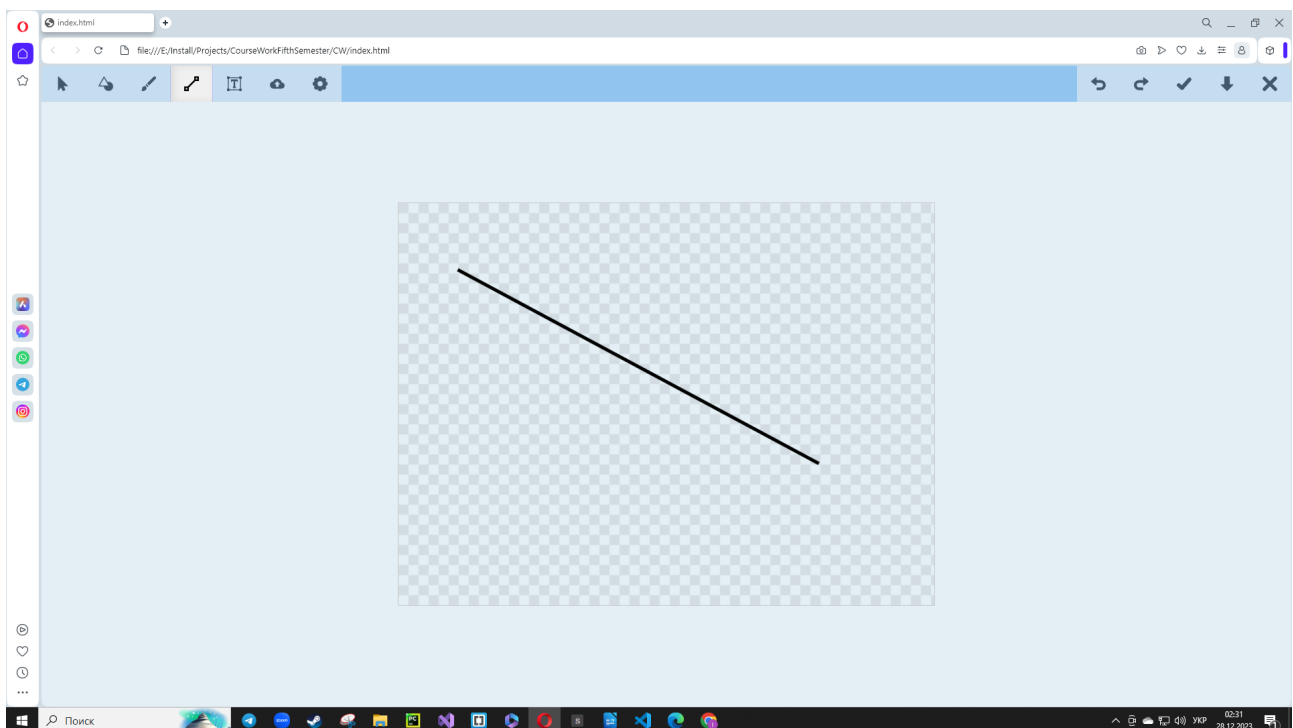


Рисунок 3.8 Ручна перевірка малювання прямої лінії на полотні.

Перевіримо працездатність функції додавання текстового поля на полотно. Процес показано в таблиці 3.9.

Таблиця 3.9 Тестування функції додавання текстового поля

Тест	Додавання текстового поля на полотно
Номер тесту	9
Початковий стан	Користувач знаходиться в застосунку
Вхідні дані	-
Опис проведення тесту	Натиснути на кнопку «Текстове поле» на панелі інструментів та натиснути на довільну точку на полотні.
Очікуваний результат	На полотні з'явиться текстове поле, відкриється панель налаштувань
Фактичний результат	На полотні з'явилося текстове поле, відкрилась панель налаштувань

Як можна побачити на рисунку 3.9, текстове поле було успішно додано на полотно зображення.

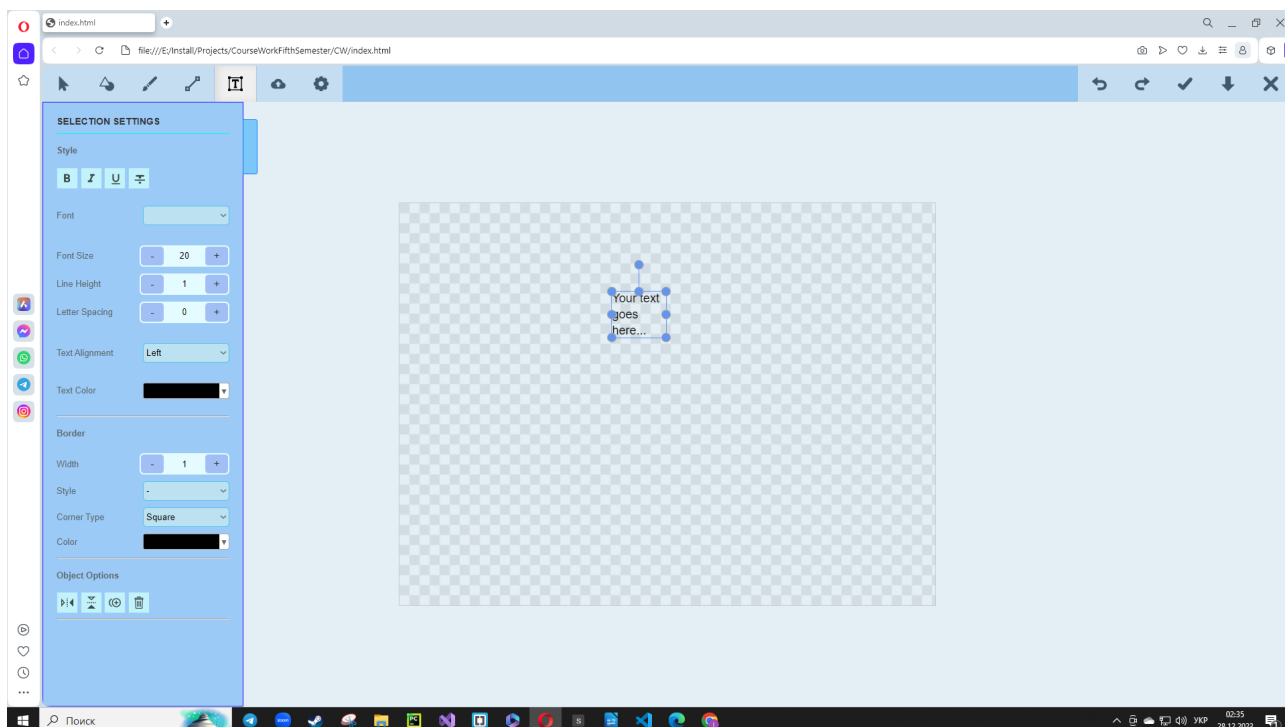


Рисунок 3.9 Ручна перевірка додавання текстового поля на полотно.

Перевіримо працездатність функції додавання зображення на полотно. Процес показано в таблиці 3.10.

Таблиця 3.10 Тестування функції додавання зображення

Тест	Додавання зображення на полотно
Номер тесту	10
Початковий стан	Користувач знаходиться в застосунку
Вхідні дані	-
Опис проведення тесту	Натиснути на кнопку «Завантажити зображення» на панелі інструментів, натиснути на форму додання, вибрати зображення.
Очікуваний результат	Обране зображення додається на полотно.
Фактичний результат	Обране зображення додалося на полотно.

Як можна побачити на рисунках 3.10 та 3.11, зображення було успішно додане на полотно зображення.

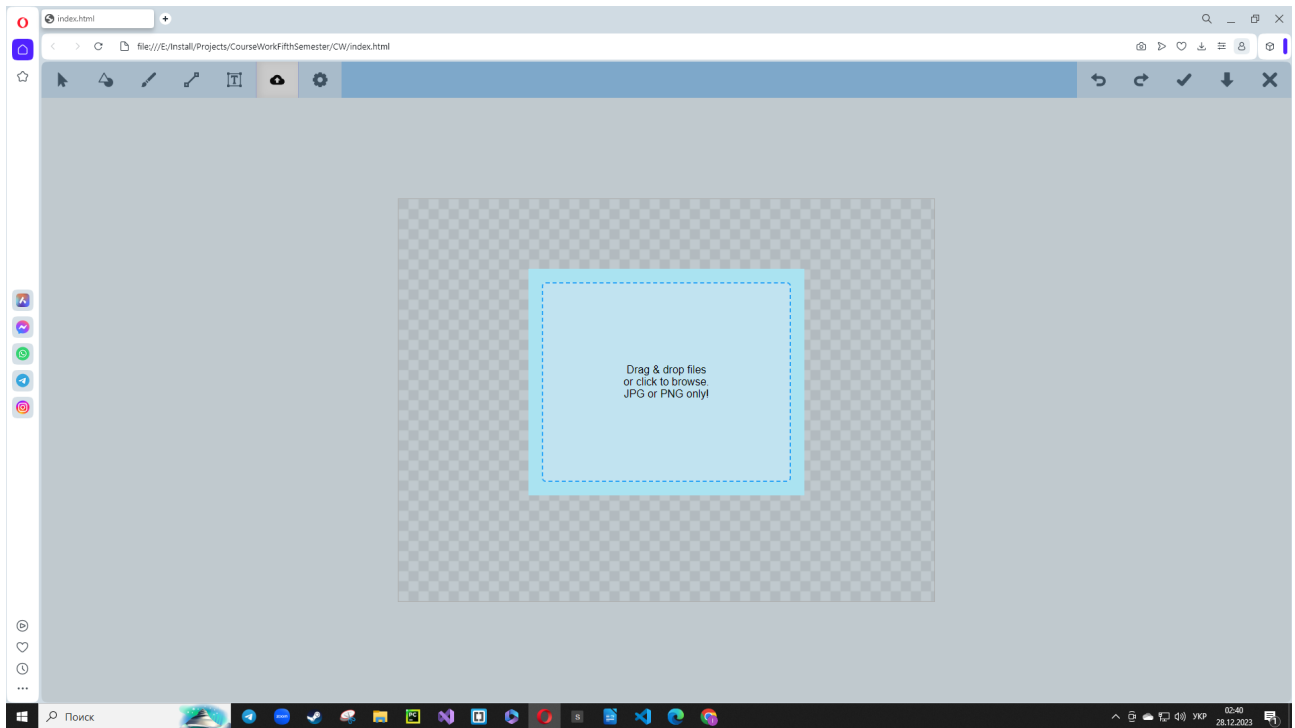


Рисунок 3.10 Відкриття форми додання зображення.

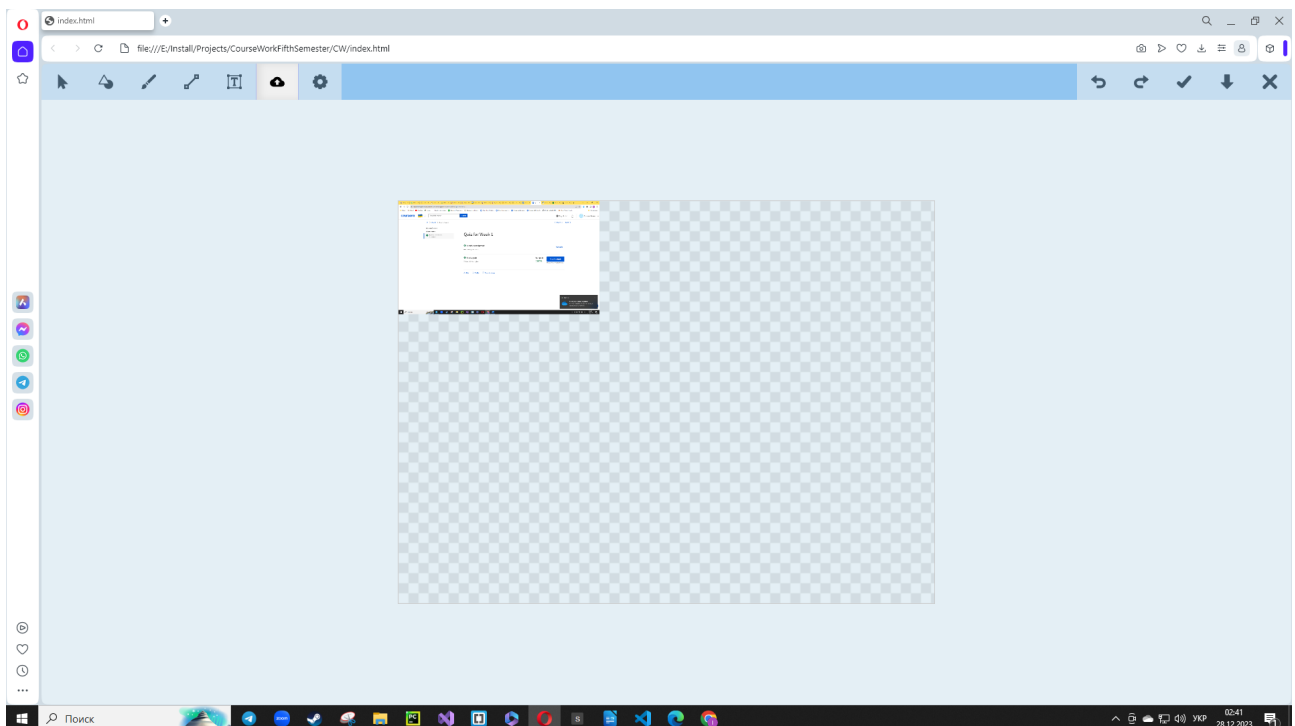


Рисунок 3.11 Ручна перевірка додавання зображення на полотно.

Перевіримо працездатність функції масштабування зображення на полотні. Процес показано в таблиці 3.11.

Таблиця 3.11 Тестування функції масштабування зображення

Тест	Вибір об'єкта на полотні
Номер тесту	11
Початковий стан	Користувач знаходиться в застосунку, наявне зображення на полотні
Вхідні дані	-
Опис проведення тесту	Натиснути на кнопку «Вибір» на панелі інструментів та натиснути на наявне зображення на полотні, потягнути за кут зображення.
Очікуваний результат	Обране зображення масштабується.
Фактичний результат	Обране зображення масштабувалося.

Як можна побачити на рисунку 3.12, фігура була успішно додана на полотно зображення.

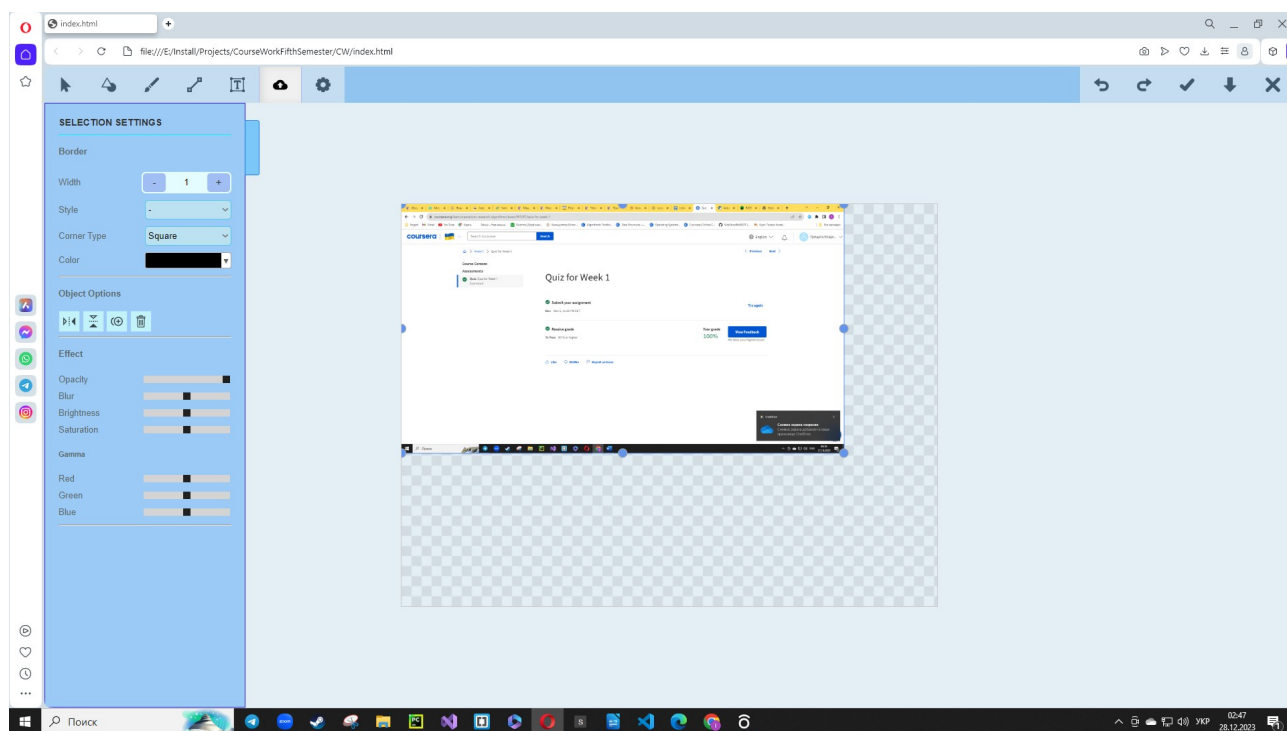


Рисунок 3.12 Ручна перевірка додавання зображення на полотно.

Перевіримо працездатність функції спотворення зображення. Процес показано в таблиці 3.12.

Таблиця 3.12 Тестування функції спотворення зображення

Тест	Спотворення зображення
Номер тесту	12
Початковий стан	Користувач знаходиться в застосунку, наявне зображення на полотні
Вхідні дані	-
Опис проведення тесту	Натиснути на кнопку «Вибір» на панелі інструментів та натиснути на наявне зображення, потягнути за кола на серединах граней.
Очікуваний результат	Обране зображення витягнеться.
Фактичний результат	Обране зображення витягнулося.

Як можна побачити на рисунку 3.13, фігура була успішно додана на полотно зображення.

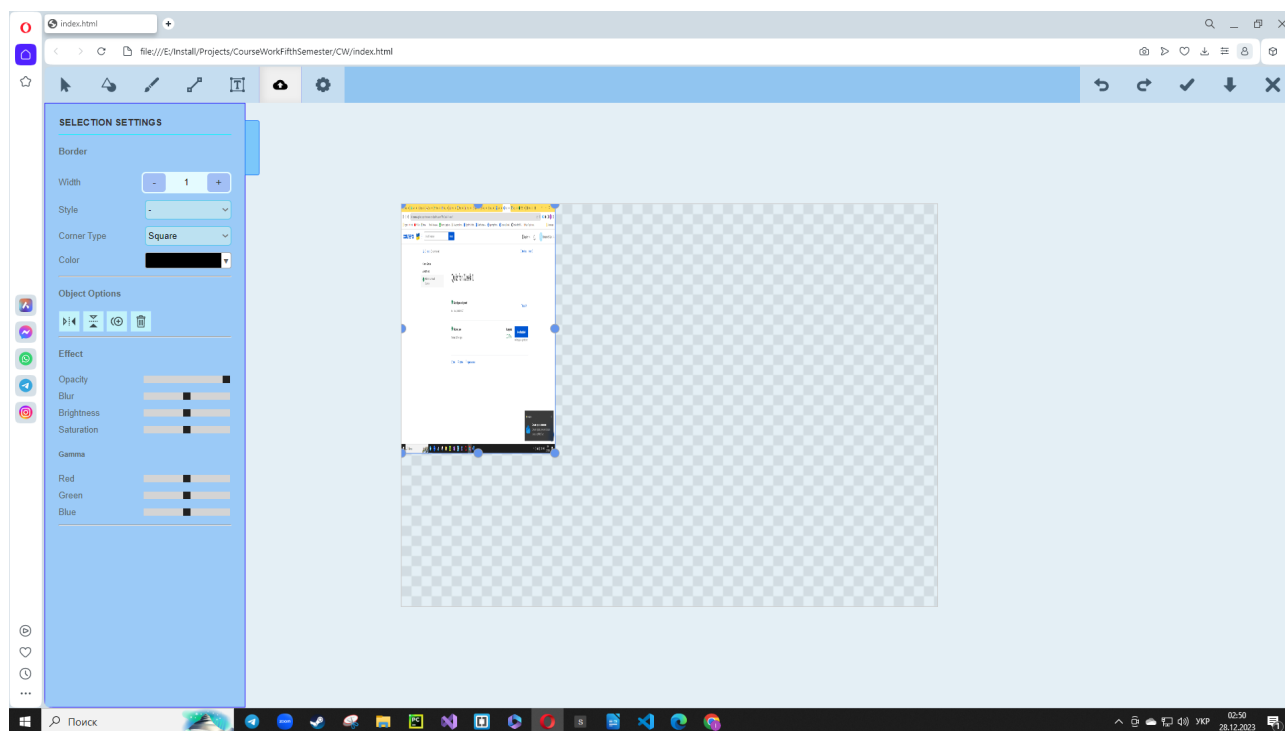


Рисунок 3.13 Ручна перевірка спотворення зображення.

Перевіримо працездатність функції повороту зображення на полотні.
Процес показано в таблиці 3.13.

Таблиця 3.13 Тестування функції повороту зображення

Тест	Поворот зображення
Номер тесту	13
Початковий стан	Користувач знаходиться в застосунку, наявне зображення на полотні
Вхідні дані	-
Опис проведення тесту	Натиснути на кнопку «Вибір» на панелі інструментів та натиснути на наявне зображення на полотні, затиснути коло над зображенням та прокрутити на довільний кут.
Очікуваний результат	Обране зображення змінить кут нахилу.
Фактичний результат	Обране зображення змінило кут нахилу.

Як можна побачити на рисунку 3.14, зображення було успішно прокручено.

