МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет

імені Петра Могили

Факультет комп'ютерних наук

Кафедра інтелектуальних інформаційних систем

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувач кафедри інтелектуальних

інформаційних систем, доцент, к.т.н.

_____Є. В. Сіденко

«____»_____2023 р.

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

СИСТЕМА ПІДБОРУ СЦЕНАРІЇВ РЕАЛІЗАЦІЇ БРАУЗЕРНИХ ЗАСТОСУНКІВ ДЛЯ АВТОМАТИЗАЦІЇ ДІЙ КОРИСТУВАЧА

Спеціальність 122 «Комп'ютерні науки»

122 – МКР – 601.1710231

Студент _____ Є. О. Шлапак

«14» лютого 2023 р.

Консультант _____ І.В.Кулаковська

к.ф.м.н., доцент

«__» лютого 2023 р.

Миколаїв – 2023

Чорноморський національний університет ім. Петра Могили
Факультет комп'ютерних наук
Кафедра інтелектуальних інформаційних систем

Освітньо-кваліфікаційний рівень **магістр**

Галузь знань **12 «Інформаційні технології»**
(шифр і назва)

Спеціальність **122 «Комп'ютерні науки»**
(шифр і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри інтелектуальних інформаційних систем, доцент, к.т.н.
_____ Є. В. Сіденко
«____» _____2022р.

ЗАВДАННЯ
на магістерську кваліфікаційну роботу

Шлапак Євгеній Олегович
(прізвище, ім'я, по батькові)

1. Тема магістерської кваліфікаційної роботи
"Система підбіру сценаріїв реалізації браузерних застосунків для автоматизації дій користувача"

Керівник роботи К.ф.м.н., доцент І.В.Кулаковська.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затв. наказом Ректора ЧНУ ім. Петра Могили від «03» 11. 2022 р.№199

2. Строк подання студентом роботи «14» лютого 2023 р.

3. Вхідні (початкові) дані до роботи: існуючи системи організації та запуску користувацьких сценаріїв, результати опитування користувачів скріптів. Очікуваний результат роботи: функціонально готова система підбіру сценаріїв реалізації браузерних застосунків для автоматизації дій користувача та сайт-бібліотека з користувацькими сценаріями.

4. Зміст пояснювальної записки (перелік питань, які потрібно розглянути):

1) аналіз предметної сфери та розгляд альтернативних систем організації та запуску користувацьких сценаріїв; 2) дослідження потреб користувачів та атуальності ствовення системи індивідуальної системи кастомізації користувацького досвіду; 3) аналіз та вивчення технологій для створення

функціонуючого продукту; 4) створення та тестування системи організації та відтворення користувацьких сценаріїв та сайту-бібліотеки скриптів.

5. Перелік графічних матеріалів. Дипломну роботу виконано на 84 аркушах. У роботі наведено 32 рисунки.

6. Завдання до спеціальної частини. Аналіз і оцінка стану умов та безпеки праці.

7. Консультанти:

Розділ	**Прізвище, ініціали та посада консультанта**	**Підпис**
Спеціальна частина з охорони праці	Григор'єва Л.І. докт. біол. наук, професор	
Методична частина	Кулаковська І.В. канд. фіз. мат. наук, доц.	

Керівник роботи __к.ф.м.н., доцент І.В.Кулаковська _____

(наук. ступінь, вчене звання, прізвище та ініціали) *(підпис)*

Завдання прийнято до виконання ___Шлапак Є.О. _____

(прізвище та ініціали) *(підпис)*

Дата видачі завдання «_07_» __листопада__ 2022 р.

КАЛЕНДАРНИЙ ПЛАН

Виконання магістерської кваліфікаційної роботи

Тема: Система підбіру сценаріїв реалізації браузерних застосунків для автоматизації дій користувача.

№	Найменування роботи	Початок	Закінчення	Примітки
1	Визначення керівника і теми МКР. Подання заяви на затвердження теми МКР	01.09.2022	20.10.2022	Виконано
2	Отримання завдання на виконання МКР	21.10.2022	10.11.2022	Виконано
3	Складання календарного плану на період виконання МКР	11.11.2022	15.11.2022	Виконано
4	Огляд літератури за темою дослідження	16.11.2022	27.11.2022	Виконано
5	Проходження переддипломної практики, збір та аналіз матеріалів до МКР	28.11.2022	18.12.2022	Виконано
6	Аналіз предметної області та розробка технічного завдання. Моделювання результатів	19.12.2022	12.01.2023	Виконано
7	Опис фахової частини МКР, зокрема дослідження публікацій щодо переміщень осіб належних до соціально незахищених груп, огляд існуючих аналогів розроблюваного додатку для вирішення поставленої задачі, реалізація обраних технологій з аналізом отриманих результатів	13.01.2023	25.01.2023	Виконано
8	Розробка спеціальної частини з охорони праці та методичної частини	26.01.2023	02.02.2023	Виконано
9	Попередній захист МКР на засіданні комісії кафедри	03.02.2023	03.02.2023	Виконано
10	Корегування роботи за результатами попереднього захисту	04.02.2023	06.02.2023	Виконано

11	Остаточне оформлення пояснювальної записки та слайдів доповіді для захисту	07.02.2023	09.02.2023	Виконано
12	Подання МКР рецензенту	09.02.2023	10.02.2023	Виконано
13	Рецензування МКР	11.02.2023	12.02.2023	
14	Подання МКР, її електронної копії та інших документів (відгуку, рецензії) до захисту	15.02.2023	16.02.2023	
15	Захист МКР перед екзаменаційною комісією (ЕК)	22.02.2023	23.02.2023	

Розробив студент _____Шлапак Є.О._____

(прізвище та ініціали) (*підпис*)

Керівник роботи _____Кулаковська І. В._____

(наук. ступінь, вчене звання, прізвище та ініціали) *(підпис)*

«_12»листопада 2022 р.

