КВАЛІФІКАЦІЙНА РОБОТА

ІПЗд-20-2

Скоробогатий Д.І

2024

ЗВО УНІВЕРСИТЕТ КОРОЛЯ ДАНИЛА

Факультет суспільних та прикладних наук

Кафедра інформаційних технологій

на правах рукопису

Скоробогатий Денис Ігорович

УДК 004.4

Розробка простого SVG - редактора

Спеціальність 121 – «Інженерія програмного забезпечення»

Кваліфікаційна робота на здобуття кваліфікації бакалавр

Нормоконтроль

_____Стисло О.В.

(підпис, дата, розшифрування підпису)

Студент

_____ Скоробогатий Д.І.

(підпис, дата, розшифрування підпису)

Допускається до захисту

Завідувач кафедри

_____к.т.н., доц. Ващишак С.П.

(підпис, дата, розшифрування підпису)

Керівник роботи

к.т.н., проф. каф. ІТ

_____ Пашкевич О.П.

(підпис, дата, розшифрування підпису)

Івано-Франківськ – 2024

ЗВО УНІВЕРСИТЕТ КОРОЛЯ ДАНИЛА

Факультет суспільних та прикладних наук

Кафедра інформаційних технологій

Освітній ступінь: «бакалавр»

Спеціальність: 121 «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____

«____» _____ 2024 року

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Скоробогатий Денис Ігорович

(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи:
Розробка простого SVG-редактора

керівник роботи:
Пашкевич Олег Петрович, к.т.н., проф. каф. ІТ

затверджена наказом вищого навчального закладу від « 12 » березня 2024 року № 19/1

2. Термін подання студентом роботи 05.06.2024

3. Вихідні дані роботи: мова програмування TypeScript, React,Redux.

4. Зміст кваліфікаційної роботи (перелік питань, які потрібно розробити)

1. Огляд існуючих SVG редакторів
2. Розробка та реалізація SVG редактору
3. Реалізація інтерфейсу та функціоналу редактора

5. Дата видачі завдання 14.03.2024

КОНСУЛЬТАНТИ РОЗДІЛІВ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Розділ	Консультант (прізвище, ініціали та посада)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

	Назва етапів кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
	Огляд існуючих аналогів	20.03.2024	Виконано
	Проектування та розробка прототипу	29.03.2024	Виконано
	Розробка дизайну редактора	12.04.2024	Виконано
	Проектування та розробка редактора	25.04.2024	Виконано
	Оформлення пояснювальної записки	03.05.2024	Виконано
	Оформлення графічного матеріалу та підготовка до захисту роботи	20.05.2024	Виконано

Студент _____ Скоробогатий Д.І.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Пашкевич О.П.
(підпис) (прізвище та ініціали)

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

Сторінка	Опис графічного матеріалу
13	Інтерфейс Adobe Illustrator
13	Інтерфейс Inkscape
14	Інтерфейс Gravit Designer
14	Інтерфейс Sketch
21	Scrum vs Kanban
22	Інтерфейс Simple SVG Editor
23	React, Redux
25	Інтерфейс робочої області Simple SVG Editor
25	Приклад використання fabric.js
29	Інструкція на стартовій сторінці для нових користувачів
32	UploadComponent

АНОТАЦІЯ

Кваліфікаційна робота присвячена розробці простого SVG редактора на платформі React з використанням TypeScript, бібліотеки fabric.js та Redux Toolkit. У роботі детально описані основні етапи створення та функціональність редактора, зокрема можливість перетягування SVG-елементів, зміна їхнього розміру та обертання, налаштування кольорів та borders об'єктів.

Також проведено аналіз сучасних тенденцій у розробці веб-додатків та особливостей ринку графічних редакторів. Результати роботи можуть бути використані для покращення користувацького досвіду у веб-редакторах та розширення можливостей веб-розробки з використанням сучасних технологій.
Проект включає в себе аналіз вимог користувачів до функціоналу редактора SVG, детальний опис розроблених компонентів та їх взаємодії, а також результати тестування та висновки щодо досягнутих цілей.

Результати цієї роботи можуть бути корисними для розробників, які планують створити власний SVG редактор на основі платформи React з використанням TypeScript та інших сучасних інструментів. Також робота може послужити основою для подальших досліджень у галузі розробки веб-додатків та графічного дизайну.

КЛЮЧОВІ СЛОВА: REACT, TYPESCRIPT, SVG, FABRIC.JS, REDUX TOOLKIT, ВЕБ-РОЗРОБКА, ГРАФІЧНИЙ РЕДАКТОР, КОРИСТУВАЦЬКИЙ ДОСВІД.

SUMMARY

The qualification work is devoted to the development of a simple SVG editor on the React platform using TypeScript, the fabric.js library, and the Redux Toolkit. The paper describes in detail the main stages of creation and functionality of the editor, including the ability to drag and drop SVG elements, resize and rotate them, set colors and borders of objects.

The article also analyzes current trends in web application development and the features of the graphic editor market. The results of the work can be used to improve the user experience in web editors and expand the capabilities of web development using modern technologies.

The project includes an analysis of user requirements for the SVG editor functionality, a detailed description of the developed components and their interaction, as well as test results and conclusions about the achieved goals.

The results of this work can be useful for developers who plan to create their own SVG editor based on the React platform using TypeScript and other modern tools. The work can also serve as a basis for further research in the field of web application development and graphic design.

KEYWORDS: REACT, TYPESCRIPT, SVG, FABRIC.JS, REDUX TOOLKIT, WEB DEVELOPMENT, GRAPHIC EDITOR, USER EXPERIENCE.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ..... 7

ВСТУП..... 8

РОЗДІЛ 1. ОГЛЯД ІСНУЮЧИХ SVG РЕДАКТОРІВ..... 11

1.1 Загальна характеристика та опис простого SVG редактора..... 11

1.2 Огляд та аналіз існуючих аналогів..... 12

1.3 Вибір та опис використаних технологій (React, fabric.js, RTK)..... 15

1.4 Методи дослідження..... 17

Висновок до розділу 1..... 18

РОЗДІЛ 2. РОЗРОБКА ТА РЕАЛІЗАЦІЯ SVG РЕДАКТОРУ..... 19

2.1 Створення user stories для функціоналу редактору..... 19

2.2 Огляд методу розробки та дизайну інтерфейсу..... 21

2.3 Детальний огляд бібліотеки Fabric.js..... 25

Висновок до розділу 2..... 26

РОЗДІЛ 3. РЕАЛІЗАЦІЯ ІНТЕРФЕЙСУ ТА ФУНКЦІОНАЛУ РЕДАКТОРА..... 27

3.1 Реалізація інтерфейсу користувача..... 27

3.2 Реалізація основних функцій редактора (маніпуляції з фігурами, зміна параметрів)..... 32

3.3 Реалізація маніпуляції даними за допомогою RTK..... 39

3.4 Висновки щодо якості роботи редактору та його можливостей..... 45

Висновок до розділу 3..... 46

ВИСНОВКИ..... 47

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ..... 48

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

TS – TypeScript

SVG – Scalable Vector Graphics

React – JavaScript-бібліотека для створення користувацького інтерфейсу

RTK – Бібліотека управління станом додатку в React

HTML – Мова розмітки гіпертексту

CSS – Каскадні таблиці стилів

UI – Інтерфейс користувача

API – Інтерфейс програмування додатків

DOM – Об’єктна модель документа

IDE – Інтегроване середовище розробки

ВСТУП

Актуальність теми полягає в зростаючій потребі у створенні і редагуванні векторних зображень у веб-розробці та дизайні. SVG (масштабована векторна графіка) стала популярним форматом для представлення графічної інформації в Інтернеті через свою масштабованість і здатність пристосовуватись до різних розмірів екранів без втрати якості.

Особливо зростає попит на прості та легкі у використанні інструменти для редагування SVG-зображень, що дозволяють веб-розробникам та дизайнерам швидко створювати і адаптувати графічні елементи для веб-сайтів і веб-додатків. У зв'язку з поширенням мобільних пристроїв і різноманіттям їх розмірів екранів, векторна графіка стала надзвичайно важливою, а простий SVG-редактор може стати корисним інструментом для широкого кола користувачів, які працюють у вебі.

Такий редактор дозволить не лише створювати нові векторні зображення, а й легко вносити зміни у вже існуючі SVG-файли без необхідності великих програм або спеціалізованих навичок. Його створення відповідає потребам широкого кола користувачів в створенні інструментів для веб-розробки, що забезпечують ефективність та продуктивність у процесі створення веб-контенту.

Мета роботи. Мета цієї кваліфікаційної роботи полягає у створенні простого SVG редактора, спрямованого на створення векторних зображень без складнощів, що часто виникають при використанні великих або складних графічних програм. Основна мета полягає в розробці інструменту, який буде легким у використанні, зрозумілим для широкого кола користувачів, і при цьому забезпечить достатній функціонал для створення векторних зображень.

Створення простого SVG редактора важливе у контексті сучасних веб-технологій та тенденцій у розробці веб-контенту. Завдяки зростанню інтересу до векторної графіки на веб-сторінках та веб-додатках, інструменти

для створення такого контенту стають дедалі потрібнішими. Зокрема, для веб-розробників, дизайнерів та інших фахівців, які створюють інтерактивний вміст, простий інструмент для створення векторних зображень може забезпечити зручність та продуктивність у роботі.

Мета даного проекту також включає дослідження технологій, для реалізації SVG редактора, таких як JavaScript, HTML5 і CSS. Важливо розуміти, як ці технології можуть бути ефективно використані для створення інтерфейсу редактора, забезпечуючи його швидкість, простоту та функціональність.

Отже, розробка простого SVG редактора стане значним внеском у сферу веб-розробки та дизайну, дозволяючи користувачам легко створювати векторні зображення без складних інструментів, і сприятиме подальшому розвитку інтерактивного вмісту в Інтернеті.

Об’єктом дослідження: є створення простого SVG редактора, спрямованого на забезпечення користувачам можливості швидко створювати, редагувати та адаптувати векторні зображення без використання складних графічних програм. Досліджується створення інструменту, який поєднує в собі простоту використання, широкий функціонал для створення векторних зображень та оптимальне використання сучасних технологій у веб-розробці.

Завдання роботи полягає у розробці простого SVG редактора з такими конкретними завданнями:

1. Проведення аналізу сучасних існуючих SVG редакторів та їх функціональності.
2. Вибір оптимальних інструментів, мов програмування (наприклад, JavaScript), технологій та бібліотек для розробки SVG редактора.
3. Розробка прототипу SVG редактора з основним функціоналом, включаючи малювання простих векторних форм, редагування та видалення елементів, зміну кольору та розміру.
4. Створення зручного, привабливого та сучасного дизайну інтерфейсу редактора, з урахуванням простоти використання та естетичних аспектів.