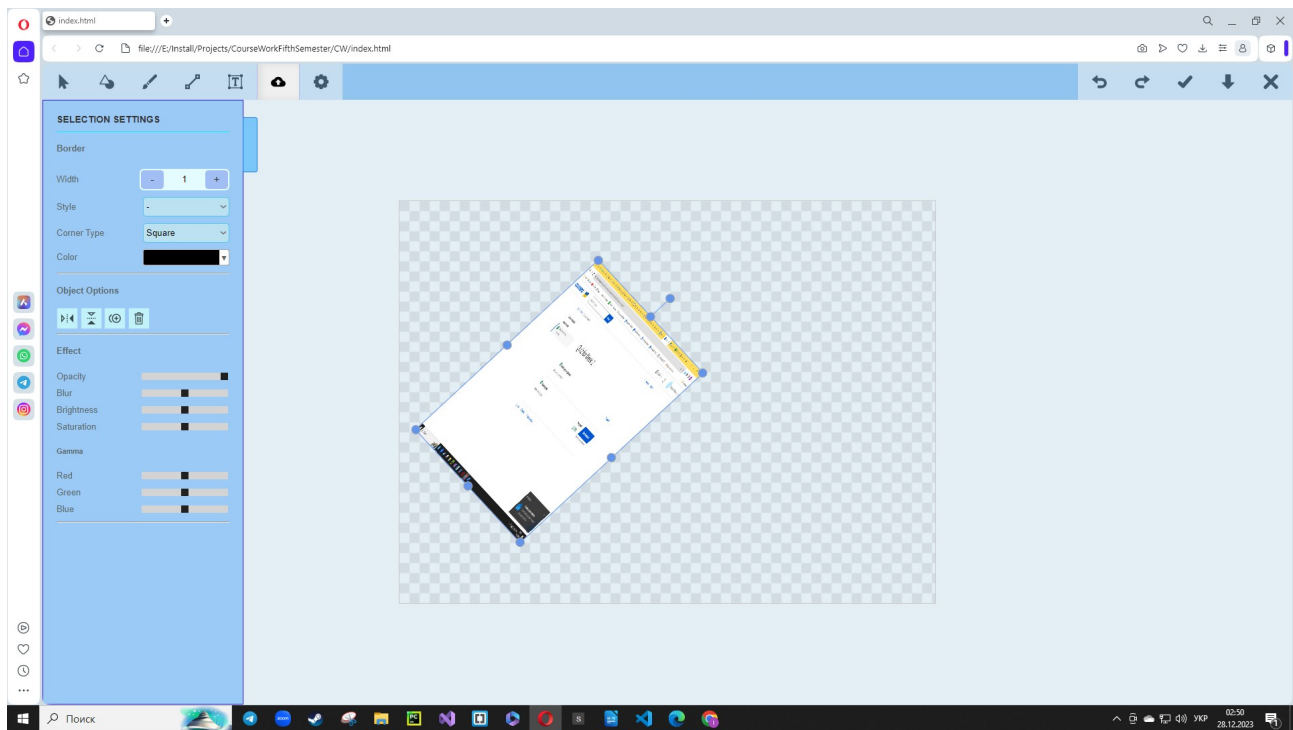


Рисунок 3.14 Ручна перевірка повороту зображення.

Перевіримо працездатність функції застосування кольоро-корекційних фільтрів над зображенням. Процес показано в таблиці 3.14.

Таблиця 3.14 Тестування функції застосування кольоро-корекційних фільтрів над зображенням

Тест	Застосування кольорокорекційних фільтрів
Номер тесту	14
Початковий стан	Користувач знаходиться в застосунку, наявне зображення на полотні
Вхідні дані	-
Опис проведення тесту	Натиснути на кнопку «Вибір» на панелі інструментів та натиснути на наявне зображення, змінити параметри ефектів в розділі «Ефекти» в панелі налаштування.
Очікуваний результат	Зміни параметрів ефектів відобразяться на

	зображенні.
Фактичний результат	Зміни параметрів ефектів відобразилися на зображенні.

Як можна побачити на рисунку 3.15, зображення успішно змінилося.

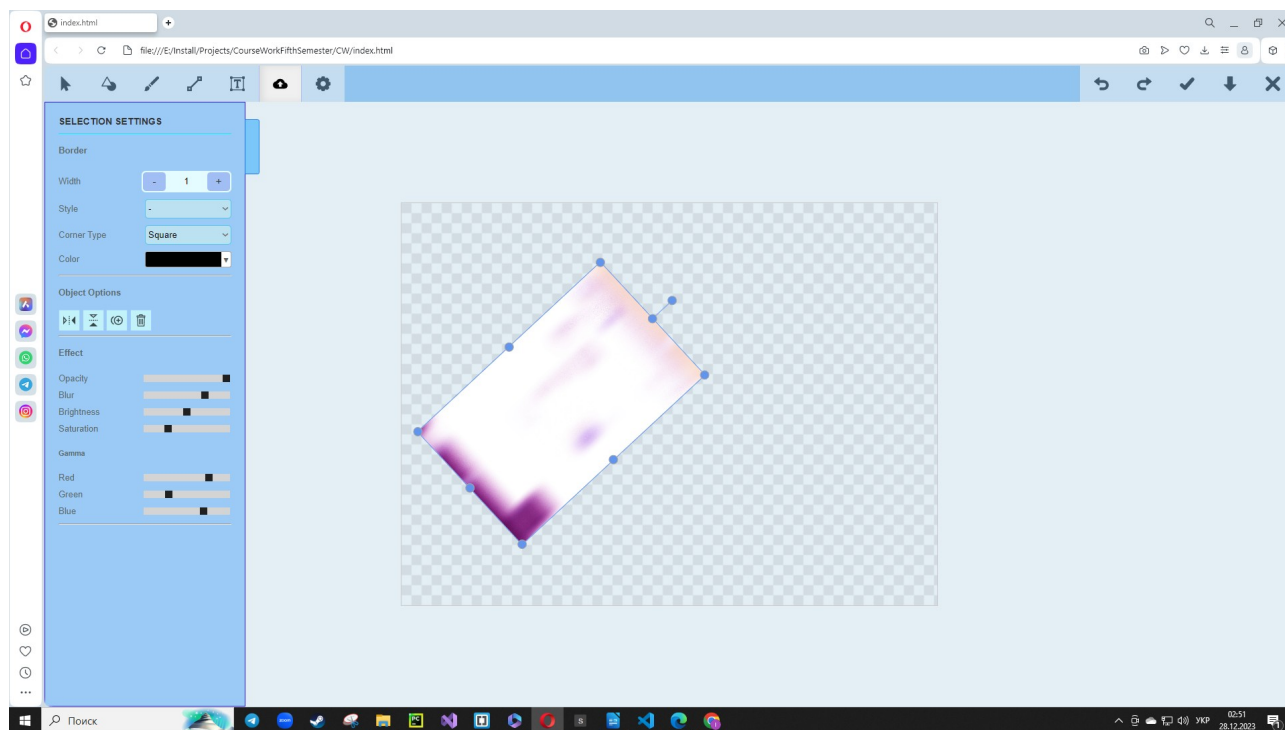


Рисунок 3.15 Ручна перевірка застосування кольорокорекційних фільтрів.

Перевіримо працездатність функції налаштування полотна. Процес показано в таблиці 3.15.

Таблиця 3.15 Тестування функції налаштування полотна

Тест	Налаштування полотна
Номер тесту	15
Початковий стан	Користувач знаходиться в застосунку
Вхідні дані	-
Опис проведення тесту	Натиснути на кнопку «Налаштування полотна» на панелі інструментів та внести зміни у параметри

	полотна.
Очікуваний результат	Полотно зміниться відповідно до параметрів.
Фактичний результат	Полотно змінилося відповідно до параметрів.

Як можна побачити на рисунку 3.2, полотно змінилося відповідно до вказаних параметрів.

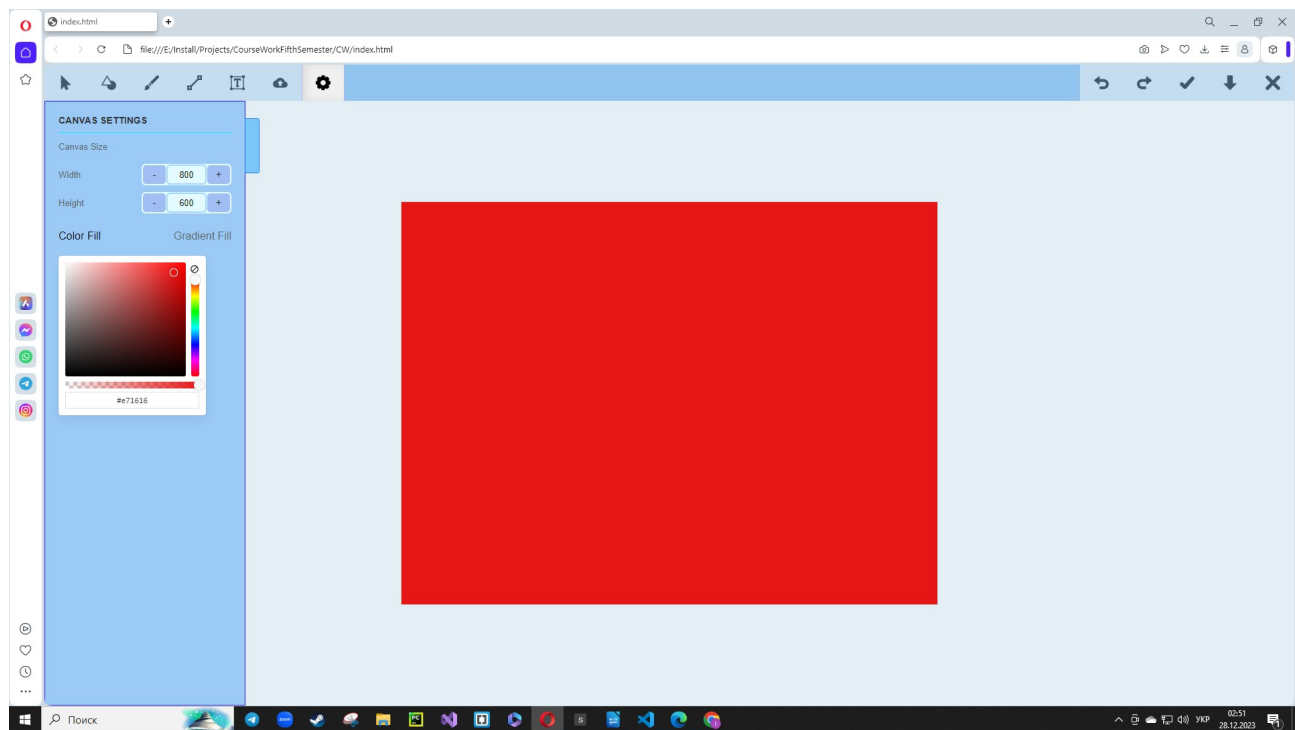


Рисунок 3.16 Ручна перевірка налаштування полотна.

Перевіримо працездатність функцій undo та redo. Процес показано в таблиці 3.16.

Таблиця 3.16 Тестування функцій undo та redo

Тест	undo та redo
Номер тесту	16
Початковий стан	Користувач знаходиться в застосунку, наявний об'єкт на полотні
Вхідні дані	-

Опис проведення тесту	Натиснути на кнопку «Вибір» на панелі інструментів та перетягнути наявний об'єкт. Потім на панелі інструментів натиснути «Undo», потім - «Redo».
Очікуваний результат	Спочатку зображення прийме попередній стан, потім — наступний.
Фактичний результат	Спочатку зображення прийняло попередній стан, потім — наступний.

Як можна побачити на рисунку 3.17 — 3.20, функції успішно спрацювали.

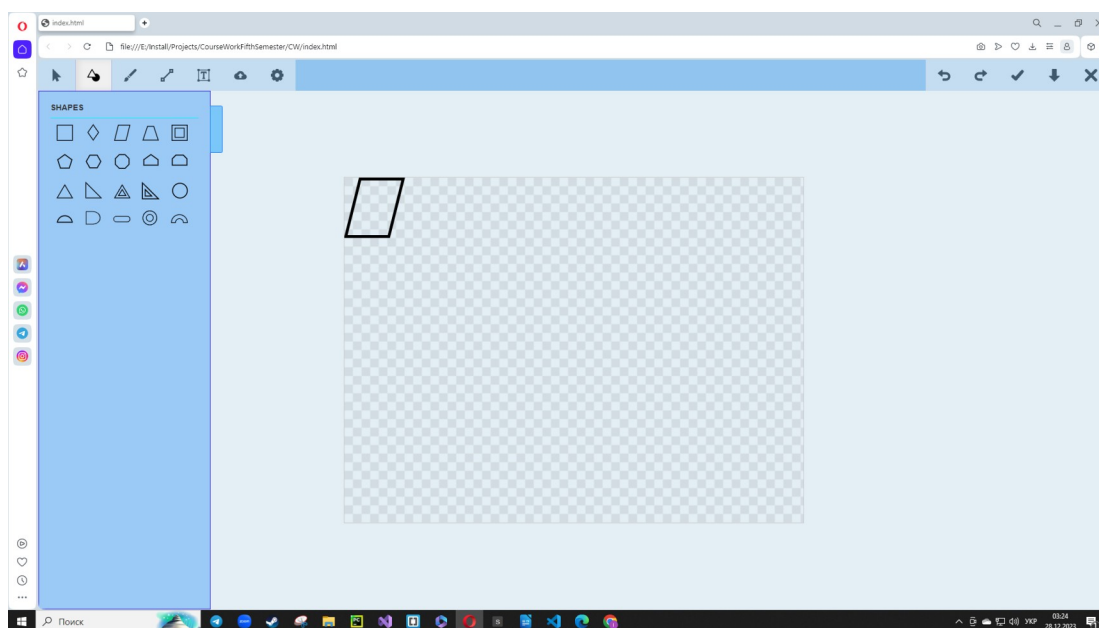


Рисунок 3.17 Початковий стан полотна.

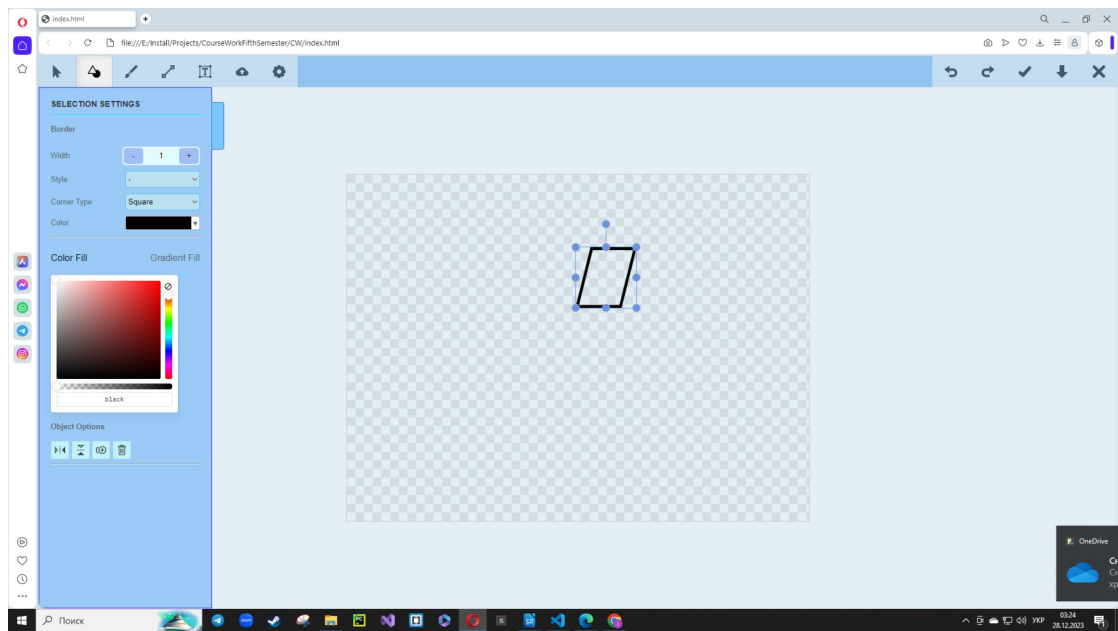


Рисунок 3.18 наступний стан полотна.

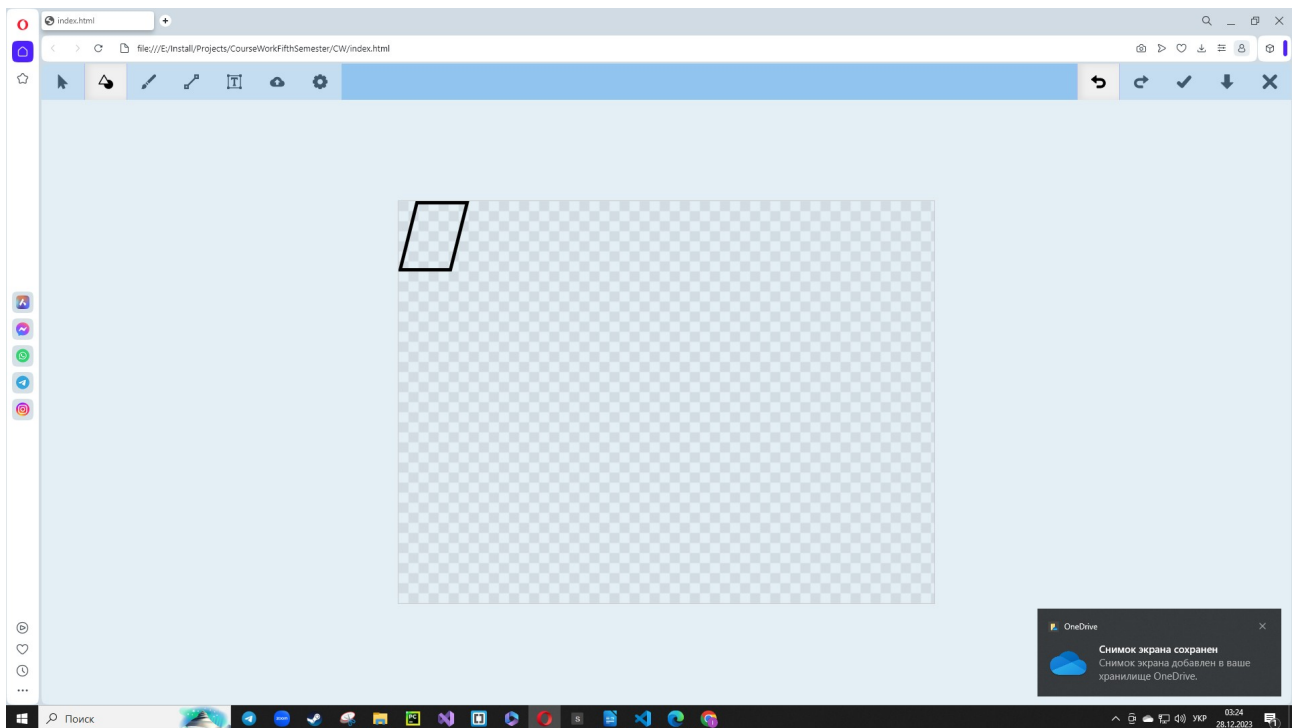


Рисунок 3.19 Ручна перевірка функції undo.

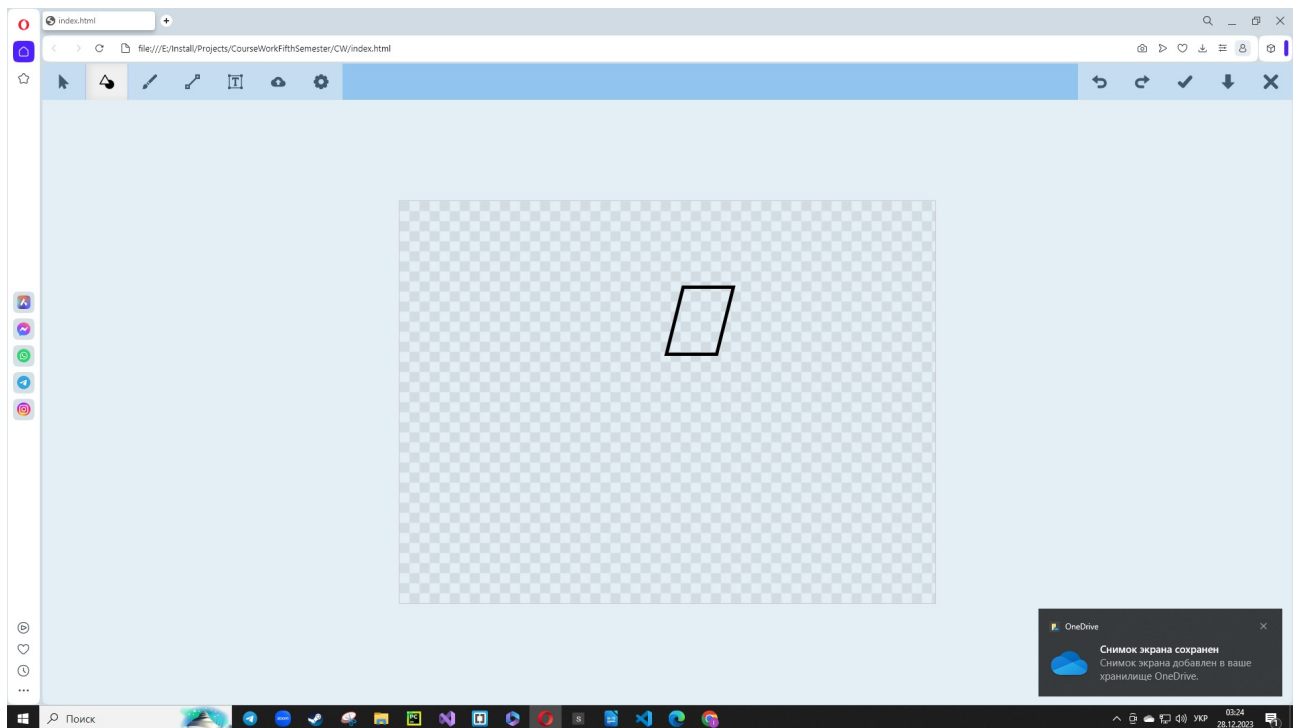


Рисунок 3.20 Ручна перевірка функції redo.

Перевіримо працездатність функції збереження стану полотна. Процес показано в таблиці 3.17.