Анотація

Google Chrome, який пропонує ряд корисних функцій і переваг для користувачів та його розширень – невеликих програмних застосунків, які можна додати до Chrome, щоб налаштувати та покращити веб-перегляд користувача. Розширення доступні для різноманітних цілей, включаючи продуктивність, розваги, безпеку та конфіденційність.

Об'єктом дослідження є процеси функціонування браузерних застосунків.

Предметом є методи та технології створення програмних елементів для web-застосунків.

Таким чином **метою** магістерської дипломної роботи є покращення індивідуального користувацького досвіду методом створення спеціальної бібліотеки сценаріїв і програми, яка її запускає, оскільки вони дозволяють користувачам налаштовувати роботу з певними веб-сторінками.

Програма функціонує як розширення, яке можна встановити у веб-браузері та запускає код на певних сторінках, дозволяючи вносити зміни, визначені користувачем. Це може включати зміни у зовнішньому вигляді, відчуттях і функціональності веб-сторінки, що дозволяє користувачеві адаптувати свій досвід у спосіб, який найкраще відповідає його потребам.

Магістерську кваліфікаційну роботу роботу виконано на 84 аркушах. У роботі наведено 32 рисунки.

Ключові слова: додаток, браузер, Google Chrome, скрипт, JavaScript, сайт, користувацький досвід, jQuery, WEB-розробка.

ABSTRACT

Google Chrome, which offers a number of useful features and benefits for users and its extensions - small software applications that can be added to Chrome to customize and enhance the user's web browsing experience. Extensions are available for a variety of purposes, including productivity, entertainment, security, and privacy.

The object of research is the functioning processes of browser applications.

The subject is methods and technologies for creating software elements for web applications.

Thus, the goal of the master's thesis is to improve the individual user experience by creating a special script library and the program that runs it, as they allow users to customize the work with certain web pages.

The program functions as an extension that can be installed in a web browser and runs code on specific pages, allowing user-defined changes to be made. This may include changes to the look, feel and functionality of the web page, allowing the user to tailor their experience in a way that best suits their needs.

The master's qualification work was completed on 84 sheets. The work contains 32 drawings.

Keywords: application, browser, Google Chrome, script, JavaScript, site, user experience, jQuery, WEB development.

Пояснювальна записка

до магістерської кваліфікаційної роботи

на тему:

«СИСТЕМА ПІДБОРУ СЦЕНАРІЇВ РЕАЛІЗАЦІЇ БРАУЗЕРНИХ ЗАСТОСУНКІВ ДЛЯ АВТОМАТИЗАЦІЇ ДІЙ КОРИСТУВАЧА»

Спеціальність 122 «Системний аналіз»

122 – МКР – 601.21710131

Виконав студент 6-го курсу, групи 601

_____ Є. О. Шлапак

«16» лютого 2023 р.

Керівник: к.ф.м.н., доцент

_____І. В. Кулаковська

«16» лютого 2023 р.

Миколаїв – 2023

ЗМІСТ

ВСТУП

Об’єктом магістерської дипломної роботи є процеси функціонування браузерних застосунків. Розширення – це невеликі програмні програми, які можна додати до Chrome, щоб налаштувати та покращити веб-перегляд користувача. Розширення доступні для різноманітних цілей, включаючи продуктивність, розваги, безпеку та конфіденційність.

Предметом магістерської дипломної роботи є технології створення розширень для браузерів та сучасних web-застосунків.

Однією з найбільших переваг розширень Google Chrome є підвищена продуктивність, яку вони пропонують. Надаючи користувачам легкий доступ до додаткових функцій, розширення можуть допомогти користувачам заощадити час, оптимізувати робочий процес і швидко отримати доступ до потрібної інформації. Популярні приклади розширень продуктивності включають LastPass, який безпечно зберігає паролі та дозволяє користувачам швидко входити в декілька облікових записів. Іншим прикладом є Grammarly, який перевіряє орфографію та граматику, коли користувачі вводять текст.

Розширення Google Chrome також можуть допомогти підвищити безпеку та конфіденційність під час перегляду в Інтернеті. До популярних розширень конфіденційності належать Ghostery, який блокує трекери з веб-сайтів, і Adblock Plus, який блокує нав’язливу рекламу. Розширення безпеки, такі як Web of Trust, надають користувачам дані про безпеку веб-сайтів і попереджають користувачів про будь-який шкідливий вміст.

Розширення Google Chrome також надають користувачам покращений досвід розваг. Такі розширення, як Netflix Enhancer, дозволяють користувачам швидко отримувати доступ до улюблених шоу та фільмів, а розширення, такі як Dark Reader, можна використовувати для створення темного режиму для веб-сайтів. Існують також розширення, які пропонують доступ до потокових сервісів, таких як Spotify і Pandora.

