**Національний технічний університет України**

**“Київський політехнічний інститут імені Ігоря Сікорського”**

**Кафедра інженерії програмного забезпечення в енергетиці**

***КУРСОВА РОБОТА***

**з дисципліни:**

**«Основи Веб-програмування»**

**тема: «Додаток для малювання»**

***2025***

**Національний технічний університет України**

**“Київський політехнічний інститут імені Ігоря Сікорського”**

**Кафедра інженерії програмного забезпечення в енергетиці**

***КУРСОВА РОБОТА***

**з дисципліни:**

**«Розробка програмного забезпечення мобільних пристроїв»**

**тема: Тестова документація проекту**

**«Додаток для календаря»**

***2024***

**Національний технічний університет України**

**“Київський політехнічний інститут імені Ігоря Сікорського”**

**Кафедра інженерії програмного забезпечення в енергетиці**

***КУРСОВА РОБОТА***

**з дисципліни:**

**«Розробка програмного забезпечення мобільних пристроїв»**

**тема: Тестова документація проекту**

**«Додаток для календаря»**

***2024***

**Національний технічний університет України**

**“Київський політехнічний інститут імені Ігоря Сікорського”**

**Кафедра інженерії програмного забезпечення в енергетиці**

***КУРСОВА РОБОТА***

**з дисципліни:**

**«Розробка програмного забезпечення мобільних пристроїв»**

**тема: Тестова документація проекту**

**«Додаток для календаря»**

***2024***

**Національний технічний університет України**

**“Київський політехнічний інститут імені Ігоря Сікорського”**

**Кафедра інженерії програмного забезпечення в енергетиці**

***КУРСОВА РОБОТА***

**з дисципліни:**

**«Розробка програмного забезпечення мобільних пристроїв»**

**тема: Тестова документація проекту**

**«Додаток для календаря»**

***2024***

**Національний технічний університет України**

**“Київський політехнічний інститут імені Ігоря Сікорського”**

**Кафедра інженерії програмного забезпечення в енергетиці**

***КУРСОВА РОБОТА***

**з дисципліни:**

**«Розробка програмного забезпечення мобільних пристроїв»**

**тема: Тестова документація проекту**

**«Додаток для календаря»**

***2024***

**Національний технічний університет України**

**“Київський політехнічний інститут імені Ігоря Сікорського”**

**Кафедра інженерії програмного забезпечення в енергетиці**

***КУРСОВА РОБОТА***

**з дисципліни:**

**«Розробка програмного забезпечення мобільних пристроїв»**

**тема: Тестова документація проекту**

**«Додаток для календаря»**

***2024***

НAЦIOНAЛЬНИЙ ТEXНIЧНИЙ УНIВEРCИТEТ УКРAЇНИ

«КИЇВCЬКИЙ ПOЛIТEXНIЧНИЙ IНCТИТУТ iмeнi IГOРЯ CIКOРCЬКOГO»

Кафедра інженерії програмного забезпечення в енергетиці

**КУРСОВА РОБОТА**

з дисципліни: « Основи Веб-програмування »

(назва дисципліни)

на тему: «Додадок для малювання»

Студентки 2 курсу групи ТВ-32

напряму підготовки ***121 Програмна інженерія***

\_\_\_\_\_\_\_\_\_ Недяк. О.С

(прізвище та ініціали)

Керівник \_\_д.т.н., доцент, Недашківський О. Л. \_\_

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

Національна оцінка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кількість балів: \_\_\_\_\_\_\_\_\_\_Оцінка: ECTS \_\_\_\_\_

Член комісії

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

Київ – 2025 рік

**Національний технічний університет України**

**“Київський політехнічний інститут імені Ігоря Сікорського”**

Навчально-науковий інститут атомної та теплової енергетики \_

( повна назва )

Кафедра інженерії програмного забезпечення в енергетиці\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

( повна назва )

Напрям підготовки \_121\_\_\_\_ Програмна інженерія\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(шифр і назва)

**З А В Д А Н Н Я**

**НА КУРСОВУ РОБОТУ СТУДЕНТУ**

Недяк Олександр Сергійович

(прізвище, ім’я, по батькові)

1. Тема роботи – «Додадок для малювання»

Керівник курсової роботи – Недашківський О.Л.

(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

2. Строк подання студентом роботи: «22» червня 2025р.

3. Вихідні дані до проекту (роботи): мова програмування – JavaScritp, дизайн – CSS.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити): розробити веб-додаток для малювання.

5. Дата видачі завдання: 14 січня 2025

**КАЛЕНДАРНИЙ ПЛАН**

|  |  |  |  |
| --- | --- | --- | --- |
| №  з/п | Назва етапів виконання дипломного  проекту (роботи) | Строк виконання етапів проекту  ( роботи ) | Примітка |
| 1. | Затвердження теми роботи | 14.01.2025 |  |
| 2. | Вивчення та аналіз задачі | 21.05.2023 |  |
| 3. | Тестування ідеї гри | 24.05.2023 |  |
| 4. | Створення тест плану | 27.05.2023 |  |
| 5. | Розробка програмного коду | 02.06.2023 |  |
| 6. | Розробка інтерфейсу програми | 09.06.2023 |  |
| 7. | Тестування програми | 16.06.2023 |  |
| 8. | Оформлення пояснювальної записки | 22.06.2023 |  |

**Студент \_\_\_\_\_\_\_\_\_** \_\_Недяк О.С.

( підпис ) (прізвище та ініціали)

**Керівник курсової роботи \_\_\_\_\_\_\_\_\_** Недашківський О. Л.

( підпис ) (прізвище та ініціали)

**АНОТАЦІЯ**

У ході виконання курсової роботи було освоєно основи розробки веб-додатків із використанням HTML, CSS та JavaScript, з акцентом на створення інтерактивного графічного редактора. Використані знання застосовано для реалізації та вдосконалення проекту, який дозволяє користувачам створювати та редагувати зображення з підтримкою шарів, інструментів малювання та ефектів. Під час виконання роботи увага приділялася кожному етапу розробки — від планування функціоналу до реалізації готового рішення. Основною метою роботи було розроблення функціоналу додатку, що забезпечує зручне малювання, редагування зображень, роботу з шарами та збереження результатів, а також створення інтуїтивно зрозумілого інтерфейсу. Графічний редактор допомагає користувачам створювати унікальні зображення, застосовувати фільтри та управляти проєктами.

Обсяг пояснювальної записки 77 аркушів, кількість ілюстрацій — 18, 3 додатки.

**ANNOTATION**

During the coursework, the basics of web application development using HTML, CSS, and JavaScript were mastered, with an emphasis on creating an interactive graphic editor. The knowledge gained was applied to the implementation and improvement of a project that allows users to create and edit images with support for layers, drawing tools, and effects. During the work, attention was paid to each stage of development — from planning the functionality to implementing the finished solution. The main goal of the work was to develop the functionality of the application, which provides convenient drawing, image editing, working with layers, and saving results, as well as creating an intuitive interface. The graphic editor helps users create unique images, apply filters, and manage projects.

The volume of the explanatory note is 77 sheets, the number of illustrations is 18, 3 appendices.

**ЗМІСТ**

[АНОТАЦІЯ 3](#_Toc201518743)

[РОЗДІЛ 1. ОПИС ПРОЕКТУ 5](#_Toc201518744)

[1.1 Формулювання задач та визначення ролей 6](#_Toc201518745)

[1.2 Огляд проекту та його структура 6](#_Toc201518746)

[**Функціональність додатку** 7](#_Toc201518747)

[**1.2.1. Бібліотеки та інструменти** 8](#_Toc201518748)

[**1.2.2. Структура додатку** 8](#_Toc201518749)

[**1.2.3. Реалізація функцій** 9](#_Toc201518750)

[РОЗДІЛ 2. ПРОЦЕС РОЗРОБКИ ДОДАТКУ 10](#_Toc201518751)

[2.1 Використані технології 10](#_Toc201518752)

[**2.1.1 HTML, CSS та JavaScript** 10](#_Toc201518753)

[**2.1.2 Canvas API** 10](#_Toc201518754)

[**2.1.3 Модульний підхід** 10](#_Toc201518755)

[2.2 Структура проекту 10](#_Toc201518756)

[2.3 Етапи розробки 10](#_Toc201518757)

[**2.3.1 Проектування інтерфейсу** 10](#_Toc201518758)

[**2.3.2 Реалізація логіки малювання** 11](#_Toc201518759)

[**2.3.3 Управління шарами** 11](#_Toc201518760)

[**2.3.4 Оптимізація та стилізація** 11](#_Toc201518761)

[2.4 Переваги технологій 11](#_Toc201518762)

[2.5 Ключові файли проекту 11](#_Toc201518763)

[**2.5.1 index.html** 11](#_Toc201518764)

[**2.5.2 script.js** 11](#_Toc201518765)

[**2.5.3 utils.js** 11](#_Toc201518766)

[**2.5.4 tools.js** 11](#_Toc201518767)

[**2.5.5 shapes.js** 12](#_Toc201518768)

[**2.5.6 layers.js** 12](#_Toc201518769)

[**2.5.7 styles.css** 12](#_Toc201518770)

[РОЗДІЛ 3. ХІД ВИКОНАННЯ РОБОТИ 12](#_Toc201518771)

[**MainActivity** 12](#_Toc201518772)

[**Layers Management** 13](#_Toc201518773)

[**Tools Management** 14](#_Toc201518774)

[**Shapes Management** 15](#_Toc201518775)

[**Utils Management** 16](#_Toc201518776)

[**Styles Management** 16](#_Toc201518777)

[**User Interface** 17](#_Toc201518778)

[РОЗДІЛ 4. КЕРІВНИЦТВО КОРИСТУВАЧА 18](#_Toc201518779)

[4.1 Головний екран додатку 18](#_Toc201518780)

[4.2 Використання інструментів малювання 18](#_Toc201518781)

[**4.2.1 Малювання вільною рукою** 18](#_Toc201518782)

[**4.2.2 Малювання фігур** 19](#_Toc201518783)

[**4.2.3 Використання заливки та аерографа** 19](#_Toc201518784)

[4.3 Управління шарами 20](#_Toc201518785)

[4.4 Налаштування кольору та градієнта 21](#_Toc201518786)

[4.5 Застосування фільтрів 21](#_Toc201518787)

[4.6 Збереження та скасування дій 22](#_Toc201518788)

[4.7 Завантаження зображень 22](#_Toc201518789)

[ВИСНОВКИ 23](#_Toc201518790)

[CПИCOК ВИКOРИCТAНИX ДЖEРEЛ 25](#_Toc201518791)

[ДОДАТОК 1 26](#_Toc201518792)

[ДОДАТОК 2 29](#_Toc201518793)

[ДОДАТОК 3 32](#_Toc201518794)

У сучасному світі, де творчість є важливим способом самовираження та реалізації ідей, цифрові інструменти для малювання стають дедалі популярнішими. З цією метою було створено веб-застосунок для малювання, який поєднує сучасні веб-технології з інтуїтивно зрозумілим інтерфейсом для забезпечення комфортного творчого процесу.

Основна мета додатку – надати користувачам зручний інструмент для створення цифрових малюнків за допомогою широкого набору функцій, таких як різноманітні пензлі, палітри кольорів, шари та фільтри. Додаток дозволяє створювати, редагувати та зберігати малюнки, а також підтримує інтеграцію з хмарними сервісами для синхронізації робіт між пристроями. Модуль авторизації забезпечує безпечну реєстрацію через електронну пошту, логін, логаут та інтеграцію з Google-акаунтом, що гарантує захист даних і зручність використання.

Актуальність цього веб-застосунку зумовлена зростаючою потребою в доступних і гнучких інструментах для творчості, які дозволяють як новачкам, так і професіоналам втілювати свої ідеї в цифровому форматі, підвищуючи продуктивність і якість творчого процесу.

# РОЗДІЛ 1. ОПИС ПРОЕКТУ

Ця курсова робота має на меті розробку календарного додатку, який допомагає користувачам в організації та плануванні їх повсякденного життя. Основна ідея проекту полягає у створенні зручного та інтуїтивно зрозумілого інструменту для планування подій, синхронізації з різними онлайн-календарями, такими як Google Calendar, та таким чином зручне управління особистим часом.

Основний функціонал додатку включатиме створення, перегляд, редагування та видалення подій у календарі. Ці можливості дають користувачам повний контроль над своїм часом і дозволяють ефективно планувати свої дні.

Метою даного проекту є створення не лише функціонального, але й безпечного та надійного додатку для управління часом, який відповідає сучасним вимогам та сприяє кращій організації життя користувачів. Сконцентрований на інтеграції з онлайн-сервісами та забезпеченні високої якості користувацького досвіду, цей додаток стане важливим помічником у плануванні та досягненні особистих та професійних цілей.

## **Формулювання задач та визначення ролей**

Перед початком розробки проекту першою задачею є визначення обов’язків серед учасників команди. У таблиці 1 зазначено роль та ім’я кожної людини.

Таблиця 1. Ролі

|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **Роль** | **Ім’я** | **Контакти** |
| 1 | Головний розробник | Гнилицький Дмитро Дмитрович | [dmitrijgnilickij7@gmail.com](mailto:dmitrijgnilickij7@gmail.com) |
| 2 | Дизайнер та тестувальник | Недяк Олександр Сергійович | [sashabelik085@gmail.com](mailto:sashabelik085@gmail.com) |

Головний розробник керує процесом розробки, виступаючи програмістом і координатором. Його обов’язки охоплюють визначення напрямку розробки, розподіл завдань між учасниками команди, написання коду, налаштування інфраструктури проекту, а також участь у вирішенні технічних питань і прийнятті стратегічних рішень.

Дизайнер та тестувальник відповідає за створення дизайну інтерфейсу користувача та забезпечення якості функціоналу. Його завдання включають розробку інтуїтивно зрозумілого й естетично привабливого інтерфейсу, а також проведення тестування для виявлення та усунення помилок чи недоліків, що можуть виникнути під час розробки.

## **1.2 Огляд проекту та його структура**

Основні характеристики додатку

Проект являє собою веб-додаток для малювання, який дозволяє користувачам створювати, редагувати та зберігати графічні зображення за допомогою різноманітних інструментів і функцій. Основна мета — надати зручний, інтуїтивний інструмент для творчості з підтримкою шарів, різноманітних інструментів малювання, фігур і фільтрів.

Ключові можливості:

1. **Інструменти малювання:**
   * Олівець, маркер, аерограф, гумка, заливка.
   * Налаштування кольору, товщини ліній, прозорості та форми пензля (кругла чи квадратна).
   * Підтримка градієнтів для інструментів і фігур.
2. **Фігури:**
   * Лінія, прямокутник, коло, трикутник, шестикутник, п’ятикутник, зірка, серце, ромб, стрілка, крива.
   * Можливість малювання фігур з градієнтним або однотонним заповненням.
3. **Шари:**
   * Створення, видалення та об’єднання шарів.
   * Налаштування видимості, прозорості, режимів накладання (normal, multiply, screen).
   * Перетягування шарів для зміни їхнього порядку.
4. **Інтерфейс:**
   * Інтуїтивна панель інструментів і панель шарів.
   * Підтримка світлого та темного режимів.
   * Масштабування полотна та повноекранний режим.
5. **Редагування та збереження:**
   * Скасування та повторення дій (undo/redo).
   * Застосування фільтрів: відтінки сірого, інверсія кольорів.
   * Збереження малюнків у форматах PNG, JPEG, SVG.
   * Завантаження зображень для редагування.

### **Функціональність додатку**

Додаток розроблено як інструмент для малювання, який поєднує простоту використання з розширеними можливостями. Ось детальний опис його функціональності:

* **Малювання:**
  + Користувачі можуть створювати малюнки за допомогою інструментів, таких як олівець, маркер, аерограф і гумка. Кожен інструмент має налаштування товщини, прозорості та форми пензля.
  + Інструмент заливки дозволяє заповнювати області кольором або градієнтом.
  + Підтримка малювання фігур із заданими параметрами (колір, товщина, прозорість).
* **Робота з шарами:**
  + Користувачі можуть додавати нові шари, видаляти їх або об’єднувати для створення складних композицій.
  + Кожен шар має налаштування видимості, прозорості та режиму накладання, а також можливість блокування для захисту від змін.
  + Перетягування шарів у панелі дозволяє легко змінювати їхній порядок.
* **Інтерфейс користувача:**
  + Панель інструментів забезпечує швидкий доступ до всіх функцій малювання, налаштувань кольору та режимів.
  + Панель шарів відображає список шарів із можливістю їх редагування.
  + Темний і світлий режими дозволяють адаптувати інтерфейс до вподобань користувача.
  + Масштабування полотна (від 10% до 300%) і повноекранний режим забезпечують комфортну роботу.
* **Редагування:**
  + Функції скасування (undo) та повторення (redo) дозволяють виправляти помилки (до 30 кроків).
  + Фільтри (відтінки сірого, інверсія кольорів) застосовуються до активного шару для стилізації малюнків.
  + Завантаження зображень із буфера обміну або файлів додає можливість редагувати існуючі картинки.
* **Збереження:**
  + Малюнки можна зберігати у популярних форматах (PNG, JPEG, SVG) для подальшого використання.