Таблиця 3.17 Тестування функції збереження

Тест	Збереження стану полотна
Номер тесту	17
Початковий стан	Користувач знаходиться в застосунку, наявний об'єкт на полотні
Вхідні дані	-
Опис проведення тесту	Натиснути на кнопку «Зберегти» на панелі інструментів та перезавантажити сторінку.
Очікуваний результат	Редактор зображень завантажиться із збереженим станом.
Фактичний результат	Редактор зображень завантажився із збереженим станом.

Як можна побачити на рисунку 3.21, фігура була успішно додана на полотно зображення.

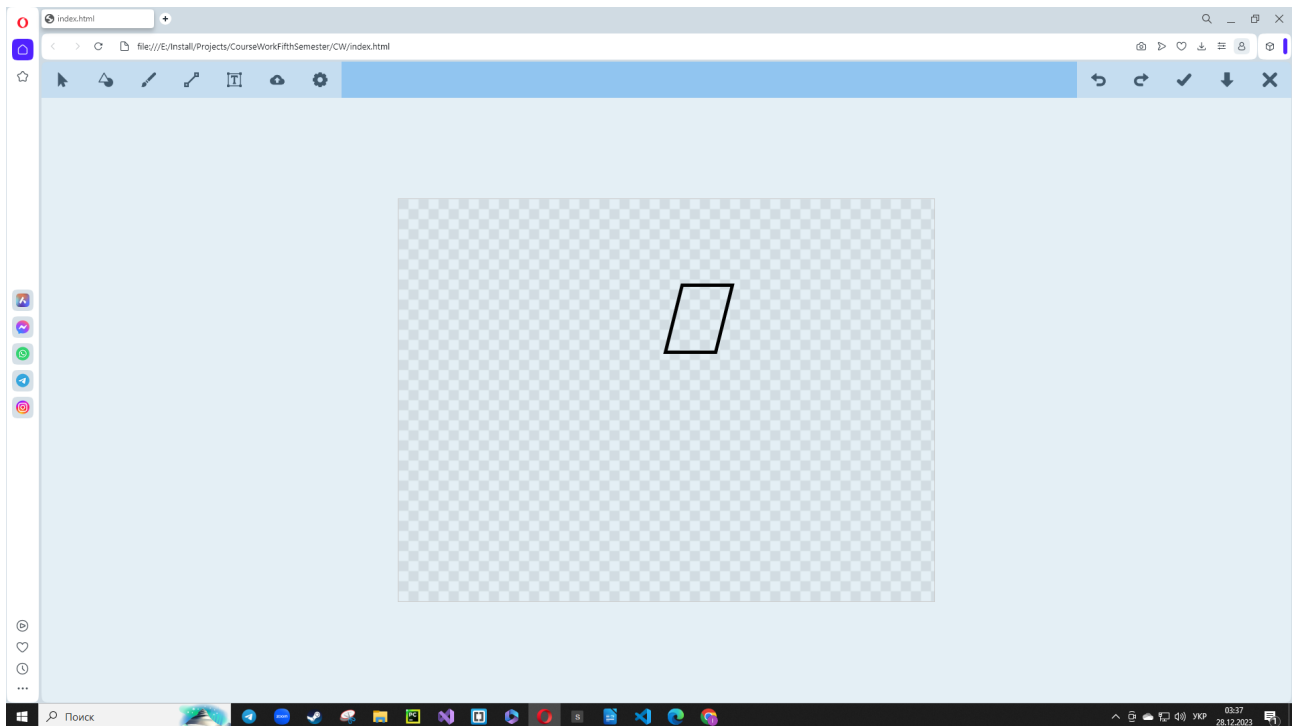


Рисунок 3.21 Ручна перевірка функції збереження.

Перевіримо працездатність функції очищення полотна. Процес показано в таблиці 3.18.

Таблиця 3.18 Тестування функції очищення полотна

Тест	Очищення полотна
Номер тесту	18
Початковий стан	Користувач знаходиться в застосунку, наявний об'єкт на полотні, наявне збереження стану
Вхідні дані	-
Опис проведення тесту	Натиснути на кнопку «Очистити» на панелі інструментів та перезавантажити сторінку.
Очікуваний результат	Редактор зображень очистить полотно і завантажиться із пустим полотном.
Фактичний результат	Редактор зображень очистив полотно і завантажився

із пустим полотном.

Як можна побачити на рисунку 3.22, полотно успішно очистилось.

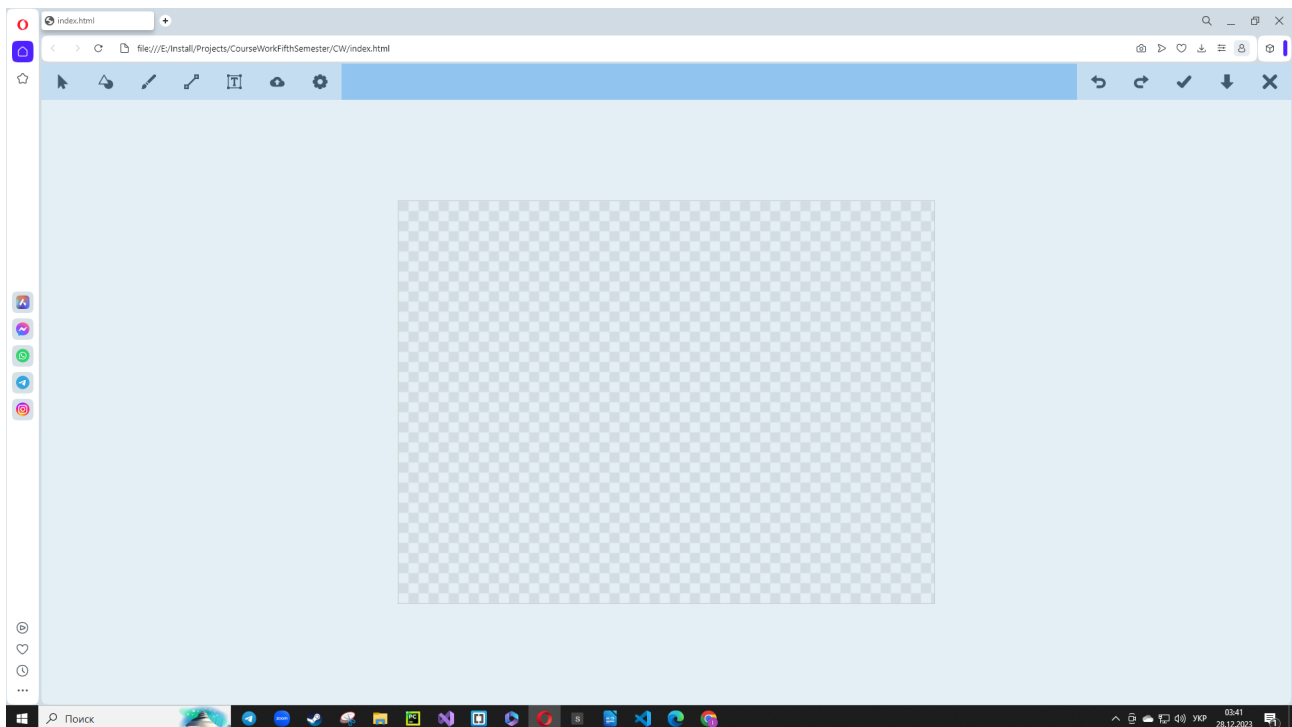


Рисунок 3.22 Ручна перевірка функції очищення.

Перевіримо працездатність функції завантаження остаточного зображення полотна. Процес показано в таблиці 3.19.

Таблиця 3.19 Тестування функції завантаження

Тест	Завантаження остаточного зображення
Номер тесту	19
Початковий стан	Користувач знаходиться в застосунку, полотно змінене.
Вхідні дані	-
Опис проведення тесту	Натиснути на кнопку «Завантажити» на панелі інструментів, в формі зображення вибрати розширення завантажуваного файлу зображення.
Очікуваний результат	Полотно у вигляді зображення завантажиться на

	сервер користувача.
Фактичний результат	Полотно у вигляді зображення завантажилося на сервер користувача.

Як можна побачити на рисунку 3.23 та 3.24, зображення успішно завантажилося.

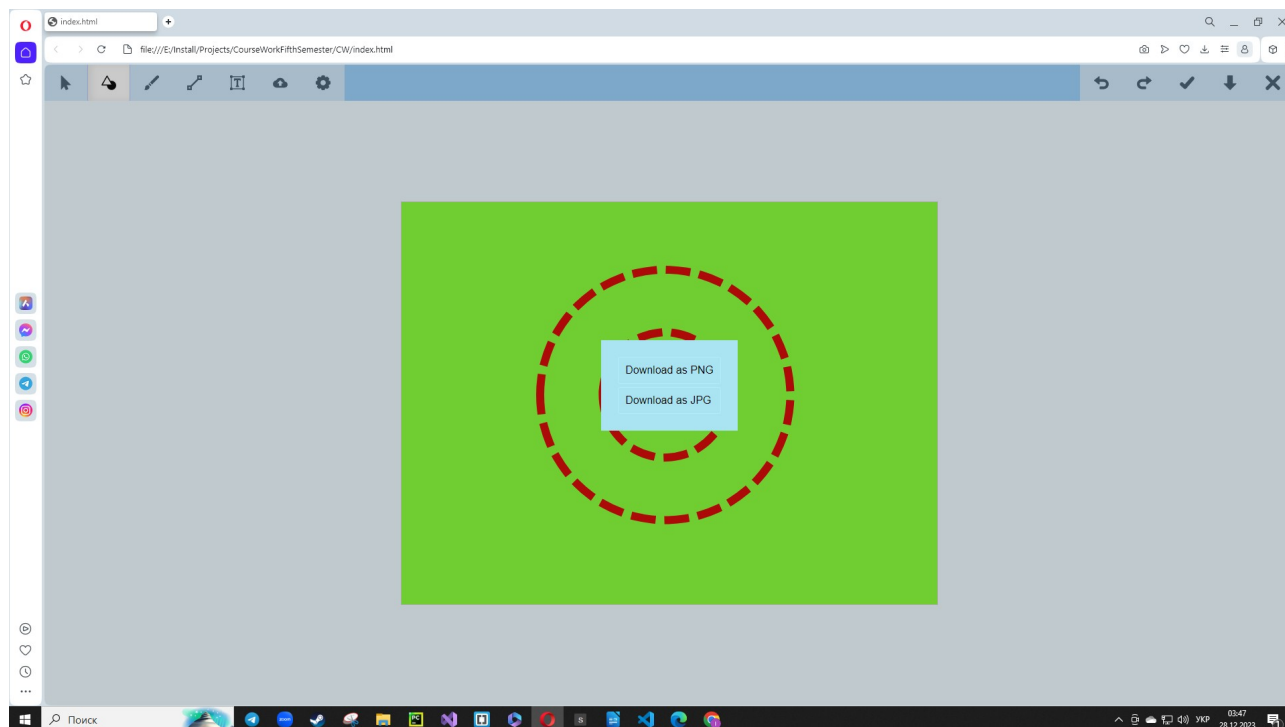


Рисунок 3.23 Форма вибору розширення зображення.

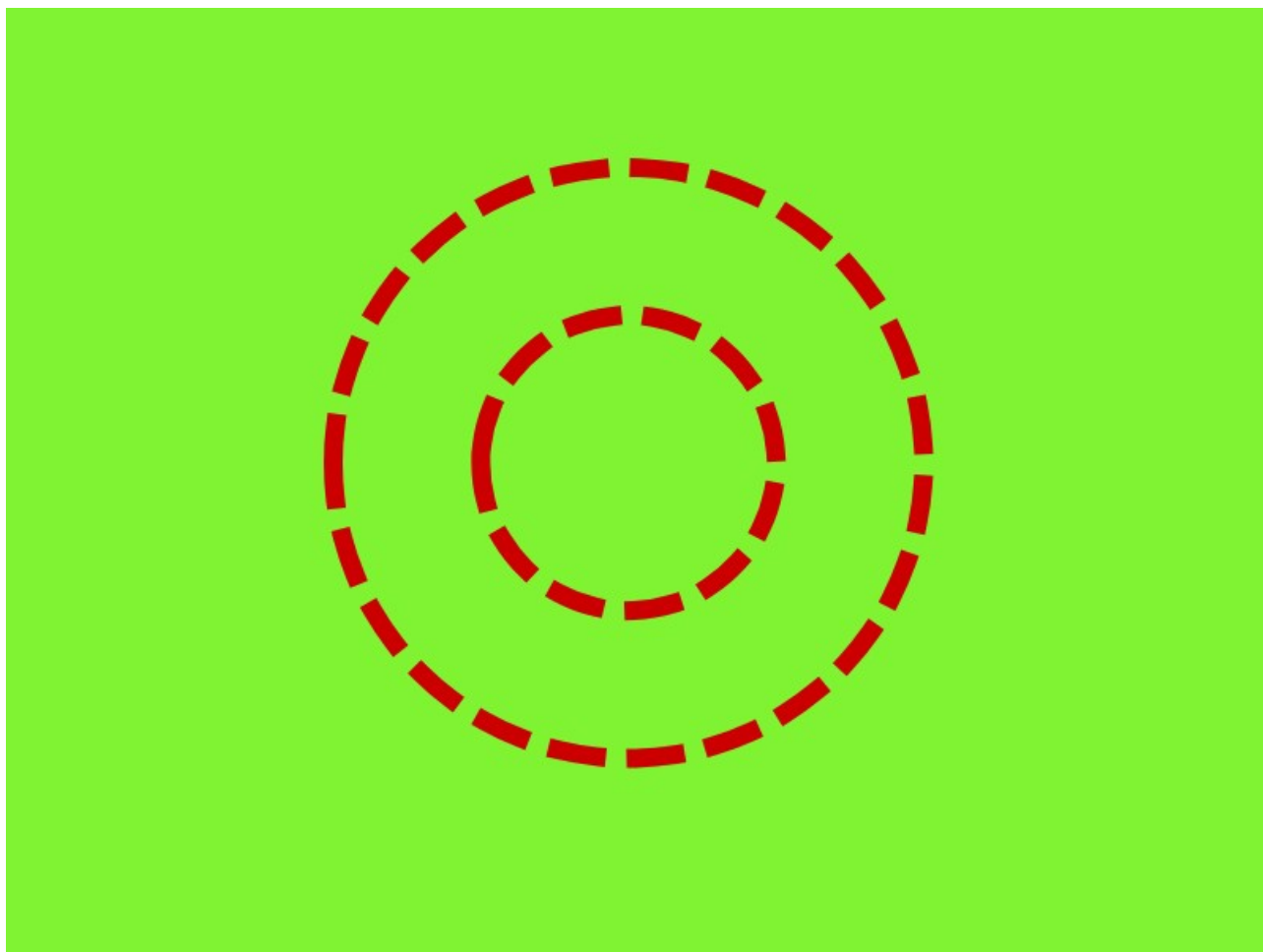


Рисунок 3.24 Результат завантаження остаточного зображення.

Перевіримо працездатність функції обрізання зображення. Процес показано в таблиці 3.20.

Таблиця 3.20 Тестування функції обрізання зображення

Тест	Вибір об'єкта на полотні
Номер тесту	20
Початковий стан	Користувач знаходиться в застосунку, наявне зображення на полотні
Вхідні дані	-
Опис проведення тесту	Масштабувати, спотворити, нахилити та розташувати зображення, виконати завантаження остаточного зображення полотна.

Очікуваний результат	Завантажиться обрізана частина зображення.
Фактичний результат	Завантажилась обрізана частина зображення.

Як можна побачити на рисунку 3.25 та 3.26, зображення було успішно обрізано.

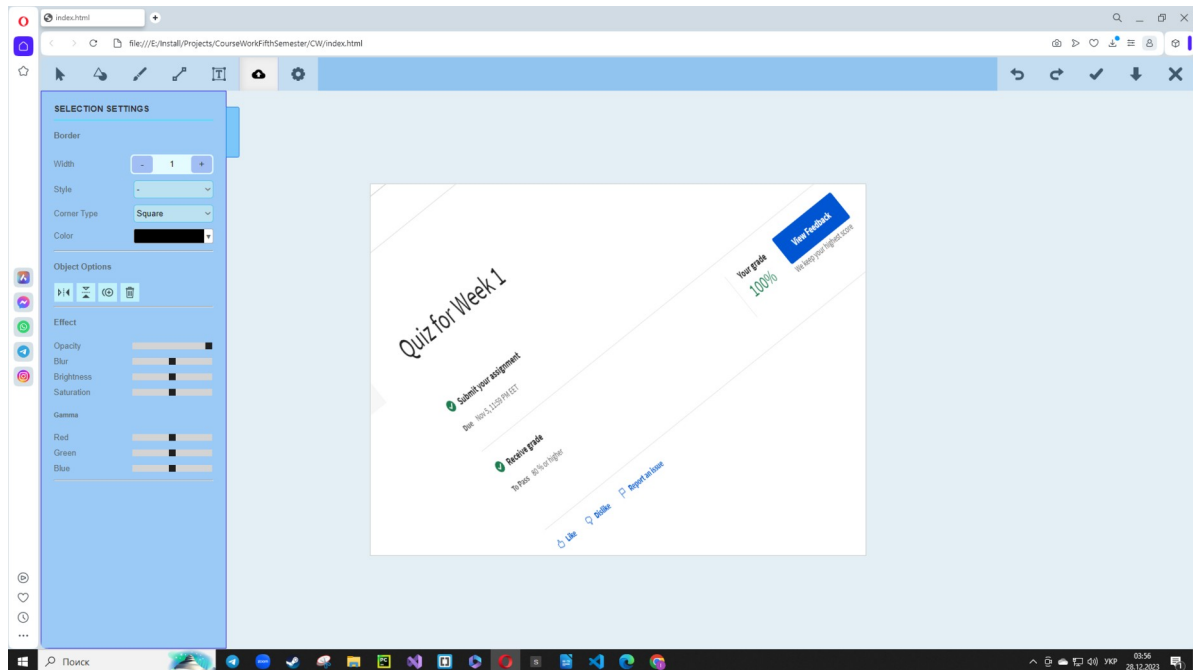


Рисунок 3.25 Підготовка зображення до обрізання.

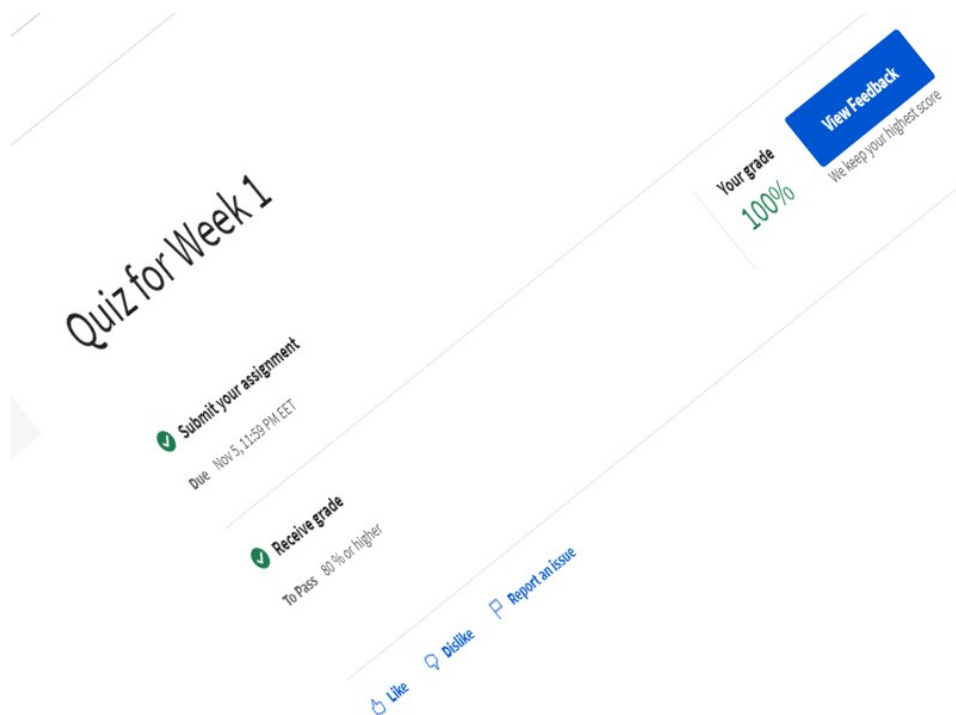


Рисунок 3.26 Результат обрізання зображення.

3.2 Висновки до розділу

Під час проведення мануального тестування було проведено тести на реагування системи на дії. Побачили, що система коректно відпрацьовує запити та відображає усі дії коректно.

Веб-редактор зображень задовільняє усі функціональні вимоги, відповідає вимогам надійності та зручності інтерфейсу.

4 ВПРОВАДЖЕННЯ І СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Розгортання програмного забезпечення

Для розгортання програмного забезпечення вам потрібно:

- Склонувати CW5_Image_Editor репозиторій:
https://github.com/Vladosichkek/CW5_Image_Editor
- Запустити файл index.html

4.2 Висновки до розділу:

У даному розділі було показано, що для того, щоб розгорнути додаток, треба клонувати репозиторій та запустити необхідний файл. Жодних бібліотек встановлювати не треба, адже всі необхідні бібліотеки або знаходяться в папках репозиторію, або файл index.html посилається на їх репозиторії в інтернеті.

ВИСНОВКИ

У процесі роботи над курсовою роботою та розробкою веб-редактора зображень підвищив ряд важливих навичок та знань:

- вивчив принципи розробки ефективних та гнучких архітектур програмного забезпечення;
- навчився використовувати паттерни проектування для досягнення гнучкості та розширюваності коду;
- здобув досвід використання JavaScript у розробці програмного забезпечення;
- освоїв техніки тестування різних аспектів програмного забезпечення, забезпечивши високу стабільність та надійність;
- вивчив процес оптимізації програмного забезпечення для підвищення його ефективності;
- отримав досвід встановлення пріоритетів та планування робочих завдань.

У процесі роботи над проектом та курсовою роботою я вдосконалив та поглибив свої знання в області розробки програмного забезпечення, освоїв нові інструменти та техніки, що покращило мої навички у галузі веб програмування.

У процесі аналізу вимог до веб-редактору зображень визначено ключові функціональні та нефункціональні вимоги. Серед них — продуктивність, надійність та зручність інтерфейсу. Спроектовано основні взаємодії з редактором зображень, визначено формати даних та розглянуто основні сценарії використання.