Загалом розширення Google Chrome пропонують користувачам ряд функцій і переваг, від підвищення продуктивності до покращення безпеки та розваг. За допомогою правильних розширень користувачі можуть налаштувати та покращити свій досвід перегляду, роблячи його простішим і приємнішим.

Проекти з відкритим кодом є важливими для успіху Інтернету. Вони надають можливість розробникам співпрацювати, ділитися кодом і створювати інноваційні програми. Розширення Chrome — чудовий спосіб створювати проекти з відкритим кодом, оскільки вони дозволяють розробникам швидко й легко створювати та поширювати свій код. Крім того, розширення Chrome надають багато функцій, які полегшують розробку та розгортання, наприклад вбудовану підтримку для налагодження та тестування, доступ до широкого спектру API та можливість інтеграції з іншими службами та продуктами.

Для розробників важливо прислухатися до відгуків про їхні продукти, оскільки це може допомогти їм удосконалити та покращити свої продукти. Однак неможливо охопити всі побажання клієнта, оскільки клієнти мають різноманітні потреби та побажання, які розробник може бути неможливим задовольнити. Спеціальні сценарії сайтів можуть надати користувачам можливість налаштовувати сайти на свій розсуд. Це дозволяє користувачам налаштовувати свої веб-сайти відповідно до своїх унікальних потреб і уподобань, що може надати їм більш індивідуальний досвід.

Таким чином **метою** магістерської дипломної роботи стало покращення індивідуального користувацького досвіду та створення спеціальної бібліотеки сценаріїв і програми, яка її запускає, оскільки вони дозволяють користувачам налаштовувати роботу з певними веб-сторінками. Програма функціонує як розширення, яке можна встановити у веб-браузері та запускає код на певних сторінках, дозволяючи вносити зміни, визначені користувачем. Це може включати зміни у зовнішньому вигляді, відчуттях і функціональності

веб-сторінки, що дозволяє користувачеві адаптувати свій досвід у спосіб, який найкраще відповідає його потребам.

Для досягнення мети магістерської дипломної роботи було поставлено такі **завдання**:

- проаналізувати предметну сферу та розглянути альтернативні системи організації та запуску користувацьких сценаріїв;
- дослідити потреб користувачів та актуальність створення системи індивідуальної системи кастомізації користувацького досвіду;
- проаналізувати та вивчити технологій для створення функціонуючого продукту;
- створити та протестувати систему організації та відтворення користувацьких сценаріїв та сайт-бібліотеку скриптів.

1 СФЕРИ ЗАСТОСУВАНЬ БРАУЗЕНИХ РОЗШИРЕНЬ

Розширення браузера – це програми, які покращують функціональність веб-браузерів, надаючи додаткові функції. Зазвичай вони розробляються сторонніми розробниками та використовуються веб-користувачами, щоб налаштувати свій досвід перегляду. Деякі з найпоширеніших прикладів розширень браузера – блокувальники реклами, програми безпеки та конфіденційності та менеджери паролів.

Блокувальники реклами – це розширення, які дозволяють користувачам блокувати рекламу, що з’являється у їхніх веб-переглядачах. Це корисно з багатьох причин, у тому числі для зменшення безладу на веб-сторінках і захисту користувачів від небажаного відстеження. Блокувальники реклами також корисні для покращення часу завантаження сторінки, оскільки реклама може значно сповільнити час завантаження сторінки.

Програми безпеки та конфіденційності – це розширення, які захищають користувачів від шкідливого вмісту під час перегляду веб-сторінок. Ці програми можна використовувати для сканування веб-сайтів на наявність підозрілого чи шкідливого змісту та попередження користувачів у разі будь-яких загроз. Крім того, їх можна використовувати для блокування файлів cookie відстеження та інших форм небажаного відстеження.

Менеджери паролів — це розширення, які дозволяють користувачам безпечно зберігати свої паролі у веб-переглядачі. Це корисно для користувачів, яким часто потрібно використовувати різні паролі для різних веб-сайтів. Менеджери паролів надають користувачам безпечний спосіб зберігати свої паролі та швидко отримувати до них доступ у разі потреби.

Розширення веб-переглядача можуть бути неймовірно корисними для веб-користувачів, оскільки вони дозволяють їм налаштовувати свій досвід перегляду та забезпечують додаткову безпеку та конфіденційність.

Блокувальники реклами, програми безпеки та конфіденційності, а також менеджери паролів – це приклади корисних розширень браузера, які можна знайти в більшості веб-браузерів.

Google Chrome пропонують ряд корисних функцій і переваг для користувачів. Розширення – це невеликі програмні програми, які можна додати до Chrome, щоб налаштувати та покращити веб-перегляд користувача. Розширення доступні для різноманітних цілей, включаючи продуктивність, розваги, безпеку та конфіденційність.

Розширення Google Chrome також можуть допомогти підвищити безпеку та конфіденційність під час перегляду в Інтернеті. До популярних розширень конфіденційності належать Ghostery, який блокує трекери з веб-сайтів, і Adblock Plus, який блокує нав’язливу рекламу. Розширення безпеки, такі як Web of Trust, надають користувачам дані про безпеку веб-сайтів і попереджають користувачів про будь-який шкідливий вміст.