### **1.2.1. Бібліотеки та інструменти**

Для створення додатку використано такі технології:

* **HTML, CSS, JavaScript:**
  + Основний стек для розробки веб-додатку.
  + HTML (index.html) визначає структуру інтерфейсу.
  + CSS (styles.css) відповідає за стилізацію, включаючи підтримку темного режиму.
  + JavaScript (script.js, utils.js, tools.js, shapes.js, layers.js) забезпечує всю логіку роботи.
* **Canvas API:**
  + Використовується для реалізації малювання, роботи з шарами та застосування фільтрів.

### **1.2.2. Структура додатку**

Додаток складається з таких ключових компонентів:

* **Скрипти:**
  + script.js — головний файл, що ініціалізує додаток, обробляє події та координує роботу всіх модулів.
  + utils.js — утилітні функції (наприклад, генерація ID, конвертація кольорів, інтерполяція градієнтів).
  + tools.js — логіка інструментів малювання (олівець, аерограф, заливка, гумка, фільтри).
  + shapes.js — функції для малювання фігур (лінія, коло, зірка тощо).
  + layers.js — управління шарами (створення, видалення, об’єднання, перетягування).
* **Стилі:**
  + styles.css — стилі для інтерфейсу, включаючи адаптивний дизайн і підтримку тем.
* **HTML:**
  + index.html — основна сторінка з розміткою для полотна, панелі інструментів, шарів і налаштувань.

### **1.2.3. Реалізація функцій**

 **Інструменти малювання:**

* Олівець, маркер і аерограф реалізовано через Canvas API з підтримкою плавного малювання та градієнтів.
* Гумка видаляє пікселі на активному шарі, використовуючи режим destination-out.
* Заливка (floodFill) заповнює область вибраним кольором або градієнтом.

 **Фігури:**

* Кожна фігура (лінія, прямокутник, зірка тощо) малюється з урахуванням координат початку та кінця, заданих користувачем.
* Попередній перегляд фігур відображається на окремому полотні (previewCanvas).

 **Шари:**

* Створення нового шару додає прозорий канвас із початковими параметрами.
* Об’єднання шарів комбінує видимі шари в один із збереженням їхніх властивостей (прозорість, режим накладання).
* Перетягування шарів реалізовано через Drag-and-Drop API.

 **Інтерфейс:**

* Панель інструментів містить кнопки для вибору інструментів, кольору, товщини та інших налаштувань.
* Панель шарів дозволяє керувати шарами (видимість, прозорість, блокування).
* Масштабування реалізовано через CSS-трансформації (scale), а повноекранний режим — через Fullscreen API.

 **Редагування та збереження:**

* Скасування/повторення дій зберігає стан полотна у стеку (до 30 кроків).
* Фільтри застосовуються до активного шару через маніпуляцію пікселями (imageData).
* Збереження малюнка створює тимчасовий канвас із усіма видимими шарами і конвертує його в обраний формат.

**РОЗДІЛ 2. ПРОЦЕС РОЗРОБКИ ДОДАТКУ**

Процес створення цього веб-додатку для малювання складався з кількох ключових етапів, спрямованих на розробку зручного, функціонального та інтуїтивного інструменту для творчості. Кожен етап був ретельно продуманий і реалізований, щоб забезпечити стабільну роботу програми та відповідність потребам користувачів.

**2.1 Використані технології**

**2.1.1 HTML, CSS та JavaScript**

Основою додатку є стандартні веб-технології:

* **HTML** (index.html) — визначає структуру сторінки, включаючи полотно для малювання, панель інструментів, панель шарів і налаштування.
* **CSS** (styles.css) — відповідає за стилізацію інтерфейсу, адаптивність і підтримку темного режиму.
* **JavaScript** — основна мова програмування, що забезпечує всю логіку роботи додатку, від малювання до управління шарами.

**2.1.2 Canvas API**

* Використовується для рендерингу графіки, малювання фігур, роботи з шарами та застосування фільтрів.
* Надає низькорівневі можливості для маніпуляції пікселями, що необхідно для інструментів, таких як олівець, гумка чи аерограф.

**2.1.3 Модульний підхід**

* Код розбито на окремі модулі (utils.js, tools.js, shapes.js, layers.js), що підвищує читабельність, полегшує підтримку та дозволяє легко додавати нові функції.

**2.2 Структура проекту**

Проект має логічну та модульну структуру:

* **index.html** — точка входу, що містить розмітку інтерфейсу.
* **script.js** — головний файл, який ініціалізує додаток і координує роботу модулів.
* **utils.js** — утилітні функції для обробки кольорів, генерації ID тощо.
* **tools.js** — логіка інструментів малювання (олівець, маркер, гумка, заливка).
* **shapes.js** — функції для створення фігур (лінія, коло, зірка тощо).
* **layers.js** — управління шарами (створення, видалення, налаштування).
* **styles.css** — стилі для всіх елементів інтерфейсу.

**2.3 Етапи розробки**

**2.3.1 Проектування інтерфейсу**

* Розроблено структуру сторінки в index.html, що включає полотно (<canvas>), панель інструментів і панель шарів.
* Стилі в styles.css забезпечують адаптивність і підтримку темного режиму.

**2.3.2 Реалізація логіки малювання**

* У script.js налаштовано обробку подій миші та клавіатури для взаємодії з полотном.
* У tools.js і shapes.js реалізовано інструменти та фігури з урахуванням налаштувань (колір, товщина, прозорість).

**2.3.3 Управління шарами**

* Модуль layers.js забезпечує створення, видалення, об’єднання шарів, а також їх перетягування та налаштування властивостей.

**2.3.4 Оптимізація та стилізація**

* Додаток оптимізовано для швидкої роботи з великою кількістю шарів і складними малюнками.
* Стилі в styles.css адаптовано для різних розмірів екранів.

**2.4 Переваги технологій**

* **HTML, CSS, JavaScript**: Кросплатформність і доступність у браузері без необхідності встановлення.
* **Canvas API**: Висока продуктивність і гнучкість для графічних операцій.
* **Модульність**: Легке розширення та підтримка коду завдяки чіткому поділу на файли.

**2.5 Ключові файли проекту**

**2.5.1 index.html**

* Містить розмітку для полотна малювання, панелі інструментів і шарів.
* Включає елементи управління: вибір кольору, товщини, прозорості, фільтрів, масштабування.

**2.5.2 script.js**

* Ініціалізує додаток, обробляє події (миші, клавіатури) і зв’язує модулі.
* Містить логіку для малювання, масштабування, збереження та скасування дій.

**2.5.3 utils.js**

* Утилітні функції: clamp, getRandomColor, rgbToHex, generateId тощо.
* Використовуються для обробки кольорів і генерації унікальних ідентифікаторів.

**2.5.4 tools.js**

* Логіка інструментів: drawFreehand, drawSpray, erasePixels, floodFill.
* Дозволяє малювати, стирати та заповнювати області.

**2.5.5 shapes.js**

* Функції для фігур: drawLine, drawCircle, drawStar, drawHeart тощо.
* Реалізує малювання з урахуванням поточних налаштувань.

**2.5.6 layers.js**

* Управління шарами: createLayer, removeCurrentLayer, mergeLayers, redrawAllLayers.
* Підтримує налаштування видимості, прозорості та режимів накладання.

**2.5.7 styles.css**

* Стилі для інтерфейсу: полотно, кнопки, панелі, слайдери.
* Забезпечує темний режим і адаптивність.

## **РОЗДІЛ 3. ХІД ВИКОНАННЯ РОБОТИ**

### **MainActivity**

Вхідною точкою в наш додаток є файл script.js. Цей файл відповідає за ініціалізацію основних компонентів програми, таких як полотно для малювання, інструменти, шари та обробники подій. При запуску додатку викликається функція init(), яка налаштовує розміри полотна, створює перший шар, ініціалізує інструменти та забезпечує базову взаємодію з користувачем.

Функція init() встановлює фіксовані розміри полотна (1000x1000 пікселів, масштабовано до 2000x2000 для високої роздільної здатності), створює початковий шар, якщо його немає, і налаштовує обробники подій для малювання, вибору інструментів, зміни кольорів, масштабування тощо.

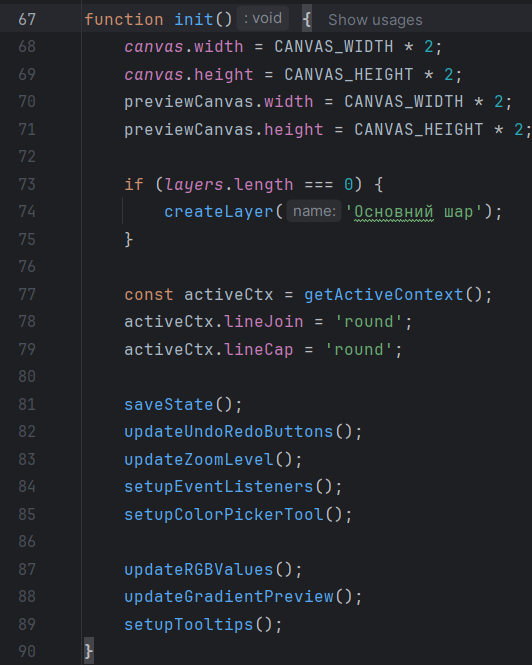


Рисунок 3.1 – Ініціалізація полотна в script.js

### **Layers Management**

Управління шарами реалізовано у файлі layers.js. Цей модуль відповідає за створення, видалення, об’єднання та редагування шарів. Кожен шар представлений окремим елементом <canvas>, що дозволяє ізолювати малюнки та застосовувати до них різні ефекти, такі як прозорість чи режими накладання.

При створенні нового шару функція createLayer() генерує прозорий канвас, додає його до списку шарів і оновлює інтерфейс користувача. Користувач може змінювати порядок шарів за допомогою Drag-and-Drop, видаляти шари, об’єднувати їх або налаштовувати видимість і прозорість.

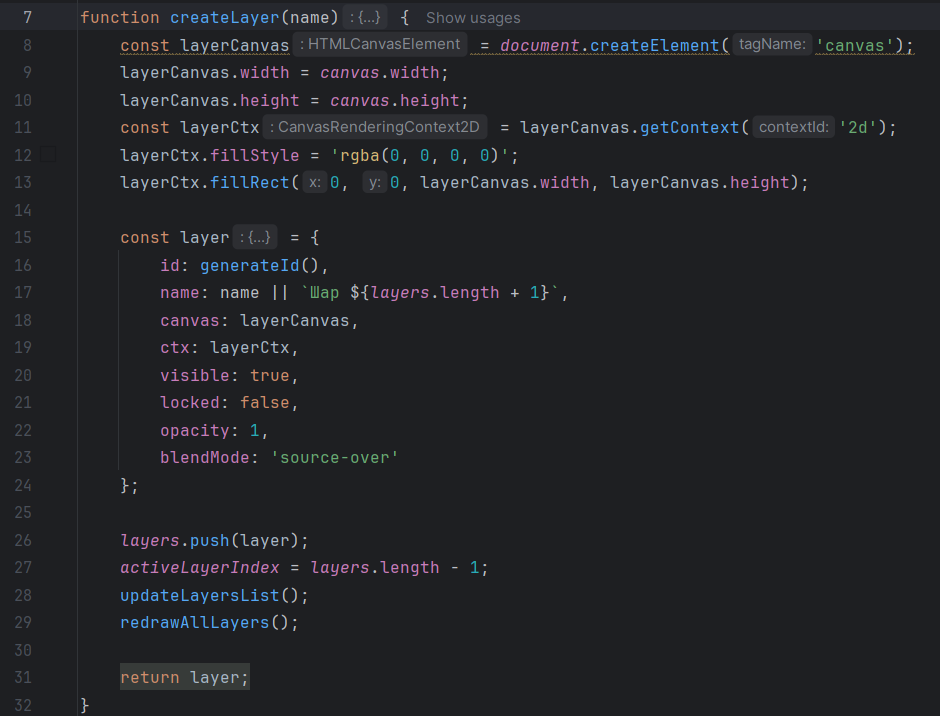


Рисунок 3.2 – Створення нового шару в layers.js

### **Tools Management**

Файл tools.js містить реалізацію основних інструментів малювання, таких як олівець, аерограф, гумка та заливка. Кожен інструмент має власну логіку, яка враховує поточні налаштування кольору, товщини ліній і прозорості.

Наприклад, інструмент заливки (floodFill) використовує алгоритм заповнення для заміни кольору в межах вибраної області. Інструменти зберігають попередні налаштування контексту, щоб забезпечити коректне відновлення стану після малювання.



Рисунок 3.3.1 – Реалізація заливки в tools.js



Рисунок 3.3.2 – Реалізація заливки в tools.js

### **Shapes Management**

Модуль shapes.js відповідає за малювання геометричних фігур, таких як лінії, прямокутники, кола, трикутники, зірки тощо. Кожна функція малює фігуру на основі початкових і кінцевих координат, заданих користувачем, і підтримує заповнення градієнтом або однотонним кольором.



Рисунок 3.4 – Малювання кола в shapes.js

### **Utils Management**

Файл utils.js містить утилітні функції, які використовуються в усьому проекті. Наприклад, функція generateId() створює унікальні ідентифікатори для шарів, а rgbToHex() конвертує RGB-колір у HEX-формат.

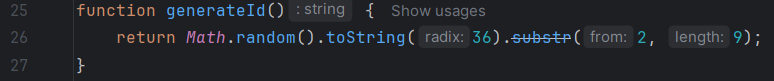


Рисунок 3.5 – Генерація унікального ID в utils.js

### **Styles Management**

Файл styles.css визначає зовнішній вигляд додатку, включаючи підтримку темного режиму, адаптивність і стилі для всіх елементів інтерфейсу. CSS-змінні дозволяють легко перемикатися між світлим і темним режимами, а медіа-запити забезпечують коректне відображення на різних пристроях.



Рисунок 3.6 – Налаштування темного режиму в styles.css

### **User Interface**

Файл index.html містить розмітку інтерфейсу користувача, включаючи полотно, панель інструментів, панель шарів і елементи управління. Панель інструментів дозволяє вибирати інструменти, кольори, товщину ліній, а панель шарів відображає список шарів із можливістю їх редагування.



Рисунок 3.7 – Панель інструментів в index.html

# РОЗДІЛ 4. КЕРІВНИЦТВО КОРИСТУВАЧА

Після запуску веб-додатку користувач зустрічається з інтуїтивним інтерфейсом для малювання. Цей розділ описує основні елементи інтерфейсу та можливості взаємодії з додатком.

## **4.1 Головний екран додатку**

Після завантаження сторінки користувач бачить головний екран із полотном для малювання, панеллю інструментів у верхній частині та панеллю шарів (за замовчуванням прихована). Полотно має фіксований розмір 1000x1000 пікселів і білий фон. Панель інструментів містить кнопки для вибору інструментів (олівець, ручка, гумка, заливка, аерограф, фігури), налаштування кольору, товщини ліній, прозорості, форми пензля, а також функції для збереження, скасування дій, масштабування та перемикання темного режиму.