У розділі моделювання та конструювання була розроблена архітектура застосунку. Вона включає класи та компоненти, представлені для графічних об'єктів та полотен. Використано паттерн проектування, такий як Module для підвищенню безпеки та об'єднанню схожих функцій та змінних в одному місці для зручності управління та розвитку програми.

Тестування застосунка було проведено мануально. Тестування охопило різні аспекти функціональності, забезпечивши стабільність та надійність програмного забезпечення.

У розділі впровадження були детально описані кроки для розгортання веб-редактору зображень. Це включало клонування необхідного репозиторію та запуску необхідного файлу.

Майбутній розвиток веб-редактора зображень може включати розширення функціональності для підтримки нових операцій, покращення ефективності та роботи з великим обсягом графічних об'єктів. Інтеграція в інші веб-застосунки та підтримка нових типів файлів можуть стати напрямками подальшого розвитку. Постійна оптимізація та удосконалення можуть допомогти веб-редактору зображень вдосконалюватися та відповідати зростаючим вимогам.

ПЕРЕЛІК ПОСИЛАНЬ

1. Online Image Editor [Електронний ресурс] - <https://www.online-image-editor.com/>
2. Canva [Електронний ресурс] - <https://www.canva.com/>
3. Fotor [Електронний ресурс] - <https://www.fotor.com/>
4. I Love IMG [Електронний ресурс] - <https://www.iloveimg.com/ru>
5. BeFunky [Електронний ресурс] - <https://www.befunky.com/>
6. Business Process Model and Notation [Електронний ресурс] - https://en.wikipedia.org/wiki/Business_Process_Model_and_Notation
7. MVC (Model-View-Controller) [Електронний ресурс] - <https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller>
8. Fabric.js [Електронний ресурс] - <http://fabricjs.com/>
9. JavaScript [Електронний ресурс] - <https://ru.wikipedia.org/wiki/JavaScript>
10. Unified Modeling Language [Електронний ресурс] - https://en.wikipedia.org/wiki/Unified_Modeling_Language
11. “Module” патерн [Електронний ресурс] - https://en.wikipedia.org/wiki/Module_pattern
12. jQuery [Електронний ресурс] - <https://jquery.com/>
13. Структурна схема інтерфейсу [Електронний ресурс] - <https://agilemodeling.com/artifacts/uiflowdiagram.htm>

ДОДАТОК А

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”

Керівник роботи

Світлана ПОПЕРЕШНЯК

“2” листопада 2023 р.

Файлова система з консольним інтерфейсом

Технічне завдання

КП.ІІ-1125.045440.02.91

“ПОГОДЖЕНО”

Керівник роботи:

Світлана ПОПЕРЕШНЯК

Консультант:

Максим ГОЛОВЧЕНКО

Виконавець:

Владислав ПРИЩЕПА

Київ – 2023

ЗМІСТ

1	НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ.....	3
2	ПІДСТАВА ДЛЯ РОЗРОБКИ.....	4
3	ПРИЗНАЧЕННЯ РОЗРОБКИ.....	5
4	ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ.....	6
4.1	Вимоги до функціональних характеристик.....	6
4.1.1	Створення та збереження зображень.....	6
4.1.2	Обробка зображень як об'єктів.....	6
4.1.3	Малювання.....	6
4.1.4	Застосування специфічних дій над зображеннями.....	6
4.1.5	Обробка тексту.....	6
4.2	Вимоги до надійності.....	6
4.3	Умови експлуатації.....	7
4.3.1	Вид обслуговування.....	7
4.3.2	Обслуговуючий персонал.....	7
4.4	Вимоги до складу і параметрів технічних засобів.....	7
4.5	Вимоги до інформаційної та програмної сумісності.....	7
4.5.1	Вимоги до вхідних даних.....	7
4.5.2	Вимоги до вихідних даних.....	7
4.5.3	Вимоги до мови розробки.....	8
4.5.4	Вимоги до середовища розробки.....	8
4.5.5	Вимоги до представленню вихідних кодів.....	8
4.6	Вимоги до маркування та пакування.....	8
4.7	Вимоги до транспортування та зберігання.....	8
4.8	Спеціальні вимоги.....	8
5	ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ.....	9
5.1	Попередній склад програмної документації.....	9
5.2	Спеціальні вимоги до програмної документації.....	9
6	СТАДІЇ І ЕТАПИ РОЗРОБКИ.....	10
7	ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ.....	11

1 НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: Веб-застосунок для створення та редагування зображень.

Галузь застосування:

Наведене технічне завдання поширюється на розробку веб-застосунку для створення та редагування зображень, котрий використовується для створення та редагування зображень та призначений для загального користування у веб-середовищі.

2 ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки веб-застосунку для створення та редагування зображень є завдання на курсову роботу, затверджене кафедрою інформатики та програмної інженерії Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського».

3 ПРИЗНАЧЕННЯ РОЗРОБКИ

Розробка призначена для створення та редагування зображень у веб-середовищі.

Метою розробки є розширення функціональних можливостей обробки зображень у веб-застосунках шляхом додавання можливостей вирізання фрагменту із зображення неправильної форми та спотворення зображення.

4 ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Вимоги до функціональних характеристик

Програмне забезпечення повинно забезпечувати виконання наступних основних функцій:

4.1.1 Створення та збереження зображень

- Завантаження початкового зображення (uploadimg);
- Створення нового зображення (createimg);
- Збереження змін (saveimg);
- Завантаження зображення (downloadimg);

4.1.2 Обробка зображень як об'єктів

- Обрізання зображення (cutimg);
- Вирізання із зображення (cutfromimg);
- Вставлення в зображення (insertinimg);
- Додання фігур (insertfigure);
- Видалення із зображення (deletefromimg);
- Зміна розміру зображення (resizeimg);
- Встановлення черговості слоїв (rearangelayer);

4.1.3 Малювання

- Малювання ліній (drawline);
- Стирання (eraseobj);

4.1.4 Застосування специфічних дій над зображеннями

- Спотворення зображення (confusionimg);
- Застосування фільтрів (usefilter);

4.1.5 Обробка тексту

- Додання тексту (addtext);
- Зміна тексту (changetext).

4.2 Вимоги до надійності

Передбачити контроль введення інформації та захист від некоректних дій користувача.

4.3 Умови експлуатації

Умови експлуатації згідно СанПін 2.2.2.542 – 96.

4.3.1 Вид обслуговування

Вимоги до виду обслуговування не висуваються.

4.3.2 Обслуговуючий персонал

Вимоги до обслуговуючого персоналу не висуваються.

4.4 Вимоги до складу і параметрів технічних засобів

Програмне забезпечення повинно функціонувати на IBM-сумісних персональних комп'ютерах.

Мінімальна конфігурація технічних засобів:

- тип процесору: Intel Core i5;
- об'єм ОЗП: 4 Гб;
- підключення до мережі Інтернет зі швидкістю від 20 мегабіт;
- назва операційної системи: Windows XP;
- тип операційної системи: 32-бітна.

Рекомендована конфігурація технічних засобів:

- тип процесору: AMD Ryzen 7 2700X Eight-Core Processor;
- об'єм ОЗП: 16 Гб;
- підключення до мережі Інтернет зі швидкістю від 100 мегабіт;
- назва операційної системи: Windows 10 версії 22H2;
- тип операційної системи: 64-бітна.

4.5 Вимоги до інформаційної та програмної сумісності

Програмне забезпечення повинно працювати під управлінням операційних систем сімейства WIN32 (Windows®XP, Windows NT і т.д.) або WIN64 (Windows 10 і т.д.).

4.5.1 Вимоги до вхідних даних

Вхідні дані повинні бути представлені в наступному форматі: .png, .jpg.

4.5.2 Вимоги до вихідних даних

Результати повинні бути представлені в наступному форматі: .png, .jpg.

4.5.3 Вимоги до мови розробки

Розробку виконати на мові програмування Java.

4.5.4 Вимоги до середовища розробки

Розробку виконати на платформі Windows 10 версії 22H2 за допомогою середовища розробки Visual Studio Code.

4.5.5 Вимоги до представлення вихідних кодів

Вихідний код програми має бути представлений у вигляді JavaScript, CSS та HTML файлів.

4.6 Вимоги до маркування та пакування

Вимоги до маркування та пакування не висуваються.

4.7 Вимоги до транспортування та зберігання

Вимоги до транспортування та зберігання не висуваються.

4.8 Спеціальні вимоги

Налаштовувати HTML-файл для збірки бінарного файлу із програмою.

5 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

5.1 Попередній склад програмної документації

У склад супроводжувальної документації повинні входити наступні документи на аркушах формату А4:

- пояснювальна записка;
- технічне завдання;
- керівництво користувача;
- програма та методика тестування;
- текст програми.

Графічна частина повинна бути виконана на аркушах формату А3 та містити наступні документи:

- схема структурна варіантів використання;
- схеми взаємодії об'єктів, об'єктна декомпозиція;
- схема структурна класів програмного забезпечення;
- схема структурна станів інтерфейсу;
- креслення вигляду екранних форм.

5.2 Спеціальні вимоги до програмної документації

Програмні модулі, котрі розробляються, повинні бути задокументовані, тобто тексти програм повинні містити всі необхідні коментарі.

6 СТАДІЇ І ЕТАПИ РОЗРОБКИ

№	Назва етапу	Строк	Звітність
1.	Вивчення літератури за тематикою роботи	6.11	
2.	Розробка технічного завдання	13.11	Технічне завдання
3.	Аналіз вимог та уточнення специфікацій	04.12	Специфікації програмного забезпечення
4.	Проектування структури програмного забезпечення, проектування компонентів	04.12	Схема структурна програмного забезпечення та специфікація компонентів (діаграма класів, схема алгоритму)
5.	Програмна реалізація програмного забезпечення	04.12	Тексти програмного забезпечення
6.	Тестування програмного забезпечення	11.12	Тести, результати тестування
7.	Розробка матеріалів текстової частини роботи	18.12	Пояснювальна записка
8.	Розробка матеріалів графічної частини роботи	25.12	Графічний матеріал проекту
9.	Оформлення технічної документації роботи	25.12	Технічна документація

7 ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ

Тестування розробленого програмного продукту виконується відповідно до “Програми та методики тестування”.

ДОДАТОК Б

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”

Керівник роботи

Світлана ПОПЕРЕШНЯК

“28” грудня 2023 р.

Файлова система з консольним інтерфейсом

Опис програми

КПІ.ІІ-1125.045440.03.34

“ПОГОДЖЕНО”

Керівник роботи:

Світлана ПОПЕРЕШНЯК

Консультант:

Максим ГОЛОВЧЕНКО

Виконавець:

Владислав ПРИЩЕПА

Київ – 2023

```
// ../index.html

<html>

<head>

  <link rel="stylesheet" href="/lib/style.css">

  <script src="https://code.jquery.com/jquery-3.5.1.min.js"></script>

  <script>

    jQuery.fn.outerHTML = function() {
      return jQuery('<div />').append(this.eq(0).clone()).html();
    };

    Number.prototype.countDecimals = function () {
      if(Math.floor(this.valueOf()) === this.valueOf()) return 0;
      return this.toString().split(".")[1].length || 0;
    }

  </script>

  <script src="https://cdnjs.cloudflare.com/ajax/libs/fabric.js/3.6.3/fabric.min.js"></
script>

  <script
src="https://cdn.jsdelivr.net/npm/spectrum-colorpicker2/dist/spectrum.min.js"></
script>

  <link rel="stylesheet" type="text/css" href="https://cdn.jsdelivr.net/npm/spectrum-
colorpicker2/dist/spectrum.min.css">

  <script src="/external/grapick.min.js"></script>

  <link rel="stylesheet" href="/external/grapick.min.css">

  <script src="/external/undo-redo-stack.js"></script>

  <script src="/lib/core.js"></script>

  <script src="/lib/toolbar.js"></script>

  <script src="/lib/canvas.js"></script>

  <script src="/lib/shapes.js"></script>

  <script src="/lib/freeDrawSettings.js"></script>
```



```

<script src="./lib/canvasSettings.js"></script>
<script src="./lib/selectionSettings.js"></script>
<script src="./lib/drawingLine.js"></script>
<script src="./lib/drawingText.js"></script>
<script src="./lib/upload.js"></script>
<script src="./lib/copyPaste.js"></script>
<script src="./lib/utils.js"></script>
<script src="./lib/zoom.js"></script>
<script src="./lib/saveInBrowser.js"></script>
</head>
<body>
  <div id="image-editor-container"></div>
</body>
<script src="./script.js"></script>
</html>
// ../script.js
try {
  // define toolbar buttons to show
  // if this value is undefined or its length is 0, default toolbar buttons will be shown
  const buttons = [
    'select',
    'shapes',
    'draw',
    'line',
    'textbox',
    'upload',
    'background',
    'undo',
    'redo',

```

```
'save',  
'download',  
'clear'  
];
```

```
var imgEditor = new ImageEditor('#image-editor-container', buttons, []);  
console.log('initialize image editor');
```

```
} catch ( ) {  
  const browserWarning = document.createElement('div')  
  browserWarning.innerHTML = '<p style="line-height: 26px; margin-top: 100px;  
font-size: 16px; color: #555">Your browser is out of date!<br/>Please update to a  
modern browser, for example:<a href="https://www.google.com/chrome/"  
target="_blank">Chrome</a>!</p>';
```

```
  browserWarning.setAttribute(  
    'style',  
    'position: fixed; z-index: 1000; width: 100%; height: 100%; top: 0; left: 0;  
background-color: #f9f9f9; text-align: center; color: #555;'  
  )
```

```
// check for flex and grid support
```

```
let divGrid = document.createElement('div')  
divGrid.style['display'] = 'grid'  
let supportsGrid = divGrid.style['display'] === 'grid'
```

```

let divFlex = document.createElement('div')
divFlex.style['display'] = 'flex'
let supportsFlex = divFlex.style['display'] === 'flex'

if (!supportsGrid || !supportsFlex) {
  document.body.appendChild(browserWarning)
}
}

// ../lib/canvas.js
//Canvas section management of image editor
(function () {
  'use strict';
  var canvas = function () {
    try {
      $('` ${this.containerSelector} .main-panel`').append(`<div class="canvas-holder"
id="canvas-holder"><div class="content"><canvas id="c"></canvas></div></div>`);
      const fabricCanvas = new fabric.Canvas('c').setDimensions({
        width: 800,
        height: 600
      })
      fabricCanvas.originalW = fabricCanvas.width;
      fabricCanvas.originalH = fabricCanvas.height;
      // set up selection style
      fabric.Object.prototype.transparentCorners = false;
      fabric.Object.prototype.cornerStyle = 'circle';
      fabric.Object.prototype.borderColor = '#6495ED';
      fabric.Object.prototype.cornerColor = '#6495ED';
      fabric.Object.prototype.cornerStrokeColor = '#A9A9A9';
    }
  }
}());

```

```

fabric.Object.prototype.padding = 0;
// retrieve active selection to react state
fabricCanvas.on('selection:created', (e) => this.setActiveSelection(e.target))
fabricCanvas.on('selection:updated', (e) => this.setActiveSelection(e.target))
fabricCanvas.on('selection:cleared', (e) => this.setActiveSelection(null))
// snap to an angle on rotate
fabricCanvas.on('object:rotating', (e) => {
  e.target.snapAngle = false;
})
fabricCanvas.on('object:modified', () => {
  console.log('trigger: modified')
  let currentState = this.canvas.toJSON();
  this.history.push(JSON.stringify(currentState));
})
const savedCanvas = saveInBrowser.load('canvasEditor');
if (savedCanvas) {
  fabricCanvas.loadFromJSON(savedCanvas,
fabricCanvas.renderAll.bind(fabricCanvas));
}
// delete object on del key
(() => {
  document.addEventListener('keydown', (e) => {
    const key = e.which || e.keyCode;
    if (
      key === 46 &&
      document.querySelectorAll('textarea:focus, input:focus').length === 0
    ) {
      fabricCanvas.getActiveObjects().forEach(obj => {
        fabricCanvas.remove(obj);
      });
    }
  });
})

```

```

    });
    fabricCanvas.discardActiveObject().requestRenderAll();
    fabricCanvas.trigger('object:modified')
  }
})
})();

setTimeout(() => {
  let currentState = fabricCanvas.toJSON();
  this.history.push(JSON.stringify(currentState));
}, 1000);
return fabricCanvas;
} catch ( ) {
  console.error("can't create canvas instance");
  return null;
}
}

window.ImageEditor.prototype.initializeCanvas = canvas;
})();

```

```
// ../lib/canvasSettings.js
```

```
//initialize canvas setting panel
```

```

(function () {
  'use strict';
  var canvasSettings = function () {
    const _self = this;
    $('` ${this.containerSelector} .main-panel `').append(`<div class="toolpanel"
id="background-panel"><div class="content"><p class="title">Canvas
Settings</p></div></div>`);
    // set dimension section

```

```

() => {
  $('${this.containerSelector} .toolpanel#background-panel .content').append(
    <div class="canvas-size-setting">
      <p>Canvas Size</p>
      <div class="input-container">
        <label>Width</label>
        <div class="custom-number-input">
          <button class="decrease">-</button>
          <input type="number" min="100" id="input-width" value="800"/>
          <button class="increase">+</button>
        </div>
      </div>
      <div class="input-container">
        <label>Height</label>
        <div class="custom-number-input">
          <button class="decrease">-</button>
          <input type="number" min="100" id="input-height" value="600"/>
          <button class="increase">+</button>
        </div>
      </div>
    </div>
  ');
  var setDimension = () => {
    try {
      let width = $('${this.containerSelector} .toolpanel#background-panel .content
#input-width').val();
      let height = $('${this.containerSelector} .toolpanel#background-
panel .content #input-height').val();
      _self.canvas.setWidth(width)
    }
  }
}

```

```

        _self.canvas.originalW = width
        _self.canvas.setHeight(height)
        _self.canvas.originalH = height
        _self.canvas.renderAll()
        _self.canvas.trigger('object:modified')
    } catch (e) {}
}

$(`${this.containerSelector} .toolpanel#background-panel .content #input-
width`).change(setDimension)

$(`${this.containerSelector} .toolpanel#background-panel .content #input-
height`).change(setDimension)
}))();
// end set dimension section

// background color
(() => {
    `${this.containerSelector} .toolpanel#background-panel .content`).append(
<div class="color-settings">
    <div class="tab-container">
        <div class="tabs">
            <div class="tab-label" data-value="color-fill">Color Fill</div>
            <div class="tab-label" data-value="gradient-fill">Gradient Fill</div>
        </div>
        <div class="tab-content" data-value="color-fill">
            <input id="color-picker" value="black"/><br>
        </div>
        <div class="tab-content" data-value="gradient-fill">
            <div id="gradient-picker"></div>
            <div class="gradient-orientation-container">

```

```

<div class="input-container">
  <label>Orientation</label>
  <select id="select-orientation">
    <option value="linear">Linear</option>
    <option value="radial">Radial</option>
  </select>
</div>
<div id="angle-input-container" class="input-container">
  <label>Angle</label>
  <div class="custom-number-input">
    <button class="decrease">-</button>
    <input type="number" min="0" max="360" value="0" id="input-angle">
    <button class="increase">+</button>
  </div>
</div>
</div>
</div>
</div>
</div>
')
$(`${this.containerSelector} .toolpanel#background-panel .content .tab-
label`).click(function () {
  `${$_self.containerSelector} .toolpanel#background-panel .content .tab-
label`).removeClass('active');
  $(this).addClass('active');
  let target = $(this).data('value');
  $(this).closest('.tab-container').find('.tab-content').hide();
  $(this).closest('.tab-container').find(`.tab-content[data-value=$
{target}]`).show();

```



```

    if (target === 'color-fill') {
        let color = $('${_self.containerSelector} .toolpanel#background-
panel .content #color-picker').val();
        try {
            _self.canvas.backgroundColor = color;
            _self.canvas.renderAll();
        } catch (_) {
            console.log("can't update background color")
        }
    } else {
        updateGradientFill();
    }
})

$('${this.containerSelector} .toolpanel#background-panel .content .tab-
label[data-value=color-fill]').click();

$('${this.containerSelector} .toolpanel#background-panel .content #color-
picker').spectrum({
    flat: true,
    showPalette: false,
    showButtons: false,
    type: "color",
    showInput: "true",
    allowEmpty: "false",
    move: function (color) {
        let hex = 'transparent';
        color && (hex = color.toRgbString()); // #ff0000
        _self.canvas.backgroundColor = hex;
        _self.canvas.renderAll();
    }
}

```

```

});
const gp = new Grapick({
  el: `${this.containerSelector} .toolpanel#background-panel .content #gradient-
picker`,
  colorEl: '<input id="colorpicker"/>'
});
gp.setColorPicker(handler => {
  const el = handler.getEl().querySelector('#colorpicker');
  $(el).spectrum({
    showPalette: false,
    showButtons: false,
    type: "color",
    showInput: "true",
    allowEmpty: "false",
    color: handler.getColor(),
    showAlpha: true,
    change(color) {
      handler.setColor(color.toRgbString());
    },
    move(color) {
      handler.setColor(color.toRgbString(), 0);
    }
  });
});
gp.addHandler(0, 'red');
gp.addHandler(100, 'blue');
const updateGradientFill = () => {
  let stops = gp.getHandlers();

```

```

    let orientation = $('${this.containerSelector} .toolpanel#background-
panel .content .gradient-orientation-container #select-orientation').val();

    let angle = parseInt($('${this.containerSelector} .toolpanel#background-
panel .content .gradient-orientation-container #input-angle').val());

    let gradient = generateFabricGradientFromColorStops(stops,
_self.canvas.width, _self.canvas.height, orientation, angle);
    _self.canvas.setBackgroundColor(gradient)
    _self.canvas.renderAll()
}

// Do stuff on change of the gradient
gp.on('change', complete => {
    updateGradientFill();
})

$('${this.containerSelector} .toolpanel#background-panel .content .gradient-
orientation-container #select-orientation').change(function () {
    let type = $(this).val();
    if (type === 'radial') {
        $(this).closest('.gradient-orientation-container').find('#angle-input-
container').hide();
    } else {
        $(this).closest('.gradient-orientation-container').find('#angle-input-
container').show();
    }
    updateGradientFill();
})

$('${this.containerSelector} .toolpanel#background-panel .content .gradient-
orientation-container #input-angle').change(function () {
    updateGradientFill();