Розширення Google Chrome також надають користувачам покращений досвід розваг. Такі розширення, як Netflix Enhancer, дозволяють користувачам швидко отримувати доступ до улюблених шоу та фільмів, а розширення, такі як Dark Reader, можна використовувати для створення темного режиму для веб-сайтів. Існують також розширення, які пропонують доступ до потокових сервісів, таких як Spotify і Pandora.

Загалом розширення Google Chrome пропонують користувачам ряд функцій і переваг, від підвищення продуктивності до покращеної безпеки та розваг. За допомогою правильних розширень користувачі можуть налаштувати та покращити свій досвід перегляду, роблячи його простішим і приємнішим.

Проекти з відкритим кодом є важливими для успіху Інтернету. Вони надають можливість розробникам співпрацювати, ділитися кодом і створювати інноваційні програми. Розширення Chrome — чудовий спосіб створювати проекти з відкритим кодом, оскільки вони дозволяють розробникам швидко й легко створювати та поширювати свій код. Крім того,

розширення Chrome надають багато функцій, які полегшують розробку та розгортання, наприклад вбудовану підтримку для налагодження та тестування, доступ до широкого спектру API та можливість інтеграції з іншими службами та продуктами.

Для розробників важливо прислухатися до відгуків про їхні продукти, оскільки це може допомогти їм удосконалити та покращити свої продукти. Однак неможливо охопити всі побажання клієнта, оскільки клієнти мають різноманітні потреби та побажання, які розробник може бути неможливим задовольнити. Спеціальні сценарії сайтів можуть надати користувачам можливість налаштовувати сайти на свій розсуд. Це дозволяє користувачам налаштовувати свої веб-сайти відповідно до своїх унікальних потреб і уподобань, що може надати їм більш індивідуальний досвід.

Наразі проблема у сучасному світі інформаційних технологій полягає в недостатній персоналізації користувацького досвіду та неможливість задовольнити всі потреби користувачів. Незважаючи на технологічний прогрес і зростання складності комп’ютерних систем, досвід користувача не завжди залишається повністю задовільним. З величезною кількістю доступних опцій неможливо створити ідеальну взаємодію з користувачем, яка б відповідала всім потребам. Завжди є компроміси та рішення, які необхідно прийняти, щоб задовольнити найважливіші потреби користувачів. У цій роботі досліджено деякі недоліки користувальницького досвіду та труднощі спроби створити ідеальний користувальницький досвід. Віно підкреслює, як обмеження технологій і необхідність визначення пріоритетів потреб користувачів, створюючи групи незадоволених користувачів.

Створити інтерфейс обо функціонал, який буде привабливим для кожного користувача, практично неможливо через широкий діапазон уподобань, потреб і можливостей користувачів. Розробка системи, яка б відповідала потребам кожного користувача, вимагала б величезних ресурсів і часу, щоб налаштувати досвід для кожного окремо. Ось чому важливо

розробляти інформаційні продукти, задовольняючи користувальницький досвід, щоб задовольнити потреби більшості користувачів, водночас дозволяючи певний рівень персоналізації для тих, хто має більш специфічні потреби.

Розробка розширень для веб-браузера є важливою сферою розробки інформаційних технологій, як ці розширення покращують взаємодію з користувачем і роблять перегляд веб-сторінок більш ефективним, безпечним і приємним. Розширення можуть надавати користувачам додаткові функції, такі як блокування реклами, швидкий доступ до часто використовуваних веб-сайтів і навіть захист паролем. З розвитком Інтернету зростає потреба у більшій безпеці для захисту інформації користувачів від хакерів, зловмисного програмного забезпечення та інших цифрових загроз. Розширення — це ефективний спосіб забезпечити користувачам безпеку, необхідну для безпечного та безпечного користування Інтернетом.

Окрім забезпечення безпеки, багато розширень також можуть зробити роботу користувача швидшою та зручнішою. Наприклад, розширення можуть забезпечити швидкий доступ до часто відвідуваних веб-сайтів, автозаповнення форм і навіть автовиправлення помилок. Завдяки цьому користувачам легше та швидше знаходити те, що вони шукають, а перегляд веб-сторінок стає приємнішим. Багато розширень також дозволяють користувачам налаштовувати свій досвід перегляду, надаючи доступ до тем, плагінів та інших функцій.

Зрештою, розробка розширень веб-браузера є важливою, оскільки вона робить роботу користувачів у мережі більш ефективною, безпечною та приємною. Це особливо важливо для компаній, які покладаються на свої веб-сайти для залучення клієнтів і нових. Оскільки Інтернет продовжує розвиватися, розширення будуть необхідними для захисту користувачів і забезпечення кращого загального досвіду перегляду.

Більшість розширень веб-переглядача призначені для досягнення певних цілей, наприклад підвищення безпеки веб-переглядача, надання додаткових функцій або додавання налаштувань. У деяких випадках це може вимагати від користувачів встановлення додаткового програмного забезпечення у своїх браузерах. Це може включати таке програмне забезпечення, як панелі інструментів, плагіни або програми сторонніх розробників. У деяких випадках може знадобитися додаткове програмне забезпечення для доступу до певних функцій або функцій. Крім того, деякі розширення веб-переглядача можуть містити зловмисні або небажані програми, такі як рекламне або шпигунське програмне забезпечення, яке може призвести до подальших збоїв у роботі користувача.