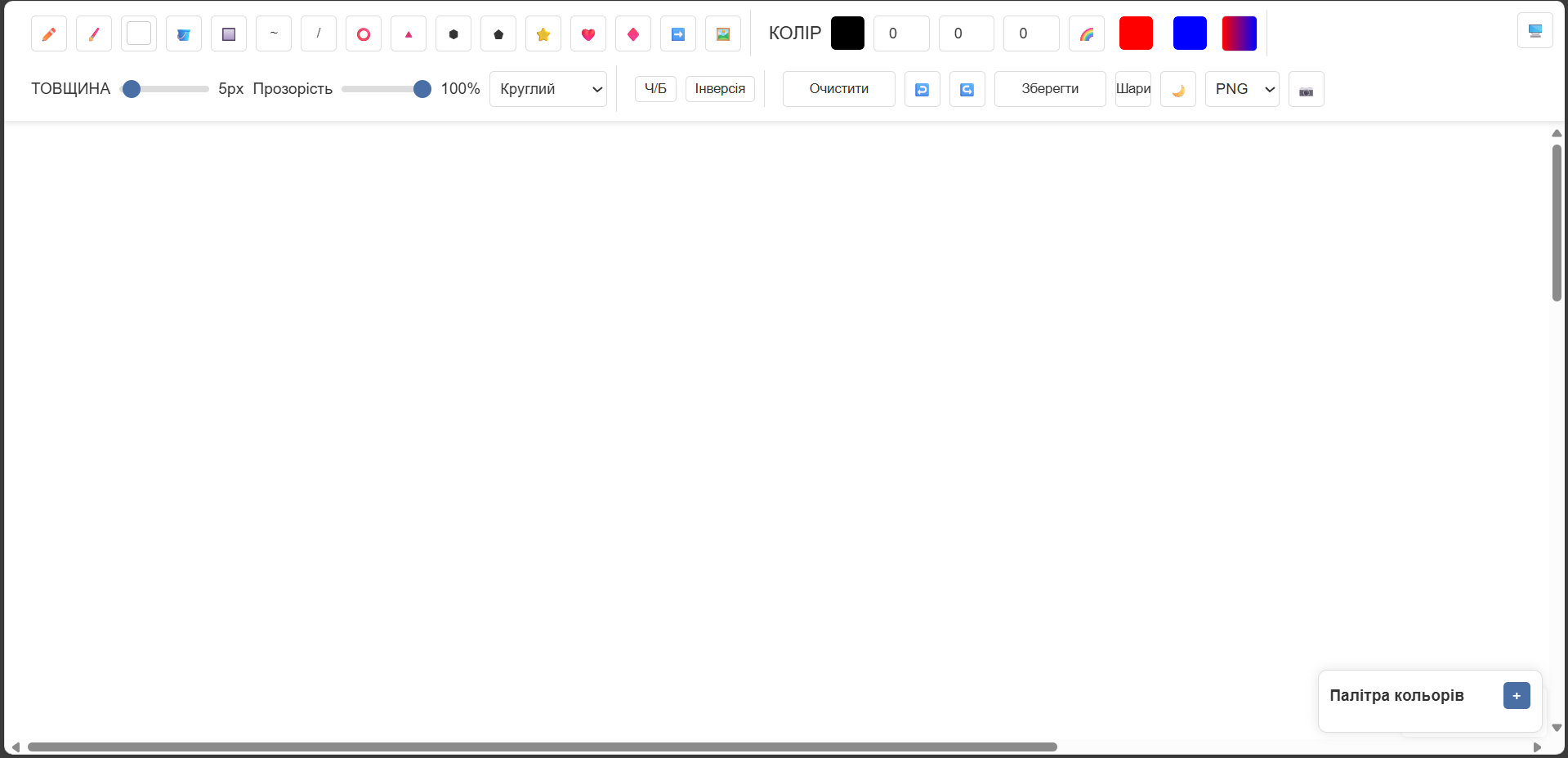


Рисунок 4.1 – Головний екран додатку

## **4.2 Використання інструментів малювання**

### **4.2.1 Малювання вільною рукою**

Користувач може обрати інструмент "Олівець", "Ручка" або "Маркер" для малювання вільною рукою. Після вибору інструменту (наприклад, натиснувши на кнопку з іконкою ✏️) користувач може малювати, утримуючи ліву кнопку миші та переміщаючи курсор по полотну. Колір, товщина ліній і прозорість налаштовуються в панелі інструментів.

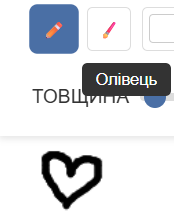
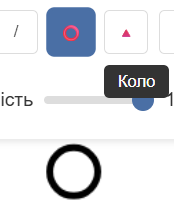


Рисунок 4.2 – Використання інструменту "Олівець"

### **4.2.2 Малювання фігур**

Для створення геометричних фігур користувач обирає відповідний інструмент (лінія, прямокутник, коло, трикутник, шестикутник, п’ятикутник, зірка, серце, діамант, стрілка). Після вибору інструменту, наприклад, "Коло" (іконка ⭕), користувач клацає на полотні, утримує мишу і перетягує, щоб задати розмір фігури. Попередній перегляд відображається на полотні в реальному часі, а після відпускання кнопки миші фігура фіксується на активному шарі.

****

**Рисунок 4.3 – Малювання кола**

### **4.2.3 Використання заливки та аерографа**

Інструмент "Заливка" (іконка 🪣) дозволяє заповнити область одним кольором або градієнтом. Користувач клацає на потрібній області, і вона заповнюється вибраним кольором. Інструмент "Розпилювач" (іконка 🖼️) створює ефект розпилення фарби, коли користувач утримує кнопку миші.

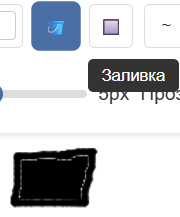


Рисунок 4.4 – Використання інструменту "Заливка"

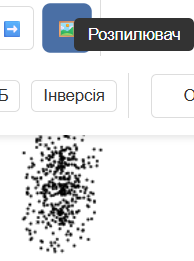


Рисунок 4.4.1– Використання інструменту "Розпилювач"

## **4.3 Управління шарами**

Користувач може відкрити панель шарів, натиснувши кнопку "Шари" у панелі інструментів. Панель шарів відображає список усіх шарів, дозволяючи:

* Створювати нові шари (кнопка "+").
* Видаляти активний шар (кнопка "−").
* Об’єднувати шари (кнопка "Об’єднати").
* Змінювати видимість, прозорість, режим накладання та назву шару.
* Перетягувати шари для зміни їх порядку.

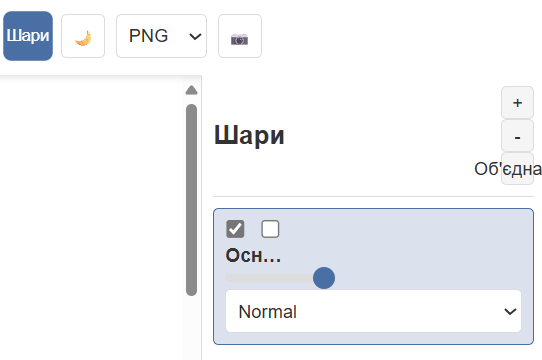
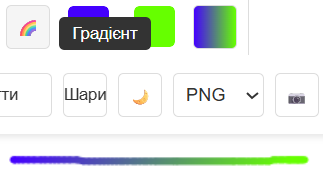


Рисунок 4.5 – Панель шарів

## **4.4 Налаштування кольору та градієнта**

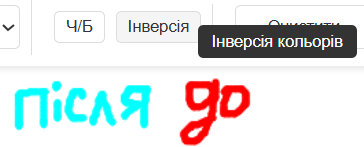
Користувач може вибрати колір за допомогою кольорового пікера або ввести значення RGB у відповідні поля. Для використання градієнта потрібно активувати режим градієнта (кнопка 🌈), після чого вибрати два кольори для градієнта. Градієнт застосовується до інструментів малювання або заливки.



**Рисунок 4.6 – Налаштування градієнта**

## **4.5 Застосування фільтрів**

Додаток підтримує фільтри "Ч/Б" (чорно-білий) та "Інверсія". Користувач може застосувати ці фільтри до активного шару, натиснувши відповідні кнопки в панелі інструментів. Фільтри змінюють кольори шару, зберігаючи його вміст.

****

**Рисунок 4.7 – Застосування фільтра "Інверсія"**

## **4.6 Збереження та скасування дій**

Користувач може скасувати (↩️) або повторити (↪️) дії за допомогою відповідних кнопок. Максимальна кількість кроків для скасування — 30. Для збереження малюнка користувач обирає формат (PNG, JPEG, SVG) у випадаючому меню та натискає кнопку "Зберегти". Малюнок завантажується як файл.

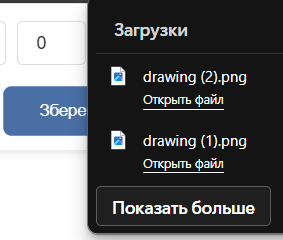
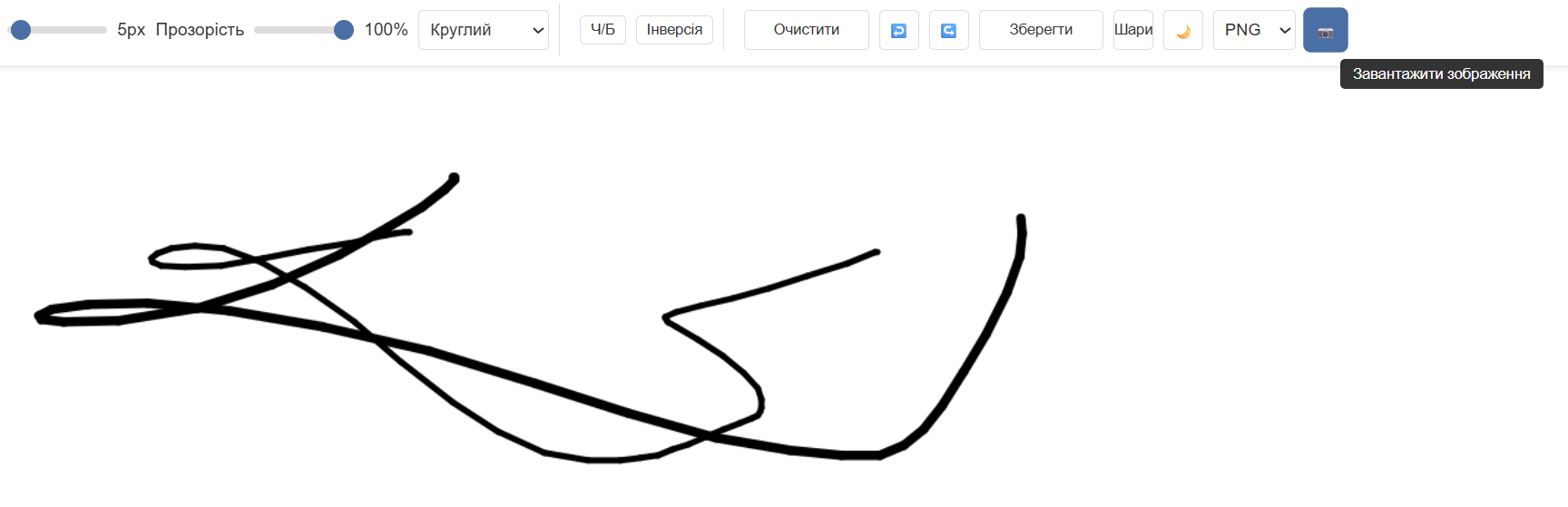


Рисунок 4.8 – Збереження малюнка

## **4.7 Завантаження зображень**

Користувач може завантажити зображення, натиснувши кнопку "📷". Зображення автоматично масштабується до розмірів полотна та додається на новий шар із назвою "Завантажене зображення". Також підтримується вставка зображень із буфера обміну (Ctrl+V).

****

**Рисунок 4.9 – Завантаження зображення**

# ВИСНОВКИ

У ході виконання цієї курсової роботи було створено веб-додаток для малювання, який є значним внеском у сферу цифрових інструментів для творчості. Використання сучасних веб-технологій, таких як HTML, CSS, JavaScript та Canvas API, дозволило реалізувати функціональний, інтуїтивний та адаптивний інтерфейс, доступний у будь-якому сучасному браузері без необхідності встановлення додаткового програмного забезпечення.

Ключовим досягненням проекту є успішне вирішення завдань, пов’язаних із створенням інструментів для малювання, управлінням шарами, застосуванням фільтрів, підтримкою градієнтів та збереженням малюнків у різних форматах (PNG, JPEG, SVG). Додаток також підтримує такі функції, як скасування/повторення дій, масштабування полотна, темний режим і повноекранний режим, що підвищує зручність використання.

Проект є актуальним у сучасному цифровому світі, де інструменти для графічного дизайну та малювання набувають дедалі більшої популярності. Реалізований додаток надає користувачам зручний і гнучкий інструмент для створення цифрових малюнків, який може бути використаний як для особистих творчих проєктів, так і для професійних цілей.

Розробка цього додатку є важливим кроком у напрямку вдосконалення веб-технологій для творчих застосунків. Проект відкриває перспективи для подальшого розвитку, зокрема додавання нових інструментів, інтеграції з хмарними сервісами для збереження малюнків або реалізації функцій спільного редагування.

Ось деякі конкретні досягнення, яких вдалося досягти в рамках цього проекту:

* **Інтуїтивний та адаптивний інтерфейс**: Додаток має простий і зрозумілий інтерфейс із підтримкою темного режиму та адаптивним дизайном, що забезпечує комфортну роботу на різних пристроях, від настільних комп’ютерів до планшетів.
* **Розширені функції малювання**: Реалізовано широкий набір інструментів (олівець, ручка, гумка, заливка, аерограф, геометричні фігури), підтримку градієнтів, фільтрів (чорно-білий, інверсія) та багатошарове малювання, що дозволяє створювати складні композиції.
* **Гнучке управління шарами**: Користувачі можуть створювати, видаляти, об’єднувати шари, змінювати їх порядок, видимість, прозорість і режими накладання, що забезпечує професійний рівень роботи з малюнками.
* **Експорт і завантаження**: Додаток підтримує збереження малюнків у популярних форматах і можливість завантаження зображень для подальшого редагування.

Загалом, розробка цього веб-додатку була успішною. Додаток є функціональним, доступним і зручним для широкого кола користувачів, від початківців до досвідчених художників. Він має потенціал стати цінним інструментом для цифрового мистецтва та відкриває можливості для подальшого розширення функціональності.

# CПИCOК ВИКOРИCТAНИX ДЖEРEЛ

1. MDN Web Docs: HTML [Електронний ресурс] – Режим доступу до ресурсу: <https://developer.mozilla.org/en-US/docs/Web/HTML>
2. MDN Web Docs: CSS [Електронний ресурс] – Режим доступу до ресурсу: <https://developer.mozilla.org/en-US/docs/Web/CSS>
3. MDN Web Docs: JavaScript [Електронний ресурс] – Режим доступу до ресурсу: <https://developer.mozilla.org/en-US/docs/Web/JavaScript>
4. MDN Web Docs: Canvas API [Електронний ресурс] – Режим доступу до ресурсу: <https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API>
5. Web.dev: Responsive Web Design [Електронний ресурс] – Режим доступу до ресурсу: <https://web.dev/learn/design/responsive-web-design>
6. W3C: Drag and Drop API [Електронний ресурс] – Режим доступу до ресурсу: [https://www.w3.org/TR/html52/editing.html#drag-and-drop](https://www.w3.org/TR/html52/editing.html%23drag-and-drop)
7. JavaScript.info: Event Listeners [Електронний ресурс] – Режим доступу до ресурсу: <https://javascript.info/event-listeners>

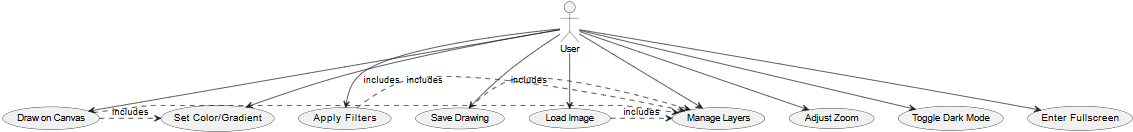
# ДОДАТОК 1

Діаграми

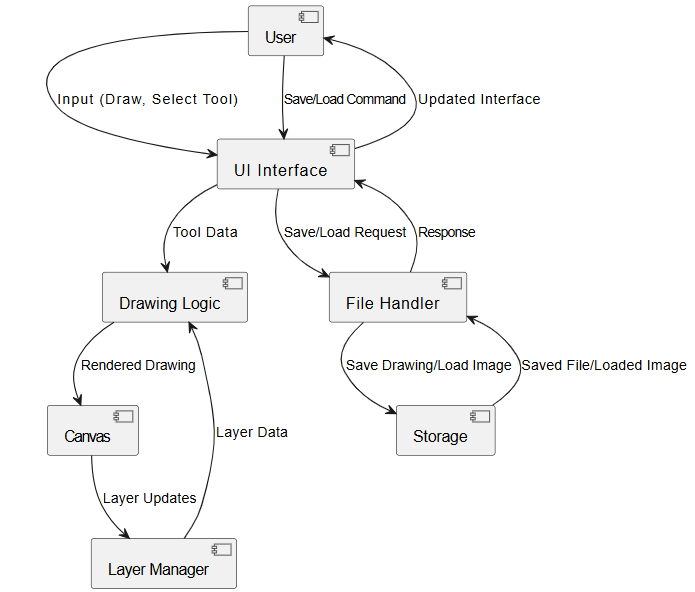
НТУУ «КПІ ім. І. Сікорського» ТЕФ АПЕПС ТВ-32

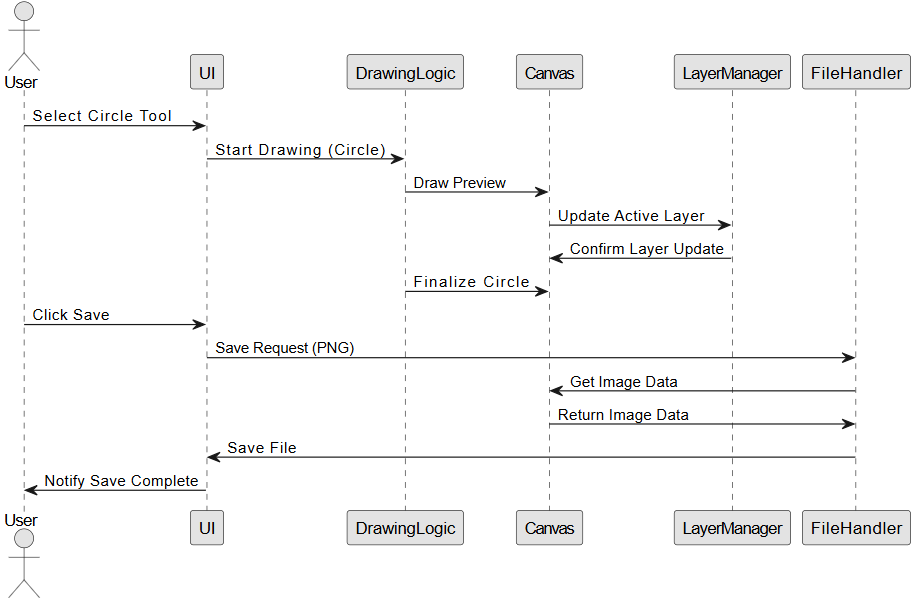
Листів 3

Київ – 2024

**Use case diagram**

**Dfd diagram**



**Sequence diagram**

# ДОДАТОК 2

Опис програми

НТУУ «КПІ ім. І. Сікорського» ТЕФ АПЕПС ТВ-32

Листів 2

Київ – 2024

У ході виконання цієї курсової роботи було створено веб-додаток для малювання, який включає широкий набір інструментів та функцій для створення цифрових малюнків. Програма реалізована за допомогою сучасних веб-технологій, таких як HTML, CSS, JavaScript та Canvas API.