```

```

    })
  })0;
}
window.ImageEditor.prototype.initializeCanvasSettingPanel = canvasSettings;
})0

```

```
// ../lib/copyPaste.js
```

```
//Define copy/paste actions on fabric js canvas
```

```

(function () {
  'use strict';
  const copyPaste = (canvas) => {
    // copy
    document.addEventListener('copy', (e) => {
      if (!canvas.getActiveObject()) return
      // copy image as dataUrl
      if (canvas.getActiveObject().type === 'image') {
        e.preventDefault()
        e.clipboardData.setData('text/plain', canvas.getActiveObject().toDataURL())
      }
      // if selection is not an image, copy as JSON
      if (canvas.getActiveObject().type !== 'image') {
        e.preventDefault()
        canvas.getActiveObject().clone((cloned) => {
          e.clipboardData.setData('text/plain', JSON.stringify(cloned.toJSON()))
        })
      }
    })
  }
})

```

```

// JSON string validator
const isJSONObjectString = (s) => {
  try {
    const o = JSON.parse(s);
    return !!o && (typeof o === 'object') && !Array.isArray(o)
  } catch {
    return false
  }
}

// base64 validator
const isBase64String = (str) => {
  try {
    str = str.split('base64,').pop()
    window.atob(str)
    return true
  } catch (e) {
    return false
  }
}

// paste
document.addEventListener('paste', (e) => {
  let pasteTextData = e.clipboardData.getData('text')
  // check if base64 image
  if (pasteTextData && isBase64String(pasteTextData)) {
    fabric.Image.fromURL(pasteTextData, (img) => {
      img.set({
        left: 0,
        top: 0
      })
    })
  }
})

```

```

    img.scaleToHeight(100)
    img.scaleToWidth(100)
    canvas.add(img)
    canvas.setActiveObject(img)
    canvas.trigger('object:modified')
  })
  return
}

// check if there's an image in clipboard items
if (e.clipboardData.items.length > 0) {
  for (let i = 0; i < e.clipboardData.items.length; i++) {
    if (e.clipboardData.items[i].type.indexOf('image') === 0) {
      let blob = e.clipboardData.items[i].getAsFile()
      if (blob !== null) {
        let reader = new FileReader()
        reader.onload = (f) => {
          fabric.Image.fromURL(f.target.result, (img) => {
            img.set({
              left: 0,
              top: 0
            })
            img.scaleToHeight(100)
            img.scaleToWidth(100)
            canvas.add(img)
            canvas.setActiveObject(img)
            canvas.trigger('object:modified')
          })
        }
        reader.readAsDataURL(blob)
      }
    }
  }
}

```

```

    }
  }
}
}
// check if JSON and type is valid
let validTypes = ['rect', 'circle', 'line', 'path', 'polygon', 'polyline', 'textbox',
'group']
if (isJSONObjectString(pasteTextData)) {
  let obj = JSON.parse(pasteTextData)
  if (!validTypes.includes(obj.type)) return
  // insert and select
  fabric.util.enlivenObjects([obj], function (objects) {
    objects.forEach(function (o) {
      o.set({
        left: 0,
        top: 0
      })
      canvas.add(o)
      o.setCoords()
      canvas.setActiveObject(o)
    })
    canvas.renderAll()
    canvas.trigger('object:modified')
  })
}
})
}
window.ImageEditor.prototype.initializeCopyPaste = copyPaste;
})()

```

```

// ../lib/core.js

//The Core of Image Editor
(function () {
  'use strict';
  /**
   * Image Editor class
   * @param {String} containerSelector jquery selector for image editor container
   * @param {Array} buttons define toolbar buttons
   * @param {Array} shapes define shapes
   */
  var ImageEditor = function (containerSelector, buttons, shapes) {
    this.containerSelector = containerSelector;
    this.containerEl = $(containerSelector);
    this.buttons = buttons;
    this.shapes = shapes;
    this.containerEl.addClass('default-container');
    this.canvas = null;
    this.activeTool = null;
    this.activeSelection = null;
    /**
     * Get current state of canvas as object
     * @returns {Object}
     */
    this.getCanvasJSON = () => {
      return this.canvas.toJSON();
    }
  }
  /**

```



```

* Set canvas status by object
* @param {Object} current the object of fabric canvas status
*/
this.setCanvasJSON = (current) => {
  current && this.canvas.loadFromJSON(JSON.parse(current),
this.canvas.renderAll.bind(this.canvas))
}
/**
* Event handler to set active tool
* @param {String} id tool id
*/
this.setActiveTool = (id) => {
  this.activeTool = id;
  $(' ${containerSelector} .toolpanel').removeClass('visible');
  if (id !== 'select' || (id === 'select' && this.activeSelection)) {
    $(' ${containerSelector} .toolpanel#${id}-panel').addClass('visible');
    if (id === 'select') {
      console.log('selection')
      $(' ${containerSelector} .toolpanel#${id}-panel').attr('class', `toolpanel
visible type-${this.activeSelection.type}`)
    }
  }
  if (id !== 'select') {
    this.canvas.discardActiveObject();
    this.canvas.renderAll();
    this.activeSelection = null;
  }
  this.canvas.isDrawingLineMode = false;
  this.canvas.isDrawingMode = false;

```

```

this.canvas.isDrawingTextMode = false;
this.canvas.defaultCursor = 'default';
this.canvas.selection = true;
this.canvas.forEachObject(o => {
  o.selectable = true;
  o.evented = true;
})
switch (id) {
  case 'draw':
    this.canvas.isDrawingMode = true;
    break;
  case 'line':
    this.canvas.isDrawingLineMode = true
    this.canvas.defaultCursor = 'crosshair'
    this.canvas.selection = false
    this.canvas.forEachObject(o => {
      o.selectable = false
      o.evented = false
    });
    break;
  case 'textbox':
    this.canvas.isDrawingTextMode = true
    this.canvas.defaultCursor = 'crosshair'
    this.canvas.selection = false
    this.canvas.forEachObject(o => {
      o.selectable = false
      o.evented = false
    });
    break;

```

```

    case 'upload':
        this.openDragDropPanel();
        break;
    default:
        break;
}
}
/**
 * Event handler when perform undo
 */
this.undo = () => {
    console.log('undo')
    try {
        let undoList = this.history.getValues().undo;
        if (undoList.length) {
            let current = undoList[undoList.length - 1];
            this.history.undo();
            current && this.canvas.loadFromJSON(JSON.parse(current),
this.canvas.renderAll.bind(this.canvas))
        }
    } catch (_) {
        console.error("undo failed")
    }
}
/**
 * Event handler when perform redo
 */
this.redo = () => {
    console.log('redo')

```

```

try {
  let redoList = this.history.getValues().redo;
  if (redoList.length) {
    let current = redoList[redoList.length - 1];
    this.history.redo();
    current && this.canvas.loadFromJSON(JSON.parse(current),
this.canvas.renderAll.bind(this.canvas))
  }
} catch (_) {
  console.error("redo failed")
}
}
/**
 * Event handler when select objects on fabric canvas
 * @param {Object} activeSelection fabric js object
 */
this.setActiveSelection = (activeSelection) => {
  this.activeSelection = activeSelection;
  this.setActiveTool('select');
}
/**
 * Initialize undo/redo stack
 */
this.configUndoRedoStack = () => {
  this.history = window.UndoRedoStack();
  const ctrZY = (e) => {
    const key = e.which || e.keyCode;

```

```

        if (e.ctrlKey && document.querySelectorAll('textarea:focus,
input:focus').length === 0) {
            if (key === 90) this.undo()
            if (key === 89) this.redo()
        }
    }
    document.addEventListener('keydown', ctrZY)
}
/**
 * Initialize zoom events
 */
this.initializeZoomEvents = () => {
    this.applyZoom = (zoom) => {
        this.canvas.setZoom(zoom)
        this.canvas.setWidth(this.canvas.originalW * this.canvas.getZoom())
        this.canvas.setHeight(this.canvas.originalH * this.canvas.getZoom())
    }
    // zoom out/in/reset (ctr + -/+ /0)
    const keyZoom = (e) => zoomWithKeys(e, this.canvas, this.applyZoom)
    document.addEventListener('keydown', keyZoom)
    // zoom out/in with mouse
    const mouseZoom = (e) => zoomWithMouse(e, this.canvas, this.applyZoom)
    document.addEventListener('wheel', mouseZoom, {
        passive: false
    })
}
/**
 * Initialize image editor
 */

```

```

this.init = () => {
  this.configUndoRedoStack();
  this.initializeToolbar();
  this.initializeMainPanel();
  this.initializeShapes();
  this.initializeFreeDrawSettings();
  this.initializeCanvasSettingPanel();
  this.initializeSelectionSettings();
  this.canvas = this.initializeCanvas();
  this.initializeLineDrawing(this.canvas);
  this.initializeTextBoxDrawing(this.canvas);
  this.initializeUpload(this.canvas);
  this.initializeCopyPaste(this.canvas);
  this.initializeZoomEvents();
  this.extendHideShowToolPanel();
  this.extendNumberInput();
}
/**
 * Initialize main panel
 */
this.initializeMainPanel = () => {
  $('` ${containerSelector} `').append('<div class="main-panel"></div>');
}
/**
 * Add features to hide/show tool panel
 */
this.extendHideShowToolPanel = () => {
  $('` ${this.containerSelector} .toolpanel .content `').each(function () {
    $(this).append('<div class="hide-show-handler"></div>')
  })
}

```

```

    })
    $('`{$this.containerSelector} .toolpanel .content .hide-show-
handler`').click(function () {
        let panel = $(this).closest('.toolpanel');
        panel.toggleClass('closed');
    })
}
/**
 * Extend custom number input with increase/decrease button
 */
this.extendNumberInput = () => {
    $('`{$containerSelector} .decrease`').click(function () {
        let input = $(this).closest('.custom-number-input').find('input[type=number]')
        let step = input.attr('step');
        if (!step) step = 1;
        else {
            step = parseFloat(step);
        }
        let val = parseFloat(input.val());
        input.val((val - step).toFixed(step.countDecimals()));
        input.change();
    })
    $('`{$containerSelector} .increase`').click(function () {
        let input = $(this).closest('.custom-number-input').find('input[type=number]')
        let step = input.attr('step');
        if (!step) step = 1;
        else {
            step = parseFloat(step);
        }
    })
}

```

```

        let val = parseFloat(input.val());
        input.val((val + step).toFixed(step.countDecimals()));
        input.change();
    })
}
this.init();
}
window.ImageEditor = ImageEditor;
})();

```

```
// ../lib/drawingLine.js
```

```
//Define action to draw line by mouse actions
```

```

(function () {
    'use strict';
    var lineDrawing = function (fabricCanvas) {
        let color = '#000000';
        let width = 5;
        let isDrawingLine = false,
            lineToDraw, pointer, pointerPoints
        fabricCanvas.on('mouse:down', (o) => {
            if (!fabricCanvas.isDrawingLineMode) return
            isDrawingLine = true
            pointer = fabricCanvas.getPointer(o.e)
            pointerPoints = [pointer.x, pointer.y, pointer.x, pointer.y]
            lineToDraw = new fabric.Line(pointerPoints, {
                strokeWidth: width,
                stroke: color
            });

```



```

    lineToDraw.selectable = false
    lineToDraw.evented = false
    lineToDraw.strokeUniform = true
    fabricCanvas.add(lineToDraw)
  });
  fabricCanvas.on('mouse:move', (o) => {
    if (!isDrawingLine) return
    pointer = fabricCanvas.getPointer(o.e)
    lineToDraw.set({
      x2: pointer.x,
      y2: pointer.y
    })
    fabricCanvas.renderAll()
  });
  fabricCanvas.on('mouse:up', () => {
    if (!isDrawingLine) return
    lineToDraw.setCoords()
    isDrawingLine = false
    fabricCanvas.trigger('object:modified')
  });
}
window.ImageEditor.prototype.initializeLineDrawing = lineDrawing;
})();

// ../lib/drawingText.js

//Define action to draw text

(function () {

```

```

const textBoxDrawing = function (fabricCanvas) {
  let isDrawingText = false,
      textboxRect, origX, origY, pointer;
  fabricCanvas.on('mouse:down', (o) => {
    if (!fabricCanvas.isDrawingTextMode) return;
    isDrawingText = true;
    pointer = fabricCanvas.getPointer(o.e);
    origX = pointer.x;
    origY = pointer.y;
    textboxRect = new fabric.Rect({
      left: origX,
      top: origY,
      width: pointer.x - origX,
      height: pointer.y - origY,
      strokeWidth: 1,
      stroke: '#C00000',
      fill: 'rgba(192, 0, 0, 0.2)',
      transparentCorners: false
    });
    fabricCanvas.add(textboxRect);
  });
  fabricCanvas.on('mouse:move', (o) => {
    if (!isDrawingText) return;
    pointer = fabricCanvas.getPointer(o.e);
    if (origX > pointer.x) {
      textboxRect.set({
        left: Math.abs(pointer.x)
      });
    }
  })
}

```

```

if (origY > pointer.y) {
  textboxRect.set({
    top: Math.abs(pointer.y)
  });
}
textboxRect.set({
  width: Math.abs(origX - pointer.x)
});
textboxRect.set({
  height: Math.abs(origY - pointer.y)
});

fabricCanvas.renderAll();
});
fabricCanvas.on('mouse:up', () => {
  if (!isDrawingText) return;
  isDrawingText = false;
  // get final rect coords and replace it with textbox
  let textbox = new fabric.Textbox('Your text goes here...', {
    left: textboxRect.left,
    top: textboxRect.top,
    width: textboxRect.width < 80 ? 80 : textboxRect.width,
    fontSize: 18,
    fontFamily: "'Open Sans', sans-serif"
  });
  fabricCanvas.remove(textboxRect);
  fabricCanvas.add(textbox).setActiveObject(textbox)
  textbox.setControlsVisibility({
    'mb': false

```

```

    });
    fabricCanvas.trigger('object:modified')
  });
}
window.ImageEditor.prototype.initializeTextBoxDrawing = textBoxDrawing;
})();

// ../lib/freeDrawSettings.js

//Define action to pen draw by mouse action
(function () {
  'use strict';
  var freeDrawSettings = function () {
    let width = 1;
    let color = 'black';
    const _self = this;
    $(' ${this.containerSelector} .main-panel`).append(`<div class="toolpanel"
id="draw-panel"><div class="content"><p class="title">Free
Draw</p></div></div>`);
    // set dimension section
    $(' ${this.containerSelector} .toolpanel#draw-panel .content`).append(`
<div>
  <div class="input-container">
    <label>Brush Width</label>
    <div class="custom-number-input">
      <button class="decrease">-</button>
      <input type="number" min="1" value="1" id="input-brush-width"/>
      <button class="increase">+</button>
    </div>

```

```

</div>
<div class="input-container">
  <label>Brush Color</label>
  <input id="color-picker" value='black' />
</div>
</div>
`);
let updateBrush = () => {
  try {
    _self.canvas.freeDrawingBrush = new fabric.PencilBrush(_self.canvas)
    _self.canvas.freeDrawingBrush.width = width;
    _self.canvas.freeDrawingBrush.color = color;
  } catch ( ) {}
}
$(`${this.containerSelector} .toolpanel#draw-panel .content #input-brush-
width`).change(function () {
  try {
    width = parseInt($(this).val());
    updateBrush();
  } catch ( ) {}
})
$(`${this.containerSelector} .toolpanel#draw-panel .content #color-
picker`).spectrum({
  type: "color",
  showInput: "true",
  showInitial: "true",
  allowEmpty: "false",
});

```

```

    $(' ${this.containerSelector} .toolpanel#draw-panel .content #color-
picker').change(function () {
    try {
        color = $(this).val();
        updateBrush();
    } catch ( ) {}
    })
}
window.ImageEditor.prototype.initializeFreeDrawSettings = freeDrawSettings;
})();

```

```

// ../lib/saveInBrowser.js

```

```

//Define utils to save/load canvas status with local storage

```

```

window.saveInBrowser = {
    save: (name, value) => {
        // if item is an object, stringify
        if (value instanceof Object) {
            value = JSON.stringify(value);
        }
        localStorage.setItem(name, value);
    },
    load: (name) => {
        let value = localStorage.getItem(name);
        value = JSON.parse(value);
        return value;
    },
    remove: (name) => {
        localStorage.removeItem(name);
    }
}

```

```
}  
}
```

```
// ../lib/selectionSettings.js
```

```
//initialize selection setting panel
```

```
(function () {  
  'use strict';  
  const BorderStyleList = [{  
    value: {  
      strokeDashArray: [],  
      strokeLineCap: 'butt'  
    },  
    label: "-"  
  }, {  
    value: {  
      strokeDashArray: [1, 10],  
      strokeLineCap: 'butt'  
    },  
    label: "|||"  
  }, {  
    value: {  
      strokeDashArray: [1, 10],  
      strokeLineCap: 'round'  
    },  
    label: '...'  
  }, {  
    value: {  
      strokeDashArray: [15, 15],
```

```

    strokeLineCap: 'square'
  },
  label: '---(s)'
}, {
  value: {
    strokeDashArray: [15, 15],
    strokeLineCap: 'round'
  },
  label: '---(r)'
}, {
  value: {
    strokeDashArray: [25, 25],
    strokeLineCap: 'square'
  },
  label: '- -(s)',
}, {
  value: {
    strokeDashArray: [25, 25],
    strokeLineCap: 'round'
  },
  label: '- -(r)',
}, {
  value: {
    strokeDashArray: [1, 8, 16, 8, 1, 20],
    strokeLineCap: 'square'
  },
  label: '-. (s)',
}, {
  value: {

```



```

strokeDashArray: [1, 8, 16, 8, 1, 20],
strokeLineCap: 'round'
},
label: '-. (r)',
}]

var selectionSettings = function () {
  const _self = this;

  $('${this.containerSelector} .main-panel').append('<div class="toolpanel"
id="select-panel"><div class="content"><p class="title">Selection
Settings</p></div></div>');

  // font section

  (() => {
    $('${this.containerSelector} .toolpanel#select-panel .content').append(
      <div class="text-section">
        <h4>Style</h4>
        <div class="style">
          <button id="bold"><svg id="Capa_1" x="0px" y="0px" viewBox="-70 -70
450 450" xml:space="preserve"><path d="M218.133,144.853c20.587-14.4,35.2-
37.653,35.2-59.52C253.333,37.227,216.107,0,168,0H34.667v298.667h150.187
c44.693,0,79.147-36.267,79.147-
80.853C264,185.387,245.547,157.76,218.133,144.853z
M98.667,53.333h64c17.707,0,32,14.293,32,32 s-14.293,32-32,32h-64V53.333z
M173.333,245.333H98.667v-
64h74.667c17.707,0,32,14.293,32,32S191.04,245.333,173.333,245.333z"></
path></svg></button>

          <button id="italic"><svg id="Capa_1" x="0px" y="0px" viewBox="-70 -70
450 450" xml:space="preserve"><polygon points="106.667,0 106.667,64 153.92,64
80.747,234.667 21.333,234.667 21.333,298.667 192,298.667 192,234.667
144.747,234.667 217.92,64 277.333,64 277.333,0 "></polygon></svg></button>

```

```
<button id="underline"><svg id="Capa_1" x="0px" y="0px" viewBox="-70 -70 450 450" xml:space="preserve"><path d="M192,298.667c70.72,0,128-57.28,128-128V0h-53.333v170.667c0,41.28-33.387,74.667-74.667,74.667 s-74.667-33.387-74.667-74.667V0H64v170.667C64,241.387,121.28,298.667,192,298.667z"></path><rect x="42.667" y="341.333" width="298.667" height="42.667"></rect></svg></button>
```

```
<button id="linethrough"><svg id="Capa_1" x="0px" y="0px" viewBox="-70 -70 450 450" xml:space="preserve"><polygon points="149.333,160 234.667,160 234.667,96 341.333,96 341.333,32 42.667,32 42.667,96 149.333,96"></polygon><rect x="149.333" y="288" width="85.333" height="64"></rect><rect x="0" y="202.667" width="384" height="42.667"></rect></svg></button>
```

```
</div>
```

```
<div class="family">
```

```
<div class="input-container">
```

```
<label>Font</label>
```

```
<select id="font-family">
```

```
<option value=""></option>
```

```
<option value="'Open Sans', sans-serif">Open Sans</option>
```

```
<option value="'Oswald', sans-serif">Oswald</option>
```

```
<option value="'Playfair Display', serif">Playfair Display</option>
```

```
<option value="'Cormorant Garamond', serif">Cormorant
```

```
Garamond</option>
```

```
<option value="Impact, Charcoal, sans-serif">Impact</option>
```

```
<option value="'Lucida Console', Monaco, monospace">Lucida
```

```
Console</option>
```

```
<option value="'Comic Sans MS', 'Comic Sans', cursive, sans-serif">Comic Sans</option>
```

```
<option value="'Dancing Script', cursive">Dancing Script</option>
```

```

    <option value="'Indie Flower', cursive">Indie Flower</option>
    <option value="'Amatic SC', cursive">Amatic SC</option>
    <option value="'Permanent Marker', cursive">Permanent Marker</option>
</select>
</div>
</div>
<div class="sizes">
  <div class="input-container"><label>Font Size</label>
    <div class="custom-number-input">
      <button class="decrease">-</button>
      <input type="number" min="1" value="20" id="fontSize">
      <button class="increase">+</button>
    </div>
  </div>
  <div class="input-container"><label>Line Height</label>
    <div class="custom-number-input">
      <button class="decrease">-</button>
      <input type="number" min="0" max="3" value="1" step="0.1"
id="lineHeight">
      <button class="increase">+</button>
    </div>
  </div>
  <div class="input-container"><label>Letter Spacing</label>
    <div class="custom-number-input">
      <button class="decrease">-</button>
      <input type="number" min="0" max="2000" step="100" value="0"
id="charSpacing">
      <button class="increase">+</button>
    </div>
  </div>