Scriptify — це розширення, розроблене, щоб допомогти користувачам писати додатковий код для будь-якої веб-сторінки. Він надає зручний інтерфейс для редагування коду веб-сторінки та дозволяє користувачам включати власний код JavaScript, HTML і CSS у веб-сторінки. Scriptify також містить бібліотеку готових сценаріїв, які можна використовувати для швидкого налаштування веб-сторінок за допомогою додаткових функцій. Scriptify дозволяє користувачам зберігати сценарії та застосовувати їх до кількох сторінок, а також забезпечує простий спосіб налагодження та тестування коду.

2 ПІДГОТОВКА ДО РОЗРОБКИ РОЗШИРЕННЯ ДЛЯ БРАУЗЕРУ

2.1 Аналіз процесу розробки додатку до браузеру

У Chrome Extensions Marketplace доступно багато розширень Chrome, які дозволяють користувачам запускати код на веб-сторінках. Приклади включають інструменти веб-розробників, які дозволяють користувачам налагоджувати та перевіряти код, сценарії вмісту, які дозволяють користувачам вставляти власний JavaScript на сторінку, і букмарклети, які дозволяють користувачам швидко отримати доступ до попередньо визначеного фрагмента JavaScript. Крім того, існують розширення, які дозволяють користувачам отримувати доступ до власних бібліотек коду, дозволяючи їм швидко вставляти фрагменти коду на веб-сторінки.

Асортимент інтернет-магазину Chrome залежить від мови та місця розташування, зазначеного в браузері. Але для створення свого додатку було прийнято рішення не використовувати вже готові рішення, які пропонує інтернет-магазин Chrome. Замість цього, можна створити своє розширення за допомогою технологій веб-розробки, таких як HTML, CSS та JavaScript. Розширення зустрічаються з різними компонентами, які залежатимуть від необхідної функціональності.

Розширення - це програми, засновані на подіях, які використовуються для модифікації або покращення роботи веб-переглядача Chrome. Події - це активатори браузера, такі як перехід на нову сторінку, видалення закладки або закриття вкладки. Розширення відстежують ці події, використовуючи скрипти у своєму фоновому службовці, які потім реагують за вказаними інструкціями.

Застосунок фонової служби завантажується, коли це потрібно, і вивантажується, коли він не працює. Деякі приклади сценарів подій включають такі дії:

- спочатку розширення встановлюється або оновлюється до нової версії;
- фонова сторінка прослуховувала подію, і подія відправляється;
- скрипт вмісту або інше розширення надсилає повідомлення;
- інший вигляд у розширенні, такий як спливаюче вікно, викликає runtime.getBackgroundPage.

Життєвий цикл розширення Google Chrome складається з кількох етапів, які мають пройти розробники, щоб створити й опублікувати успішне розширення. Життєвий цикл починається зі стадії розробки, коли розробники створюють код розширення та дизайн. Після завершення розробки розробники виконають тестування та налагодження, щоб переконатися, що їх розширення працює належним чином.

Коли розширення буде готове до випуску, команда розробників надішле його до веб-магазину Chrome для перевірки. Google перевірить розширення, щоб переконатися, що воно відповідає вимогам Веб-магазину Chrome. Якщо розширення відповідає вимогам, Google опублікує його у веб-магазині Chrome, щоб користувачі могли завантажити та встановити.

Коли користувач завантажить і встановить розширення, браузер почне керувати життєвим циклом розширення. Браузер перевірить наявність оновлень і завантажить усі доступні нові. Він також керуватиме налаштуваннями дозволів, дозволяючи користувачам контролювати, що може робити розширення. Користувач також може будь-коли видалити розширення.

Зрештою, розширення буде вилучено, коли розробник припинить надавати підтримку та оновлення. Користувач може продовжувати використовувати розширення, але воно більше не отримуватиме жодних оновлень або виправлень помилок.

Фонові сценарії є основними компонентами розширення Chrome і використовуються для забезпечення основних функцій розширення. Вони завантажуються у фоновому режимі та відповідають за налаштування функції

основного розширення, а також за зв’язок із браузером та іншими веб-сторінками. Фонові сценарії можуть прослуховувати події, взаємодіяти з API браузера та керувати вмістом розширення.

Сценарії вмісту – це файли JavaScript, які запускаються разом із веб-сторінками та взаємодіють із DOM сторінки. Сценарії вмісту можуть отримувати доступ і змінювати DOM веб-сторінки, і вони використовуються для налаштування зовнішнього вигляду або вмісту веб-сторінки. Сценарії вмісту також можуть взаємодіяти з фоновим сценарієм та іншими сценаріями вмісту, а також їх можна динамічно завантажувати на веб-сторінку.

Після завантаження працівник служби продовжує працювати, доки виконує певну дію, наприклад, викликає API Chrome або видає мережевий запит. Крім того, службовець не розвантажується, доки не закриються всі видимі подання та всі порти повідомлень.

Скрипти вмісту - це файли, що працюють у контексті веб-сторінок. Використовуючи стандартну об'єктну модель документа (DOM), вони можуть читати деталі веб-сторінок, які відвідує браузер, вносити до них зміни та передавати інформацію своєму батьківському розширенню.