Метою цього курсового проекту було розробити веб-додаток для малювання, який відповідатиме потребам сучасних користувачів у сфері цифрового мистецтва. Додаток створено з використанням веб-стандартів і не потребує встановлення, що робить його доступним через будь-який браузер.

Основні завдання додатку включали в себе:

* **Зручність використання**: Додаток має інтуїтивно зрозумілий інтерфейс, який дозволяє легко освоїти його навіть новачкам.
* **Функціональність**: Додаток пропонує широкий спектр інструментів для малювання, управління шарами та редагування зображень.
* **Надійність**: Програма протестована на сумісність із сучасними браузерами та не містить критичних помилок.

Актуальність даної теми зумовлена зростаючою популярністю цифрових інструментів для малювання та графічного дизайну. Такі додатки стають незамінними для художників, дизайнерів та ентузіастів, які прагнуть створювати мистецтво в цифровому середовищі. Розробка нового веб-додатку для малювання, що відповідає сучасним вимогам, є важливим внеском у розвиток цієї галузі.

Уся програма була реалізована за допомогою HTML для структури, CSS для стилізації та JavaScript для логіки, з використанням інтегрованого середовища розробки, такого як Visual Studio Code, і тестуванням у браузерах (Chrome, Firefox, Edge).

Конкретні функціональні можливості додатку:

* **Створення та редагування малюнків**: Користувачі можуть малювати за допомогою інструментів (олівець, ручка, геометричні фігури, заливка, аерограф), застосовувати фільтри та редагувати шари.
* **Управління шарами**: Додаток дозволяє створювати, видаляти, об’єднувати шари, змінювати їх видимість, прозорість і порядок.
* **Експорт і завантаження**: Користувачі можуть зберігати малюнки у форматах PNG, JPEG, SVG та завантажувати зображення для подальшого редагування.

Очікувані результати:  
Після завершення розробки додаток відповідає наступним вимогам:

* **Простота у використанні**: Інтерфейс є інтуїтивним, із підтримкою темного режиму, масштабування та повноекранного режиму.
* **Широкі можливості**: Додаток забезпечує різноманітні інструменти для малювання, градієнтів, фільтрів і багатошарового редагування.
* **Надійність**: Програма працює стабільно в сучасних браузерах без значних збоїв.

Розробка веб-додатку для малювання є важливим завданням, яке має потенціал для вдосконалення творчого процесу сучасних користувачів. Додаток, створений у рамках цього курсового проекту, відповідає потребам цифрових художників і забезпечує широкий спектр функціональних можливостей для створення та редагування малюнків.

# ДОДАТОК 3

Код програми

НТУУ «КПІ ім. І. Сікорського» ТЕФ АПЕПС ТВ-32

Листів 37

Київ – 2024

**index.html**

<!DOCTYPE html>

<html lang="uk">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Редактор Малювання</title>

<link rel="stylesheet" href="styles.css">

</head>

<body>

<div class="app-container">

<header class="app-header">

<div class="toolbar">

<div class="tool-group">

<button class="tool-btn" data-tool="pencil" title="Олівець">✏️</button>

<button class="tool-btn" data-tool="pen" title="Ручка">🖌️</button>

<button class="tool-btn" data-tool="eraser" title="Гумка">🧽</button>

<button class="tool-btn" data-tool="fill" title="Заливка">🪣</button>

<button class="tool-btn" data-tool="rectangle" title="Прямокутник">🔲</button>

<button class="tool-btn" data-tool="curve" title="Крива">~</button>

<button class="tool-btn" data-tool="line" title="Лінія">/</button>

<button class="tool-btn" data-tool="circle" title="Коло">⭕</button>

<button class="tool-btn" data-tool="triangle" title="Трикутник">🔺</button>

<button class="tool-btn" data-tool="hexagon" title="Шестикутник">⬢</button>

<button class="tool-btn" data-tool="pentagon" title="П’ятикутник">⬟</button>

<button class="tool-btn" data-tool="star" title="Зірка">⭐</button>

<button class="tool-btn" data-tool="heart" title="Серце">❤️</button>

<button class="tool-btn" data-tool="diamond" title="Діамант">♦️</button>

<button class="tool-btn" data-tool="arrow" title="Стрілка">➡️</button>

<button class="tool-btn" data-tool="spray" title="Розпилювач">🖼️</button>

</div>

<div class="tool-group">

<label>КОЛІР</label>

<input type="color" id="color-picker" value="#000000">

<div class="rgb-picker">

<input type="number" id="r-value" min="0" max="255" value="0">

<input type="number" id="g-value" min="0" max="255" value="0">

<input type="number" id="b-value" min="0" max="255" value="0">

</div>

<button class="tool-btn" id="gradient-btn" title="Градієнт">🌈</button>

<input type="color" id="gradient-color1" class="gradient-color-picker" value="#FF0000">

<input type="color" id="gradient-color2" class="gradient-color-picker" value="#0000FF">

<div id="gradient-preview" class="gradient-preview"></div>

</div>

<div class="tool-group">

<div class="slider-container">

<label>ТОВЩИНА</label>

<input type="range" id="line-width" min="1" max="100" value="5">

<span id="width-value">5px</span>

</div>

<div class="slider-container">

<label>Прозорість</label>

<input type="range" id="opacity-slider" min="0" max="1" step="0.01" value="1">

<span id="opacity-value">100%</span>

</div>

<select id="brush-shape">

<option value="round">Круглий</option>

<option value="square">Квадратний</option>

</select>

</div>

<div class="tool-group filter-controls">

<button class="filter-btn" id="grayscale-btn" title="Чорно-білий">Ч/Б</button>

<button class="filter-btn" id="invert-btn" title="Інверсія кольорів">Інверсія</button>

</div>

<div class="tool-group">

<button class="tool-btn wide-btn" id="clear-btn" title="Очистити">Очистити</button>

<button class="tool-btn" id="undo-btn" title="Скасувати">↩️</button>

<button class="tool-btn" id="redo-btn" title="Повернути">↪️</button>

<button class="tool-btn wide-btn" id="save-btn" title="Зберегти">Зберегти</button>

<button class="tool-btn" id="layers-btn" title="Шари">Шари</button>

<button class="tool-btn" id="dark-theme-btn" title="Темна тема">🌙</button>

<select id="save-format">

<option value="png">PNG</option>

<option value="jpeg">JPEG</option>

<option value="svg">SVG</option>

</select>

<input type="file" id="load-image-btn" accept="image/\*" style="display: none;">

<button class="tool-btn" onclick="document.getElementById('load-image-btn').click();" title="Завантажити зображення">📷</button>

</div>

</div>

</header>

<div class="main-content">

<div id="canvas-container" class="canvas-container">

<canvas id="drawing-canvas"></canvas>

<canvas id="preview-canvas" class="preview-canvas"></canvas>

</div>

<div class="layers-panel" id="layers-panel">

<div class="layers-header">

<h3>Шари</h3>

<div>

<button id="add-layer" title="Додати шар">+</button>

<button id="remove-layer" title="Видалити шар">-</button>

<button id="merge-layers" title="Об'єднати шари">Об'єднати</button>

</div>

</div>

<div id="layers-list" class="layers-list"></div>

</div>

</div>

<div class="zoom-controls">

<button id="zoom-out">-</button>

<span id="zoom-level">100%</span>

<button id="zoom-in">+</button>

</div>

<div class="color-swatches-panel" id="color-swatches">

<div class="swatches-header">

<h4>Палітра кольорів</h4>

<button id="add-swatch" title="Додати колір">+</button>

</div>

<div id="swatches-grid" class="swatches-grid"></div>

</div>

<button class="tool-btn fullscreen-btn" id="fullscreen-btn" title="Повноекранний режим">🖥️</button>

</div>

<script type="module" src="script.js"></script>

</body>

</html>

**script.js**

import { clamp, getRandomColor, rgbToHex, interpolateColor, colorToRgb, generateId } from './utils.js';  
import { drawFreehand, drawSpray, drawMarker, erasePixels, floodFill, applyFilter } from './tools.js';  
import { drawLine, drawRectangle, drawCircle, drawTriangle, drawHexagon, drawPentagon, drawStar, drawHeart, drawDiamond, drawArrow, drawCurve } from './shapes.js';  
import { *layers*, *activeLayerIndex*, createLayer, removeCurrentLayer, addNewLayer, updateLayersList, setActiveLayer, redrawAllLayers, getActiveContext, getActiveCanvas, drawOnActiveLayer, mergeLayers } from './layers.js';  
  
export const *canvas* = *document*.getElementById('drawing-canvas');  
export const *ctx* = *canvas*.getContext('2d');  
const previewCanvas = *document*.getElementById('preview-canvas');  
const previewCtx = previewCanvas.getContext('2d');  
const colorPicker = *document*.getElementById('color-picker');  
const lineWidth = *document*.getElementById('line-width');  
const widthValue = *document*.getElementById('width-value');  
const opacitySlider = *document*.getElementById('opacity-slider');  
const opacityValue = *document*.getElementById('opacity-value');  
const clearBtn = *document*.getElementById('clear-btn');  
const undoBtn = *document*.getElementById('undo-btn');  
const redoBtn = *document*.getElementById('redo-btn');  
const saveBtn = *document*.getElementById('save-btn');  
const layersBtn = *document*.getElementById('layers-btn');  
const addLayerBtn = *document*.getElementById('add-layer');  
const removeLayerBtn = *document*.getElementById('remove-layer');  
const mergeLayersBtn = *document*.getElementById('merge-layers');  
const layersPanel = *document*.getElementById('layers-panel');  
const layersList = *document*.getElementById('layers-list');  
const zoomInBtn = *document*.getElementById('zoom-in');  
const zoomOutBtn = *document*.getElementById('zoom-out');  
const zoomLevel = *document*.getElementById('zoom-level');  
const saveFormat = *document*.getElementById('save-format');  
const toolButtons = *document*.querySelectorAll('.tool-btn');  
const rValue = *document*.getElementById('r-value');  
const gValue = *document*.getElementById('g-value');  
const bValue = *document*.getElementById('b-value');  
const gradientBtn = *document*.getElementById('gradient-btn');  
const gradientColor1 = *document*.getElementById('gradient-color1');  
const gradientColor2 = *document*.getElementById('gradient-color2');  
const gradientPreview = *document*.getElementById('gradient-preview');  
const grayscaleBtn = *document*.getElementById('grayscale-btn');  
const invertBtn = *document*.getElementById('invert-btn');  
const darkThemeBtn = *document*.getElementById('dark-theme-btn');  
const colorSwatches = *document*.getElementById('color-swatches');  
const loadImageBtn = *document*.getElementById('load-image-btn');  
const brushShape = *document*.getElementById('brush-shape');  
const fullscreenBtn = *document*.getElementById('fullscreen-btn');  
  
let currentTool = 'pencil';  
let isDrawing = false;  
let startX = 0;  
let startY = 0;  
let currentColor = '#000000';  
let currentOpacity = 1;  
let currentLineWidth = 5;  
let undoStack = [];  
let redoStack = [];  
let maxUndoSteps = 30;  
let zoom = 1;  
let isGradientMode = false;  
let lastX = 0;  
let lastY = 0;  
let curvePoints = [];  
let sprayInterval = null;  
let currentBrushShape = 'round';  
  
// Фиксированные размеры холста  
const CANVAS\_WIDTH = 1000;  
const CANVAS\_HEIGHT = 1000;  
  
function init() {  
 *canvas*.width = CANVAS\_WIDTH \* 2;  
 *canvas*.height = CANVAS\_HEIGHT \* 2;  
 previewCanvas.width = CANVAS\_WIDTH \* 2;  
 previewCanvas.height = CANVAS\_HEIGHT \* 2;  
  
 if (*layers*.length === 0) {  
 createLayer('Основний шар');  
 }  
  
 const activeCtx = getActiveContext();  
 activeCtx.lineJoin = 'round';  
 activeCtx.lineCap = 'round';  
  
 saveState();  
 updateUndoRedoButtons();  
 updateZoomLevel();  
 setupEventListeners();  
 setupColorPickerTool();  
  
 updateRGBValues();  
 updateGradientPreview();  
 setupTooltips();  
}  
  
function setupTooltips() {  
 const tooltips = *document*.querySelectorAll('[title]');  
 tooltips.forEach(tool => {  
 tool.addEventListener('mouseenter', (e) => {  
 const tooltip = *document*.createElement('div');  
 tooltip.className = 'custom-tooltip';  
 tooltip.textContent = tool.title;  
 tooltip.style.left = `${e.clientX + 10}px`;  
 tooltip.style.top = `${e.clientY + 10}px`;  
 *document*.body.appendChild(tooltip);  
  
 tool.dataset.tooltipId = tooltip.textContent;  
 tool.addEventListener('mouseleave', () => {  
 tooltip.remove();  
 });  
 });  
 });  
}  
  
function getCanvasCoordinates(e) {  
 const rect = *canvas*.getBoundingClientRect();  
 const scaleX = *canvas*.width / rect.width;  
 const scaleY = *canvas*.height / rect.height;  
 return {  
 x: (e.clientX - rect.left) \* scaleX,  
 y: (e.clientY - rect.top) \* scaleY  
 };  
}  
  
function startDrawing(e) {  
 if (e.target !== *canvas*) return;  
  
 const { x, y } = getCanvasCoordinates(e);  
 startX = x;  
 startY = y;  
 lastX = startX;  
 lastY = startY;  
  
 isDrawing = true;  
  
 if (currentTool === 'fill') {  
 const fillColor = isGradientMode ?  
 getGradientColorAtPoint(startX, startY, startX, startY, gradientColor1.value, gradientColor2.value) :  
 currentColor;  
  
 floodFill(startX, startY, fillColor, getActiveContext());  
 redrawAllLayers();  
 saveState();  
 isDrawing = false;  
 } else if (currentTool === 'spray') {  
 sprayInterval = setInterval(() => {  
 const sprayColor = isGradientMode ?  
 getGradientColorAtPoint(lastX, lastY, startX, startY, gradientColor1.value, gradientColor2.value) :  
 currentColor;  
 drawSpray(lastX, lastY, currentLineWidth, 5, sprayColor, getActiveContext());  
 redrawAllLayers();  
 }, 30);  
 } else if (currentTool === 'eraser') {  
 const activeCtx = getActiveContext();  
 activeCtx.globalCompositeOperation = 'destination-out';  
 activeCtx.strokeStyle = 'rgba(0,0,0,1)';  
 activeCtx.lineWidth = currentLineWidth;  
 activeCtx.lineJoin = currentBrushShape;  
 activeCtx.lineCap = currentBrushShape;  
 activeCtx.beginPath();  
 activeCtx.moveTo(startX, startY);  
 } else if (['pencil', 'pen', 'marker'].includes(currentTool)) {  
 const activeCtx = getActiveContext();  
 activeCtx.beginPath();  
 activeCtx.moveTo(startX, startY);  
 activeCtx.strokeStyle = isGradientMode ?  
 createGradient(startX, startY, startX + 100, startY + 100, activeCtx) :  
 currentColor;  
 activeCtx.lineWidth = currentTool === 'pencil' ? currentLineWidth :  
 currentTool === 'pen' ? currentLineWidth \* 1.5 : currentLineWidth;  
 activeCtx.globalAlpha = currentOpacity;  
 activeCtx.lineJoin = currentBrushShape;  
 activeCtx.lineCap = currentBrushShape;  
 } else if (currentTool === 'curve') {  
 curvePoints = [{ x: startX, y: startY }];  
 }  
}  
  
function draw(e) {  
 if (!isDrawing) return;  
  