```

```

</div>
</p>
</div>
<div class="align">
  <div class="input-container">
    <label>Text Alignment</label>
    <select id="text-align">
      <option value="left">Left</option>
      <option value="center">Center</option>
      <option value="right">Right</option>
      <option value="justify">Justify</option>
    </select>
  </div>
</div>
<div class="color">
  <div class="input-container">
    <label>Text Color</label>
    <input id="color-picker" value="black">
  </div>
</div>
<hr>
</div>

```

```

`);

```

```

$(`${this.containerSelector} .toolpanel#select-panel .style button`).click(function

```

```

() {

```

```

  let type = $(this).attr('id');

```

```

  switch (type) {

```

```

    case 'bold':

```

```

        setActiveFontStyle(_self.activeSelection, 'fontWeight',
getActiveFontStyle(_self.activeSelection, 'fontWeight') === 'bold' ? " : 'bold')
        break;
    case 'italic':
        setActiveFontStyle(_self.activeSelection, 'fontStyle',
getActiveFontStyle(_self.activeSelection, 'fontStyle') === 'italic' ? " : 'italic')
        break;
    case 'underline':
        setActiveFontStyle(_self.activeSelection, 'underline', !
getActiveFontStyle(_self.activeSelection, 'underline'))
        break;
    case 'linethrough':
        setActiveFontStyle(_self.activeSelection, 'linethrough', !
getActiveFontStyle(_self.activeSelection, 'linethrough'))
        break;
    default:
        break;
}
_self.canvas.renderAll(), _self.canvas.trigger('object:modified');
}))
$('`${this.containerSelector}` .toolbar#select-panel .family #font-
family`).change(function () {
    let family = $(this).val();
    setActiveFontStyle(_self.activeSelection, 'fontFamily', family)
    _self.canvas.renderAll(), _self.canvas.trigger('object:modified');
})
$('`${this.containerSelector}` .toolbar#select-panel .size
input').change(function () {
    let value = parseFloat($(this).val());

```

```

    let type = $(this).attr('id');
    setActiveFontStyle(_self.activeSelection, type, value);
    _self.canvas.renderAll(), _self.canvas.trigger('object:modified');
  })

  $('${this.containerSelector} .toolpanel#select-panel .align #text-align').change(function () {
    let mode = $(this).val();
    setActiveFontStyle(_self.activeSelection, 'textAlign', mode);
    _self.canvas.renderAll(), _self.canvas.trigger('object:modified');
  })

  $('${this.containerSelector} .toolpanel#select-panel .color #color-picker').spectrum({
    type: "color",
    showInput: "true",
    allowEmpty: "false"
  });

  $('${this.containerSelector} .toolpanel#select-panel .color #color-picker').change(function () {
    let color = $(this).val();
    setActiveFontStyle(_self.activeSelection, 'fill', color)
    _self.canvas.renderAll(), _self.canvas.trigger('object:modified');
  })
  })();

  // end font section

  // border section
  (() => {
    $('${this.containerSelector} .toolpanel#select-panel .content').append(
      <div class="border-section">

```

```

<h4>Border</h4>
<div class="input-container"><label>Width</label>
  <div class="custom-number-input">
    <button class="decrease">-</button>
    <input type="number" min="1" value="1" id="input-border-width">
    <button class="increase">+</button>
  </div>
</div>

<div class="input-container"><label>Style</label><select id="input-border-
style">${BorderStyleList.map(item => `<option value='${
JSON.stringify(item.value)}'>${item.label}</option>`)}</select></div>

  <div class="input-container"><label>Corner Type</label><select id="input-
corner-type"><option value="miter" selected>Square</option><option
value="round">Round</option></select></div>

  <div class="input-container"><label>Color</label><input id="color-picker"
value="black"></div>

  <hr>
</div>

`);
$(`${this.containerSelector} .toolpanel#select-panel .border-section #color-
picker`).spectrum({
  showButtons: false,
  type: "color",
  showInput: "true",
  allowEmpty: "false",
  move: function (color) {
    let hex = 'transparent';
    color && (hex = color.toRgbString()); // #ff0000
    _self.canvas.getActiveObjects().forEach(obj => obj.set('stroke', hex))
  }
});

```

```

        _self.canvas.renderAll(), _self.canvas.trigger('object:modified')
    }
});

$(`${this.containerSelector} .toolpanel#select-panel .border-section #input-
border-width`).change(function () {
    let width = parseInt($(this).val());
    _self.canvas.getActiveObjects().forEach(obj => obj.set({
        strokeUniform: true,
        strokeWidth: width
    })))
    _self.canvas.renderAll(), _self.canvas.trigger('object:modified')
})

$(`${this.containerSelector} .toolpanel#select-panel .border-section #input-
border-style`).change(function () {
    try {
        let style = JSON.parse($(this).val());
        _self.canvas.getActiveObjects().forEach(obj => obj.set({
            strokeUniform: true,
            strokeDashArray: style.strokeDashArray,
            strokeLineCap: style.strokeLineCap
        })))
        _self.canvas.renderAll(), _self.canvas.trigger('object:modified')
    } catch (_) {}
})

$(`${this.containerSelector} .toolpanel#select-panel .border-section #input-
corner-type`).change(function () {
    let corner = $(this).val();
    _self.canvas.getActiveObjects().forEach(obj => obj.set('strokeLineJoin', corner))
    _self.canvas.renderAll(), _self.canvas.trigger('object:modified')
})

```



```

    })
  })();
// end border section

// fill color section
(() => {
  $('` ${this.containerSelector} .toolpanel#select-panel .content`').append(`
    <div class="fill-section">
      <div class="tab-container">
        <div class="tabs">
          <div class="tab-label" data-value="color-fill">Color Fill</div>
          <div class="tab-label" data-value="gradient-fill">Gradient Fill</div>
        </div>
        <div class="tab-content" data-value="color-fill">
          <input id="color-picker" value='black' /><br>
        </div>
        <div class="tab-content" data-value="gradient-fill">
          <div id="gradient-picker"></div>
          <div class="gradient-orientation-container">
            <div class="input-container">
              <label>Orientation</label>
              <select id="select-orientation">
                <option value="linear">Linear</option>
                <option value="radial">Radial</option>
              </select>
            </div>
            <div id="angle-input-container" class="input-container">
              <label>Angle</label>
              <div class="custom-number-input">

```

```

        <button class="decrease">-</button>

        <input type="number" min="0" max="360" value="0" id="input-angle">

        <button class="increase">+</button>

    </div>

</div>

</div>

</div>

</div>

</div>

</div>

');
$(`${this.containerSelector} .toolpanel#select-panel .content .tab-
label`).click(function () {
    `${_self.containerSelector} .toolpanel#select-panel .content .tab-
label`).removeClass('active');
    $(this).addClass('active');
    let target = $(this).data('value');
    $(this).closest('.tab-container').find('.tab-content').hide();
    $(this).closest('.tab-container').find(`.tab-content[data-value=$
{target}]`).show();
    if (target === 'color-fill') {
        let color = `${_self.containerSelector} .toolpanel#select-panel .fill-section
#color-picker`).val();
        try {
            _self.canvas.getActiveObjects().forEach(obj => obj.set('fill', color))
            _self.canvas.renderAll(), _self.canvas.trigger('object:modified')
        } catch (_) {
            console.log("can't update background color")
        }
    } else {

```

```

        updateGradientFill();
    }
})
$(`$_self.containerSelector} .toolpanel#select-panel .content .tab-label[data-
value=color-fill]`).click();
$(`$_self.containerSelector} .toolpanel#select-panel .fill-section #color-
picker`).spectrum({
    flat: true,
    showPalette: false,
    showButtons: false,
    type: "color",
    showInput: "true",
    allowEmpty: "false",
    move: function (color) {
        let hex = 'transparent';
        color && (hex = color.toRgbString()); // #ff0000
        _self.canvas.getActiveObjects().forEach(obj => obj.set('fill', hex))
        _self.canvas.renderAll(), _self.canvas.trigger('object:modified')
    }
});
const gp = new Grapick({
    el: `$_self.containerSelector} .toolpanel#select-panel .fill-section #gradient-
picker`,
    colorEl: '<input id="colorpicker"/>'
});
gp.setColorPicker(handler => {
    const el = handler.getEl().querySelector('#colorpicker');
    $(el).spectrum({
        showPalette: false,

```

```

showButtons: false,
type: "color",
color: handler.getColor(),
showAlpha: true,
change(color) {
  handler.setColor(color.toRgbString());
},
move(color) {
  handler.setColor(color.toRgbString(), 0);
}
});
gp.addHandler(0, 'red');
gp.addHandler(100, 'blue');
const updateGradientFill = () => {
  let stops = gp.getHandlers();
  let orientation = $('${this.containerSelector} .toolpanel#select-
panel .content .gradient-orientation-container #select-orientation').val();
  let angle = parseInt($('${this.containerSelector} .toolpanel#select-panel .content
.gradient-orientation-container #input-angle').val());

  let gradient = generateFabricGradientFromColorStops(stops,
_self.activeSelection.width, _self.activeSelection.height, orientation, angle);
  _self.activeSelection.set('fill', gradient);
  _self.canvas.renderAll()
}
gp.on('change', complete => {
  updateGradientFill();
})

```

```

    $('${this.containerSelector} .toolpanel#select-panel .content .gradient-orientation-container #select-orientation`).change(function () {
        let type = $(this).val();
        console.log('orientation', type)
        if (type === 'radial') {
            $(this).closest('.gradient-orientation-container').find('#angle-input-container').hide();
        } else {
            $(this).closest('.gradient-orientation-container').find('#angle-input-container').show();
        }
        updateGradientFill();
    })

    $('${this.containerSelector} .toolpanel#select-panel .content .gradient-orientation-container #input-angle`).change(function () {
        updateGradientFill();
    })
    })());

// end fill color section

// object options section
(() => {
    $('${this.containerSelector} .toolpanel#select-panel .content').append(
        <div class="object-options">
            <h4>Object Options</h4>
            <button id="flip-h"><svg width="512" height="512" enable-background="new
0 0 16 16" viewBox="0 0 16 20" xml:space="preserve"><g transform="matrix(0
1.5365 1.5385 0 -5.0769 1.5495)"><rect x="5" y="8" width="1"
height="1"></rect><rect x="7" y="8" width="1" height="1"></rect><rect x="9"

```

```
y="8" width="1" height="1"></rect><rect x="1" y="8" width="1"
height="1"></rect><rect x="3" y="8" width="1" height="1"></rect><path d="M 1,2
5.5,6 10,2 Z M 7.37,3 5.5,4.662 3.63,3 Z"></path><polygon points="10 15 5.5 11 1
15"></polygon></g></svg></button>
```

```
<button id="flip-v"><svg width="512" height="512" enable-background="new
0 0 16 16" viewBox="0 0 16 20" xml:space="preserve"><g
transform="matrix(1.5365 0 0 1.5385 -.45052 -3.0769)"><rect x="5" y="8"
width="1" height="1"></rect><rect x="7" y="8" width="1" height="1"></rect><rect
x="9" y="8" width="1" height="1"></rect><rect x="1" y="8" width="1"
height="1"></rect><rect x="3" y="8" width="1" height="1"></rect><path d="M 1,2
5.5,6 10,2 Z M 7.37,3 5.5,4.662 3.63,3 Z"></path><polygon points="5.5 11 1 15 10
15"></polygon></g></svg></button>
```

```
<button id="duplicate"><svg id="Capa_1" x="0px" y="0px" viewBox="0 0
512 512" xml:space="preserve"><g><g><g><path d="M42.667,256c0-
59.52,35.093-110.827,85.547-
134.827V75.2C53.653,101.44,0,172.48,0,256s53.653,154.56,128.213,180.8 v-
45.973C77.76,366.827,42.667,315.52,42.667,256z"></path><path d="M320,64c-
105.92,0-192,86.08-192,192s86.08,192,192,192s192-86.08,192-
192S425.92,64,320,64z M320,405.333 c-82.347,0-149.333-66.987-149.333-
149.333S237.653,106.667,320,106.667S469.333,173.653,469.333,256
S402.347,405.333,320,405.333z"></path><polygon points="341.333,170.667
298.667,170.667 298.667,234.667 234.667,234.667 234.667,277.333
298.667,277.333 298.667,341.333 341.333,341.333 341.333,277.333
405.333,277.333 405.333,234.667 341.333,234.667
"></polygon></g></g></g></svg></button>
```

```
<button id="delete"><svg id="Layer_1" x="0px" y="0px" viewBox="0 0 512
512" xml:space="preserve"><g><g><path d="M425.298,51.358h-91.455V16.696c0-
9.22-7.475-16.696-16.696-16.696H194.855c-9.22,0-16.696,7.475-
16.696,16.696v34.662 H86.704c-9.22,0-16.696,7.475-
```

16.696,16.696v51.357c0,9.22,7.475,16.696,16.696,16.696h5.072l15.26,359.906
c0.378,8.937,7.735,15.988,16.68,15.988h264.568c8.946,0,16.302-7.051,16.68-
15.989l15.259-359.906h5.073 c9.22,0,16.696-7.475,16.696-
16.696V68.054C441.994,58.832,434.519,51.358,425.298,51.358z
M211.551,33.391h88.9v17.967h-88.9 V33.391z M372.283,478.609H139.719l-
14.522-342.502h261.606L372.283,478.609z M408.602,102.715c-15.17,0-296.114,0-
305.202,0 V84.749h305.202V102.715z"></path></g></g><g><g><path
d="M188.835,187.304c-9.22,0-16.696,7.475-
16.696,16.696v206.714c0,9.22,7.475,16.696,16.696,16.696 c9.22,0,16.696-
7.475,16.696-16.696V204C205.53,194.779,198.055,187.304,188.835,187.304z"></
path></g></g><g><g><path d="M255.998,187.304c-9.22,0-16.696,7.475-
16.696,16.696v206.714c0,9.22,7.474,16.696,16.696,16.696 c9.22,0,16.696-
7.475,16.696-
16.696V204C272.693,194.779,265.218,187.304,255.998,187.304z"></path></g></
g><g><g><g><path d="M323.161,187.304c-9.22,0-16.696,7.475-
16.696,16.696v206.714c0,9.22,7.475,16.696,16.696,16.696 s16.696-7.475,16.696-
16.696V204C339.857,194.779,332.382,187.304,323.161,187.304z"></path></g></
g></svg></button>

<hr>

</div>

`);

\$(`\$`{this.containerSelector} .toolbar#select-panel .object-options #flip-
h`).click(() => {

 this.activeSelection.set('flipX', !this.activeSelection.flipX);

 this.canvas.renderAll(), this.canvas.trigger('object:modified');

})

\$(`\$`{this.containerSelector} .toolbar#select-panel .object-options #flip-
v`).click(() => {

 this.activeSelection.set('flipY', !this.activeSelection.flipY);

```

    this.canvas.renderAll(), this.canvas.trigger('object:modified');
  })
  $('${this.containerSelector} .toolbar#select-panel .object-options
#duplicate').click() => {
    let clonedObjects = []
    let activeObjects = this.canvas.getActiveObjects()
    activeObjects.forEach(obj => {
      obj.clone(clone => {
        this.canvas.add(clone.set({
          strokeUniform: true,
          left: obj.aCoords.tl.x + 20,
          top: obj.aCoords.tl.y + 20
        }));

        if (activeObjects.length === 1) {
          this.canvas.setActiveObject(clone)
        }
        clonedObjects.push(clone)
      })
    })
    if (clonedObjects.length > 1) {
      let sel = new fabric.ActiveSelection(clonedObjects, {
        canvas: this.canvas,
      });
      this.canvas.setActiveObject(sel)
    }
    this.canvas.requestRenderAll(), this.canvas.trigger('object:modified')
  })

```



```

    $('${this.containerSelector} .toolpanel#select-panel .object-options
#delete`).click() => {
        this.canvas.getActiveObjects().forEach(obj => this.canvas.remove(obj))
        this.canvas.discardActiveObject().requestRenderAll(),
this.canvas.trigger('object:modified');
    })
    })();
    // end object options section

    // effect section
    (() => {
        $('${this.containerSelector} .toolpanel#select-panel .content').append(
            <div class="effect-section">
                <h4>Effect</h4>
                <div class="input-container"><label>Opacity</label><input id="opacity"
type="range" min="0" max="1" value="1" step="0.01"></div>
                <div class="input-container"><label>Blur</label><input class="effect"
id="blur" type="range" min="0" max="100" value="50"></div>
                <div class="input-container"><label>Brightness</label><input class="effect"
id="brightness" type="range" min="0" max="100" value="50"></div>
                <div class="input-container"><label>Saturation</label><input class="effect"
id="saturation" type="range" min="0" max="100" value="50"></div>
                <h5>Gamma</h5>
                <div class="input-container"><label>Red</label><input class="effect"
id="gamma.r" type="range" min="0" max="100" value="50"></div>
                <div class="input-container"><label>Green</label><input class="effect"
id="gamma.g" type="range" min="0" max="100" value="50"></div>
                <div class="input-container"><label>Blue</label><input class="effect"
id="gamma.b" type="range" min="0" max="100" value="50"></div>

```

```

    <hr>
  </div>

  `);
  $('`{$this.containerSelector} .toolpanel#select-panel .effect-section
#opacity`).change(function () {
    let opacity = parseFloat($(this).val());
    _self.activeSelection.set('opacity', opacity)
    _self.canvas.renderAll(), _self.canvas.trigger('object:modified')
  })
  $('`{$this.containerSelector} .toolpanel#select-panel .effect-
section .effect`).change(function () {
    let effect = $(this).attr('id');
    let value = parseFloat($(this).val());
    let currentEffect = getCurrentEffect(_self.activeSelection);
    _self.activeSelection.filters = getUpdatedFilter(currentEffect, effect, value);
    _self.activeSelection.applyFilters();
    _self.canvas.renderAll(), _self.canvas.trigger('object:modified')
  })
  })0;
  // end effect section
}
window.ImageEditor.prototype.initializeSelectionSettings = selectionSettings;
})0

// ../lib/shapes.js

//Define action to add shape to canvas
(function () {
  'use strict';

```

```

const defaultShapes = [
  '<svg viewBox="-10 -10 120 120"><polygon points="0 0, 0 100, 100 100, 100 0"
stroke-width="5" stroke="#000" fill="none"></polygon></svg>',
  '<svg viewBox="-8 -8 120 120"><polygon fill="none" stroke-width="5"
stroke="black" points="50 0, 85 50, 50 100, 15 50"></polygon></svg>',
  '<svg viewBox="-10 -10 120 120"><polygon points="25 0, 0 100, 75 100, 100 0"
stroke-width="5" stroke="#000" fill="none"></polygon></svg>',
  '<svg viewBox="-8 -8 120 120"><polygon points="0,100 30,10 70,10 100,100"
stroke-width="5" stroke="#000" fill="none"></polygon></svg>',
  '<svg viewBox="-10 -10 120 120"><path d="M 80,80 V 20 H 20 v 60 z m 20,20
V 0 H 0 v 100 z" stroke-width="5" stroke="#000" fill-rule="evenodd"
fill="none"></path></svg>',
  '<svg viewBox="0 0 100 100"><polygon points="26,86 11.2,40.4 50,12.2
88.8,40.4 74,86 " stroke="#000" stroke-width="5" fill="none"></polygon></svg>',
  '<svg viewBox="0 0 100 100"><polygon points="30.1,84.5 10.2,50 30.1,15.5
69.9,15.5 89.8,50 69.9,84.5" stroke-width="5" stroke="#000"
fill="none"></polygon></svg>',
  '<svg viewBox="0 0 100 100"><polygon points="34.2,87.4 12.3,65.5 12.3,34.5
34.2,12.6 65.2,12.6 87.1,34.5 87.1,65.5 65.2,87.4" stroke-width="5" stroke="#000"
fill="none"></polygon></svg>',
  '<svg viewBox="0 0 100 100"><polygon points="11.2,70 11.2,40 50,12.2
88.8,40 88.8,70" stroke="#000" stroke-width="5" fill="none"></polygon></svg>',
  '<svg viewBox="0 0 100 100"><polygon points="10.2,70 10.2,35 30.1,15
69.9,15 89.8,35 89.8,70" stroke-width="5" stroke="#000"
fill="none"></polygon></svg>',
  '<svg viewBox="-10 -10 120 120"><polygon points="50 15, 100 100, 0 100"
stroke-width="5" stroke="#000" fill="none"></polygon></svg>',
  '<svg viewBox="-10 -10 120 120"><polygon points="0 0, 100 100, 0 100"
stroke-width="5" stroke="#000" fill="none"></polygon></svg>',

```

```

`<svg viewBox="-10 -10 120 120"><path d="M 26,85 50,45 74,85 Z m -26,15
50,-85 50,85 z" stroke-width="5" stroke="#000" fill="none"></path></svg>`,
`<svg viewBox="8 50 100 100"><path d="M 62.68234,131.5107 H 26.75771 V
96.075507 Z M 11.572401,146.76255 V 59.66782 l 87.983665,87.09473 z" stroke-
width="5" stroke="#000" fill="none" fill-rule="evenodd"></path></svg>`,
`<svg viewBox="-2 -2 100 100"><circle cx="50" cy="50" r="40" stroke="#000"
stroke-width="5" fill="none"></circle></svg>`,
`<svg x="0px" y="0px" viewBox="0 0 96 120" xml:space="preserve"><path
stroke="#000" stroke-width="5" fill="none"
d="M9.113,65.022C11.683,45.575,28.302,30.978,48,30.978c19.696,0,36.316,14.598,
38.887,34.045H9.113z"></path></svg>`,
`<svg viewBox="-15 -15 152 136"><path stroke="#000000" stroke-width="5"
d="m0 0l57.952755 0l0 0c32.006428 -1.4055393E-14 57.952755 23.203636
57.952755 51.82677c0 28.623135 -25.946327 51.82677 -57.952755 51.82677l-
57.952755 0z" fill="none"></path></svg>`,
`<svg viewBox="-5 -50 140 140"><path stroke="#000000" stroke-width="5"
d="m20.013628 0l84.37401 0l0 0c11.053215 -1.04756605E-14 20.013626 9.282301
20.013626 20.7326c0 11.450296 -8.960411 20.7326 -20.013626 20.7326l-84.37401
0l0 0c-11.053222 0 -20.013628 -9.282303 -20.013628 -20.7326c-5.2380687E-15 -
11.450298 8.960406 -20.7326 20.013628 -20.7326z" fill="none"></path></svg>`,
`<svg viewBox="-8 -8 136 136"><path stroke="#000000" stroke-width="5"
d="m0 51.82677l0 0c0 -28.623135 23.203636 -51.82677 51.82677 -51.82677l0
0c13.745312 0 26.927654 5.4603047 36.64706 15.17971c9.719406 9.719404
15.17971 22.901749 15.17971 36.64706l0 0c0 28.623135 -23.203636 51.82677 -
51.82677 51.82677l0 0c-28.623135 0 -51.82677 -23.203636 -51.82677 -
51.82677zm25.913385 0l0 0c0 14.311565 11.60182 25.913387 25.913385
25.913387c14.311565 0 25.913387 -11.601822 25.913387 -25.913387c0 -14.311565
-11.601822 -25.913385 -25.913387 -25.913387c-14.311565 0 -25.913385
11.60182 -25.913385 25.913385z" fill="none"></path></svg>`,