Сценарії вмісту можуть отримати доступ до API-інтерфейсів Chrome, використовуваних їх батьківським розширенням, обмінюючись повідомленнями з розширенням. Вони також можуть отримати доступ до URL-адреси файлу розширення за допомогою chrome.runtime.getURL () і використовувати результат так само, як інші URL-адреси.

Дозволити користувачам налаштовувати поведінку розширення можна, надаючи сторінку параметрів. Користувач може переглянути параметри розширення, клацнувши правою кнопкою миші піктограму розширення на панелі інструментів, потім вибравши параметри або перейшовши на сторінку управління розширенням за адресою chrome://extensions, знайшовши потрібне розширення, клацнувши Деталі, а потім вибравши посилання параметрів.

Інтерфейс розширення повинен бути цілеспрямованим і мінімальним. Подібно до самих розширень, інтерфейс користувача повинен налаштувати або покращити досвід перегляду, не відволікаючи від цього.

2.2 Створення розширення

Розширення починаються з їх маніфесту, а саме з створення файлу із назвою manifest.json і додаванням наступного коду.

```
{
  "name": "Getting Started Example",
  "description": "Build an Extension!",
  "version": "1.0",
  "manifest_version": 3
}
```

Рисунок 2.1 — Приклад файлу manifest.json

Кожне розширення має файл маніфесту у форматі JSON з іменем manifest.json, який надає важливу інформацію.

“manifest.json” – це файл JSON, який містить важливу інформацію про розширення Google Chrome, як-от його назву, версію, опис, дозволи та фонові сценарії. Він потрібен для всіх розширень Chrome і використовується для опису основних властивостей розширення.

JSON (JavaScript Object Notation) - простий формат обміну даними, зручний для читання і написання як людиною, так і комп'ютером. Він заснований на підмножині мови програмування JavaScript, визначеного в стандарті ECMA-262 3rd Edition - December 1999. JSON - текстовий формат, повністю незалежний від мови реалізації, але він використовує угоди, знайомі програмістам C-подібних мов, таких як C, C++, C# , Java, JavaScript, Perl,

Python і багатьох інших. Ці властивості роблять JSON ідеальною мовою обміну даними.

JSON заснований на двох структурах даних:

- колекція пар ключ / значення. У різних мовах, ця концепція реалізована як об'єкт, запис, структура, словник, хеш, іменований список або асоціативний масив;
- упорядкований список значень. У більшості мов це реалізовано як масив, вектор, список або послідовність.

Це універсальні структури даних. Майже всі сучасні мови програмування підтримують їх в будь-якій формі. Логічно припустити, що формат даних, незалежний від мови програмування, повинен бути заснований на цих структурах.

Каталог, що містить файл маніфесту, можна додати як розширення в режимі розробника у його поточному стані.

1. Відкрийте сторінку управління розширеннями, перейшовши до chrome: // extensions.
    - Або відкрийте цю сторінку, натиснувши кнопку меню Розширення та вибравши Керування розширеннями внизу меню.
    - Або відкрийте цю сторінку, натиснувши меню Chrome, наведіть курсор на Інші інструменти та виберіть Розширення
2. Увімкніть режим розробника, натиснувши перемикач поруч із режимом розробника.
3. Натисніть кнопку «Завантажити розпаковану» та виберіть каталог розширень.

Та-да! Розширення успішно встановлено. Оскільки в маніфесті не було включено піктограм, для розширення буде створено загальну піктограму.

Зараз розширення встановлено, але в даний час воно нічого не робить, оскільки ми не сказали йому, що робити і коли це робити. Давайте виправимо це, додавши якийсь код для зберігання значення кольору фону.

Для цього нам потрібно буде створити фоновий скрипт і додати його до маніфесту розширення. Почніть із створення файлу з іменем background.js всередині каталогу розширення.

Також в маніфест файлі потрібно вказати шлях до фонового скрипту.

```
{
  "name": "Getting Started Example",
  "description": "Build an Extension!",
  "version": "1.0",
  "manifest_version": 3,
  "background": {
    "service_worker": "background.js"
  }
}
```

Рисунок 2.2 — Додавання шляху до фонового скрипту через manifest.json

Фонові сценарії, як і багато інших важливих компонентів, повинні бути зареєстровані в маніфесті. Реєстрація фонового сценарію в маніфесті повідомляє розширення, на який файл посилатися, і як цей файл повинен поводитися.

Chrome тепер знає, що розширення включає працівника служби. Коли ви перезавантажуєте розширення, Chrome сканує вказаний файл для отримання додаткових інструкцій, таких як важливі події, які йому потрібно прослухати [1].

Це розширення потребуватиме інформації з постійної змінної відразу після її встановлення. Почніть з включення події прослуховування для runtime.onInstalled у фоновому сценарії. Всередині слухача onInstalled розширення встановить значення за допомогою API зберігання. Це дозволить

декільком компонентам розширення отримати доступ до цього значення та оновити його.

Розширення можуть мати багато форм інтерфейсу користувача; цей використовуватиме спливаюче вікно. Створіть та додайте файл із назвою popup.html до каталогу розширення. Це розширення використовує кнопку для зміни кольору фону.