Ці завдання спрямовані на створення функціонального та ефективного інструменту для створення векторних зображень, який буде корисним для широкого кола користувачів у сфері веб-розробки та дизайну.

Предметом дослідження: функціональність та ефективність простого SVG редактора у веб-розробці та дизайні. Досліджується можливість створення векторних зображень, їхнє редагування та використання на веб-сторінках і веб-додатках. Вивчаються особливості інтерфейсу користувача, оптимальність алгоритмів обробки графіки та можливості інтеграції у сучасні веб-середовища.

Методи дослідження: дослідження базується на комплексному аналізі сучасних SVG редакторів, огляді функціональних можливостей інструментів для векторного малювання, а також експериментальному порівнянні різних технологій і бібліотек для розробки SVG редактора. Використовуються методи дослідження, такі як аналіз програмного коду, експериментальні тести та анкетування для оцінки користувацьких потреб та задоволення від використання інструменту.

Структура роботи. Розділи – 3. Обсяг основної частини – 43 сторінок. Список використаних джерел – 20.

РОЗДІЛ 1. ОГЛЯД ІСНУЮЧИХ SVG РЕДАКТОРІВ

1.1 Загальна характеристика та опис простого SVG редактора

Прості SVG редактори використовуються для створення, редагування та маніпулювання векторними графічними об'єктами у форматі SVG. Вони забезпечують доступ до базових інструментів для створення різноманітних фігур, тексту, ліній і зображень, які можна використовувати на веб-сайтах, у веб-додатках та інших проектах, де потрібні векторні зображення [1].

Типовий простий SVG редактор має інтуїтивний інтерфейс, що дозволяє швидко навчитися його використовувати навіть початківцям. Він часто має наступні функції:

- створення фігур: можливість малювати прості геометричні об'єкти, такі як круги, прямокутники, трикутники, лінії тощо:
- редагування параметрів: зміна кольору, товщини контуру, заливки, а також інших властивостей об'єктів;
- маніпуляція з об'єктами: переміщення, обертання, масштабування та видалення об'єктів на полотні;
- експорт імпорт: можливість зберігати та завантажувати створені зображення у форматі SVG.

Такий редактор корисний для веб-розробників, що створюють інтерактивні графічні елементи на веб-сайтах, а також для графічних дизайнерів, які потребують швидко створювати і змінювати векторні зображення [2]. Він дозволяє швидко втілювати ідеї у векторну графіку без необхідності використання складних програм.

Прості SVG редактори відрізняються від більш складних програм для векторного дизайну тим, що вони зазвичай мають обмежений набір інструментів і функцій, спрощений інтерфейс та менше вимог до системи. Це

робить їх ідеальними для швидкої створення простих векторних зображень без потреби в складних налаштуваннях чи великих навичках використання графічних програм.

Загальна характеристика простого SVG редактора полягає в тому, що він зосереджений на базових функціях створення та редагування векторних зображень, не перевантажуючи користувача зайвою функціональністю. Такий підхід робить редактори зручними для широкого кола користувачів, включаючи веб-розробників, графічних дизайнерів, викладачів та студентів, які шукають простий спосіб створення векторних графічних елементів.

Основна перевага простих SVG редакторів полягає в їхній легкості використання та доступності. Вони дозволяють швидко втілити ідеї в життя без великих затрат часу і зусиль. Такі редактори стають важливим інструментом для тих, хто шукає простий спосіб створити або відредагувати векторне зображення для вебу або інших цифрових проектів.

1.2 Огляд та аналіз існуючих аналогів

В сучасному світі векторна графіка відіграє важливу роль у багатьох галузях, включаючи дизайн, архітектуру, інженерію та інше. SVG (Scalable Vector Graphics) є основним форматом для векторної графіки в інтернеті. Є багато редакторів SVG, які надають користувачам можливість створювати та редагувати векторні зображення. Однак, багато з цих інструментів можуть бути занадто складними для користувачів, особливо для тих, хто тільки починає працювати з векторною графікою [3].

Ось короткий огляд деяких SVG редакторів, з акцентом на їх складність:

Adobe Illustrator: цей редактор має велику кількість функцій, що може бути перевагою для професіоналів, але занадто складним для новачків. Крім того, висока вартість ліцензії може бути недоступною користувачів (рис. 1.1).

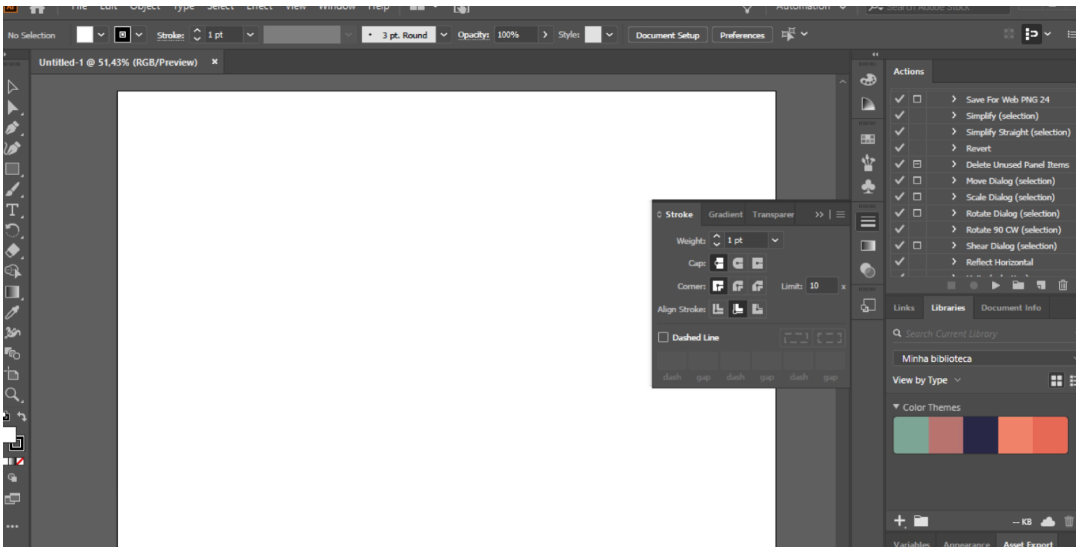


Рисунок 1.1 – Інтерфейс **Adobe Illustrator**

Inkscape: цей безкоштовний редактор з відкритим вихідним кодом має багатий функціонал, але може працювати повільно при роботі з великими файлами або складними проектами (рис. 1.2).

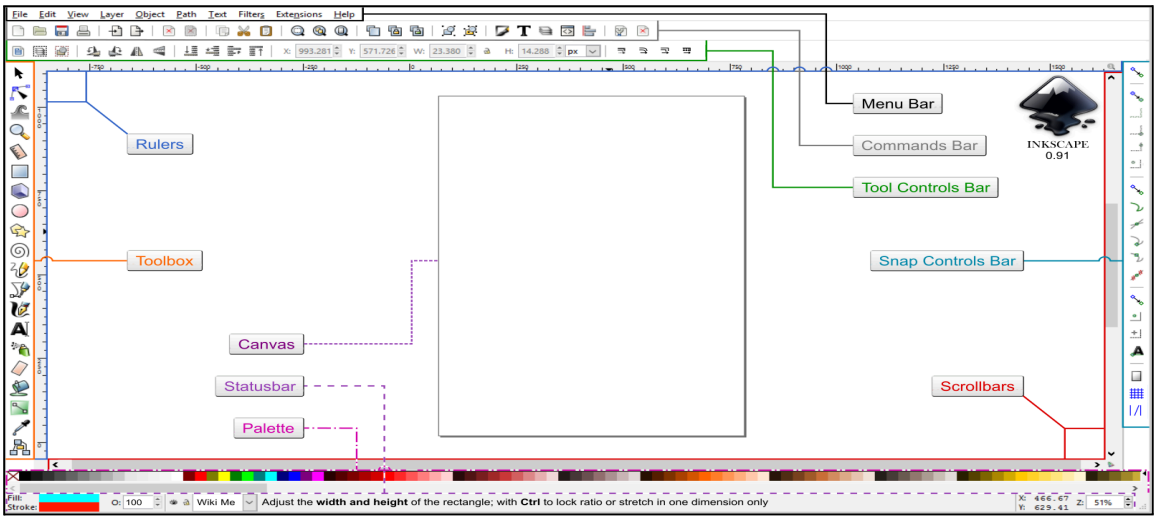


Рисунок 1.2 – І нтерфейс **Inkscape**

Gravit Designer: цей безкоштовний редактор пропонує онлайн-інструменти та настільну версію, але його онлайн-версія може мати обмеження у функціоналі та продуктивності (рис. 1.3).

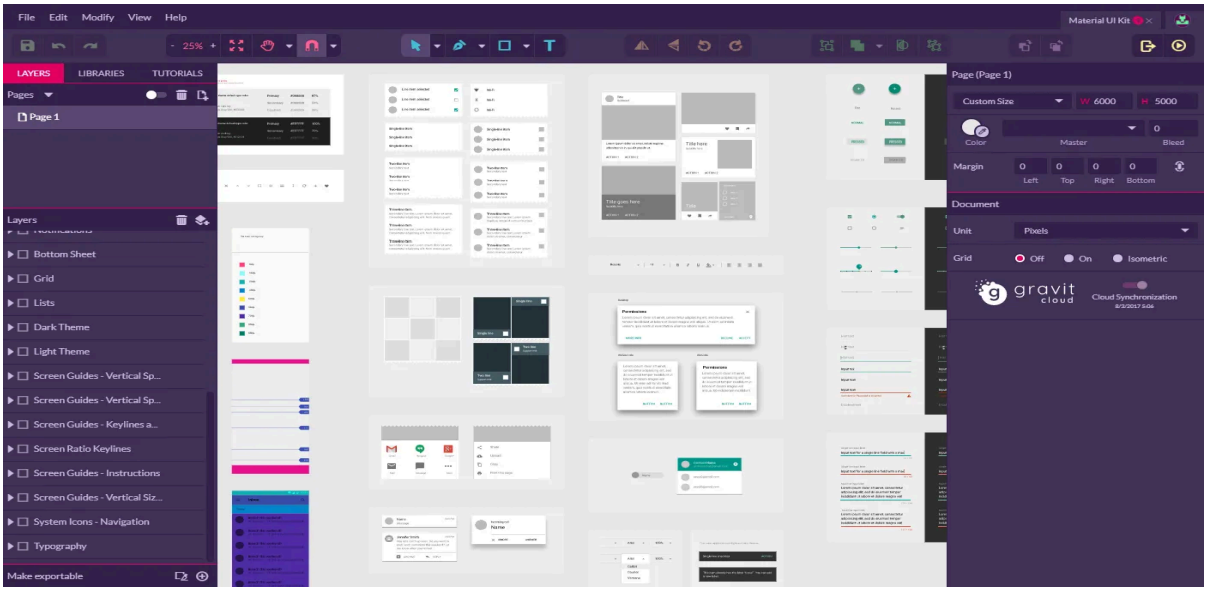


Рисунок 1.3 – Інтерфейс **Gravit Designer**

Sketch: інструмент орієнтований на веб та мобільні додатки, але також може використовуватися для роботи з векторною графікою. Проте, його обмежена підтримка векторних форматів та вартість ліцензії можуть зробити привабливішим для користувачів, що працюють з векторами (рис. 1.4).