 const { x, y } = getCanvasCoordinates(e);  
 previewCtx.clearRect(0, 0, previewCanvas.width, previewCanvas.height);  
  
 if (currentTool === 'marker') {  
 drawMarker(lastX, lastY, x, y, currentLineWidth, currentColor, currentOpacity, getActiveContext());  
 lastX = x;  
 lastY = y;  
 redrawAllLayers();  
 } else if (currentTool === 'spray') {  
 lastX = x;  
 lastY = y;  
 } else if (['pencil', 'pen', 'eraser'].includes(currentTool)) {  
 const activeCtx = getActiveContext();  
 activeCtx.lineTo(x, y);  
 activeCtx.stroke();  
 activeCtx.beginPath();  
 activeCtx.moveTo(x, y);  
 lastX = x;  
 lastY = y;  
 redrawAllLayers();  
 } else if (currentTool === 'curve') {  
 if (curvePoints.length > 0) {  
 previewCtx.strokeStyle = currentColor;  
 previewCtx.lineWidth = currentLineWidth;  
 previewCtx.globalAlpha = currentOpacity;  
 previewCtx.lineJoin = currentBrushShape;  
 previewCtx.lineCap = currentBrushShape;  
  
 previewCtx.beginPath();  
 previewCtx.moveTo(curvePoints[0].x, curvePoints[0].y);  
 for (let i = 1; i < curvePoints.length; i++) {  
 previewCtx.lineTo(curvePoints[i].x, curvePoints[i].y);  
 }  
 previewCtx.lineTo(x, y);  
 previewCtx.stroke();  
 }  
 } else {  
 previewCtx.strokeStyle = isGradientMode ? createGradient(startX, startY, x, y, previewCtx) : currentColor;  
 previewCtx.fillStyle = isGradientMode ? createGradient(startX, startY, x, y, previewCtx) : currentColor;  
 previewCtx.lineWidth = currentLineWidth;  
 previewCtx.globalAlpha = currentOpacity;  
 previewCtx.lineJoin = currentBrushShape;  
 previewCtx.lineCap = currentBrushShape;  
  
 switch (currentTool) {  
 case 'line': drawLine(startX, startY, x, y, previewCtx); break;  
 case 'rectangle': drawRectangle(startX, startY, x, y, false, previewCtx); break;  
 case 'circle': drawCircle(startX, startY, x, y, false, previewCtx); break;  
 case 'triangle': drawTriangle(startX, startY, x, y, false, previewCtx); break;  
 case 'hexagon': drawHexagon(startX, startY, x, y, previewCtx); break;  
 case 'pentagon': drawPentagon(startX, startY, x, y, previewCtx); break;  
 case 'star': drawStar(startX, startY, x, y, previewCtx); break;  
 case 'heart': drawHeart(startX, startY, x, y, previewCtx); break;  
 case 'diamond': drawDiamond(startX, startY, x, y, previewCtx); break;  
 case 'arrow': drawArrow(startX, startY, x, y, previewCtx); break;  
 }  
 }  
}  
  
function stopDrawing(e) {  
 if (!isDrawing) return;  
  
 isDrawing = false;  
 previewCtx.clearRect(0, 0, previewCanvas.width, previewCanvas.height);  
  
 if (currentTool === 'spray') {  
 clearInterval(sprayInterval);  
 sprayInterval = null;  
 saveState();  
 return;  
 }  
  
 const { x, y } = getCanvasCoordinates(e);  
  
 if (currentTool === 'curve') {  
 curvePoints.push({ x, y });  
 if (curvePoints.length > 1) {  
 drawOnActiveLayer(ctx => drawCurve(curvePoints, ctx));  
 saveState();  
 }  
 } else if (!['pencil', 'pen', 'marker', 'spray', 'eraser'].includes(currentTool)) {  
 const activeCtx = getActiveContext();  
 activeCtx.strokeStyle = isGradientMode ? createGradient(startX, startY, x, y, activeCtx) : currentColor;  
 activeCtx.fillStyle = isGradientMode ? createGradient(startX, startY, x, y, activeCtx) : currentColor;  
 activeCtx.lineWidth = currentLineWidth;  
 activeCtx.globalAlpha = currentOpacity;  
 activeCtx.lineJoin = currentBrushShape;  
 activeCtx.lineCap = currentBrushShape;  
  
 switch (currentTool) {  
 case 'line': drawOnActiveLayer(ctx => drawLine(startX, startY, x, y, ctx)); break;  
 case 'rectangle': drawOnActiveLayer(ctx => drawRectangle(startX, startY, x, y, false, ctx)); break;  
 case 'circle': drawOnActiveLayer(ctx => drawCircle(startX, startY, x, y, false, ctx)); break;  
 case 'triangle': drawOnActiveLayer(ctx => drawTriangle(startX, startY, x, y, false, ctx)); break;  
 case 'hexagon': drawOnActiveLayer(ctx => drawHexagon(startX, startY, x, y, ctx)); break;  
 case 'pentagon': drawOnActiveLayer(ctx => drawPentagon(startX, startY, x, y, ctx)); break;  
 case 'star': drawOnActiveLayer(ctx => drawStar(startX, startY, x, y, ctx)); break;  
 case 'heart': drawOnActiveLayer(ctx => drawHeart(startX, startY, x, y, ctx)); break;  
 case 'diamond': drawOnActiveLayer(ctx => drawDiamond(startX, startY, x, y, ctx)); break;  
 case 'arrow': drawOnActiveLayer(ctx => drawArrow(startX, startY, x, y, ctx)); break;  
 }  
 saveState();  
 } else if (currentTool === 'eraser') {  
 const activeCtx = getActiveContext();  
 activeCtx.globalCompositeOperation = 'source-over';  
 redrawAllLayers();  
 saveState();  
 } else {  
 redrawAllLayers();  
 saveState();  
 }  
  
 if (currentTool !== 'curve') curvePoints = [];  
}  
  
function createGradient(x1, y1, x2, y2, context) {  
 const gradient = context.createLinearGradient(x1, y1, x2, y2);  
 gradient.addColorStop(0, gradientColor1.value);  
 gradient.addColorStop(1, gradientColor2.value);  
 return gradient;  
}  
  
function getGradientColorAtPoint(x, y, x1, y1, color1, color2) {  
 const distance = *Math*.sqrt(*Math*.pow(x - x1, 2) + *Math*.pow(y - y1, 2));  
 const maxDistance = *Math*.sqrt(*Math*.pow(*canvas*.width, 2) + *Math*.pow(*canvas*.height, 2));  
 const ratio = distance / maxDistance;  
 return interpolateColor(color1, color2, ratio);  
}  
  
function handleKeyDown(e) {  
 if (e.ctrlKey) {  
 if (e.key === 'z') {  
 e.preventDefault();  
 undo();  
 } else if (e.key === 'y') {  
 e.preventDefault();  
 redo();  
 }  
 }  
  
 if (e.ctrlKey && (e.key === '+' || e.key === '=')) {  
 e.preventDefault();  
 changeZoom(0.1);  
 }  
 if (e.ctrlKey && e.key === '-') {  
 e.preventDefault();  
 changeZoom(-0.1);  
 }  
 if (e.ctrlKey && e.key === 'v') handlePasteImage(e);  
}  
  
function handleWheel(e) {  
 if (e.ctrlKey) {  
 e.preventDefault();  
 const delta = e.deltaY > 0 ? -0.1 : 0.1;  
 changeZoom(delta);  
 }  
}  
  
export function saveState() {  
 if (undoStack.length >= maxUndoSteps) undoStack.shift();  
 const state = *canvas*.toDataURL();  
 undoStack.push(state);  
 redoStack = []; // Очищаємо redo при новому дії  
 updateUndoRedoButtons();  
}  
  
function undo() {  
 if (undoStack.length > 1) {  
 const currentState = undoStack.pop();  
 redoStack.push(currentState);  
 const previousState = undoStack[undoStack.length - 1];  
 updateUndoRedoButtons();  
 restoreState(previousState);  
 saveState();  
 }  
}  
  
function redo() {  
 if (redoStack.length > 0) {  
 const nextState = redoStack.pop();  
 undoStack.push(nextState);  
 restoreState(nextState);  
 updateUndoRedoButtons();  
 }  
}  
  
function restoreState(state) {  
 const img = new Image();  
 img.onload = function() {  
 *ctx*.clearRect(0, 0, *canvas*.width, *canvas*.height);  
 *ctx*.drawImage(img, 0, 0);  
 };  
 img.src = state;  
}  
  
function updateUndoRedoButtons() {  
 undoBtn.disabled = undoStack.length <= 1;  
 redoBtn.disabled = redoStack.length === 0;  
}  
  
function clearCanvas() {  
 if (confirm('Ви впевнені, що хочете очистити полотно?')) {  
 const activeCtx = getActiveContext();  
 activeCtx.clearRect(0, 0, activeCtx.canvas.width, activeCtx.canvas.height);  
 redrawAllLayers();  
 saveState();  
 }  
}  
  
function saveDrawing() {  
 const format = saveFormat.value;  
 const mimeType = format === 'png' ? 'image/png' : format === 'jpeg' ? 'image/jpeg' : 'image/svg+xml';  
  
 const link = *document*.createElement('a');  
 link.download = `drawing.${format}`;  
  
 const tempCanvas = *document*.createElement('canvas');  
 tempCanvas.width = *canvas*.width;  
 tempCanvas.height = *canvas*.height;  
 const tempCtx = tempCanvas.getContext('2d');  
  
 tempCtx.fillStyle = '#FFFFFF';  
 tempCtx.fillRect(0, 0, tempCanvas.width, tempCanvas.height);  
  
 *layers*.forEach(layer => {  
 if (layer.visible) {  
 tempCtx.globalAlpha = layer.opacity;  
 tempCtx.globalCompositeOperation = layer.blendMode;  
 tempCtx.drawImage(layer.canvas, 0, 0);  
 tempCtx.globalAlpha = 1;  
 tempCtx.globalCompositeOperation = 'source-over';  
 }  
 });  
  
 link.href = tempCanvas.toDataURL(mimeType);  
 link.click();  
}  
  
function handlePasteImage(e) {  
 const items = (e.clipboardData || *window*.clipboardData).items;  
 for (let i = 0; i < items.length; i++) {  
 if (items[i].type.indexOf('image') !== -1) {  
 const blob = items[i].getAsFile();  
 const reader = new FileReader();  
 reader.onload = function(event) {  
 const img = new Image();  
 img.onload = function() {  
 const ratio = *Math*.min(*canvas*.width / img.width, *canvas*.height / img.height);  
 const width = img.width \* ratio;  
 const height = img.height \* ratio;  
 const x = (*canvas*.width - width) / 2;  
 const y = (*canvas*.height - height) / 2;  
  
 const layer = createLayer('Вставлене зображення');  
 layer.ctx.drawImage(img, x, y, width, height);  
 redrawAllLayers();  
 saveState();  
 };  
 img.src = event.target.result;  
 };  
 reader.readAsDataURL(blob);  
 break;  
 }  
 }  
}  
  
function handleLoadImage(e) {  
 const file = e.target.files[0];  
 if (file) {  
 const reader = new FileReader();  
 reader.onload = function(event) {  
 const img = new Image();  
 img.onload = function() {  
 const ratio = *Math*.min(*canvas*.width / img.width, *canvas*.height / img.height);  
 const width = img.width \* ratio;  
 const height = img.height \* ratio;  
 const x = (*canvas*.width - width) / 2;  
 const y = (*canvas*.height - height) / 2;  
  
 const layer = createLayer('Завантажене зображення');  
 layer.ctx.drawImage(img, x, y, width, height);  
 redrawAllLayers();  
 saveState();  
 };  
 img.src = event.target.result;  
 };  
 reader.readAsDataURL(file);  
 }  
 e.target.value = '';  
}  
  
function toggleLayersPanel() {  
 layersPanel.classList.toggle('active');  
 layersBtn.classList.toggle('active', layersPanel.classList.contains('active'));  
}  
  
function toggleDarkTheme() {  
 *document*.body.classList.toggle('dark-theme');  
 saveState();  
}  
  
function toggleFullscreen() {  
 if (!*document*.fullscreenElement) {  
 *document*.documentElement.requestFullscreen();  
 } else {  
 *document*.exitFullscreen();  
 }  
}  
  
function changeZoom(delta) {  
 zoom = *Math*.max(0.1, *Math*.min(3, zoom + delta));  
 updateZoomLevel();  
  
 const container = *document*.getElementById('canvas-container');  
 container.style.transform = `scale(${zoom})`;  
 container.style.transformOrigin = '0 0';  
}  
  
function updateZoomLevel() {  
 zoomLevel.textContent = `${*Math*.round(zoom \* 100)}%`;  
}  
  
function updateRGBValues() {  
 const hex = currentColor.startsWith('#') ? currentColor : `#${currentColor}`;  
 const r = parseInt(hex.substr(1, 2), 16);  
 const g = parseInt(hex.substr(3, 2), 16);  
 const b = parseInt(hex.substr(5, 2), 16);  
  
 rValue.value = r;  
 gValue.value = g;  
 bValue.value = b;  
}  
  
function updateColorFromRGB() {  
 const r = parseInt(rValue.value) || 0;  
 const g = parseInt(gValue.value) || 0;  
 const b = parseInt(bValue.value) || 0;  
  
 currentColor = `rgb(${r}, ${g}, ${b})`;  
 colorPicker.value = `#${((1 << 24) + (r << 16) + (g << 8) + b).toString(16).slice(1)}`;  
}  
  
function toggleGradientMode() {  
 isGradientMode = !isGradientMode;  
 gradientBtn.classList.toggle('active', isGradientMode);  
 updateGradientPreview();  
}  
  
function updateGradientPreview() {  
 gradientPreview.style.background = `linear-gradient(to right, ${gradientColor1.value}, ${gradientColor2.value})`;  
}  
  
function createColorSwatches() {  
 *document*.querySelectorAll('.color-swatch').forEach(swatch => {  
 swatch.addEventListener('click', function() {  
 currentColor = this.style.backgroundColor;  
 colorPicker.value = rgbToHex(currentColor);  
 updateRGBValues();  
 });  
 });  
  
 *document*.getElementById('add-swatch').addEventListener('click', function() {  
 const swatch = *document*.createElement('div');  
 swatch.className = 'color-swatch';  
 swatch.style.backgroundColor = currentColor;  
 swatch.addEventListener('click', function() {  
 currentColor = this.style.backgroundColor;  
 colorPicker.value = rgbToHex(currentColor);  
 updateRGBValues();  
 });  
 *document*.getElementById('swatches-grid').appendChild(swatch);  
 });  
}  
  
function setupEventListeners() {  
 *canvas*.addEventListener('mousedown', startDrawing);  
 *canvas*.addEventListener('mousemove', draw);  
 *canvas*.addEventListener('mouseup', stopDrawing);  
 *canvas*.addEventListener('mouseout', stopDrawing);  
 *canvas*.addEventListener('wheel', handleWheel, { passive: false });  
  
 *document*.addEventListener('keydown', handleKeyDown);  
  
 colorPicker.addEventListener('input', (e) => {  
 currentColor = e.target.value;  
 updateRGBValues();  
 });  
  
 lineWidth.addEventListener('input', (e) => {  
 currentLineWidth = e.target.value;  
 widthValue.textContent = `${currentLineWidth}px`;  
 });  
  
 opacitySlider.addEventListener('input', (e) => {  
 currentOpacity = e.target.value;  
 opacityValue.textContent = `${*Math*.round(currentOpacity \* 100)}%`;  
 });  
  
 rValue.addEventListener('input', updateColorFromRGB);  
 gValue.addEventListener('input', updateColorFromRGB);  
 bValue.addEventListener('input', updateColorFromRGB);  
  
 gradientBtn.addEventListener('click', toggleGradientMode);  
 gradientColor1.addEventListener('input', updateGradientPreview);  
 gradientColor2.addEventListener('input', updateGradientPreview);  
  
 grayscaleBtn.addEventListener('click', () => {  
 const activeCtx = getActiveContext();  
 applyFilter('grayscale', activeCtx);  
 redrawAllLayers();  
 saveState();  
 });  
  
 invertBtn.addEventListener('click', () => {  
 const activeCtx = getActiveContext();  
 applyFilter('invert', activeCtx);  
 redrawAllLayers();  
 saveState();  
 });  
  
 darkThemeBtn.addEventListener('click', toggleDarkTheme);  
  
 clearBtn.addEventListener('click', clearCanvas);  
 undoBtn.addEventListener('click', undo);  
 redoBtn.addEventListener('click', redo);  
 saveBtn.addEventListener('click', saveDrawing);  
  
 layersBtn.addEventListener('click', toggleLayersPanel);  
 addLayerBtn.addEventListener('click', () => {  
 addNewLayer();  
 saveState();  
 });  
 removeLayerBtn.addEventListener('click', () => {  
 removeCurrentLayer();  
 saveState();  
 });  
 mergeLayersBtn.addEventListener('click', () => {  
 mergeLayers();  
 saveState();  
 });  
  
 zoomInBtn.addEventListener('click', () => {  
 changeZoom(0.1);  
 });  
 zoomOutBtn.addEventListener('click', () => {  
 changeZoom(-0.1);  
 });  
  
 loadImageBtn.addEventListener('change', handleLoadImage);  
 fullscreenBtn.addEventListener('click', toggleFullscreen);  
  
 toolButtons.forEach(button => {  
 button.addEventListener('mousedown', (e) => {  
 e.preventDefault();  
 toolButtons.forEach(btn => btn.classList.remove('active'));  
 button.classList.add('active');  
 currentTool = button.dataset.tool;  
 previewCtx.clearRect(0, 0, previewCanvas.width, previewCanvas.height);  
  
 if (currentTool === 'curve') curvePoints = [];  
 });  
 });  
  
 brushShape.addEventListener('change', (e) => {  
 currentBrushShape = e.target.value;  
 });  
}  
  
function setupColorPickerTool() {  
 createColorSwatches();  
}  
  
// Initialize the application  
init();