Як і фоновий сценарій, цей файл повинен бути оголошений у маніфесті, щоб Chrome міг представити його у спливаючому вікні розширення. Для цього додайте об’єкт дії до маніфесту та встановіть popup.html як спливаюче вікно дії за замовчуванням.

```
{
  "name": "Getting Started Example",
  "description": "Build an Extension!",
  "version": "1.0",
  "manifest_version": 3,
  "background": {
    "service_worker": "background.js"
  },
  "permissions": ["storage"],
  "action": {
    "default_popup": "popup.html"
  }
}
```

Рисунок 2.3 — Додавання інтерфейсу розширення через manifest.json

Позначення піктограм на панелі інструментів також включено під дію в поле default_icons. Завантажте папку зображень сюди, розпакуйте її та помістіть у каталог розширення. Оновіть маніфест, щоб розширення знало, як використовувати зображення.

```
    "default_icon": {
      "16": "/images/get_started16.png",
      "32": "/images/get_started32.png",
      "48": "/images/get_started48.png",
      "128": "/images/get_started128.png"
    }
```

Рисунок 2.4 — Додавання іконок для розширення через manifest.json

Розширення також відображають зображення на сторінці управління розширеннями, попередження про дозволи та значок. Ці зображення позначені в маніфесті під піктограмами.[2]

```
  "icons": {
    "16": "/images/get_started16.png",
    "32": "/images/get_started32.png",
    "48": "/images/get_started48.png",
    "128": "/images/get_started128.png"
  }
```

Рисунок 2.4 — Додавання іконок для розширення на сторінці управління через manifest.json

Маніфесту знадобиться дозвіл activeTab, щоб дозволити розширенню тимчасовий доступ до поточної сторінки, а також дозвіл на сценарії для використання методу ExecuteScript API сценаріїв.

```
  "permissions": ["storage", "activeTab", "scripting"],
```

Рисунок 2.5 — Додавання необхідних дозволів для розширення через manifest.json

Щоб використовувати більшість API-інтерфейсів chrome. Розширення або програма повинні оголосити про свій намір у полях дозволів маніфесту. Розширення можуть вимагати трьох категорій дозволів, зазначених за допомогою відповідних ключів у маніфесті:

- «permissions» містять елементи зі списку відомих рядків (наприклад, "геолокація");
- «optional_permissions» схожі на звичайні дозволи, але надаються користувачем розширення під час виконання, а не заздалегідь;
- «host_permissions» містять один або кілька шаблонів збігів, що дають доступ одному або декільком хостам.

Дозволи допомагають обмежити шкоду, якщо ваше розширення або додаток скомпрометовано шкідливим програмним забезпеченням. Деякі дозволи відображаються користувачам для отримання згоди перед встановленням або під час виконання, як це потрібно.

Приклад частини дозволів файлу маніфесту наведено на рисунку 2.6.

```
"permissions": [
  "tabs",
  "bookmarks",
  "unlimitedStorage"
],
"optional_permissions": [
  "unlimitedStorage"
],
"host_permissions": [
  "http://www.blogger.com/",
  "http://*.google.com/"
],
```

Рисунок 2.6 — Приклад дозволів в файлі manifest.json

3 ПРОЦЕС РОЗРОБКИ РОЗШИРЕННЯ ДЛЯ БРАУЗЕРУ

3.1 Організація процесів.

Розробка будь-якого продукту являється непростим завданням. А для отримання гарного результату роботи, потрібно роботу добре спланувати. Планування має велике значення для проекту, оскільки проект містить те, що раніше не виконувалося, і включає порівняно багато процесів, які охоплюють всі етапи проектного циклу: створення концепції проекту; вибір стратегічного рішення щодо виконання проекту і розробка деталей проекту. Існує неймовірна кількість методик розробки програмних продуктів, і кожен з них має як свої переваги, так і свої недоліки. Також деякі методики можуть бути більш релевантними для одного типу задач або продуктів, а деякі – для інших. Для більш ефективного процесу виконання дипломної роботи було розглянуто декілька методологій та обрано найбільш привабливу та відповідну до типу поставленої задачі.[3] Серед розглянутих методик були:

- Agile Modeling - набір концепцій, принципів і методів (практик), який дозволяє швидко і легко виконувати проектування та документацію для проектів з розробки програмного забезпечення. Не включає в себе докладні інструкції з проектування, містить описи, як побудувати діаграми UML. Основна мета – ефективне моделювання та документація, але не включає в себе програмування і тестування, управління проектом, розгортання та обслуговування системи;
- OpenUP. Цей метод розробки є ітераційно-інкрементний. Позиціонується як одна із версій RUP. OpenUP ділить життя проекту на чотири фази: початковий етап, фаза специфікації, розробка та передача готового фунціоналу. Життєвий цикл проекту надає зацікавленим особам та членам колективних точку відліку і прийняття рішень протягом всього проекту. В цьому випадку можливо ефективно контролювати процес розробки;

- Kanban software development. Канбан реалізує принцип «точно в строк» і урівноважує робоче навантаження між усіма членами команди. За допомогою цього методу весь процес розробки зрозумілий для всіх членів команди. Канбан є візуальною моделлю розвитку, яка показує те, що потрібно виробляти, коли і скільки;
- Scrum. Встановлює правила для процесу управління розробки програмного забезпечення і дозволяє використовувати існуючу практику кодування, регулювати вимоги або приймати тактичні зміни. Використовуючи цю методику можливо виявити і усунути відхилення від бажаного результату на більш ранніх стадіях розробки програмного забезпечення;
- Scrumban - є структурою управління і гібрид Scrum і Kanban. Розробники працюють з історіями користувачів і намагаються зберігати ітерації якомога коротшими. Тут не існує конкретних ролей, як наприклад в Scum. Кожен член команди зберігає свою існуючу роль в проекті.
- Тут також деякі інші методології розробки програмного забезпечення по Agile: Agile Unified Process (AUP), Agile Data Method, Essential Unified Process (EssUP), Getting Real.