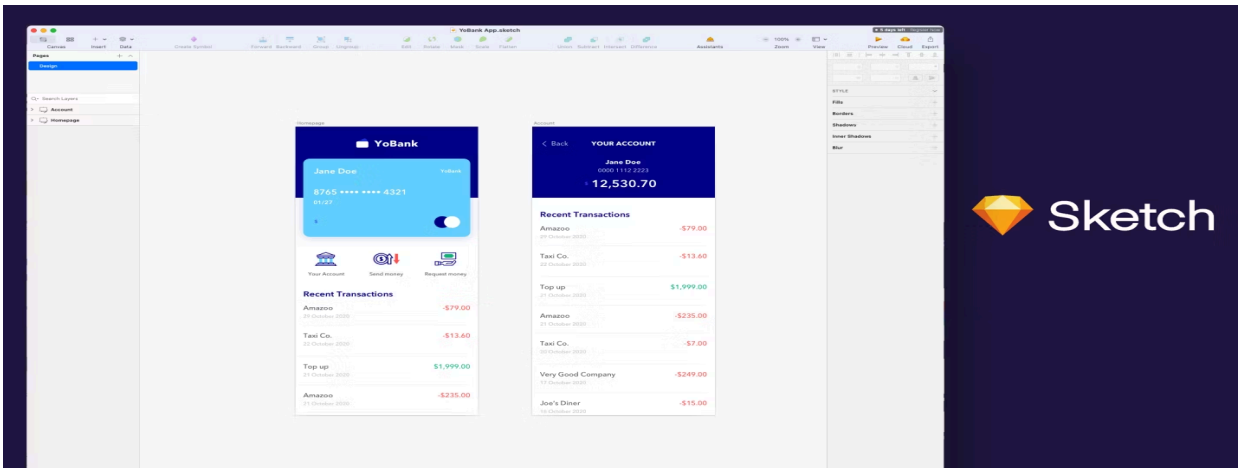


Рисунок 1.4 – Інтерфейс **Sketch**

Ці редактори мають свої переваги та недоліки, тому вибір конкретного інструменту залежить від конкретних потреб та вимог користувача. Однак, є

потреба в простому, але потужному SVG редакторі, який би був доступним та зручним для користувачів різного рівня.

1.3 Вибір та опис використаних технологій (React, fabric.js, RTK)

Для розробки нашого Simple SVG Editor ми обрали набір технологій, які дозволяють нам ефективно працювати з векторною графікою та забезпечують зручний інтерфейс для користувачів. Основні компоненти стеку включають:

React TS (TypeScript): React TS був обраний як основною бібліотекою для розробки користувацького інтерфейсу SVG редактора. TypeScript дозволяє покращити якість коду та забезпечує типізацію, що спрощує відлагодження і підтримку додатку. Він допомагає нам створювати надійні та безпечні додатки.

fabric.js: fabric.js є потужною бібліотекою JavaScript, спеціалізованою на роботі з векторною графікою. Ця бібліотека надає зручний інтерфейс для маніпулювання графічними об’єктами, створенням фігур, редагуванням параметрів та іншими операціями. Вона дозволяє нам легко і швидко створювати та редагувати SVG зображення [4].

Redux Toolkit: Redux Toolkit використовується для керування станом додатку та забезпечення передбачуваності взаємодії компонентів. Використання Redux спрощує управління станом додатку та дозволяє ефективно синхронізувати дані між різними частинами редактора. Це допомагає нам забезпечити стабільність та надійність нашого SVG редактора.

SCSS (Sassy CSS): SCSS є препроцесором CSS, який додає багато корисних функцій до звичайного CSS, таких як змінні, вкладеність, міксини, спадкування та інше [5]. Він допомагає розробникам писати більш організований та читабельний код, що полегшує розробку та підтримку стилів. У контексті нашого Simple SVG Editor, SCSS може бути використаний для створення та управління стилями інтерфейсу користувача.

Vite: Vite є сучасним інструментом для збірки та розробки, який надає швидке середовище розробки з гарячим перезавантаженням модулів та багатьма іншими функціями. Він використовує ES Modules (ESM) для надання швидкого часу завантаження та багатьох оптимізацій для продакшну. У контексті нашого Simple SVG Editor, Vite може бути використаний для ефективної розробки та збірки проекту.

Ant Design: це одна з найпопулярніших бібліотек компонентів для розробки інтерфейсів, зокрема на основі React [6]. Вона надає велику кількість готових до використання компонентів, що дозволяє значно прискорити процес розробки та забезпечити консистентність інтерфейсу.

Основні переваги Ant Design включають:

1. Велику кількість компонентів: Ant Design має більше 50 готових до використання компонентів, що включають кнопки, меню, форми, таблиці, модальні вікна, тултипи та багато іншого. Це дозволяє розробникам створювати складні інтерфейси без необхідності створювати багато компонентів з нуля.

2. Консистентність дизайну: всі компоненти в Ant Design розроблені згідно з одними дизайн-правилами, що забезпечує консистентність інтерфейсу. Це дозволяє створювати професійно виглядаючі додатки без необхідності використовувати додаткові CSS-стилі [7].

3. Інтеграція з React: Ant Design ідеально підходить для розробки React-додатків, оскільки всі його компоненти розроблені з використанням React. Це дозволяє легко інтегрувати Ant Design у будь-який React-додаток.

4. Підтримка інтернаціоналізації: Ant Design підтримує багато мов, що дозволяє розробникам створювати додатки для міжнародної аудиторії.

У контексті нашого Simple SVG Editor, Ant Design може бути використаний для створення інтерфейсу редактора. Його компоненти можуть бути використані для створення панелей інструментів, меню, діалогових вікон та інших елементів інтерфейсу. Крім того, завдяки його інтеграції з React, Ant Design може бути легко інтегрований з іншими частинами нашого додатку, які

розроблені з використанням React і Redux Toolkit. Загалом, Ant Design допомагає нам створити зручний, привабливий та консистентний інтерфейс для нашого SVG редактора.

Ці технології були обрані через їхню потужність, гнучкість та популярність у веб-розробці. Комбінація React TS, fabric.js і Redux Toolkit надає зручність у розробці та забезпечує оптимальний досвід користувача при роботі з нашим Simple SVG Editor [8]. Вони допомагають нам створити продукт, який відповідає потребам користувачів і є простим у використанні, незважаючи на складність задач, які він вирішує.

1.4 Методи дослідження

Для розробки та аналізу простого SVG редактора було використано різноманітні методи дослідження, спрямовані на оцінку можливостей, визначення вимог і аналіз конкурентного середовища. Основні методи дослідження включають:

1. **Огляд літератури та аналіз існуючих рішень:** цей метод включає в себе детальний огляд наукових статей, технічних документів, блогів, відео та інших ресурсів, що стосуються SVG редакторів. Мета цього методу - зрозуміти основні принципи роботи SVG редакторів, визначити їхні основні функції та можливості, а також виявити найкращі практики та загальні підходи до розробки таких редакторів [9].
2. **Експертна оцінка інструментів:** цей метод включає в себе детальний аналіз різних інструментів та технологій, які можуть бути використані для розробки SVG редактора. Мета цього методу - визначити найкращі інструменти для розробки нашого SVG редактора, враховуючи такі фактори, як їхні можливості, продуктивність, надійність, підтримка спільноти та інше.

3. **Аналіз конкурентів існуючих рішень:** цей метод включає в себе порівняльний аналіз існуючих SVG редакторів. Мета цього методу - визначити їхні сильні та слабкі сторони, виявити унікальні функції та можливості, а також зрозуміти, як вони вирішують проблеми користувачів. Це допомагає нам краще зрозуміти потреби користувачів та визначити, які функції та можливості ми повинні включити в наш SVG редактор.

Користувацький опитувальник і спостереження за користувачами: цей метод включає в себе проведення опитувань серед потенційних користувачів SVG редактора, а також спостереження за їхніми діями під час використання існуючих SVG редакторів. Мета цього методу - визначити основні вимоги користувачів до функціоналу та інтерфейсу SVG редактора, а також виявити потенційні проблеми та виклики, з якими вони можуть зіткнутися.

Висновок до розділу 1

У першому розділі роботи було проведено огляд існуючих SVG редакторів, аналіз їхніх можливостей та обмежень. Було виявлено, що більшість сучасних редакторів забезпечують основні функції для створення та редагування векторних зображень, проте не завжди мають інтуїтивно зрозумілий інтерфейс і достатню гнучкість для інтеграції в сучасні веб-додатки. Вивчено основні технології, які використовуються для розробки таких редакторів, зокрема React, TypeScript, Fabric.js та Redux Toolkit, що дозволяє забезпечити високу продуктивність та розширюваність додатків.

РОЗДІЛ 2. РОЗРОБКА ТА РЕАЛІЗАЦІЯ SVG РЕДАКТОРУ

2.1 Створення user stories для функціоналу редактору

Для розробки простого SVG редактора були сформульовані user stories, які визначають вимоги та очікувані можливості програмного забезпечення з точки зору користувача [10]. User stories допомагають чітко описати, які можливості повинен мати редактор і як вони повинні взаємодіяти з користувачем.

User story (історія користувача) – це короткий опис функціональної вимоги до програмного забезпечення, яка формулюється з точки зору користувача. Це інструмент управління вимогами, який допомагає команді розробників розуміти потреби і очікування користувачів продукту.

Основна мета user story полягає в описі конкретного сценарію використання програмного продукту з погляду його користувачів. Кожна історія користувача включає в себе короткий текст, що описує конкретну функцію або можливість, яку користувач хоче бачити в програмі. User story допомагає визначити, що саме потрібно реалізувати розробникам для того, щоб задовольнити потреби своїх користувачів.

Основні складові user story включають:

Роль користувача: хто є користувачем програмного продукту і які є його потреби чи очікування.

Дія користувача: що саме користувач хоче зробити або яку функцію хоче використати в програмі.

Бажаний результат: який результат очікується від виконання функції.