```

```

`<svg viewBox="-7 -35 133 105"><path stroke="#000000" stroke-width="5"
d="m0 57.952755l0 0c0 -32.006424 25.946333 -57.952755 57.952755 -
57.952755c32.006428 0 57.952755 25.946333 57.952755 57.952755l-28.97638 0c0 -
16.003212 -12.97316 -28.976377 -28.976376 -28.976377c-16.003212 0 -28.976377
12.9731655 -28.976377 28.976377z" fill="none"></path></svg>`
]

var shapes = function () {
  const _self = this;
  let ShapeList = defaultShapes;
  if (Array.isArray(this.shapes) && this.shapes.length) ShapeList = this.shapes;
  $('`${this.containerSelector} .main-panel`).append(`<div class="toolpanel"
id="shapes-panel"><div class="content"><p
class="title">Shapes</p></div></div>`);
  ShapeList.forEach(svg => {
    $('`${this.containerSelector} .toolpanel#shapes-panel .content`).append(`<div
class="button">${svg}</div>`)
  })
  $('`${this.containerSelector} .toolpanel#shapes-
panel .content .button`).click(function () {
    let svg = $(this).html();
    try {
      fabric.loadSVGFromString(
        svg,
        (objects, options) => {
          var obj = fabric.util.groupSVGElements(objects, options)
          obj.strokeUniform = true
          obj.strokeLineJoin = 'miter'
          obj.scaleToWidth(100)
          obj.scaleToHeight(100)
        }
      )
    } catch (e) {
      console.error(e)
    }
  })
}

```

```

        obj.set({
            left: 0,
            top: 0
        })
        _self.canvas.add(obj).renderAll()
        _self.canvas.trigger('object:modified')
    }
)
} catch (e) {
    console.error("can't add shape");
}
})
}
window.ImageEditor.prototype.initializeShapes = shapes;
})();

```

```
// ../lib/style.css
```

```

body {
    margin: 0;
    font-family: "Open Sans", sans-serif;
}

```

```

.grp-handler-cp-c {
    margin-left: -20px;
}

```

```

.default-container {
    width: 100%;
}

```

```
height: 100%;  
/* display: flex; */  
}
```

```
.toolbar {  
  line-height: 0;  
  background-color: #91c5f0;  
  box-shadow: 0 0 3px 0 rgba(50, 50, 50, .25);  
  display: flex;  
  justify-content: space-between;  
  align-items: center;  
}
```

```
.toolbar button {  
  width: 64px;  
  height: 54px;  
  opacity: .55;  
  clear: both;  
  border: 0;  
  border-radius: unset;  
  outline: none;  
}
```

```
.toolbar button.active,  
.toolbar button:hover {  
  opacity: 1;  
  border-left: 1px solid #b0bcee;  
  border-right: 1px solid #b0bcee;  
  box-shadow: inset 5px 0 10px 0 rgba(50, 50, 50, .1);  
}
```

```
}
```

```
.toolbar button img,  
.toolbar button svg {  
  width: 22px;  
  height: 22px;  
}
```

```
.main-panel {  
  height: calc(100% - 54px);  
  display: flex;  
  position: relative;  
}
```

```
.canvas-holder {  
  margin: auto;  
  width: 100%;  
  height: 100%;  
  overflow: auto;  
  display: flex;  
  align-items: center;  
  justify-content: center;  
  background: #e4eff5;  
}
```

```
.canvas-container {  
  background-image:  
url("data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAABAAAAAAQCAy
```


AAAAf8/9hAAAAHUlEQVQ4jWNgYGAQIYAJglEDhoUBg9+FowbQ2gAARjwK
ARjtnN8AAAAASUVORK5CYII=");

```
background-size: 30px 30px;
border: 1px solid #ccc;
margin: auto;
}
```

```
.toolpanel {
background-color: #9bcaf7;
width: 300px;
top: 0;
left: 0;
height: 100%;
border: 1px solid #2c1efa;
transition: all .4s;
box-sizing: border-box;
text-align: left;
font-size: 13px;
color: #5a6a6d;
display: none;
position: absolute;
z-index: 9999;
}
```

```
.toolpanel.closed {
left: -300px;
}
```

```
.toolpanel.visible {
```

```
display: initial;
}
```

```
.toolpanel .content {
padding: 20px;
position: relative;
height: -webkit-fill-available;
height: -ms-fill-available;
height: fill-available;
height: -moz-fill-available;
}
```

```
.toolpanel .title {
font-size: 14px;
font-weight: 700;
margin: 0;
padding-bottom: 10px;
width: 100%;
border-bottom: 1px solid #04f5fd;
color: #333;
text-transform: uppercase;
}
```

```
.toolpanel .content .hide-show-handler {
position: absolute;
top: calc(5% - 20px);
right: -22px;
width: 20px;
height: 80px;
```

```
background: #7bc7fa;
border: 1px solid #1c7bf8;
border-top-right-radius: 3px;
border-bottom-right-radius: 3px;
cursor: pointer;
```

```
background-size: 10px;
background-repeat: no-repeat;
background-position: center center;
}
```

```
.toolpanel.closed .content .hide-show-handler {
  background: #7bc7fa;
  border: 1px solid #1c7bf8;
}
```

```
.spectrum.with-add-on {
  width: 40px;
}
```

```
#shapes-panel .button {
  cursor: pointer;
  line-height: 0;
  overflow: hidden;
  padding: 0;
  width: 32px;
  height: 32px;
  display: inline-block;
  margin: 9px;
```

```
}
```

```
#background-panel .canvas-size-setting input {  
  width: 60px;  
  background-color: #a0e6fc;  
  border-radius: 6px;  
  border: 2px solid #a4d6ff;  
  padding: 4px 10px;  
  line-height: 18px;  
  font-size: 13px;  
}
```

```
#select-panel .text-section .style button,  
#select-panel .object-options button {  
  padding: 0;  
  width: 32px;  
  height: 32px;  
  background-color: #c2f3fa;  
  border: 1px solid #a3ceff;  
  text-align: center;  
  outline: none;  
}
```

```
#select-panel button svg {  
  opacity: .7;  
  width: 18px;  
  height: 18px;  
  vertical-align: middle;  
}
```

```
#select-panel .text-section .style,  
#select-panel .text-section .family,  
#select-panel .text-section .sizes,  
#select-panel .text-section .align,  
#select-panel .text-section .color {  
    margin-bottom: 20px;  
}
```

```
#select-panel .text-section .sizes input {  
    width: 50px;  
}
```

```
.toolpanel#select-panel .text-section,  
.toolpanel#select-panel .effect-section {  
    display: none;  
}
```

```
.toolpanel#select-panel.type-group .border-section {  
    display: none;  
}
```

```
.toolpanel#select-panel.type-group .fill-section {  
    display: none;  
}
```

```
.toolpanel#select-panel.type-textbox .text-section {  
    display: block;  
}
```

```
.toolpanel#select-panel.type-textbox .fill-section {  
    display: none;  
}
```

```
.toolpanel#select-panel.type-image .effect-section {  
    display: block;  
}
```

```
.toolpanel#select-panel.type-image .fill-section {  
    display: none;  
}
```

```
.custom-modal-container {  
    position: absolute;  
    width: 100%;  
    height: 100%;  
    top: 0;  
    left: 0;  
    background: #333333;  
    display: flex;  
    justify-content: center;  
    align-items: center;  
}
```

```
.custom-modal-content {  
    background: rgb(170, 227, 241);  
    width: max-content;  
    padding: 20px;
```

```
}
```

```
.custom-modal-content .button-download {  
  border: 1px solid #b1eff8;  
  padding: 10px;  
  cursor: pointer;  
  margin: 5px;  
  border-radius: 3px;  
}
```

```
.custom-modal-content .button-download:hover {  
  background: #bed7f8;  
  transition: 0.3s;  
}
```

```
.toolpanel .input-container {  
  display: flex;  
  align-items: center;  
  padding-top: 5px;  
  padding-bottom: 5px;  
}
```

```
.toolpanel .input-container label {  
  width: 50%;  
}
```

```
.toolpanel .input-container select {  
  width: 50%;  
  height: 29px;
```

```
background: #bce1f1;
border: 1px solid #52c6ec;
border-radius: 5px;
outline: none;
}
```

```
.toolpanel .input-container .sp-replacer {
width: 50%;
}
```

```
.toolpanel .input-container .custom-number-input {
background: #cdf9ff;
display: flex;
align-items: center;
padding: 1px;
height: 30px;
background-color: #e4fbff;
border-radius: 6px;
text-align: center;
}
```

```
.toolpanel .input-container .custom-number-input button {
width: 36px !important;
height: 30px !important;
background-color: #a2bfff;
background-clip: padding-box;
border-radius: 6px;
color: #333;
border: 1px solid transparent;
```



```
font-size: 16px;
cursor: pointer;
outline: none;
}
```

```
.toolpanel .input-container .custom-number-input input {
height: 30px !important;
width: 60px !important;
background: transparent !important;
border: none;
outline: none;
text-align: center;
}
```

```
.toolpanel .input-container .custom-number-input input::-webkit-outer-spin-button,
.toolpanel .input-container .custom-number-input input::-webkit-inner-spin-button {
-webkit-appearance: none;
margin: 0;
}
```

```
.tab-container .tabs {
padding-top: 20px;
padding-bottom: 20px;
display: flex;
justify-content: space-between;
}
```

```
.tab-container .tabs .tab-label {
font-size: 16px;
```

```
    cursor: pointer;
}
```

```
.tab-container .tabs .tab-label.active {
    color: black
}
```

```
.gradient-orientation-container {
    padding-top: 40px;
}
```

```
.drag-drop-input {
    background-color: #c1e3f0;
    width: 100%;
    box-sizing: border-box;
    border: 2px dashed #2aa5f7;
    border-radius: 6px;
    text-align: center;
    padding: 120px;
}
```

```
.drag-drop-input.dragging {
    border-color: #4368a9;
}
```

```
// ../lib/toolbar.js
```

```
//Initialize toolbar
```

```
(function () {
```

```

'use strict';
var defaultButtons = [{
  name: 'select',
  title: 'Select/move object (V)',
  icon: `69
```

```

64.152344 512 50.628906 C 512 37.105469 506.734375 24.390625 497.171875
14.828125 C 487.609375 5.265625 474.894531 0 461.371094 0 C 447.847656 0
435.132812 5.265625 425.570312 14.828125 L 198.296875 242.105469 L
269.894531 313.703125 Z M 497.171875 86.429688 " style="stroke: none; fill-rule:
nonzero; fill: rgb(0, 0, 0); fill-opacity: 1;"></path><path d="M 65.839844 506.65625
C 92.171875 507.21875 130.371094 496.695312 162.925781 459.074219 C
164.984375 456.691406 166.894531 454.285156 168.664062 451.855469 C
179.460938 435.875 184.695312 418.210938 183.855469 400.152344 C 182.945312
380.5625 174.992188 362.324219 161.460938 348.796875 C 150.28125 337.613281
134.722656 331.457031 117.648438 331.457031 C 95.800781 331.457031
73.429688 341.296875 56.277344 358.449219 C 31.574219 383.152344 31.789062
404.234375 31.976562 422.839844 C 32.15625 440.921875 32.316406 456.539062
11.101562 480.644531 L 0 493.257812 C 0 493.257812 26.828125 505.820312
65.839844 506.65625 Z M 65.839844 506.65625 " style="stroke: none; fill-rule:
nonzero; fill: rgb(0, 0, 0); fill-opacity: 1;"></path><path d="M 209.980469
373.621094 L 248.496094 335.101562 L 176.894531 263.503906 L 137.238281
303.160156 C 154.691406 306.710938 170.464844 315 182.859375 327.394531 C
195.746094 340.285156 205.003906 356.1875 209.980469 373.621094 Z M
209.980469 373.621094 " style="stroke: none; fill-rule: nonzero; fill: rgb(0, 0, 0);
fill-opacity: 1;"></path></g></svg>`

```

```

}, {

```

```

    name: 'line',

```

```

    title: 'Line',

```

```

    icon: `<svg id="Capa_1" x="0px" y="0px" viewBox="0 0 512 512"

```

```

xml:space="preserve"><path

```

```

d="M349.091,0v124.516L124.516,349.091H0V512h162.909V387.484l224.574-

```

```

224.574H512V0H349.091z M54.303,457.696v-54.303 h54.303v54.303H54.303z

```

```

M457.696,108.605h-54.303V54.303h54.303V108.605z"></path></svg>`

```

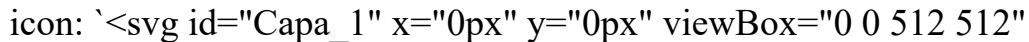
```

}, {

```

name: 'textbox',

title: 'Text box',

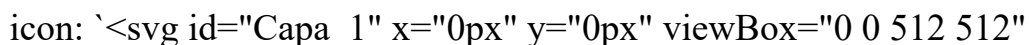
icon: ``

```
xml:space="preserve"><g><g><path d="M497,90c8.291,0,15-6.709,15-15V15c0-8.291-6.709-15-15-15h-60c-8.291,0-15,6.709-15,15v15H90V15c0-8.401-6.599-15-15-15 H15C6.599,0,0,6.599,0,15v60c0,8.399,6.599,15,15,15h15v332H15c-8.291,0-15,6.709-15,15v60c0,8.291,6.709,15,15,15h60 c8.291,0,15-6.709,15-15v-15h332v15c0,8.399,6.599,15,15,15h60c8.401,0,15-6.601,15-15v-60c0-8.401-6.599-15-15-15h-15V90H497z M452,422h-15c-8.401,0-15,6.599-15,15v15H90v-15c0-8.291-6.709-15-15-15H60V90h15c8.401,0,15-6.601,15-15V60h332v15c0,8.291,6.709,15,15,15h15V422z"></path></g></g><g><g><path d="M361,105H151c-8.291,0-15,6.709-15,15v60c0,6.064,3.647,11.543,9.258,13.857c5.625,2.329,12.056,1.04,16.348-3.252 L187.211,165H226v176.459l-27.48,42.221c-3.062,4.6-3.354,10.518-0.747,15.396S205.463,407,211,407h90 c5.537,0,10.62-3.047,13.228-7.925c2.608-4.878,2.314-10.796-0.747-15.396L286,341.459V165h38.789l25.605,25.605 c4.307,4.307,10.781,5.596,16.348,3.252c5.61-2.314,9.258-7.793,9.258-13.857v-60C376,111.709,369.291,105,361,105z"></path></g></g></svg>`
```

}, {

name: 'upload',

title: 'Upload image',

icon: ``

```
xml:space="preserve"><path d="M412.907,214.08C398.4,140.693,333.653,85.333,256,85.333c-61.653,0-115.093,34.987-141.867,86.08 C50.027,178.347,0,232.64,0,298.667c0,70.72,57.28,128,128,128h277.333C464.213,426.667,512,378.88,512,320 C512,263.68,468.16,218.027,412.907,214.08z M298.667,277.333v85.333h-85.333v-85.333h-64L256,170.667l106.667,106.667H298.667z"></path></svg>`
```

}, {

name: 'background',

title: 'Canvas option',

icon: `d="m499.953125 197.703125-39.351563-8.554687c-3.421874-10.476563-7.660156-
20.695313-12.664062-30.539063l21.785156-33.886719c3.890625-6.054687
3.035156-14.003906-2.050781-19.089844l-61.304687-61.304687c-5.085938-
5.085937-13.035157-5.941406-19.089844-2.050781l-33.886719 21.785156c-
9.84375-5.003906-20.0625-9.242188-30.539063-12.664062l-8.554687-39.351563c-
1.527344-7.03125-7.753906-12.046875-14.949219-12.046875h-86.695312c-
7.195313 0-13.421875 5.015625-14.949219 12.046875l-8.554687 39.351563c-
10.476563 3.421874-20.695313 7.660156-30.539063 12.664062l-33.886719-
21.785156c-6.054687-3.890625-14.003906-3.035156-19.089844 2.050781l-
61.304687 61.304687c-5.085937 5.085938-5.941406 13.035157-2.050781
19.089844l21.785156 33.886719c-5.003906 9.84375-9.242188 20.0625-12.664062
30.539063l-39.351563 8.554687c-7.03125 1.53125-12.046875 7.753906-12.046875
14.949219v86.695312c0 7.195313 5.015625 13.417969 12.046875
14.949219l39.351563 8.554687c3.421874 10.476563 7.660156 20.695313
12.664062 30.539063l-21.785156 33.886719c-3.890625 6.054687-3.035156
14.003906 2.050781 19.089844l61.304687 61.304687c5.085938 5.085937
13.035157 5.941406 19.089844 2.050781l33.886719-21.785156c9.84375 5.003906
20.0625 9.242188 30.539063 12.664062l8.554687 39.351563c1.527344 7.03125
7.753906 12.046875 14.949219 12.046875h86.695312c7.195313 0 13.421875-
5.015625 14.949219-12.046875l8.554687-39.351563c10.476563-3.421874
20.695313-7.660156 30.539063-12.664062l33.886719 21.785156c6.054687
3.890625 14.003906 3.039063 19.089844-2.050781l61.304687-
61.304687c5.085937-5.085938 5.941406-13.035157 2.050781-19.089844l-
21.785156-33.886719c5.003906-9.84375 9.242188-20.0625 12.664062-
30.539063l39.351563-8.554687c7.03125-1.53125 12.046875-7.753906 12.046875-

```

14.949219v-86.695312c0-7.195313-5.015625-13.417969-12.046875-14.949219zm-
152.160156 58.296875c0 50.613281-41.179688 91.792969-91.792969 91.792969s-
91.792969-41.179688-91.792969-91.792969 41.179688-91.792969 91.792969-
91.792969 91.792969 41.179688 91.792969 91.792969zm0 0"></path></svg>`
    ]]
    const defaultExtendedButtons = [{
      name: 'undo',
      title: 'Undo',
      icon: `73
```

```

c41.856,0,80,30.08,84.192,71.712c4.832,47.872-32.704,88.288-
79.584,88.288H208c-8.832,0-16,7.168-16,16v64
c0,8.832,7.168,16,16,16h128C438.816,480.015,521.472,391.151,511.136,286.255z"
></path></svg>`
  }, {
    name: 'save',
    title: 'Save',
    icon: `

```



```

    name: 'clear',
    title: 'Clear',
    icon: `

```

```

    this.containerEl.append('<div class="toolbar" id="toolbar"><div class="main-
buttons"></div><div class="extended-buttons"></div></div>');

    // main buttons
    () => {
        buttons.forEach(item => {
            $(' ${this.containerSelector} #toolbar .main-buttons').append('<button id="$
{item.name}">${item.icon}</button>');
        })
        $(' ${this.containerSelector} #toolbar .main-buttons button').click(function () {
            let id = $(this).attr('id');
            $(' ${_self.containerSelector} #toolbar button').removeClass('active');
            $(' ${_self.containerSelector} #toolbar button#${id}').addClass('active');
            _self.setActiveTool(id);
        })
    })();

    // extended buttons
    () => {
        extendedButtons.forEach(item => {
            $(' ${this.containerSelector} #toolbar .extended-buttons').append('<button
id="${item.name}">${item.icon}</button>');
        })
        $(' ${this.containerSelector} #toolbar .extended-buttons button').click(function
() {
            let id = $(this).attr('id');
            if (id === 'save') {
                if (window.confirm('The current canvas will be saved in your local! Are you
sure?')) {
                    saveInBrowser.save('canvasEditor', _self.canvas.toJSON());
                }
            }
        })
    }

```

```

    } else if (id === 'clear') {
        if (window.confirm('This will clear the canvas! Are you sure?')) {
            _self.canvas.clear(), saveInBrowser.remove('canvasEditor');
        }
    } else if (id === 'download') {
        $('body').append(`<div class="custom-modal-container">
            <div class="custom-modal-content">
                <div class="button-download" id="png">Download as PNG</div>
                <div class="button-download" id="jpg">Download as JPG</div>
            </div>
        </div>`)
        $(".custom-modal-container").click(function () {
            $(this).remove();
        })
        $(".custom-modal-container .button-download").click(function (e) {
            let type = $(this).attr('id');
            if (type === 'png') downloadImage(_self.canvas.toDataURL())
            else if (type === 'jpg') downloadImage(_self.canvas.toDataURL({
                format: 'jpeg'
            }), 'jpg', 'image/jpeg');
        })
    } else if (id === 'undo') _self.undo();
    else if (id === 'redo') _self.redo();
    })
    })()
} catch (e) {
    console.error("can't create toolbar");
}
}

```

```

window.ImageEditor.prototype.initializeToolbar = toolbar;
})();

// ../lib/upload.js

//Define action to upload, drag & drop images into canvas
(function () {
    var upload = function (canvas) {
        const _self = this;
        this.openDragDropPanel = function () {
            console.log('open drag drop panel')
            $('body').append('<div class="custom-modal-container">
                <div class="custom-modal-content">
                    <div class="drag-drop-input">
                        <div>Drag & drop files<br>or click to browse.<br>JPG or PNG only!</div>
                    </div>
                </div>
            </div>`)
            $('.custom-modal-container').click(function () {
                $(this).remove()
            })
            $('.drag-drop-input').click(function () {
                console.log('click drag drop')
                $('_${_self.containerSelector} #btn-image-upload').click();
            })
            $(".drag-drop-input").on("dragover", function (event) {
                event.preventDefault();
                event.stopPropagation();
                $(this).addClass('dragging');
            });
        }
    }