**styles.css**

:root {  
 --primary-color: #4a6fa5;  
 --secondary-color: #6c8fc7;  
 --dark-color: #333;  
 --light-color: #f5f5f5;  
 --border-color: #ddd;  
 --shadow-color: rgba(0, 0, 0, 0.1);  
 --grid-color: rgba(0, 0, 0, 0.05);  
 --bg-color: #f0f0f0;  
 --text-color: #333;  
 --panel-bg: white;  
 --canvas-bg: #FFFFFF;  
}  
  
.dark-theme {  
 --primary-color: #6c8fc7;  
 --secondary-color: #4a6fa5;  
 --dark-color: #f5f5f5;  
 --light-color: #333;  
 --border-color: #555;  
 --shadow-color: rgba(255, 255, 255, 0.1);  
 --grid-color: rgba(255, 255, 255, 0.05);  
 --bg-color: #222;  
 --text-color: #f5f5f5;  
 --panel-bg: #333;  
 --canvas-bg: #333;  
}  
  
\* {  
 margin: 0;  
 padding: 0;  
 box-sizing: border-box;  
 font-family: 'Arial', sans-serif;  
}  
  
body {  
 background-color: var(--bg-color);  
 color: var(--text-color);  
 height: 100vh;  
 overflow: hidden;  
 transition: background-color 0.3s, color 0.3s;  
}  
  
.app-container {  
 display: flex;  
 flex-direction: column;  
 height: 100vh;  
}  
  
.app-header {  
 background-color: var(--panel-bg);  
 padding: 0.5rem 1rem;  
 box-shadow: 0 2px 5px var(--shadow-color);  
 z-index: 10;  
 transition: background-color 0.3s;  
}  
  
.toolbar {  
 display: flex;  
 flex-wrap: wrap;  
 gap: 0.5rem;  
 align-items: center;  
}  
  
.tool-group {  
 display: flex;  
 align-items: center;  
 gap: 0.5rem;  
 padding: 0.3rem 0.5rem;  
 border-right: 1px solid var(--border-color);  
 transition: border-color 0.3s;  
}  
  
.tool-group:last-child {  
 border-right: none;  
}  
  
.tool-btn {  
 width: 32px;  
 height: 32px;  
 min-width: 32px;  
 max-width: 120px;  
 display: flex;  
 align-items: center;  
 justify-content: center;  
 background-color: var(--panel-bg);  
 border: 1px solid var(--border-color);  
 border-radius: 4px;  
 cursor: pointer;  
 color: var(--text-color);  
 font-size: 12px;  
 transition: all 0.2s;  
 position: relative;  
 overflow: hidden;  
 white-space: nowrap;  
 text-overflow: ellipsis;  
 padding: 0 3px;  
}  
  
.tool-btn.wide-btn {  
 min-width: 100px;  
 width: auto;  
 padding: 0 10px;  
}  
  
.tool-btn span {  
 font-size: 16px;  
 line-height: 1;  
 white-space: nowrap;  
 overflow: hidden;  
 text-overflow: ellipsis;  
 max-width: 100%;  
}  
  
.tool-btn:hover {  
 background-color: var(--light-color);  
 color: var(--dark-color);  
}  
  
.tool-btn:hover::after {  
 content: attr(title);  
 position: absolute;  
 bottom: -30px;  
 left: 50%;  
 transform: translateX(-50%);  
 background-color: var(--dark-color);  
 color: white;  
 padding: 4px 8px;  
 border-radius: 4px;  
 font-size: 12px;  
 white-space: nowrap;  
 z-index: 100;  
}  
  
.tool-btn.active {  
 background-color: var(--primary-color);  
 color: white;  
 border-color: var(--primary-color);  
}  
  
.tool-btn.active[data-tool="marker"] {  
 background-color: rgba(74, 111, 165, 0.3);  
}  
  
.tool-btn[data-tool="eraser"]:after {  
 content: '';  
 position: absolute;  
 width: 20px;  
 height: 20px;  
 background-color: white;  
 border: 1px solid #ccc;  
 border-radius: 3px;  
}  
  
input[type="color"] {  
 width: 32px;  
 height: 32px;  
 border: 1px solid var(--border-color);  
 border-radius: 4px;  
 padding: 0;  
 cursor: pointer;  
 background-color: var(--panel-bg);  
}  
  
.gradient-color-picker {  
 width: 30px;  
 height: 30px;  
 margin: 0 5px;  
 border: 1px solid var(--border-color);  
 border-radius: 4px;  
 cursor: pointer;  
}  
  
input[type="range"] {  
 width: 80px;  
 cursor: pointer;  
 background-color: var(--panel-bg);  
}  
  
select, input[type="number"] {  
 height: 32px;  
 padding: 0 0.5rem;  
 border: 1px solid var(--border-color);  
 border-radius: 4px;  
 background-color: var(--panel-bg);  
 color: var(--text-color);  
}  
  
.slider-container {  
 display: flex;  
 align-items: center;  
 gap: 0.5rem;  
 font-size: 14px;  
}  
  
.slider-container span {  
 white-space: nowrap;  
}  
  
.main-content {  
 display: flex;  
 flex: 1;  
 overflow: hidden;  
}  
  
.canvas-container {  
 flex: 1;  
 position: relative;  
 overflow: auto;  
 background-color: var(--canvas-bg);  
 display: flex;  
 justify-content: center;  
 align-items: center;  
 background-image:  
 linear-gradient(45deg, var(--grid-color) 25%, transparent 25%),  
 linear-gradient(-45deg, var(--grid-color) 25%, transparent 25%),  
 linear-gradient(45deg, transparent 75%, var(--grid-color) 75%),  
 linear-gradient(-45deg, transparent 75%, var(--grid-color) 75%);  
 background-size: 20px 20px;  
 background-position: 0 0, 0 10px, 10px -10px, -10px 0px;  
}  
  
#drawing-canvas, #preview-canvas {  
 position: absolute;  
 top: 0;  
 left: 0;  
 background-color: transparent;  
 box-shadow: 0 0 10px var(--shadow-color);  
}  
  
.preview-canvas {  
 pointer-events: none;  
 opacity: 0.7;  
}  
  
.zoom-controls {  
 position: absolute;  
 bottom: 1rem;  
 right: 1rem;  
 display: flex;  
 align-items: center;  
 gap: 0.5rem;  
 background-color: var(--panel-bg);  
 padding: 0.5rem;  
 border-radius: 4px;  
 box-shadow: 0 0 5px var(--shadow-color);  
}  
  
.zoom-controls button {  
 width: 28px;  
 height: 28px;  
 display: flex;  
 align-items: center;  
 justify-content: center;  
 background-color: var(--panel-bg);  
 border: 1px solid var(--border-color);  
 border-radius: 4px;  
 cursor: pointer;  
 color: var(--text-color);  
}  
  
.layers-panel {  
 width: 250px;  
 background-color: var(--panel-bg);  
 border-left: 1px solid var(--border-color);  
 padding: 0.5rem;  
 overflow-y: auto;  
 display: none;  
 transition: background-color 0.3s, border-color 0.3s;  
}  
  
.layers-panel.active {  
 display: block;  
}  
  
.layers-header {  
 display: flex;  
 justify-content: space-between;  
 align-items: center;  
 margin-bottom: 0.5rem;  
 padding-bottom: 0.5rem;  
 border-bottom: 1px solid var(--border-color);  
}  
  
.layers-header button {  
 width: 24px;  
 height: 24px;  
 display: flex;  
 align-items: center;  
 justify-content: center;  
 background-color: var(--light-color);  
 border: 1px solid var(--border-color);  
 border-radius: 4px;  
 cursor: pointer;  
 color: var(--text-color);  
}  
  
.layers-list {  
 display: flex;  
 flex-direction: column;  
 gap: 0.3rem;  
}  
  
.layer-item {  
 display: flex;  
 align-items: center;  
 padding: 8px;  
 border: 1px solid var(--border-color);  
 border-radius: 4px;  
 cursor: pointer;  
 background-color: var(--panel-bg);  
 transition: all 0.2s;  
 margin-bottom: 5px;  
}  
  
.layer-item.active {  
 border-color: var(--primary-color);  
 background-color: rgba(74, 111, 165, 0.2);  
}  
  
.layer-item.dragging {  
 opacity: 0.5;  
 background-color: rgba(74, 111, 165, 0.1);  
}  
  
.layer-controls {  
 display: grid;  
 grid-template-columns: 20px 20px 1fr;  
 gap: 5px;  
 align-items: center;  
 width: 100%;  
}  
  
.layer-name {  
 flex: 1;  
 white-space: nowrap;  
 overflow: hidden;  
 text-overflow: ellipsis;  
 max-width: 100%;  
 grid-column: span 2;  
 font-size: 14px;  
 font-weight: bold;  
}  
  
.layer-item input[type="checkbox"] {  
 margin-right: 0.3rem;  
}  
  
.layer-opacity {  
 width: 50px;  
 grid-column: span 3;  
 width: 100%;  
}  
  
.blend-mode {  
 grid-column: span 3;  
 width: 100%;  
}  
  
.rgb-picker {  
 display: flex;  
 gap: 0.5rem;  
 align-items: center;  
}  
  
.rgb-picker input {  
 width: 50px;  
 height: 32px;  
 text-align: center;  
}  
  
.gradient-preview {  
 width: 32px;  
 height: 32px;  
 border: 1px solid var(--border-color);  
 border-radius: 4px;  
 cursor: pointer;  
 background: linear-gradient(to right, #ff0000, #0000ff);  
}  
  
.filter-controls {  
 display: flex;  
 gap: 0.5rem;  
}  
  
.filter-btn {  
 padding: 0.25rem 0.5rem;  
 border: 1px solid var(--border-color);  
 border-radius: 4px;  
 cursor: pointer;  
 font-size: 12px;  
 background-color: var(--panel-bg);  
 color: var(--text-color);  
}  
  
.filter-btn:hover {  
 background-color: var(--light-color);  
}  
  
.custom-tooltip {  
 position: fixed;  
 background-color: var(--dark-color);  
 color: white;  
 padding: 5px 10px;  
 border-radius: 4px;  
 font-size: 12px;  
 z-index: 1000;  
 pointer-events: none;  
 white-space: nowrap;  
}  
  
.color-swatches-panel {  
 position: fixed;  
 right: 20px;  
 bottom: 20px;  
 width: 200px;  
 background-color: var(--panel-bg);  
 border: 1px solid var(--border-color);  
 border-radius: 8px;  
 padding: 10px;  
 box-shadow: 0 0 10px var(--shadow-color);  
 z-index: 100;  
}  
  
.swatches-header {  
 display: flex;  
 justify-content: space-between;  
 align-items: center;  
 margin-bottom: 10px;  
}  
  
.swatches-header h4 {  
 font-size: 14px;  
 color: var(--text-color);  
}  
  
.swatches-header button {  
 width: 24px;  
 height: 24px;  
 display: flex;  
 align-items: center;  
 justify-content: center;  
 background-color: var(--primary-color);  
 color: white;  
 border: none;  
 border-radius: 4px;  
 cursor: pointer;  
}  
  
.swatches-grid {  
 display: grid;  
 grid-template-columns: repeat(5, 1fr);  
 gap: 5px;  
}  
  
.color-swatch {  
 width: 100%;  
 height: 30px;  
 border: 1px solid var(--border-color);  
 border-radius: 4px;  
 cursor: pointer;  
 transition: transform 0.2s;  
}  
  
.color-swatch:hover {  
 transform: scale(1.1);  
}  
  
.fullscreen-btn {  
 position: fixed;  
 top: 10px;  
 right: 10px;  
 z-index: 1001;  
}  
  
@keyframes fadeIn {  
 from { opacity: 0; }  
 to { opacity: 1; }  
}  
  
@media (max-width: 1200px) {  
 .toolbar {  
 flex-direction: column;  
 align-items: flex-start;  
 }  
  
 .tool-group {  
 border-right: none;  
 border-bottom: 1px solid var(--border-color);  
 width: 100%;  
 }  
  
 .main-content {  
 flex-direction: column;  
 }  
  
 .layers-panel {  
 width: 100%;  
 height: 200px;  
 border-left: none;  
 border-top: 1px solid var(--border-color);  
 }  
}  
  
.dark-theme .tool-btn:not(.active):hover {  
 background-color: #444;  
}  
  
.dark-theme .layer-item:not(.active):hover {  
 background-color: #444;  
}  
  
.tool-btn {  
 user-select: none;  
 -webkit-user-select: none;  
 touch-action: manipulation;  
}  
  
.layers-panel {  
 z-index: 100;  
 position: relative;  
}  
  
input[type="color"] {  
 -webkit-appearance: none;  
 border: none;  
 width: 30px;  
 height: 30px;  
 cursor: pointer;  
}  
input[type="color"]::-webkit-color-swatch-wrapper {  
 padding: 0;  
}  
input[type="color"]::-webkit-color-swatch {  
 border: none;  
 border-radius: 4px;  
}  
  
input[type="range"] {  
 -webkit-appearance: none;  
 height: 6px;  
 background: var(--border-color);  
 border-radius: 3px;  
}  
input[type="range"]::-webkit-slider-thumb {  
 -webkit-appearance: none;  
 width: 16px;  
 height: 16px;  
 background: var(--primary-color);  
 border-radius: 50%;  
 cursor: pointer;  
}  
  
select {  
 cursor: pointer;  
 padding: 5px;  
}  
  
input[type="number"], input[type="text"] {  
 padding: 5px;  
 border: 1px solid var(--border-color);  
 border-radius: 4px;  
}  
  
.tool-btn.active {  
 box-shadow: 0 0 0 2px var(--primary-color);  
}

**layers.js**

import { generateId } from './utils.js';  
import { *canvas*, *ctx*, saveState as saveScriptState } from './script.js';  
  
let *layers* = [];  
let *activeLayerIndex* = 0;  
  
function createLayer(name) {  
 const layerCanvas = *document*.createElement('canvas');  
 layerCanvas.width = *canvas*.width;  
 layerCanvas.height = *canvas*.height;  
 const layerCtx = layerCanvas.getContext('2d');  
 layerCtx.fillStyle = 'rgba(0, 0, 0, 0)';  
 layerCtx.fillRect(0, 0, layerCanvas.width, layerCanvas.height);  
  
 const layer = {  
 id: generateId(),  
 name: name || `Шар ${*layers*.length + 1}`,  
 canvas: layerCanvas,  
 ctx: layerCtx,  
 visible: true,  
 locked: false,  
 opacity: 1,  
 blendMode: 'source-over'  
 };  
  
 *layers*.push(layer);  
 *activeLayerIndex* = *layers*.length - 1;  
 updateLayersList();  
 redrawAllLayers();  
  
 return layer;  
}  
  
function removeCurrentLayer() {  
 if (*layers*.length <= 1) {  
 alert('Повинен залишатися хоча б один шар!');  
 return;  
 }  
  
 *layers*.splice(*activeLayerIndex*, 1);  
 if (*activeLayerIndex* >= *layers*.length) {  
 *activeLayerIndex* = *layers*.length - 1;  
 }  
  
 updateLayersList();  
 redrawAllLayers();  
 saveScriptState();  
}  
  
function addNewLayer() {  
 createLayer(`Шар ${*layers*.length + 1}`);  
 saveScriptState();  
}  
  
function mergeLayers() {  
 if (*layers*.length <= 1) return;  
  
 const newCanvas = *document*.createElement('canvas');  
 newCanvas.width = *canvas*.width;  
 newCanvas.height = *canvas*.height;  
 const newCtx = newCanvas.getContext('2d');  
  
 *layers*.forEach(layer => {  
 if (layer.visible) {  
 newCtx.globalAlpha = layer.opacity;  
 newCtx.globalCompositeOperation = layer.blendMode;  
 newCtx.drawImage(layer.canvas, 0, 0);  
 newCtx.globalAlpha = 1;  
 newCtx.globalCompositeOperation = 'source-over';  
 }  
 });  
  