Основні user stories для простого SVG редактора включають:

1. **User Story 1:** я, як користувач, хочу мати можливість створювати нові фігури. Ця user story відображає основну можливість SVG редактора -

створення нових фігур. Користувач повинен мати змогу легко та швидко створювати нові фігури, вибираючи з різних типів (наприклад, прямокутники, кола, лінії), та розміщувати їх на полотні редактора.

2. **User Story 2:** я, як користувач, хочу мати можливість змінювати параметри фігур. Користувач повинен мати змогу легко змінювати параметри фігур, такі як розмір, колір, товщина ліній тощо. Це включає можливість змінювати колір SVG та колір бордеру, а також ширину бордера.

3. **User Story 3:** я, як користувач, хочу мати можливість переміщувати та змінювати масштаб фігур. Користувач повинен мати змогу переміщувати фігури по полотні редактора, а також змінювати їх масштаб. Це дозволяє користувачу контролювати розташування та розмір фігур на полотні, що забезпечує більшу гнучкість при створенні SVG зображень.

4. **User Story 4:** я, як користувач, хочу мати можливість видаляти фігури. Ця user story відображає потребу користувача видаляти непотрібні фігури з полотна редактора. Це дозволяє користувачу легко очищувати полотно та управляти елементами SVG зображення.

5. **User Story 5:** я, як користувач, хочу мати можливість зберігати та завантажувати SVG файли. Користувач повинен мати змогу зберігати свої роботи у форматі SVG файлу та завантажувати збережені проекти для подальшої роботи. Це дозволяє користувачу продовжувати роботу над проектами після закриття редактора та ділитися SVG зображеннями з іншими.

6. **User Story 6:** я, як користувач, хочу мати можливість відкривати і редагувати існуючі SVG файли. Ця user story відображає потребу користувача відкривати та редагувати існуючі SVG файли. Це дозволяє користувачу використовувати редактор для редагування SVG зображень, створених в інших програмах або завантажених з Інтернету.

Ці user stories визначають основний функціонал редактора, який повинен бути реалізований під час розробки. Вони відображають конкретні сценарії використання редактора і важливі для забезпечення зручності та ефективності

користування продуктом. Завдяки цим user stories, ми можемо краще зрозуміти потреби користувачів та визначити, які функції та можливості ми повинні включити в наш SVG редактор.

2.2 Огляд методу розробки та дизайну інтерфейсу

Під час розробки простого SVG редактора важливо визначити ефективні методи розробки програмного забезпечення та підходи до створення інтуїтивного і привабливого інтерфейсу користувача [11]. Цей розділ допоможе краще зрозуміти, як команди розробників обирають та застосовують методики розробки та дизайну для досягнення високої якості продукту.

Методи розробки:

– огляд методологій розробки програмного забезпечення: Scrum, Kanban та інші методології розробки програмного забезпечення використовуються для ефективного управління процесом розробки. Вони допомагають командам розробників краще розуміти вимоги до продукту, планувати роботу, відстежувати прогрес та реагувати на зміни (рис. 2.1);

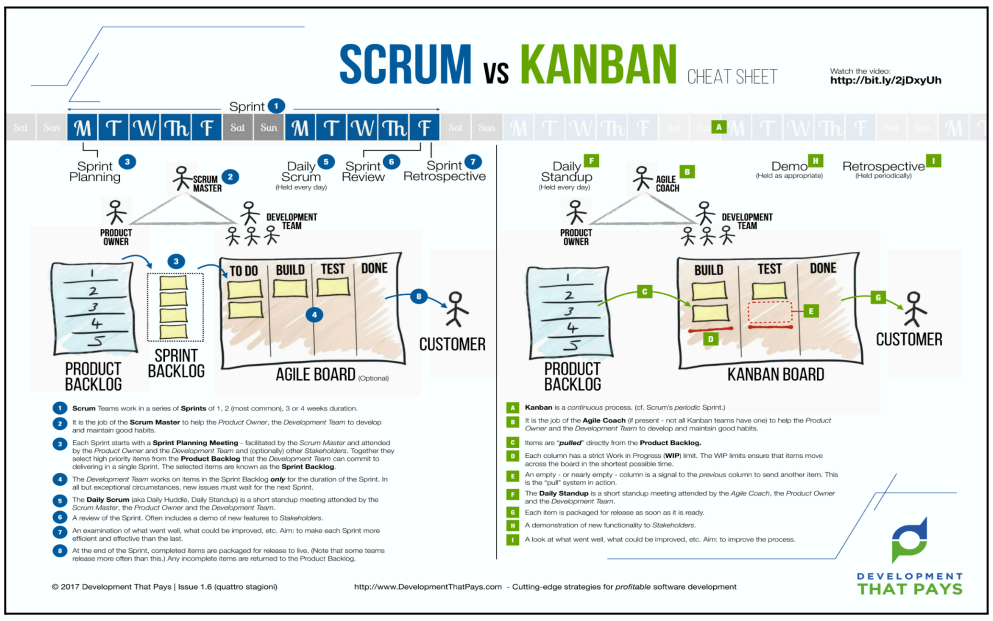


Рисунок 2.1 – Scrum vs Kanban

– розгляд підходів до розробки frontend частини програми: React та Redux Toolkit - це потужні інструменти для розробки інтерфейсу користувача. Вони дозволяють створювати інтерактивні, відповідні та веб-додатки;

– опис методів роботи з ітераціями, тестуванням, використанням компонентної архітектури: ітеративний підхід до розробки, систематичне тестування та використання компонентної архітектури допомагають забезпечити високу якість коду, гнучкість та масштабованість продукту.

Дизайн інтерфейсу:

– огляд сучасних тенденцій у дизайні веб-інтерфейсів: сучасні тенденції в дизайні веб-інтерфейсів, такі як матеріальний дизайн, плоский дизайн, градієнти, анімація та інше, використовуються для створення привабливого та зручного для користувача інтерфейсу;

– підходи до створення зручного та естетичного інтерфейсу користувача: використання принципів UX (User Experience) та UI (User Interface) дизайну допомагає створити інтерфейс, який є не тільки привабливим, але й зручним та інтуїтивно зрозумілим для користувачів (рис. 2.2);

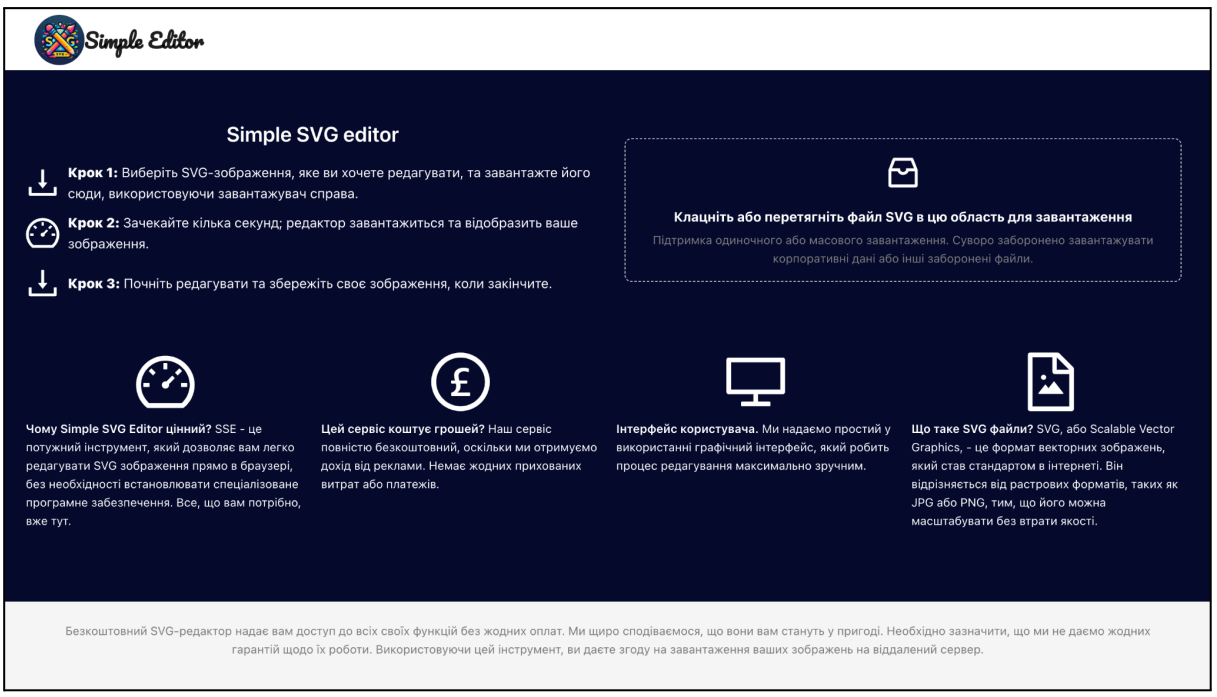


Рисунок 2.2 – Інтерфейс Simple SVG Editor

– використання бібліотеки fabric.js для реалізації графічного інтерфейсу редактора: fabric.js - це потужна бібліотека JavaScript для роботи з векторною графікою;

– розгляд та обгрунтування вибору React TS, Redux Toolkit та інших інструментів у контексті розробки SVG редактора: вибір технологій для розробки продукту залежить від багатьох факторів, включаючи вимоги до продукту, досвід команди, вимоги до продуктивності та інше;

– react TS, RTK та інші інструменти надають нам потрібну гнучкість, продуктивність та надійність для розробки SVG редактора (рис. 2.3);

–

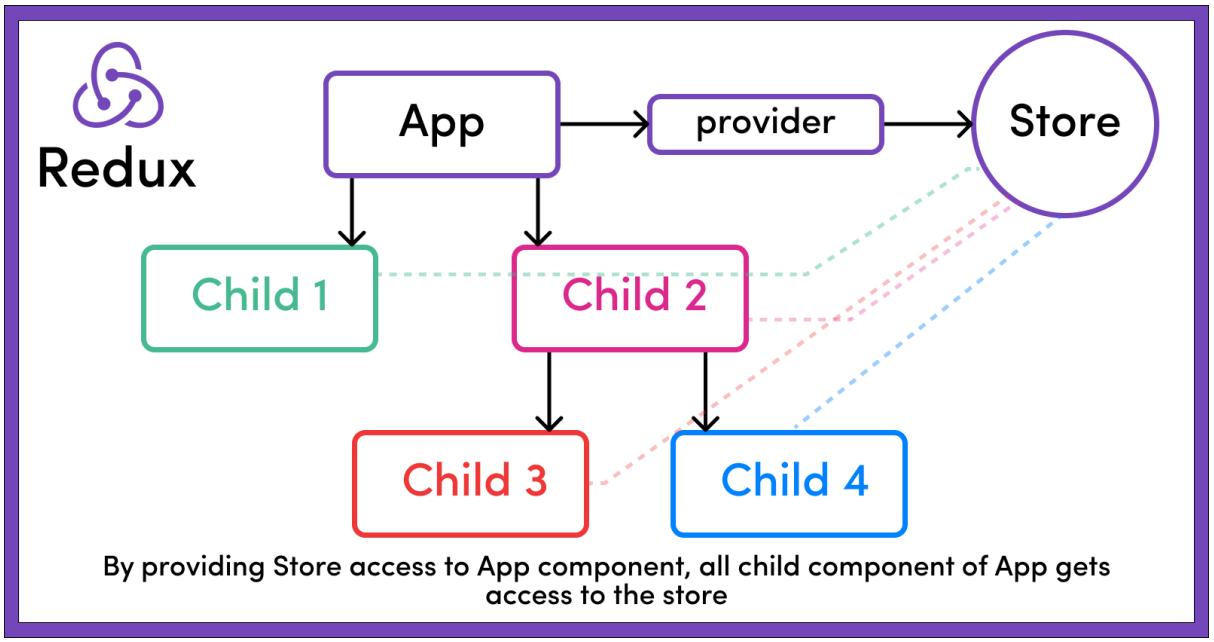


Рисунок 2.3 – React, Redux

– опис основних переваг і обмежень кожної обраної технології. Кожна технологія має свої переваги та обмеження. Наприклад, React дозволяє нам створювати високопродуктивні, відповідні веб-додатки з використанням сучасного JavaScript, але вимагає від нас глибокого розуміння JavaScript та його екосистеми. З іншого боку, Redux Toolkit допомагає нам ефективно управляти станом додатку, але може бути складним для вивчення для новачків.