```

```

});
$(".drag-drop-input").on("dragleave", function (event) {
    event.preventDefault();
    event.stopPropagation();
    $(this).removeClass('dragging');
});
$(".drag-drop-input").on("drop", function (event) {
    event.preventDefault();
    event.stopPropagation();
    $(this).removeClass('dragging');
    if (event.originalEvent.dataTransfer) {
        if (event.originalEvent.dataTransfer.files.length) {
            let files = event.originalEvent.dataTransfer.files
            processFiles(files);
            $('.custom-modal-container').remove();
        }
    }
});
}

const processFiles = (files) => {
    if (files.length === 0) return;
    const allowedTypes = ['image/jpeg', 'image/png']
    for (let file of files) {
        // check type
        if (!allowedTypes.includes(file.type)) continue
        let reader = new FileReader()
        // handle image, read file, add to canvas
        reader.onload = (f) => {
            fabric.Image.fromURL(f.target.result, (img) => {

```

```

        img.set({
            left: 0,
            top: 0
        })
        img.scaleToHeight(300)
        img.scaleToWidth(300)
        canvas.add(img)
        canvas.renderAll()
        canvas.trigger('object:modified')
    })
}
reader.readAsDataURL(file)
}
}
this.containerEl.append(`<input id="btn-image-upload" type="file"
accept="image/*" multiple hidden>`);
document.querySelector(`${this.containerSelector} #btn-image-
upload`).addEventListener('change', function (e) {
    if (e.target.files.length === 0) return;
    processFiles(e.target.files)
})
}
window.ImageEditor.prototype.initializeUpload = upload;
})()

// ../lib/utils.js

//Define util functions
/**

```

```

* Get fabric js gradient from colorstops, orientation and angle
* @param {Array} handlers array of color stops
* @param {Number} width gradient width
* @param {Number} height gradient height
* @param {String} orientation orientation type linear/radial
* @param {Number} angle the angle of linear gradient
*/

```

```

const generateFabricGradientFromColorStops = (handlers, width, height, orientation,
angle) => {
  const gradAngleToCoords = (angle) => {
    let anglePI = (-parseInt(angle, 10)) * (Math.PI / 180)
    let angleCoords = {
      'x1': (Math.round(50 + Math.sin(anglePI) * 50)) / 100,
      'y1': (Math.round(50 + Math.cos(anglePI) * 50)) / 100,
      'x2': (Math.round(50 + Math.sin(anglePI + Math.PI) * 50)) / 100,
      'y2': (Math.round(50 + Math.cos(anglePI + Math.PI) * 50)) / 100,
    }
    return angleCoords
  }
  let bgGradient = {};
  let colorStops = [];
  for (var i in handlers) {
    colorStops.push({
      id: i,
      color: handlers[i].color,
      offset: handlers[i].position / 100,
    })
  }
  if (orientation === 'linear') {

```

```

let angleCoords = gradAngleToCoords(angle)
bgGradient = new fabric.Gradient({
  type: 'linear',
  coords: {
    x1: angleCoords.x1 * width,
    y1: angleCoords.y1 * height,
    x2: angleCoords.x2 * width,
    y2: angleCoords.y2 * height
  },
  colorStops,
})
} else if (orientation === 'radial') {
  bgGradient = new fabric.Gradient({
    type: 'radial',
    coords: {
      x1: width / 2,
      y1: height / 2,
      r1: 0,
      x2: width / 2,
      y2: height / 2,
      r2: width / 2
    },
    colorStops: colorStops
  });
}
return bgGradient
}
const getRealBBox = async (obj) => {
  let tempCanv, ctx, w, h;

```



```

// we need to use a temp canvas to get imagedata
const getImageData = (dataUrl) => {
  if (tempCanv === null) {
    tempCanv = document.createElement('canvas');
    tempCanv.style.border = '1px solid blue';
    tempCanv.style.position = 'absolute';
    tempCanv.style.top = '-100%';
    tempCanv.style.visibility = 'hidden';
    ctx = tempCanv.getContext('2d');
    document.body.appendChild(tempCanv);
  }
  return new Promise(function (resolve, reject) {
    if (dataUrl === null) return reject();
    var image = new Image();
    image.addEventListener('load', () => {
      w = image.width;
      h = image.height;
      tempCanv.width = w;
      tempCanv.height = h;
      ctx.drawImage(image, 0, 0, w, h);
      var imageData = ctx.getImageData(0, 0, w, h).data.buffer;
      resolve(imageData, false);
    });
    image.src = dataUrl;
  });
}

// analyze pixels 1-by-1
const scanPixels = (imageData) => {
  var data = new Uint32Array(imageData),

```

```

x, y, y1, y2, x1 = w,
x2 = 0;
// y1
for (y = 0; y < h; y++) {
    for (x = 0; x < w; x++) {
        if (data[y * w + x] & 0xff000000) {
            y1 = y;
            y = h;
            break;
        }
    }
}
// y2
for (y = h - 1; y > y1; y--) {
    for (x = 0; x < w; x++) {
        if (data[y * w + x] & 0xff000000) {
            y2 = y;
            y = 0;
            break;
        }
    }
}
// x1
for (y = y1; y < y2; y++) {
    for (x = 0; x < w; x++) {
        if (x < x1 && data[y * w + x] & 0xff000000) {
            x1 = x;
            break;
        }
    }
}

```

```

    }
  }
  // x2
  for (y = y1; y < y2; y++) {
    for (x = w - 1; x > x1; x--) {
      if (x > x2 && data[y * w + x] & 0xff000000) {
        x2 = x;
        break;
      }
    }
  }
  return {
    x1: x1,
    x2: x2,
    y1: y1,
    y2: y2,
    width: x2 - x1,
    height: y2 - y1
  }
}

let data = await getImageData(obj.toDataURL());
return scanPixels(data);
}

/**
 * Get the filters of current image selection
 * @param {Object} activeSelection fabric js object
 */
const getCurrentEffect = (activeSelection) => {
  let updatedEffects = {

```

```

opacity: 100,
blur: 0,
brightness: 50,
saturation: 50,
gamma: {
  r: 45,
  g: 45,
  b: 45
}
}
updatedEffects.opacity = activeSelection.opacity * 100
let hasBlur = activeSelection.filters.find(x => x.blur)
if (hasBlur) {
  updatedEffects.blur = hasBlur.blur * 100
}
let hasBrightness = activeSelection.filters.find(x => x.brightness)
if (hasBrightness) {
  updatedEffects.brightness = ((hasBrightness.brightness + 1) / 2) * 100
}
let hasSaturation = activeSelection.filters.find(x => x.saturation)
if (hasSaturation) {
  updatedEffects.saturation = ((hasSaturation.saturation + 1) / 2) * 100
}
let hasGamma = activeSelection.filters.find(x => x.gamma)
if (hasGamma) {
  updatedEffects.gamma.r = Math.round(hasGamma.gamma[0] / 0.022)
  updatedEffects.gamma.g = Math.round(hasGamma.gamma[1] / 0.022)
  updatedEffects.gamma.b = Math.round(hasGamma.gamma[2] / 0.022)
}

```

```

    return updatedEffects;
}
const getUpdatedFilter = (effects, effect, value) => {
  let updatedEffects = {
    ...effects
  }
  switch (effect) {
    case 'gamma.r':
      updatedEffects.gamma.r = value
      break
    case 'gamma.g':
      updatedEffects.gamma.g = value
      break
    case 'gamma.b':
      updatedEffects.gamma.b = value
      break

    default:
      updatedEffects[effect] = value
      break
  }
  effects = updatedEffects;
  // rebuild filter array, calc values for fabric
  // blur 0-1 (def val 0), brightness, saturation -1-1 (def val: 0), gamma 0-2.2 (def
val: 1)
  let updatedFilters = []
  if (effects.blur > 0) {
    updatedFilters.push(new fabric.Image.filters.Blur({
      blur: effects.blur / 100

```

```

    }));
  }
  if (effects.brightness !== 50) {
    updatedFilters.push(new fabric.Image.filters.Brightness({
      brightness: ((effects.brightness / 100) * 2) - 1
    }));
  }
  if (effects.saturation !== 50) {
    updatedFilters.push(new fabric.Image.filters.Saturation({
      saturation: ((effects.saturation / 100) * 2) - 1
    }));
  }
  if (
    effects.gamma.r !== 45 ||
    effects.gamma.g !== 45 ||
    effects.gamma.b !== 45
  ) {
    updatedFilters.push(new fabric.Image.filters.Gamma({
      gamma: [
        Math.round((effects.gamma.r * 0.022) * 10) / 10,
        Math.round((effects.gamma.g * 0.022) * 10) / 10,
        Math.round((effects.gamma.b * 0.022) * 10) / 10
      ]
    }));
  }
  return updatedFilters;
}

const getActiveFontStyle = (activeSelection, styleName) => {
  if (activeSelection.getSelectionStyles && activeSelection.isEditing) {

```

```

let styles = activeSelection.getSelectionStyles()
if (styles.find(o => o[styleName] === "")) {
  return ""
}
return styles[0][styleName]
}
return activeSelection[styleName] || ""
}
const setActiveFontStyle = (activeSelection, styleName, value) => {
  if (activeSelection.setSelectionStyles && activeSelection.isEditing) {
    let style = {}
    style[styleName] = value;
    activeSelection.setSelectionStyles(style)
    activeSelection.setCoords()
  } else {
    activeSelection.set(styleName, value)
  }
}
const downloadImage = (data, extension = 'png', mimeType = 'image/png') => {
  const imageData = data.toString().replace(/^data:image\/(png|jpeg|jpg);base64,/, "");
  const byteCharacters = atob(imageData);
  const byteNumbers = new Array(byteCharacters.length);
  for (let i = 0; i < byteCharacters.length; i += 1) {
    byteNumbers[i] = byteCharacters.charCodeAt(i);
  }
  const byteArray = new Uint8Array(byteNumbers);
  const file = new Blob([byteArray], {
    type: mimeType + ';base64'
  });
};

```

```

const fileURL = window.URL.createObjectURL(file);
// IE doesn't allow using a blob object directly as link href
// instead it is necessary to use msSaveOrOpenBlob
if (window.navigator && window.navigator.msSaveOrOpenBlob) {
  window.navigator.msSaveOrOpenBlob(file);
  return;
}
const link = document.createElement('a');
link.href = fileURL;
link.download = 'image.' + extension;
link.dispatchEvent(new MouseEvent('click'));
setTimeout(() => {

  window.URL.revokeObjectURL(fileURL);
}, 60);
}

// ../lib/zoom.js

//Define action to zoom in/out by mouse+key events
// keyboard shortcuts and zoom calculations
const minZoom = 0.05
const maxZoom = 3
// zoom with key
const zoomWithKeys = (e, canvas, applyZoom) => {
  const key = e.which || e.keyCode
  // ctrl -
  if (key === 189 && e.ctrlKey) {
    e.preventDefault()

```



```

if (canvas.getZoom() === minZoom) return
let updatedZoom = parseInt(canvas.getZoom() * 100)
// 25% jumps
if ((updatedZoom % 25) !== 0) {
  while ((updatedZoom % 25) !== 0) {
    updatedZoom = updatedZoom - 1
  }
} else {
  updatedZoom = updatedZoom - 25
}
updatedZoom = updatedZoom / 100
updatedZoom = (updatedZoom <= 0) ? minZoom : updatedZoom
applyZoom(updatedZoom)
}
// ctrl +
if (key === 187 && e.ctrlKey) {
  e.preventDefault()
  if (canvas.getZoom() === maxZoom) return
  let updatedZoom = parseInt(canvas.getZoom() * 100)
  // 25% jumps
  if ((updatedZoom % 25) !== 0) {
    while ((updatedZoom % 25) !== 0) {
      updatedZoom = updatedZoom + 1
    }
  } else {
    updatedZoom = updatedZoom + 25
  }
  updatedZoom = updatedZoom / 100
  updatedZoom = (updatedZoom > maxZoom) ? maxZoom : updatedZoom

```

```

    applyZoom(updatedZoom)
  }
  // ctrl 0
  if ((key === 96 || key === 48 || key === 192) && e.ctrlKey) {
    e.preventDefault()
    applyZoom(1)
  }

}

// zoom with mouse
const zoomWithMouse = (e, canvas, applyZoom) => {
  if (!e.ctrlKey) return
  e.preventDefault()
  let updatedZoom = canvas.getZoom().toFixed(2)
  let zoomAmount = (e.deltaY > 0) ? -5 : 5
  updatedZoom = ((updatedZoom * 100) + zoomAmount) / 100
  if (updatedZoom < minZoom || updatedZoom > maxZoom) return
  applyZoom(updatedZoom)
}

```

ДОДАТОК В

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”

Керівник роботи

Світлана ПОПЕРЕШНЯК

“28” грудня 2023 р.

Файлова система з консольним інтерфейсом

Програма та методика тестування

КП.ІІ-1125.045440.04.51

“ПОГОДЖЕНО”

Керівник роботи:

Світлана ПОПЕРЕШНЯК

Консультант:

Максим ГОЛОВЧЕНКО

Виконавець:

Владислав ПРИЩЕПА

ЗМІСТ

1 ОБ'ЄКТ ВИПРОБУВАНЬ.....	3
2 МЕТА ТЕСТУВАННЯ.....	4
3 МЕТОДИ ТЕСТУВАННЯ.....	5
4 ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ.....	6

1 ОБ'ЄКТ ВИПРОБУВАНЬ

Об'єктом випробувань є веб-редактор зображень, що написаний на мові програмування JavaScript.

2 МЕТА ТЕСТУВАННЯ

У процесі тестування має бути перевірено наступне:

- а) працезданість веб-редактору зображень;
- б) виконання функціональних вимог;
- в) виконання нефункціональних вимог.

3 МЕТОДИ ТЕСТУВАННЯ

Тестування програмного забезпечення було здійснено мануально.

4 ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ

Працездатність файлової системи перевіряється за допомогою:

а) ручного всіх функцій, заявлених в технічному завданні.

ДОДАТОК Г

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”

Керівник роботи

Світлана ПОПЕРЕШНЯК

“28” грудня 2023 р.

Файлова система з консольним інтерфейсом

Керівництво користувача

КПІ.ІП-1125.045440.05.13

“ПОГОДЖЕНО”

Керівник роботи:

Світлана ПОПЕРЕШНЯК

Консультант:

Максим ГОЛОВЧЕНКО

Виконавець:

Владислав ПРИЩЕПА

ЗМІСТ

1 ПРИЗНАЧЕННЯ ПРОГРАМИ.....	3
2 ПІДГОТОВКА ДО РОБОТИ З ПРОГРАМНИМ ЗАБЕЗПЕЧЕННЯМ.....	4
2.1 Системні вимоги для коректної роботи.....	4
2.2 Завантаження застосунку.....	4

1 ПРИЗНАЧЕННЯ ПРОГРАМИ

Веб-редактор зображень призначений для створення та редагування зображень, які можуть використовуватися для побутових цілей. Завдяки можливостям веб-редактору зображень користувач може створити та редагувати зображення.

2 ПІДГОТОВКА ДО РОБОТИ З ПРОГРАМНИМ ЗАБЕЗПЕЧЕННЯМ

2.1 Системні вимоги для коректної роботи

Для успішного функціонування програми користувач повинен мати ІВМсумісний персональний комп'ютер.

Мінімальна конфігурація технічних засобів:

- тип процесору: Intel Core i5;
- об'єм ОЗП: 4 Гб;
- підключення до мережі Інтернет зі швидкістю від 20 мегабіт;
- назва операційної системи: Windows XP;
- тип операційної системи: 32-бітна.

Рекомендована конфігурація технічних засобів:

- тип процесору: AMD Ryzen 7 2700X Eight-Core Processor;
- об'єм ОЗП: 16 Гб;
- підключення до мережі Інтернет зі швидкістю від 100 мегабіт;
- назва операційної системи: Windows 10 версії 22H2;
- тип операційної системи: 64-бітна.

2.2 Завантаження застосунку

Склонуйте репозиторій чи завантажте архів репозиторію CW5_Image_Editor, як продемонстровано на рисунку 2.1

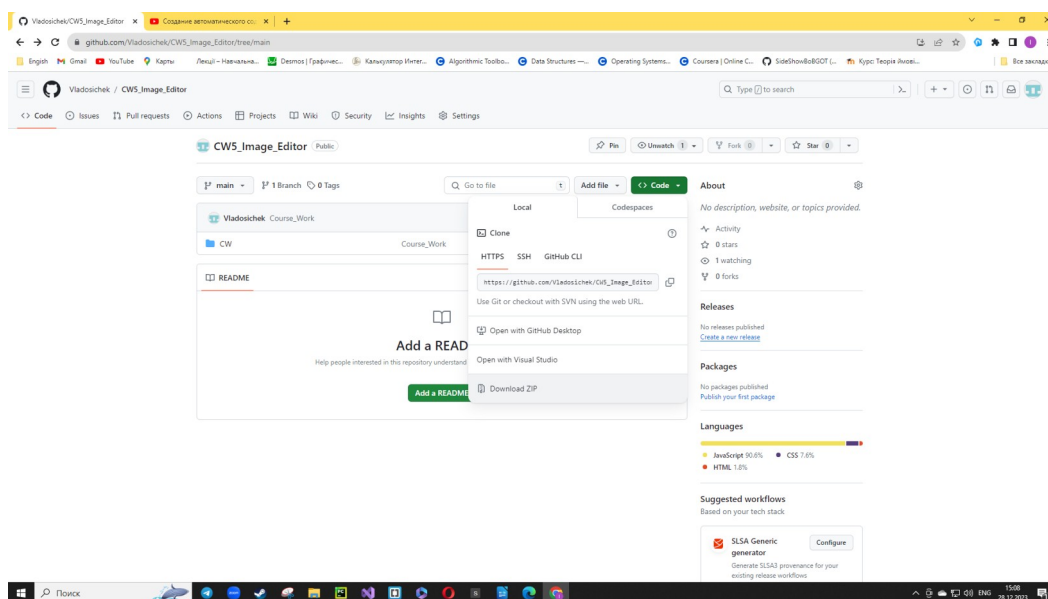


Рисунок 2.1 Завантаження архіву репозиторію

Далі розархівуйте архів в обрану папку. Матимете наступну структуру файлів, як на рисунку 2.2





 external	25.12.2023 17:05	Папка с файлами	
 lib	25.12.2023 19:40	Папка с файлами	
 index.html	26.12.2023 18:32	Opera Web Docu...	2 КБ
 script.js	27.12.2023 13:19	файл JavaScript	2 КБ

Рисунок 2.2 Результат розархівування архіву

Далі запустіть файл index.html. Результатом запуску буде веб-застосунок у вашому браузері, як на рисунку 2.3

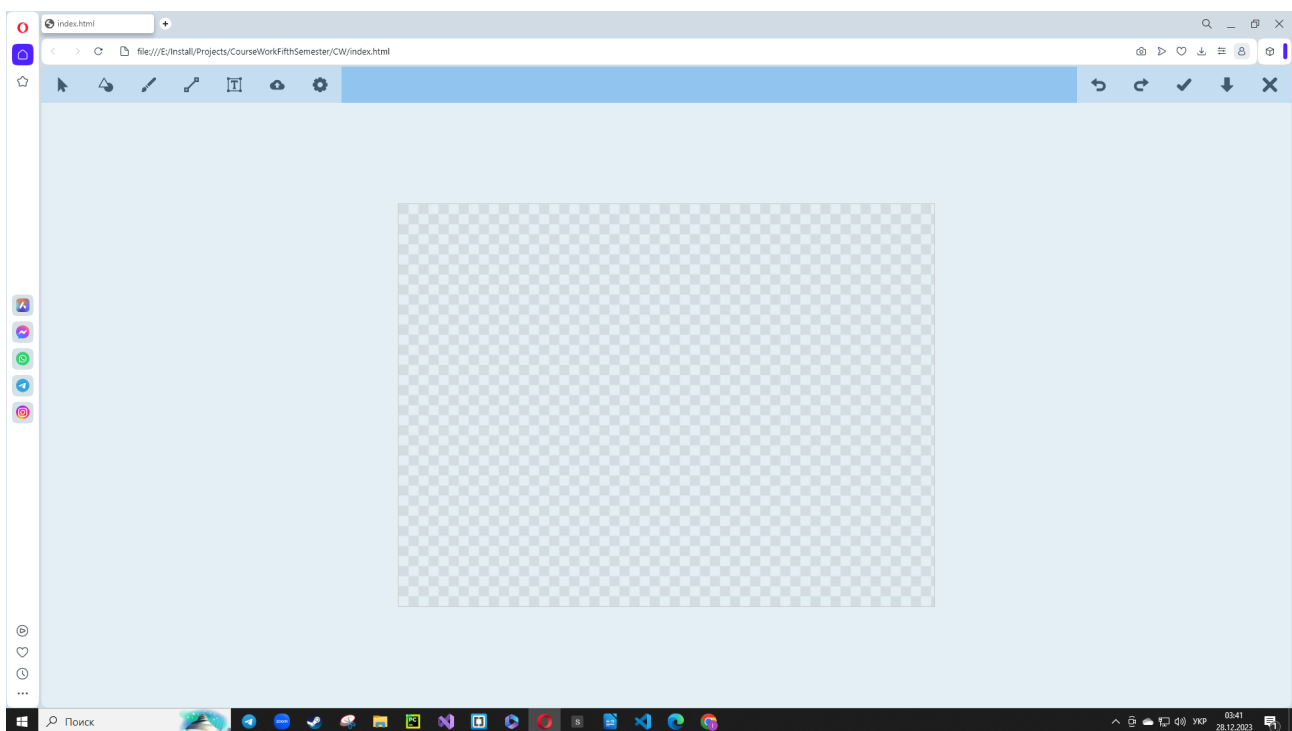


Рисунок 2.2 Результат запуску застосунку