 *layers* = [{  
 id: generateId(),  
 name: 'Об`єднаний шар',  
 canvas: newCanvas,  
 ctx: newCtx,  
 visible: true,  
 locked: false,  
 opacity: 1,  
 blendMode: 'source-over'  
 }];  
 *activeLayerIndex* = 0;  
  
 updateLayersList();  
 redrawAllLayers();  
 saveScriptState();  
}  
  
function updateLayersList() {  
 const layersList = *document*.getElementById('layers-list');  
 layersList.innerHTML = '';  
  
 *layers*.forEach((layer, index) => {  
 const layerItem = *document*.createElement('div');  
 layerItem.className = `layer-item ${index === *activeLayerIndex* ? 'active' : ''}`;  
 layerItem.dataset.layerId = layer.id;  
 layerItem.draggable = true;  
  
 layerItem.innerHTML = `  
 <div class="layer-controls">  
 <input type="checkbox" ${layer.visible ? 'checked' : ''} class="visibility-toggle">  
 <input type="checkbox" ${layer.locked ? 'checked' : ''} class="lock-toggle">  
 <span class="layer-name">${layer.name}</span>  
 <input type="range" min="0" max="1" step="0.01" value="${layer.opacity}" class="layer-opacity">  
 <select class="blend-mode">  
 <option value="source-over" ${layer.blendMode === 'source-over' ? 'selected' : ''}>Normal</option>  
 <option value="multiply" ${layer.blendMode === 'multiply' ? 'selected' : ''}>Multiply</option>  
 <option value="screen" ${layer.blendMode === 'screen' ? 'selected' : ''}>Screen</option>  
 </select>  
 </div>  
 `;  
  
 layerItem.addEventListener('click', (e) => {  
 if (!e.target.classList.contains('visibility-toggle') && !e.target.classList.contains('lock-toggle') &&  
 !e.target.classList.contains('layer-opacity') && !e.target.classList.contains('blend-mode')) {  
 setActiveLayer(index);  
 }  
 });  
  
 layerItem.addEventListener('dblclick', () => renameLayer(index));  
  
 const visibilityToggle = layerItem.querySelector('.visibility-toggle');  
 visibilityToggle.addEventListener('change', (e) => {  
 e.stopPropagation();  
 layer.visible = e.target.checked;  
 redrawAllLayers();  
 saveScriptState();  
 });  
  
 const lockToggle = layerItem.querySelector('.lock-toggle');  
 lockToggle.addEventListener('change', (e) => {  
 e.stopPropagation();  
 layer.locked = e.target.checked;  
 });  
  
 const opacitySlider = layerItem.querySelector('.layer-opacity');  
 opacitySlider.addEventListener('input', (e) => {  
 layer.opacity = parseFloat(e.target.value);  
 redrawAllLayers();  
 saveScriptState();  
 });  
  
 const blendModeSelect = layerItem.querySelector('.blend-mode');  
 blendModeSelect.addEventListener('change', (e) => {  
 layer.blendMode = e.target.value;  
 redrawAllLayers();  
 saveScriptState();  
 });  
  
 layersList.appendChild(layerItem);  
 });  
  
 setupLayerDragAndDrop();  
}  
  
function renameLayer(index) {  
 const newName = prompt('Введіть нову назву шару:', *layers*[index].name);  
 if (newName && newName.trim() !== '') {  
 *layers*[index].name = newName.trim();  
 updateLayersList();  
 saveScriptState();  
 }  
}  
  
function setActiveLayer(index) {  
 if (*layers*[index].locked) return;  
 *activeLayerIndex* = index;  
 updateLayersList();  
}  
  
function setupLayerDragAndDrop() {  
 const layerItems = *document*.querySelectorAll('.layer-item');  
 layerItems.forEach(item => {  
 item.addEventListener('dragstart', e => {  
 e.dataTransfer.setData('text/plain', item.dataset.layerId);  
 item.classList.add('dragging');  
 });  
 item.addEventListener('dragend', () => item.classList.remove('dragging'));  
 });  
  
 const layersList = *document*.getElementById('layers-list');  
 layersList.addEventListener('dragover', e => {  
 e.preventDefault();  
 const draggingElement = *document*.querySelector('.layer-item.dragging');  
 const targetElement = e.target.closest('.layer-item');  
 if (targetElement && draggingElement !== targetElement) {  
 const rect = targetElement.getBoundingClientRect();  
 const next = e.clientY > rect.top + rect.height / 2;  
 layersList.insertBefore(draggingElement, next ? targetElement.nextSibling : targetElement);  
 }  
 });  
  
 layersList.addEventListener('drop', e => {  
 e.preventDefault();  
 const layerId = e.dataTransfer.getData('text/plain');  
 const layerIndex = *layers*.findIndex(l => l.id === layerId);  
 if (layerIndex === -1) return;  
  
 const newOrder = [];  
 *document*.querySelectorAll('.layer-item').forEach(item => {  
 const id = item.dataset.layerId;  
 const layer = *layers*.find(l => l.id === id);  
 if (layer) newOrder.push(layer);  
 });  
  
 *layers* = newOrder;  
 *activeLayerIndex* = *layers*.findIndex(l => l.id === layerId);  
 redrawAllLayers();  
 saveScriptState();  
 });  
}  
  
function redrawAllLayers() {  
 if (!*ctx*) return;  
 *ctx*.fillStyle = '#FFFFFF';  
 *ctx*.fillRect(0, 0, *canvas*.width, *canvas*.height);  
  
 *layers*.forEach((layer) => {  
 if (layer.visible) {  
 *ctx*.globalAlpha = layer.opacity;  
 *ctx*.globalCompositeOperation = layer.blendMode;  
 *ctx*.drawImage(layer.canvas, 0, 0);  
 *ctx*.globalAlpha = 1;  
 *ctx*.globalCompositeOperation = 'source-over';  
 }  
 });  
}  
  
function getActiveContext() {  
 return *layers*[*activeLayerIndex*].ctx;  
}  
  
function getActiveCanvas() {  
 return *layers*[*activeLayerIndex*].canvas;  
}  
  
function drawOnActiveLayer(drawFunc) {  
 const activeCtx = getActiveContext();  
 drawFunc(activeCtx);  
 redrawAllLayers();  
}  
  
export {  
 *layers*,  
 *activeLayerIndex*,  
 createLayer,  
 removeCurrentLayer,  
 addNewLayer,  
 updateLayersList,  
 setActiveLayer,  
 redrawAllLayers,  
 getActiveContext,  
 getActiveCanvas,  
 drawOnActiveLayer,  
 mergeLayers  
};

**canvas.js**

const *canvas* = *document*.getElementById('canvas');  
if (!*canvas*) {  
 *console*.error('Canvas element not found!');  
} else {  
 const ctx = *canvas*.getContext('2d');  
 let width, height;  
 let dots = [];  
 let rotationAngle = 0;  
 const topColor = { r: 128, g: 0, b: 32 }; // #800020  
 const bottomColor = { r: 75, g: 0, b: 130 }; // #4B0082  
  
 function resizeCanvas() {  
 const headerHeight = *document*.querySelector('.app-header').offsetHeight;  
 *canvas*.width = *window*.innerWidth;  
 *canvas*.height = *window*.innerHeight - headerHeight;  
 *previewCanvas*.width = *canvas*.width;  
 *previewCanvas*.height = *canvas*.height;  
 *layers*.forEach(layer => {  
 layer.canvas.width = *canvas*.width;  
 layer.canvas.height = *canvas*.height;  
 });  
 redrawAllLayers();  
 }  
  
 *window*.addEventListener('resize', resizeCanvas);  
  
 function initDots() {  
 dots = [];  
 const gridSize = 100;  
 const scaledWidth = width \* 2;  
 const scaledHeight = height \* 2;  
 const cols = *Math*.ceil(scaledWidth / gridSize);  
 const rows = *Math*.ceil(scaledHeight / gridSize);  
  
 for (let i = 0; i < cols \* rows; i++) {  
 const x = (i % cols) \* gridSize;  
 const y = *Math*.floor(i / cols) \* gridSize;  
 const offsetX = x - width \* 0.5;  
 const offsetY = y - height \* 0.5;  
 dots.push({  
 x: offsetX,  
 y: offsetY,  
 originalX: offsetX,  
 originalY: offsetY,  
 size: 10  
 });  
 }  
 }  
  
 function getDotColor(y) {  
 const ratio = (height - y) / height;  
 const r = *Math*.round(topColor.r + (bottomColor.r - topColor.r) \* ratio);  
 const g = *Math*.round(topColor.g + (bottomColor.g - topColor.g) \* ratio);  
 const b = *Math*.round(topColor.b + (bottomColor.b - topColor.b) \* ratio);  
 return `rgb(${r}, ${g}, ${b})`;  
 }  
  
 function getNeonColor(colorStr) {  
 const match = colorStr.match(/\d+/g);  
 if (match) {  
 let r = *Math*.min(255, parseInt(match[0]));  
 let g = *Math*.min(255, parseInt(match[1]));  
 let b = *Math*.min(255, parseInt(match[2]));  
 return `rgb(${r}, ${g}, ${b})`;  
 }  
 return colorStr;  
 }  
  
 function getShadowColor(colorStr) {  
 const match = colorStr.match(/\d+/g);  
 if (match) {  
 let r = *Math*.max(0, parseInt(match[0]) \* 0.5);  
 let g = *Math*.max(0, parseInt(match[1]) \* 0.5);  
 let b = *Math*.max(0, parseInt(match[2]) \* 0.5);  
 return `rgba(${r}, ${g}, ${b}, 0.5)`;  
 }  
 return 'rgba(0, 0, 0, 0.5)';  
 }  
  
 function animate() {  
 ctx.clearRect(0, 0, width, height);  
 rotationAngle += 0.05;  
 const centerX = width / 2;  
 const centerY = height / 2;  
  
 dots.forEach(dot => {  
 const dx = dot.originalX - centerX;  
 const dy = dot.originalY - centerY;  
 const cos = *Math*.cos(rotationAngle \* *Math*.PI / 180);  
 const sin = *Math*.sin(rotationAngle \* *Math*.PI / 180);  
 dot.x = centerX + (dx \* cos - dy \* sin);  
 dot.y = centerY + (dx \* sin + dy \* cos);  
  
 dot.color = getDotColor(dot.y);  
 ctx.beginPath();  
 const gradient = ctx.createRadialGradient(dot.x, dot.y, 0, dot.x, dot.y, dot.size);  
 gradient.addColorStop(0, dot.color);  
 const neonColor = getNeonColor(dot.color);  
 gradient.addColorStop(1, neonColor);  
 ctx.fillStyle = gradient;  
  
 const shadowColor = getShadowColor(dot.color);  
 ctx.shadowOffsetX = 10;  
 ctx.shadowOffsetY = 10;  
 ctx.shadowBlur = 15;  
 ctx.shadowColor = shadowColor;  
  
 ctx.arc(dot.x, dot.y, dot.size, 0, *Math*.PI \* 2);  
 ctx.fill();  
  
 ctx.shadowOffsetX = 0;  
 ctx.shadowOffsetY = 0;  
 ctx.shadowBlur = 0;  
 ctx.closePath();  
 });  
  
 requestAnimationFrame(animate);  
 }  
  
 resizeCanvas();  
 initDots();  
 animate();  
}

**shapes.js**

import { generateId } from './utils.js';  
  
function drawLine(x1, y1, x2, y2, context) {  
 const prevStrokeStyle = context.strokeStyle;  
 const prevLineWidth = context.lineWidth;  
 const prevGlobalAlpha = context.globalAlpha;  
 const prevLineJoin = context.lineJoin;  
 const prevLineCap = context.lineCap;  
  
 context.beginPath();  
 context.moveTo(x1, y1);  
 context.lineTo(x2, y2);  
 context.strokeStyle = prevStrokeStyle;  
 context.lineWidth = prevLineWidth;  
 context.globalAlpha = prevGlobalAlpha;  
 context.lineJoin = prevLineJoin;  
 context.lineCap = prevLineCap;  
 context.stroke();  
}  
  
function drawRectangle(x1, y1, x2, y2, fill = false, context) {  
 const prevStrokeStyle = context.strokeStyle;  
 const prevFillStyle = context.fillStyle;  
 const prevLineWidth = context.lineWidth;  
 const prevGlobalAlpha = context.globalAlpha;  
 const prevLineJoin = context.lineJoin;  
 const prevLineCap = context.lineCap;  
  
 const minX = *Math*.min(x1, x2);  
 const minY = *Math*.min(y1, y2);  
 const width = *Math*.abs(x2 - x1);  
 const height = *Math*.abs(y2 - y1);  
  
 context.beginPath();  
 context.rect(minX, minY, width, height);  
 if (fill) {  
 context.fillStyle = prevFillStyle;  
 context.fill();  
 }  
 context.strokeStyle = prevStrokeStyle;  
 context.lineWidth = prevLineWidth;  
 context.globalAlpha = prevGlobalAlpha;  
 context.lineJoin = prevLineJoin;  
 context.lineCap = prevLineCap;  
 context.stroke();  
}  
  
function drawCircle(x1, y1, x2, y2, fill = false, context) {  
 const prevStrokeStyle = context.strokeStyle;  
 const prevFillStyle = context.fillStyle;  
 const prevLineWidth = context.lineWidth;  
 const prevGlobalAlpha = context.globalAlpha;  
 const prevLineJoin = context.lineJoin;  
 const prevLineCap = context.lineCap;  
  
 const radius = *Math*.sqrt(*Math*.pow(x2 - x1, 2) + *Math*.pow(y2 - y1, 2));  
 context.beginPath();  
 context.arc(x1, y1, radius, 0, *Math*.PI \* 2);  
 if (fill) {  
 context.fillStyle = prevFillStyle;  
 context.fill();  
 }  
 context.strokeStyle = prevStrokeStyle;  
 context.lineWidth = prevLineWidth;  
 context.globalAlpha = prevGlobalAlpha;  
 context.lineJoin = prevLineJoin;  
 context.lineCap = prevLineCap;  
 context.stroke();  
}  
  
function drawTriangle(x1, y1, x2, y2, fill = false, context) {  
 const prevStrokeStyle = context.strokeStyle;  
 const prevFillStyle = context.fillStyle;  
 const prevLineWidth = context.lineWidth;  
 const prevGlobalAlpha = context.globalAlpha;  
 const prevLineJoin = context.lineJoin;  
 const prevLineCap = context.lineCap;  
  
 context.beginPath();  
 context.moveTo(x1, y1);  
 context.lineTo(x2, y2);  
 context.lineTo(x1 \* 2 - x2, y2);  
 context.closePath();  
  
 if (fill) {  
 context.fillStyle = prevFillStyle;  
 context.fill();  
 }  
 context.strokeStyle = prevStrokeStyle;  
 context.lineWidth = prevLineWidth;  
 context.globalAlpha = prevGlobalAlpha;  
 context.lineJoin = prevLineJoin;  
 context.lineCap = prevLineCap;  
 context.stroke();  
}  
  
function drawHexagon(x1, y1, x2, y2, context) {  
 const prevStrokeStyle = context.strokeStyle;  
 const prevLineWidth = context.lineWidth;  
 const prevGlobalAlpha = context.globalAlpha;  
 const prevLineJoin = context.lineJoin;  
 const prevLineCap = context.lineCap;  
  
 const centerX = x1;  
 const centerY = y1;  
 const radius = *Math*.sqrt(*Math*.pow(x2 - x1, 2) + *Math*.pow(y2 - y1, 2));  
  
 context.beginPath();  
 for (let i = 0; i < 6; i++) {  
 const angle = (*Math*.PI / 3) \* i;  
 context.lineTo(centerX + radius \* *Math*.cos(angle), centerY + radius \* *Math*.sin(angle));  
 }  
 context.closePath();  
 context.strokeStyle = prevStrokeStyle;  
 context.lineWidth = prevLineWidth;  
 context.globalAlpha = prevGlobalAlpha;  
 context.lineJoin = prevLineJoin;  
 context.lineCap = prevLineCap;  
 context.stroke();  
}  
  
function drawPentagon(x1, y1, x2, y2, context) {  
 const prevStrokeStyle = context.strokeStyle;  
 const prevLineWidth = context.lineWidth;  
 const prevGlobalAlpha = context.globalAlpha;  
 const prevLineJoin = context.lineJoin;  
 const prevLineCap = context.lineCap;  
  
 const centerX = x1;  
 const centerY = y1;  
 const radius = *Math*.sqrt(*Math*.pow(x2 - x1, 2) + *Math*.pow(y2 - y1, 2));  
  
 context.beginPath();  
 for (let i = 0; i < 5; i++) {  
 const angle = (*Math*.PI \* 2 / 5) \* i - *Math*.PI / 2;  
 context.lineTo(centerX + radius \* *Math*.cos(angle), centerY + radius \* *Math*.sin(angle));  
 }  
 context.closePath();  
 context.strokeStyle = prevStrokeStyle;  
 context.lineWidth = prevLineWidth;  
 context.globalAlpha = prevGlobalAlpha;  
 context.lineJoin = prevLineJoin;  
 context.lineCap = prevLineCap;  
 context.stroke();  
}  
  