Порівняння альтернативних підходів і технологій, які розглядалися під час вибору стеку технологій для проекту. Під час вибору технологій для нашого проекту ми розглядали різні альтернативи. Наприклад, ми могли використовувати Angular замість React, або MobX замість Redux. Однак, після детального аналізу ми вирішили, що React TS та Redux Toolkit найкраще відповідають нашим потребам [12].

Процес дизайну інтерфейсу:

1. Огляд процесу проектування інтерфейсу з використанням UX/UI підходів: процес дизайну інтерфейсу включає в себе ряд етапів, включаючи дослідження користувачів, створення прототипів, тестування та ітерацію. Ми використовуємо принципи UX (User Experience) та UI (User Interface) дизайну для створення інтерфейсу, який є не тільки привабливим, але й зручним та інтуїтивно зрозумілим для користувачів.

2. Пояснення структури інтерфейсу редактора та його компонентів: інтерфейс нашого SVG редактора складається з ряду компонентів, включаючи панель інструментів, поле для малювання, панель властивостей та інше. Кожен компонент має свою роль та функції, і разом вони створюють зручний та гнучкий інтерфейс для користувачів.

3. Опис робочих процесів, які використовуються під час розробки графічного інтерфейсу: розробка графічного інтерфейсу - це ітеративний процес, який включає в себе створення прототипів, тестування з користувачами, отримання відгуків та внесення відповідних змін [13]. Ми використовуємо різні інструменти та техніки для ефективної роботи над дизайном інтерфейсу, включаючи скетчинг, wireframing, prototyping, user testing та інше.

Розділ підкреслює важливість обрання правильних підходів до розробки програмного забезпечення та створення привабливого інтерфейсу. Розглянуті методи допоможуть забезпечити ефективну роботу команди розробників та створити зручний та сучасний SVG редактор. Інтерфейс робочої області нашого редактора представлено на рисунку 2.4.

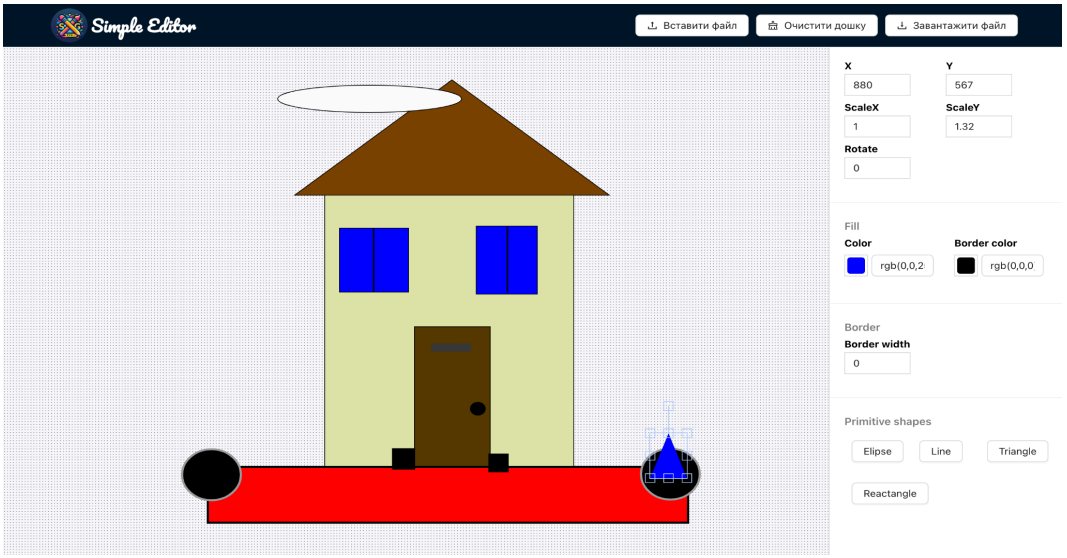


Рисунок 2.4 – Інтерфейс робочої області Simple SVG Editor

2.3 Детальний огляд бібліотеки Fabric.js

Бібліотека Fabric.js є потужним інструментом для роботи з векторною графікою у веб-додатках. Вона надає розробникам зручний API для створення і маніпулювання графічними об'єктами, що робить її ідеальним вибором для реалізації графічного інтерфейсу SVG редактора (рис. 2.5).

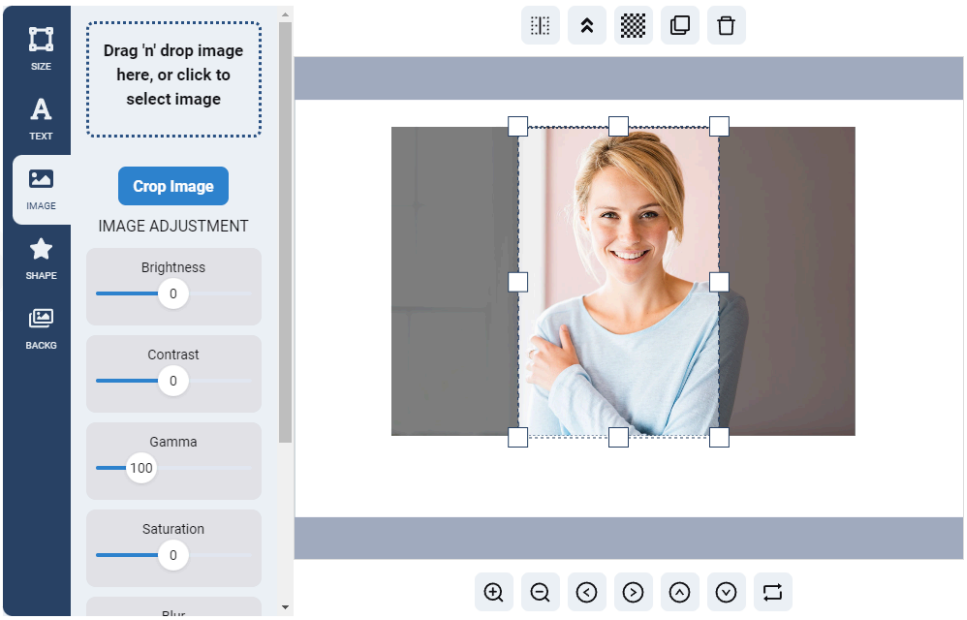


Рисунок 2.5 – Приклад використання fabric.js

Основні переваги використання Fabric.js включають:

1. **Підтримка багатошаровості:** бібліотека дозволяє працювати зі шарами, що дозволяє керувати порядком відображення графічних об'єктів. Це дозволяє створювати складні композиції та взаємодію об'єктів.

2. Великий вибір графічних ефектів: Fabric.js підтримує різні графічні ефекти, такі як заливка, обводка, трансформації та інші[14]. Це дозволяє створювати візуальні ефекти без необхідності писати складний код з нуля.

3. **Підтримка різних типів об'єктів:** Fabric.js пропонує широкий вибір готових типів графічних об'єктів, таких як прямокутники, кола, лінії, текст і багато інших. Це дозволяє швидко створювати різноманітні елементи у своєму SVG редакторі [15].

Можливості редагування і трансформації: Fabric.js використовується багатьма веб-розробниками для створення інтерактивних інтерфейсів та графічних додатків через свою потужну функціональність і зручний інтерфейс програмування[16]. Вона є відмінним інструментом для реалізації SVG редактора з багатьма можливостями для творчості та взаємодії з графікою.

Висновок до розділу 2

Другий розділ був присвячений розробці та технічній реалізації простого SVG редактора. В ході роботи було створено прототип, який включає базовий функціонал, зокрема малювання, редагування та видалення елементів, а також зміну кольору та розміру об'єктів. Розроблено зручний та сучасний дизайн інтерфейсу, що враховує простоту використання та естетичні аспекти.

РОЗДІЛ 3. РЕАЛІЗАЦІЯ ІНТЕРФЕЙСУ ТА ФУНКЦІОНАЛУ РЕДАКТОРА

3.1 Реалізація інтерфейсу користувача

Інтерфейс користувача є одним з найважливіших аспектів будь-якого програмного продукту. Він визначає спосіб взаємодії користувача з додатком і впливає на загальний досвід його використання [17]. Навіть найпотужніші функції можуть втрачати значення, якщо інтерфейс не інтуїтивний або важкий для використання. Тому створення зручного і ефективного інтерфейсу є ключовим завданням для розробників програмного забезпечення. Давайте розглянемо докладніше, чому інтерфейс користувача так важливий у контексті розробки SVG редактора.

На початковій сторінці рисунок 2.2 ми можемо спостерігати простий та зрозумілий для нових користувачів інтерфейс який складається з header, content, footer. Header містить у собі логотип. А content коротку інструкцію для нових користувачів та поле за допомогою якого можна завантажити SVG файл перетягнувши в це поле або клікнувши по ньому (рис. 3.1).

Цей код створює елемент списку з іконкою завантаження та параграфом, який містить інструкції для першого кроку:

```
<ul className="content__list">
  <li>
    <DownloadOutlined />
    <Paragraph>
      <strong>Крок 1:</strong> Виберіть
SVG-зображення, яке ви хочете
      редагувати, та завантажте його сюди,
використовуючи завантажувач
      справа.
    </Paragraph>
```

У цьому коді використовується іконка для завантаження та параграф, який пояснює, що потрібно зробити на першому кроці.

Цей код створює елемент списку з іконкою відображення процесу та параграфом, який містить інструкції для другого кроку:

```
                    </li>
                    <li>
                      <DashboardOutlined />
                      <Paragraph>
                        <strong>Крок 2:</strong> Зачекайте кілька
секунд; редактор
                        завантажиться та відобразить ваше зображення.
                      </Paragraph>
                    </li>
```

У цьому коді використовується іконка для відображення процесу та параграф, який пояснює, що потрібно зробити на другому кроці.

Цей код створює елемент списку з іконкою збереження та параграфом, який містить інструкції для третього кроку:

```
                    <li>

                      <DownloadOutlined />
                      <Paragraph>
                        <strong>Крок 3:</strong> Почніть редагувати
та збережіть своє
                        зображення, коли закінчите.
                      </Paragraph>
                    </li>
                  </ul>
                </Col>
```

У цьому коді використовується іконка для збереження та параграф, який пояснює, що потрібно зробити на третьому кроці.

Результат коду:

Рисунок 3.1 – Інструкція на стартовій сторінці для нових користувачів

Цей код представляє компонент React, який відповідає за завантаження файлів SVG в SVG редакторі. Давайте розберемо кожну частину цього

Компонент UploadComponent:

— це функціональний компонент React, який використовує хук useNavigate з бібліотеки react-router-dom для навігації;

— використано хук useDispatch з бібліотеки react-redux для отримання функції dispatch, яка дозволяє викликати дії Redux для зміни стану.

```
const UploadComponent: React.FC = () => {
   const navigate = useNavigate();
   const dispatch = useDispatch();
```

Props UploadProps:

— це об'єкт з параметрами, які передаються в компонент Dragger з Ant Design для налаштування його поведінки;

— name: ім'я, що використовується для передачі файлів у запиті;

— multiple: дозволяє завантажувати кілька файлів одночасно;

— accept: фільтр для типів файлів, які можна завантажити (тільки файли з розширенням .svg);

– showUploadList: параметр, що приховує список завантажених файлів під час відображення.

```
const props: UploadProps = {
    name: 'file',
    multiple: true,
    accept: '.svg',
    showUploadList: false,
```

Метод customRequest: це функція, яка викликається при спробі завантажити файл.

Перевіряється тип файлу. Якщо тип невірний, викликається функція onError з повідомленням про помилку.

Якщо файл валідний, його вміст читається як текст за допомогою FileReader, і результат передається до onSuccess. Після цього дані додаються в Redux store за допомогою dispatch.

```
customRequest({ file, onSuccess, onError }) {
            if (typeof file === 'string' || !(file instanceof
Blob)) {
                onError && onError(new Error('Invalid file type'));
                return;
            }
            const reader = new FileReader();
            reader.onload = (e: ProgressEvent<FileReader>) => {
                if (e.target?.result) {
                    const newSvgCode = e.target.result as string;
                    onSuccess && onSuccess('ok');
                    dispatch(add(newSvgCode));
                }
            };
            reader.onerror = () => {
                onError && onError(new Error('FileReader error'));
            };
```

```
            reader.readAsText(file);
        },
```

Метод onChange:

— це обробник події, який викликається при зміні статусу завантаження файлу.

— якщо статус done, виводиться повідомлення про успішне завантаження і відбувається перехід на сторінку /workspace.

— якщо статус error, виводиться помилку про невдале завантаження.

```
        onChange(info) {
            const { status } = info.file;
            if (status === 'done') {
                message.success(`${info.file.name} file uploaded
successfully.`);
                return navigate("/workspace");
            } else if (status === 'error') {
                message.error(`${info.file.name} file upload
failed.`);
            }
        },
```

Метод onDrop: це обробник події, який викликається при перетягуванні файлів на область завантаження .

```
    return (
        <div className='upload__container'>
            <Dragger {...props}>
                <p className="ant-upload-drag-icon">
                    <InboxOutlined />
                </p>
                <p className="ant-upload-text"><strong>Клацніть або
перетягніть файл SVG в цю область для завантаження</strong></p>
                <p className="ant-upload-hint">
```

```
                        Підтримка одиночного або масового завантаження.
Суворо заборонено завантажувати корпоративні дані або інші
                        заборонені файли.
                    </p>
                </Dragger>
            </div>
        );
```

Цей компонент дозволяє користувачам завантажувати файли SVG, перетягуючи їх у відповідну область або вибираючи їх через стандартний діалог вибору файлів (рис. 3.2). Після завантаження файлу вміст SVG читається, і його код додається до Redux store для подальшого використання у SVG редакторі. Крім того, компонент забезпечує зручний інтерфейс користувача з повідомленнями про стан завантаження та корисними порадами щодо його використання.

Результат коду:

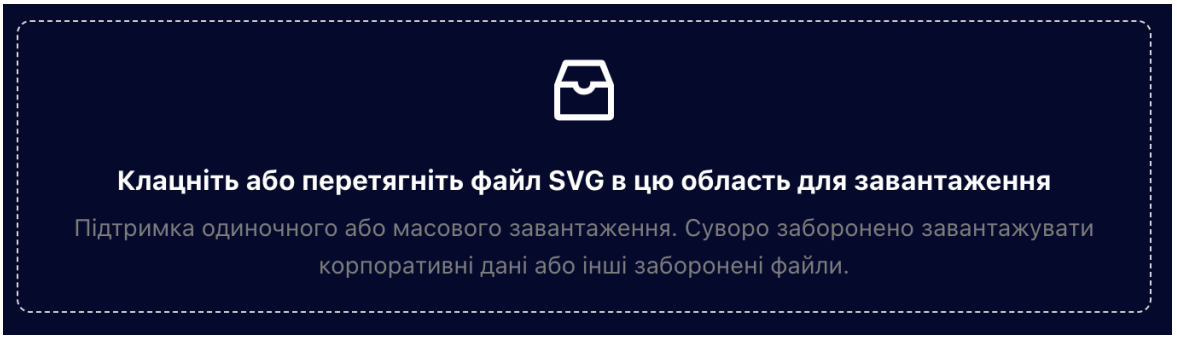


Рисунок 3.2 – UploadComponent

3.2 Реалізація основних функцій редактора (маніпуляції з фігурами, зміна параметрів)

У цьому розділі буде описано реалізацію основних функцій SVG-редактора, які дозволяють користувачеві створювати, редагувати та маніпулювати векторними фігурами. До цих функцій належать:

— створення фігур: опис інструментів для створення різних типів фігур, таких як лінії, криві, прямокутники, кола, еліпси та багатокутники;

— вибір та маніпулювання фігурами: опис методів вибору та маніпулювання окремими фігурами або групами фігур, включаючи переміщення, копіювання, видалення, обертання та масштабування;

— зміна параметрів фігур: опис інтерфейсу та методів для зміни параметрів фігур, таких як колір заливки, колір обведення, товщина обведення, стиль обведення та інші атрибути;

— робота з шарами: опис можливостей роботи з шарами, включаючи створення, видалення, перейменування та упорядкування шарів, а також відображення та редагування фігур на окремих шарах;

— відміна та повторення дій: опис функціоналу для скасування та повторення останніх дій користувача.

Під час опису реалізації буде приділено увагу використаним технологіям, застосованим алгоритмам, дизайну інтерфейсу користувача та забезпеченню зручності використання редактора[18].

Очікувані результати:

— розроблений набір інструментів для створення та редагування векторних фігур;

— реалізовані функції вибору та маніпулювання фігурами;

— інтерфейс для зміни параметрів фігур;

— можливості роботи з шарами;

— функціонал для скасування та повторення дій.

Цей розділ стане основою для розуміння ключових аспектів реалізації SVG-редактора та його можливостей.

```
    export const applySvgProperties = (canvas: fabric.Canvas | null,
svgProperties: SVGProperties) => {
        if (canvas) {
          const activeObject = canvas.getActiveObject();
```

```
      if (activeObject) {

        activeObject.set({
          left: svgProperties.x,
          top: svgProperties.y,
          scaleX: svgProperties.scaleX,
          scaleY: svgProperties.scaleY,
          angle: svgProperties.rotate,
          strokeWidth: svgProperties.borderWight,
          strokeUniform: true,
        });
        canvas.renderAll();
      }
    }
  };
```

Ця функція **applySvgProperties** призначена для застосування властивостей SVG до активного об'єкта на полотні Fabric.js. Давайте розберемо кожну частину цієї функції:

Параметри функції:

— canvas: полотно fabric.Canvas або null. Це полотно, до якого ми застосовуємо властивості SVG;

— svgProperties: об'єкт SVGProperties, який містить дані про властивості SVG, які ми застосовуємо до активного об'єкта;

Перевірка наявності полотна:

— ми перевіряємо, чи існує canvas . Це важливо, оскільки ми можемо застосувати властивості тільки до активного об'єкта на існуючому полотні;

Отримання активного об'єкта:

— ми отримуємо активний об'єкт на полотні за допомогою canvas.getActiveObject(). Активний об'єкт - це той об'єкт, який вибраний або виділений користувачем на полотні;

— Застосування властивостей до активного об'єкта:

– якщо є активний об'єкт (activeObject не є null), ми встановлюємо нові властивості для цього об'єкта за допомогою activeObject.set({...}).

Ось які властивості ми встановлюємо:

— left: позиція по горизонталі (X);

— top: позиція по вертикалі (Y);

— scaleX: масштаб по горизонталі.

— scaleY: масштаб по вертикалі.

— angle: кут повороту.

— strokeWidth: товщина контуру.

— strokeUniform: флаг, що вказує, що ширина контуру залишається постійною при зміні масштабу об'єкта.

Оновлення полотна:

– Після встановлення нових властивостей для об'єкта ми викликаємо canvas.renderAll(), щоб оновити відображення на полотні.

Ця функція дозволяє динамічно змінювати властивості активного об'єкта на полотні Fabric.js на основі переданих даних svgProperties, що дозволяє змінювати розміщення, масштаб, поворот та інші параметри об'єкта.