function drawStar(x1, y1, x2, y2, context) {  
 const prevStrokeStyle = context.strokeStyle;  
 const prevLineWidth = context.lineWidth;  
 const prevGlobalAlpha = context.globalAlpha;  
 const prevLineJoin = context.lineJoin;  
 const prevLineCap = context.lineCap;  
  
 const centerX = x1;  
 const centerY = y1;  
 const radius = *Math*.sqrt(*Math*.pow(x2 - x1, 2) + *Math*.pow(y2 - y1, 2));  
 const outerRadius = radius;  
 const innerRadius = outerRadius / 2;  
  
 context.beginPath();  
 for (let i = 0; i < 10; i++) {  
 const angle = (*Math*.PI / 5) \* i - *Math*.PI / 2;  
 const r = i % 2 === 0 ? outerRadius : innerRadius;  
 context.lineTo(centerX + r \* *Math*.cos(angle), centerY + r \* *Math*.sin(angle));  
 }  
 context.closePath();  
 context.strokeStyle = prevStrokeStyle;  
 context.lineWidth = prevLineWidth;  
 context.globalAlpha = prevGlobalAlpha;  
 context.lineJoin = prevLineJoin;  
 context.lineCap = prevLineCap;  
 context.stroke();  
}  
  
function drawHeart(x1, y1, x2, y2, context) {  
 const prevStrokeStyle = context.strokeStyle;  
 const prevLineWidth = context.lineWidth;  
 const prevGlobalAlpha = context.globalAlpha;  
 const prevLineJoin = context.lineJoin;  
 const prevLineCap = context.lineCap;  
  
 const width = *Math*.abs(x2 - x1);  
 const height = *Math*.abs(y2 - y1);  
 const centerX = x1;  
 const centerY = y1;  
 const size = *Math*.min(width, height) / 2;  
  
 context.beginPath();  
 context.moveTo(centerX, centerY - size/2);  
 context.bezierCurveTo(  
 centerX + size, centerY - size,  
 centerX + size, centerY + size/2,  
 centerX, centerY + size  
 );  
 context.bezierCurveTo(  
 centerX - size, centerY + size/2,  
 centerX - size, centerY - size,  
 centerX, centerY - size/2  
 );  
 context.strokeStyle = prevStrokeStyle;  
 context.lineWidth = prevLineWidth;  
 context.globalAlpha = prevGlobalAlpha;  
 context.lineJoin = prevLineJoin;  
 context.lineCap = prevLineCap;  
 context.stroke();  
}  
  
function drawDiamond(x1, y1, x2, y2, context) {  
 const prevStrokeStyle = context.strokeStyle;  
 const prevLineWidth = context.lineWidth;  
 const prevGlobalAlpha = context.globalAlpha;  
 const prevLineJoin = context.lineJoin;  
 const prevLineCap = context.lineCap;  
  
 const centerX = x1;  
 const centerY = y1;  
 const width = *Math*.abs(x2 - x1);  
 const height = *Math*.abs(y2 - y1);  
  
 context.beginPath();  
 context.moveTo(centerX, centerY - height/2); // Верхняя точка  
 context.lineTo(centerX + width/2, centerY); // Правая точка  
 context.lineTo(centerX, centerY + height/2); // Нижняя точка  
 context.lineTo(centerX - width/2, centerY); // Левая точка  
 context.closePath();  
  
 context.strokeStyle = prevStrokeStyle;  
 context.lineWidth = prevLineWidth;  
 context.globalAlpha = prevGlobalAlpha;  
 context.lineJoin = prevLineJoin;  
 context.lineCap = prevLineCap;  
 context.stroke();  
}  
  
function drawArrow(x1, y1, x2, y2, context) {  
 const prevStrokeStyle = context.strokeStyle;  
 const prevFillStyle = context.fillStyle;  
 const prevLineWidth = context.lineWidth;  
 const prevGlobalAlpha = context.globalAlpha;  
 const prevLineJoin = context.lineJoin;  
 const prevLineCap = context.lineCap;  
  
 // Рассчитываем длину стрелки  
 const dx = x2 - x1;  
 const dy = y2 - y1;  
 const angle = *Math*.atan2(dy, dx);  
 const length = *Math*.sqrt(dx \* dx + dy \* dy);  
  
 // Размер наконечника стрелки пропорционален толщине линии, но не более 30  
 const headLength = *Math*.min(30, prevLineWidth \* 3);  
  
 // Корректируем конечную точку, чтобы линия не заходила за наконечник  
 const adjustedX2 = x2 - headLength \* *Math*.cos(angle);  
 const adjustedY2 = y2 - headLength \* *Math*.sin(angle);  
  
 // Рисуем линию  
 context.beginPath();  
 context.moveTo(x1, y1);  
 context.lineTo(adjustedX2, adjustedY2);  
 context.strokeStyle = prevStrokeStyle;  
 context.lineWidth = prevLineWidth;  
 context.globalAlpha = prevGlobalAlpha;  
 context.lineJoin = prevLineJoin;  
 context.lineCap = prevLineCap;  
 context.stroke();  
  
 // Рисуем наконечник стрелки  
 context.beginPath();  
 context.moveTo(x2, y2);  
 context.lineTo(  
 x2 - headLength \* *Math*.cos(angle - *Math*.PI / 6),  
 y2 - headLength \* *Math*.sin(angle - *Math*.PI / 6)  
 );  
 context.lineTo(  
 x2 - headLength \* *Math*.cos(angle + *Math*.PI / 6),  
 y2 - headLength \* *Math*.sin(angle + *Math*.PI / 6)  
 );  
 context.closePath();  
 context.fillStyle = prevStrokeStyle;  
 context.fill();  
}  
  
function drawCurve(points, context) {  
 const prevStrokeStyle = context.strokeStyle;  
 const prevLineWidth = context.lineWidth;  
 const prevGlobalAlpha = context.globalAlpha;  
 const prevLineJoin = context.lineJoin;  
 const prevLineCap = context.lineCap;  
  
 context.beginPath();  
 context.moveTo(points[0].x, points[0].y);  
 for (let i = 1; i < points.length; i++) {  
 context.lineTo(points[i].x, points[i].y);  
 }  
 context.strokeStyle = prevStrokeStyle;  
 context.lineWidth = prevLineWidth;  
 context.globalAlpha = prevGlobalAlpha;  
 context.lineJoin = prevLineJoin;  
 context.lineCap = prevLineCap;  
 context.stroke();  
}  
  
export { drawLine, drawRectangle, drawCircle, drawTriangle, drawHexagon, drawPentagon, drawStar, drawHeart, drawDiamond, drawArrow, drawCurve };

**tools.js**

import { colorToRgb, rgbToHex, interpolateColor } from './utils.js';  
  
function drawFreehand(x1, y1, x2, y2, context) {  
 const prevStrokeStyle = context.strokeStyle;  
 const prevLineWidth = context.lineWidth;  
 const prevGlobalAlpha = context.globalAlpha;  
 const prevLineJoin = context.lineJoin;  
 const prevLineCap = context.lineCap;  
  
 context.beginPath();  
 context.moveTo(x1, y1);  
 context.lineTo(x2, y2);  
 context.strokeStyle = prevStrokeStyle;  
 context.lineWidth = prevLineWidth;  
 context.globalAlpha = prevGlobalAlpha;  
 context.lineJoin = prevLineJoin;  
 context.lineCap = prevLineCap;  
 context.stroke();  
}  
  
function drawSpray(x, y, radius, density, color, context) {  
 const prevFillStyle = context.fillStyle;  
 const prevGlobalAlpha = context.globalAlpha;  
  
 const points = radius \* density;  
 context.fillStyle = color;  
 context.globalAlpha = 1;  
  
 for (let i = 0; i < points; i++) {  
 const angle = *Math*.random() \* *Math*.PI \* 2;  
 const distance = *Math*.random() \* radius;  
 const px = x + *Math*.cos(angle) \* distance;  
 const py = y + *Math*.sin(angle) \* distance;  
 context.beginPath();  
 context.arc(px, py, 1, 0, *Math*.PI \* 2);  
 context.fill();  
 }  
  
 context.fillStyle = prevFillStyle;  
 context.globalAlpha = prevGlobalAlpha;  
}  
  
function drawMarker(x1, y1, x2, y2, width, color, opacity, context) {  
 const prevStrokeStyle = context.strokeStyle;  
 const prevLineWidth = context.lineWidth;  
 const prevGlobalAlpha = context.globalAlpha;  
 const prevLineJoin = context.lineJoin;  
 const prevLineCap = context.lineCap;  
  
 context.strokeStyle = color;  
 context.lineWidth = width;  
 context.globalAlpha = opacity \* 0.7;  
 context.lineJoin = 'round';  
 context.lineCap = 'round';  
  
 context.beginPath();  
 context.moveTo(x1, y1);  
 context.lineTo(x2, y2);  
 context.stroke();  
  
 const tempCanvas = *document*.createElement('canvas');  
 tempCanvas.width = context.canvas.width;  
 tempCanvas.height = context.canvas.height;  
 const tempCtx = tempCanvas.getContext('2d');  
 tempCtx.drawImage(context.canvas, 0, 0);  
  
 context.save();  
 context.globalCompositeOperation = 'copy';  
 context.filter = 'blur(3px)';  
 context.drawImage(tempCanvas, 0, 0);  
 context.restore();  
  
 context.strokeStyle = prevStrokeStyle;  
 context.lineWidth = prevLineWidth;  
 context.globalAlpha = prevGlobalAlpha;  
 context.lineJoin = prevLineJoin;  
 context.lineCap = prevLineCap;  
}  
  
function erasePixels(x, y, radius, context) {  
 const prevGlobalCompositeOperation = context.globalCompositeOperation;  
 const prevFillStyle = context.fillStyle;  
 const prevLineWidth = context.lineWidth;  
 const prevLineJoin = context.lineJoin;  
 const prevLineCap = context.lineCap;  
  
 context.globalCompositeOperation = 'destination-out';  
 context.fillStyle = 'rgba(0,0,0,1)';  
 context.lineWidth = radius \* 2;  
 context.lineJoin = 'round';  
 context.lineCap = 'round';  
  
 context.beginPath();  
 context.arc(x, y, radius, 0, *Math*.PI \* 2);  
 context.fill();  
  
 context.globalCompositeOperation = prevGlobalCompositeOperation;  
 context.fillStyle = prevFillStyle;  
 context.lineWidth = prevLineWidth;  
 context.lineJoin = prevLineJoin;  
 context.lineCap = prevLineCap;  
}  
  
function floodFill(startX, startY, fillColor, context) {  
 if (!context) return;  
 const canvas = context.canvas;  
 const imageData = context.getImageData(0, 0, canvas.width, canvas.height);  
 const data = imageData.data;  
 const startPos = (*Math*.floor(startY) \* canvas.width + *Math*.floor(startX)) \* 4;  
 const startR = data[startPos];  
 const startG = data[startPos + 1];  
 const startB = data[startPos + 2];  
 const startA = data[startPos + 3];  
  
 const fillRgb = colorToRgb(fillColor);  
 if (!fillRgb) return;  
 if (startR === fillRgb.r && startG === fillRgb.g && startB === fillRgb.b && startA === 255) return;  
  
 const pixelStack = [[*Math*.floor(startX), *Math*.floor(startY)]];  
 const width = canvas.width;  
  
 while (pixelStack.length) {  
 const newPos = pixelStack.pop();  
 const x = newPos[0];  
 let y = newPos[1];  
 let pixelPos = (y \* width + x) \* 4;  
  
 while (y-- >= 0 && matchStartColor(pixelPos, data, startR, startG, startB, startA)) {  
 pixelPos -= width \* 4;  
 }  
 pixelPos += width \* 4;  
 y++;  
 let reachLeft = false;  
 let reachRight = false;  
  
 while (y++ < canvas.height - 1 && matchStartColor(pixelPos, data, startR, startG, startB, startA)) {  
 colorPixel(pixelPos, data, fillRgb.r, fillRgb.g, fillRgb.b);  
  
 if (x > 0) {  
 if (matchStartColor(pixelPos - 4, data, startR, startG, startB, startA)) {  
 if (!reachLeft) pixelStack.push([x - 1, y]);  
 reachLeft = true;  
 } else if (reachLeft) reachLeft = false;  
 }  
  
 if (x < canvas.width - 1) {  
 if (matchStartColor(pixelPos + 4, data, startR, startG, startB, startA)) {  
 if (!reachRight) pixelStack.push([x + 1, y]);  
 reachRight = true;  
 } else if (reachRight) reachRight = false;  
 }  
  
 pixelPos += width \* 4;  
 }  
 }  
  
 context.putImageData(imageData, 0, 0);  
}  
  
function matchStartColor(pixelPos, data, r, g, b, a) {  
 return data[pixelPos] === r && data[pixelPos + 1] === g && data[pixelPos + 2] === b && data[pixelPos + 3] === a;  
}  
  
function colorPixel(pixelPos, data, r, g, b) {  
 data[pixelPos] = r;  
 data[pixelPos + 1] = g;  
 data[pixelPos + 2] = b;  
 data[pixelPos + 3] = 255;  
}  
  
function applyFilter(filterType, context) {  
 const canvas = context.canvas;  
 const imageData = context.getImageData(0, 0, canvas.width, canvas.height);  
 const data = imageData.data;  
  
 switch (filterType) {  
 case 'grayscale':  
 for (let i = 0; i < data.length; i += 4) {  
 const avg = (data[i] \* 0.3 + data[i + 1] \* 0.59 + data[i + 2] \* 0.11);  
 data[i] = avg;  
 data[i + 1] = avg;  
 data[i + 2] = avg;  
 }  
 break;  
  
 case 'invert':  
 for (let i = 0; i < data.length; i += 4) {  
 data[i] = 255 - data[i];  
 data[i + 1] = 255 - data[i + 1];  
 data[i + 2] = 255 - data[i + 2];  
 }  
 break;  
 }  
  
 context.putImageData(imageData, 0, 0);  
}  
  
export { drawFreehand, drawSpray, drawMarker, erasePixels, floodFill, applyFilter };

**utils.js**

// Утилітні функції для загального використання  
function clamp(value, min, max) {  
 return *Math*.min(*Math*.max(value, min), max);  
}  
  
function getRandomColor() {  
 const letters = '0123456789ABCDEF';  
 let color = '#';  
 for (let i = 0; i < 6; i++) {  
 color += letters[*Math*.floor(*Math*.random() \* 16)];  
 }  
 return color;  
}  
  
function formatTime(seconds) {  
 const mins = *Math*.floor(seconds / 60);  
 const secs = seconds % 60;  
 return `${mins}:${secs < 10 ? '0' : ''}${secs}`;  
}  
  
function isEmpty(obj) {  
 return *Object*.keys(obj).length === 0;  
}  
  
function generateId() {  
 return *Math*.random().toString(36).substr(2, 9);  
}  
  
function rgbToHex(color) {  
 if (color.startsWith('#')) return color;  
 const rgb = color.match(/\d+/g);  
 if (!rgb || rgb.length < 3) return '#000000';  
 const r = parseInt(rgb[0]);  
 const g = parseInt(rgb[1]);  
 const b = parseInt(rgb[2]);  
 return `#${((1 << 24) + (r << 16) + (g << 8) + b).toString(16).slice(1)}`;  
}  
  
function interpolateColor(color1, color2, factor) {  
 if (factor <= 0) return color1;  
 if (factor >= 1) return color2;  
 const r1 = parseInt(color1.substring(1, 3), 16);  
 const g1 = parseInt(color1.substring(3, 5), 16);  
 const b1 = parseInt(color1.substring(5, 7), 16);  
 const r2 = parseInt(color2.substring(1, 3), 16);  
 const g2 = parseInt(color2.substring(3, 5), 16);  
 const b2 = parseInt(color2.substring(5, 7), 16);  
 const r = *Math*.round(r1 + factor \* (r2 - r1));  
 const g = *Math*.round(g1 + factor \* (g2 - g1));  
 const b = *Math*.round(b1 + factor \* (b2 - b1));  
 return `#${((1 << 24) + (r << 16) + (g << 8) + b).toString(16).slice(1)}`;  
}  
  
function colorToRgb(color) {  
 if (color.startsWith('#')) {  
 return {  
 r: parseInt(color.substr(1, 2), 16),  
 g: parseInt(color.substr(3, 2), 16),  
 b: parseInt(color.substr(5, 2), 16)  
 };  
 } else if (color.startsWith('rgb')) {  
 const rgb = color.match(/\d+/g);  
 return {  
 r: parseInt(rgb[0]),  
 g: parseInt(rgb[1]),  
 b: parseInt(rgb[2])  
 };  
 }  
 return null;  
}  
  
// Експорт усіх функцій  
export { clamp, getRandomColor, formatTime, isEmpty, generateId, rgbToHex, interpolateColor, colorToRgb };