Ця функція **handleMouseDown** відповідає за обробку події "mousedown" на полотні canvas і виконує ряд дій при переміщенні миші. Давайте розглянемо її структуру та функціональність без використання списків.

```
export const handleMouseDown = (canvas: fabric.Canvas | null, dispatch: Dispatch) => {
  canvas?.on('mouse:down', function () {
    canvas.on('mouse:move', function (options) {
      if (options.target) {
        const selectedObject = options.target;
            const { left, top, angle, scaleX, scaleY, fill, strokeWidth, stroke, } = selectedObject;
         canvas.moveTo(selectedObject, canvas.getObjects().length - 1);
```

```
const width = scaleX?.toFixed(2);
const height = scaleY?.toFixed(2);
```

Ця функція відстежує подію "**mousedown**" на полотні і встановлює обробник подій **"mouse:move"**, який виконується при переміщенні миші над об'єктами на полотні. Коли об'єкт був вибраний (тобто options.target не є null), функція витягує властивості цього об'єкта, такі як позиція, кут, масштаб, колір тощо.

```
if (canvas) {
    const svg = canvas.toSVG();
    dispatch(setSvg(svg));
  }
  if(typeof fill === 'string'){
    dispatch(setColor(fill))
  }
  const myObject = canvas.toJSON();
  localStorage.setItem('canvas', JSON.stringify(myObject));
```

Потім вона виконує ряд дій:

— змінює z-index об'єкта, щоб він був найвищим і показувався поверх інших об'єктів на полотні;

— зберігає SVG-представлення полотна у глобальному стані за допомогою dispatch(setSvg(svg));

— встановлює колір заливки об'єкта, якщо він є строкою, за допомогою dispatch(setColor(fill)).

— зберігає дані полотна у форматі JSON у локальному сховищі браузера за допомогою localStorage.setItem('canvas', JSON.stringify(myObject));

— встановлює різні параметри (колір рамки, товщину рамки, розміри, позицію та кут обертання) об'єкта у глобальний стан за допомогою різних dispatch дій.

```
        dispatch(setBorderColor((stroke) ?? 'black'))
          dispatch(setBorderWight((strokeWidth) ?? 0));
          dispatch(setWidth(Number(width) ?? 0));
          dispatch(setHeight(Number(height) ?? 0));
          dispatch(setX(Math.round(left ?? 0)));
          dispatch(setY(Math.round(top ?? 0)));
          dispatch(setRotate(Math.round(angle ?? 0)));
        }
      })
    });
```

Також, при події "mouseup", функція відключає обробник подій "mouse:move", щоб припинити відстеження переміщення миші після того, як кнопка миші була відпущена.

```
    canvas?.on('mouse:up', function () {
      canvas?.off('mouse:move');
    });
  };
```

Ця функція важлива для реалізації взаємодії з об'єктами на полотні, виконання дій під час їх переміщення та зберігання стану полотна для подальшого використання чи збереження.

Ця функція **handleDeleteSVG** відповідає за обробку події натискання клавіші "Delete" або "Backspace" під час взаємодії з полотном canvas. Вона встановлює обробник подій keydown на вікні браузера.

```
export const handleDeleteSVG = (canvas: fabric.Canvas | null, dispatch: Dispatch) => {
```

При натисканні клавіші "Delete" або "Backspace" (при умові, що активний елемент не є полем введення) і якщо на полотні є активні об'єкти (activeObjects), функція видаляє кожен активний об'єкт з полотна.

```
        canvas?.on('mouse:down', function () {
           window.addEventListener('keydown', function (e) {
                  if ((e.key === 'Delete' || e.key === 'Backspace') &&
(e.target as Node).nodeName !== 'INPUT') {
                const activeObjects = canvas?.getActiveObjects();
                activeObjects?.forEach(function (object) {
                  canvas?.remove(object);
                });
                if (canvas) {
                    const svg = canvas.toSVG();
                    dispatch(setSvg(svg));
                    this.localStorage.removeItem('canvas');
                  }
                canvas?.discardActiveObject().renderAll();
             }
            });
         });
       };
```

Далі, після видалення об'єктів, функція витягує SVG-представлення полотна і зберігає його у глобальному стані додатка за допомогою функції dispatch(setSvg(svg)). Крім того, дані про полотно видаляються з локального сховища браузера за допомогою localStorage.removeItem('canvas').

Нарешті, активний об'єкт на полотні скасовується за допомогою canvas?.discardActiveObject().renderAll(), щоб забрати виділення з об'єкта після його видалення.

Цей компонент **ClearBtn** відповідає за створення кнопки для очищення робочої області SVG редактора. При натисканні на кнопку викликається функція handdleClearWorkspace, яка відправляє дію очищення до Redux store за

допомогою dispatch(clear()). Також зберігання даних у локальному сховищі браузера за допомогою localStorage.removeItem('canvas').

```
const ClearBtn = ({canvas} : SvgResizerProps) => {
   const dispatch = useDispatch()
```

Після цього функція перевіряє, чи існує об'єкт canvas.current, і якщо він існує, то викликає метод clear() для очищення робочої області. Крім того, відображається повідомлення про успішне очищення робочої області за допомогою message.success('Workspace clear').

```
  const handdleClearWorkspace = () => {
       dispatch(clear());
       localStorage.removeItem('canvas');
       if(canvas.current) {
           canvas.current.clear();
       }
       message.success('Workspace clear');
   }
   return (
  <Button onClick={handdleClearWorkspace}><ClearOutlined/>Очистити
дошку</Button>
    );
}
export default ClearBtn;
```

Цей підхід дозволяє користувачам легко очищати робочу область в SVG редакторі, зберігаючи при цьому можливість управління станом за допомогою Redux та локального сховища браузера для зберігання даних між сесіями роботи з редактором.

3.3 Реалізація маніпуляції даними за допомогою RTK

У цьому розділі буде описано, як використовувати Redux Toolkit (Redux TK) для реалізації маніпуляції даними в нашому додатку. Redux TK - це бібліотека з відкритим кодом, яка полегшує роботу з Redux, популярним JavaScript-фреймворком для управління станом[19].

RTK пропонує ряд переваг для реалізації маніпуляції даними, зокрема: просте визначення типів дій та редукторів: Redux TK дозволяє легко визначати типи дій та редуктори за допомогою TypeScript. Це допомагає забезпечити типізовану безпеку та покращити читабельність коду.

Автоматичне створення action creators та selectors: Redux TK автоматично генерує action creators та selectors на основі визначених типів дій. Це економить час і допомагає уникнути помилок.

Підтримка нормалізації даних: RTK пропонує зручні інструменти для нормалізації даних, що робить їх більш організованими та легкими для доступу[20].

У цьому розділі буде продемонстровано, як використовувати RTK для:

- визначення типів дій та редукторів;
- створення action creators та selectors;
- нормалізація даних;
- оновлення стану Redux у відповідь на дії користувач.

Цей код визначає слайс для керування властивостями SVG за допомогою **Redux Toolkit**. Ось розбивка:

1. Визначення інтерфейсу:

- SVGProperties - це інтерфейс, який визначає властивості SVG;
- він містить властивості, такі як scaleX, scaleY, x, y для позиціонування та масштабування, rotate для обертання та borderWight (ймовірно, помилка друку, має бути borderWidth для ширини рамки).

```
export type SVGProperties = {
 scaleX: number;
 scaleY: number;
 x: number;
 y: number;
 rotate: number;
 borderWight: number;
};

const initialState: SVGProperties = {
 scaleX: 0,
 scaleY: 0,
 x: 0,
 y: 0,
 rotate: 0,
 borderWight: 0,
};
```

2. Початковий стан:

initialState - це об'єкт, який встановлює початкові значення для всіх властивостей SVG. Всі властивості встановлюються в 0.

3. Створення слайсу:

- svgPropertiesSlice створюється за допомогою createSlice з RTK;
- name: "svgProperties" - ідентифікує цей слайс у магазині Redux;
- initialState: Початковий стан об'єкта, визначений раніше;
- reducers: Це об'єкт, який визначає функції редуктора для кожної дії, яка може змінювати властивості SVG.

```
const svgPropertiesSlice = createSlice({
 name: 'svgProperties',
 initialState,
```

4. Редуктори:

- кожна функція редуктора приймає два аргументи;

— state: поточний стан властивостей SVG;

— action: об'єкт дії, який було відправлено і який викликав редуктор;

— на основі типу дії (action.type) редуктор оновлює відповідну властивість у стані значенням, наданим навантаженні дії (action.payload).

— є функції редуктора для встановлення кожної властивості SVG: setWidth, setHeight, setX, setY, setRotate та setBorderWight.

```
reducers: {
  setWidth: (state, action: PayloadAction<number>) => {
    state.scaleX = action.payload;
  },
  setHeight: (state, action: PayloadAction<number>) => {
    state.scaleY = action.payload;
  },
  setX: (state, action: PayloadAction<number>) => {
    state.x = action.payload;
  },
  setY: (state, action: PayloadAction<number>) => {
    state.y = action.payload;
  },
  setRotate: (state, action: PayloadAction<number>) => {
    state.rotate = action.payload;
  },
  setBorderWight: (state, action: PayloadAction<number>) => {
    state.borderWight = action.payload;
  },
```

5. Експорт дій та редуктора:

— дії (setWidth, setHeight тощо) - це функції, які використовуються для відправлення дій, які оновлюють властивості SVG;

— ці дії експортуються для використання в інших частинах коду;

— основний редуктор (svgPropertiesSlice.reducer) також експортується як за замовчуванням. Цей редуктор відповідає за обробку всіх дій, пов'язаних із властивостями SVG, та відповідне оновлення стану.

```
 },
});
export const { setWidth, setHeight, setX, setY, setRotate, setBorderWight } = svgPropertiesSlice.actions;
export default svgPropertiesSlice.reducer;
```

Загалом, цей код надає спосіб керувати властивостями SVG у застосунку Redux за допомогою Redux Toolkit. Він визначає початковий стан, редуктори для оновлення стану та дії для відправлення оновлень.

Це дозволяє централізовано керувати та легко маніпулювати властивостями SVG у вашому застосунку.

```
import { createSlice, PayloadAction } from '@reduxjs/toolkit';
export type ArrayState = string[];
const svgsSlice = createSlice({
 name: 'svgsArray',
 initialState: [] as ArrayState,
 reducers: {
   add: (state, action: PayloadAction<string>) => {
     state.push(action.payload);
   },
   clear: (state) => { state.length = 0; },
 },
});
export const { add, clear } = svgsSlice.actions;
export default svgsSlice.reducer;
```

Цей код є частиною використання бібліотеки Redux Toolkit для створення Slice (частини стану) у Redux. Давай розглянемо кожен рядок по порядку:

```
export type ArrayState = string[];:
```

Тут визначається тип ArrayState, який є масивом рядків (string[]). Це буде типом стану для нашого Slice.

```
const svgsSlice = createSlice({ ... });:
```

createSlice - це функція з RTK, яка дозволяє створювати Slice (частину стану) у Redux разом із відповідними редукторами (reducers) та діями (actions).

```
name: 'svgsArray',:
```

Властивість name визначає назву для Slice. В даному випадку, назва Slice - 'svgsArray'.

```
initialState: [] as ArrayState,:
```

initialState - це початковий стан для нашого Slice. Тут ми встановлюємо початковий стан як порожній масив ([]) із типом ArrayState (масив рядків).

```
reducers: { ... }:
```

Властивість reducers містить об'єкт з редукторами для даного Slice. Редуктори - це функції, які змінюють стан залежно від дій (actions).

```
add: (state, action: PayloadAction<string>) => { state.push(action.payload); },:
```

Це редуктор для дії add. Він отримує поточний стан state та action (що містить payload, яке є рядком (string)). В редукторі add ми додаємо новий рядок до масиву стану, використовуючи action.payload.

```
clear: (state) => { state.length = 0; },
```

Це редуктор для дії clear. Він отримує поточний стан state і просто встановлює довжину масиву state рівною 0, що очищує масив.

```
export const { add, clear } = svgsSlice.actions;:
```

Тут експортуються дії (actions) зі Slice. add і clear - це функції-дії, які можна використовувати для виклику в компонентах для зміни стану.

```
export default svgsSlice.reducer;:
```

Експорт основного редуктора (reducer) з Slice. Цей редуктор відповідає за обробку дій, що визначені в reducers, та змінює відповідний стан.

Загальною метою цього коду є створення частини стану Redux для масиву рядків svgsArray, яка містить дві можливості: додавання нового рядка до масиву і очищення масиву. Це дозволяє ефективно керувати станом масиву у Redux за допомогою зручних дій і редукторів, що надає Redux Toolkit.

3.4 Висновки щодо якості роботи редактору та його можливостей

Цей простий SVG редактор з інтуїтивно зрозумілим інтерфейсом ідеально підходить для початківців та користувачів, які шукають інструмент для створення простих SVG зображень. Користувачі можуть легко масштабувати та переміщувати фігури на робочому просторі. Всі фігури можна зафарбувати в будь-який колір за допомогою палітри кольорів. Також можна додати бордер до будь-якої фігури та змінювати його товщину.

Інтерфейс редактора простий та зрозумілий, що робить його доступним для користувачів з будь-яким рівнем досвіду. Завдяки інструментам для

створення фігур та зміни їх властивостей, користувачі можуть легко створювати прості SVG зображення без поглиблених знань про SVG.

Цей редактор є чудовим стартовим майданчиком для тих, хто хоче навчитися створювати SVG зображення. Він чудово підходить для створення простих SVG зображень, таких як логотипи, іконки, діаграми та інші графічні елементи. Редактор доступний безкоштовно, що робить його доступним для широкої аудиторії. Він працює в браузері, не потребуючи завантаження та встановлення програмного забезпечення.

Загалом, цей простий SVG редактор є чудовим інструментом для початківців та користувачів, які шукають простий спосіб створювати базові SVG зображення. Його інтуїтивно зрозумілий інтерфейс, легкість використання, безкоштовність роблять його доступним для будь-якого рівня досвіду.

Висновок до розділу 3

У третьому розділі реалізовано інтерфейс та основні функції SVG редактора. Розроблено компоненти, які дозволяють користувачам здійснювати маніпуляції з фігурами, змінювати їх параметри, а також зберігати результати роботи. Інтегровано Redux Toolkit для керування станом додатку, що забезпечило зручну роботу з даними та їх маніпуляцію. Було проведено тестування, яке підтвердило високу якість роботи редактора та його відповідність вимогам користувачів.

ВИСНОВКИ

Виконання роботи з розробки простого SVG редактора дозволило досягнути важливих висновків та досягнень. Проведений аналіз сучасних SVG редакторів показав, що існуючі редактори мають значні обмеження у простоті використання та доступності для широкого кола користувачів. Це підтверджує актуальність створення простого та ефективного редактора.

Вибір технологій для розробки, зокрема JavaScript, HTML5, CSS, а також методів розробки базувався на їхній спроможності забезпечити простоту використання, а також ефективність у створенні векторних зображень. Успішна розробка прототипу SVG редактора з основним функціоналом, включаючи можливість малювання, редагування та зміну параметрів векторних об’єктів, була проведена. Тестування підтвердило працездатність та стійкість розробленого інструменту.

Розроблений простий SVG редактор має практичне значення для веб-розробників та дизайнерів, забезпечуючи їм зручний інструмент для створення векторної графіки без складнощів. Створення простого SVG редактора є важливим кроком у напрямку поліпшення доступності та ефективності роботи з векторною графікою в сучасній веб-розробці. Результати дослідження демонструють успішну реалізацію поставлених завдань та підтверджують важливість подальшого розвитку інтерактивного веб-контенту через прості та ефективні інструменти, такі як розроблений SVG редактор.

Дана робота дозволила створити функціональний інструмент, який може забезпечити користувачам не лише можливість створення векторних зображень, а й покращити продуктивність і ефективність роботи веб-розробників та дизайнерів у процесі створення веб-контенту.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. React Documentation. URL: https://reactjs.org/docs/getting-started.html (Дата звернення: 20.02.2024).

2. Fabric.js Documentation. URL: http://fabricjs.com/docs/ (Дата звернення: 24.02.2024).

3. Redux Toolkit Documentation. URL: https://redux-toolkit.js.org/introduction/getting-started (Дата звернення: 27.02.2024).

4. Pro React 16. Adam Freeman. Apress, 2019. URL: http://surl.li/uejxf (Дата звернення: 03.03.2024).

5. Learning React: Modern Patterns for Developing React Apps. Alex Banks, Eve Porcello. O'Reilly Media, 2017. URL: http://surl.li/uejxr (Дата звернення: 10.03.2024).

6. Fabric.js: Graphics and Animation for HTML5. Andrea Boggesi. Packt Publishing, 2013. URL: http://fabricjs.com/ (Дата звернення: 20.03.2024).

7. React Design Patterns and Best Practices. Michele Bertoli. Packt Publishing, 2017. URL: http://surl.li/uejym (Дата звернення: 28.03.2024).

8. Redux Quick Start Guide: A Practical Introduction to Managing State in React. Abhishek Bhardwaj. Packt Publishing, 2018. URL: http://surl.li/uejzm (Дата звернення: 01.04.2024).

9. React.js Essentials. Artemij Fedosejev. Packt Publishing, 2016. URL: http://surl.li/uekav (Дата звернення: 05.04.2024).

10. Mastering Redux. Daniel Bugl. Packt Publishing, 2017. URL: http://surl.li/ufnau (Дата звернення: 08.04.2024).

11. SVG Essentials. J. David Eisenberg. O'Reilly Media, 2014. URL: http://surl.li/ufnaz (Дата звернення: 15.04.2024).

12. Practical SVG. Chris Coyier. A Book Apart, 2016. URL: http://surl.li/ufnbc (Дата звернення: 18.04.2024).

13. SVG Animations. Sarah Drasner. O’Reilly Media, 2017. URL: http://surl.li/ufnbg (Дата звернення: 22.04.2024).

14. Using SVG with CSS3 and HTML5. Amelia Bellamy-Royds, Kurt Cagle. O’Reilly Media, 2017. URL: http://surl.li/ufnbo (Дата звернення: 26.04.2024).

15. Scalable Vector Graphics (SVG): An Overview. Andrew Watt. Springer, 2003. URL: http://surl.li/ufnbu (Дата звернення: 30.04.2024).

16. HTML5 and CSS3 All-in-One For Dummies. Andy Harris. Wiley, 2014. URL: http://surl.li/ufnbx (Дата звернення: 03.05.2024).

17. Designing Web Interfaces: Principles and Patterns for Rich Interactions. Bill Scott, Theresa Neil. O’Reilly Media, 2009. URL: http://surl.li/ufnca(Дата звернення: 07.05.2024).

18. Web Design with HTML, CSS, JavaScript and jQuery Set. Jon Duckett. Wiley, 2014. URL: http://surl.li/ufncd (Дата звернення: 10.05.2024).

19. JavaScript: The Good Parts. Douglas Crockford. O’Reilly Media, 2008. URL: http://surl.li/ufnch (Дата звернення: 15.05.2024).

20. Eloquent JavaScript: A Modern Introduction to Programming. Marijn Haverbeke. No Starch Press, 2018. URL: http://surl.li/ufncj (Дата звернення: 20.05.2024).

метадані

Заголовок

Розробка простого SVG редактора

Автор
Скоробагатий Д. І.

Науковий керівник / Експерт
кандидат технічних наук Олег Пашкевич

підрозділ
King Danylo University

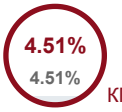
Тривога

У цьому розділі ви знайдете інформацію щодо текстових спотворень. Ці спотворення в тексті можуть говорити про МОЖЛИВІ маніпуляції в тексті. Спотворення в тексті можуть мати навмисний характер, але частіше характер технічних помилок при конвертації документа та його збереженні, тому ми рекомендуємо вам підходити до аналізу цього модуля відповідально. У разі виникнення запитань, просимо звертатися до нашої служби підтримки.

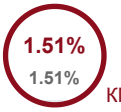
Заміна букв		0
Інтервали	A→	0
Мікропробіли	·	0
Білі знаки		0
Парафрази (SmartMarks)	a	23

Обсяг знайдених подібностей

Коефіцієнт подібності визначає, який відсоток тексту по відношенню до загального обсягу тексту було знайдено в різних джерелах. Зверніть увагу, що високі значення коефіцієнта не автоматично означають плагіат. Звіт має аналізувати компетентна / уповноважена особа.



КП 1



КП 2

25
Довжина фрази для коефіцієнта подібності 2

7558
Кількість слів

59620
Кількість символів

Подібності за списком джерел

Нижче наведений список джерел. В цьому списку є джерела із різних баз даних. Колір тексту означає в якому джерелі він був знайдений. Ці джерела і значення Коефіцієнту Подібності не відображають прямого плагіату. Необхідно відкрити кожне джерело і проаналізувати зміст і правильність оформлення джерела.

10 найдовших фраз — Колір тексту

ПОРЯДКОВИЙ НОМЕР	НАЗВА ТА АДРЕСА ДЖЕРЕЛА URL (НАЗВА БАЗИ)	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)	
1	http://repository.ukd.edu.ua/bitstream/handle/123456789/394/%D0%A0%D1%83%D0%B4%D0%B8%D0%B9%20%D0%90%D0%BD%D0%B4%D1%80%D1%96%D0%B9%20%D0%B4%D0%B8%D0%BF%D0%BB%D0%BE%D0%BC%D0%BD%D0%B0.pdf?sequence=1	60	0.79 %
2	http://repository.ukd.edu.ua/bitstream/handle/123456789/394/%D0%A0%D1%83%D0%B4%D0%B8%D0%B9%20%D0%90%D0%BD%D0%B4%D1%80%D1%96%D0%B9%20%D0%B4%D0%B8%D0%BF%D0%BB%D0%BE%D0%BC%D0%BD%D0%B0.pdf?sequence=1	54	0.71 %
3	Pavlenko_bac_rob.docx [redacted] 6/12/2022 Sumy State University (Кафедра комп'ютерних наук)	22	0.29 %