Досконало вивчивши розглянуті методики та порівнявши їх сильні та слабі сторони, проаналізувавши доречність кожного з представлених претендентів, було прийнято зважене рішення про використання методології, представленої четвертою в наведеному вище списці, Scrum. Цю методику було обрано через порівняну простоту та нейбільшу відповідність до поставленої задачі. Scrum –поширена методологія розробки програмного забезпечення. Scrum — це кістяк процесу, який включає набір методів і попередньо визначених ролей. Головні дійові особи — ScrumMaster, той хто опікується процесами, веде їх і працює як керівник проекту, Власник Продукту, людина, що представляє інтереси кінцевих користувачів та інших

зацікавлених в продукті сторін, та Команду, яка включає розробників. Обов’язковим артефактом при роботі за методологією Scrum є Product Backlog. Product backlog — це документ, який має список вимог до функціональності, які упорядковані згідно зі ступенем важливості. Product backlog представляє список того, що повинно бути реалізовано. Елементи цього списку називаються «історіями» (user story) або елементами backlog-у (backlog items). Product backlog відкритий для редагування усім учасникам Scrum-процесу.[4]

Через відсутність команди при виконанні дипломної роботи методику Scrum було спрощено та адаптовано для комфортної та ефективної роботи однієї персони. Було вилучено стендапи перед командою, адже команда налічувала лише одну людину. Було збережено ітеративність імплементування змін для розділення процесів розробки. Backlog було організовано завдяки функціоналу інтерактивної дошки планування задач Trello, розміщеної у відкритому доступі в інтернеті.

Trello — безплатна багатоплатформна система управління проектами, розроблена Fog Creek Software. Вона використовує парадигму керування проектами, відому як канбан. Проекти зображуються дошками, що містять списки. Списки містять картки, якими зображуються задачі. Картки повинні переходити з попереднього списку до наступного (за допомогою перетягування), таким чином зображаючи рух якоїсь функції від ідеї, аж до тестування. Картці може бути присвоєно відповідальних за неї користувачів. Користувачі та дошки можуть об'єднуватись в команди.

Як вже було оголошено вище, Product Backlog будо організовано у виді Trello таблиці.

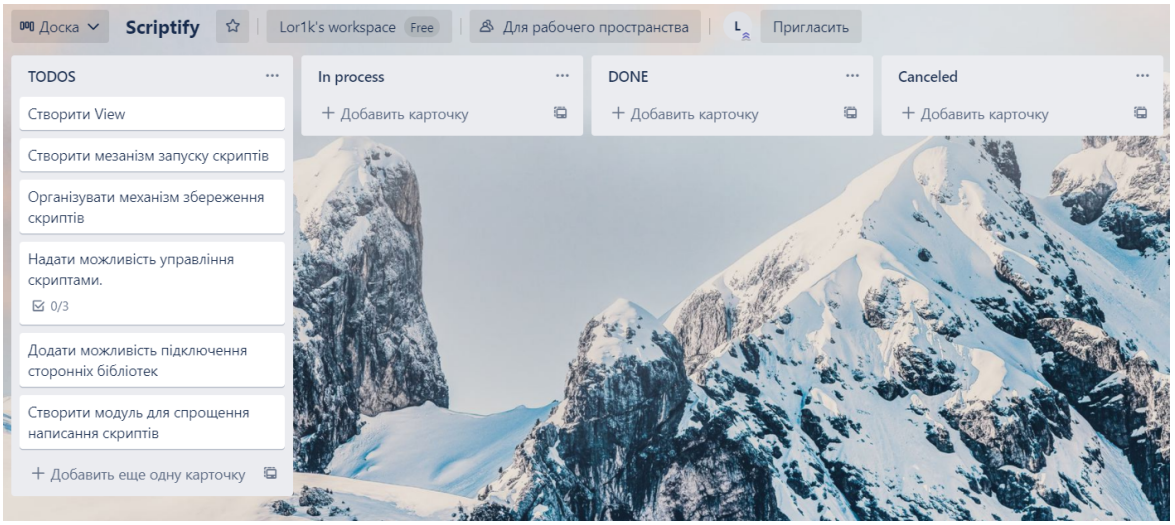

Рисунок 3.1 — Дошка – Product Backlog

Як можна помітити, на дошці (рис. 3.1) стовпець перенесення User Story було змінено но стовбець Canceled. Це ще одна відмінність застосованої методології від оригінального скраму. Як виявилося далі цей стовпець став у пригоді через неможливість виконання деяких додаткових задач. [5]

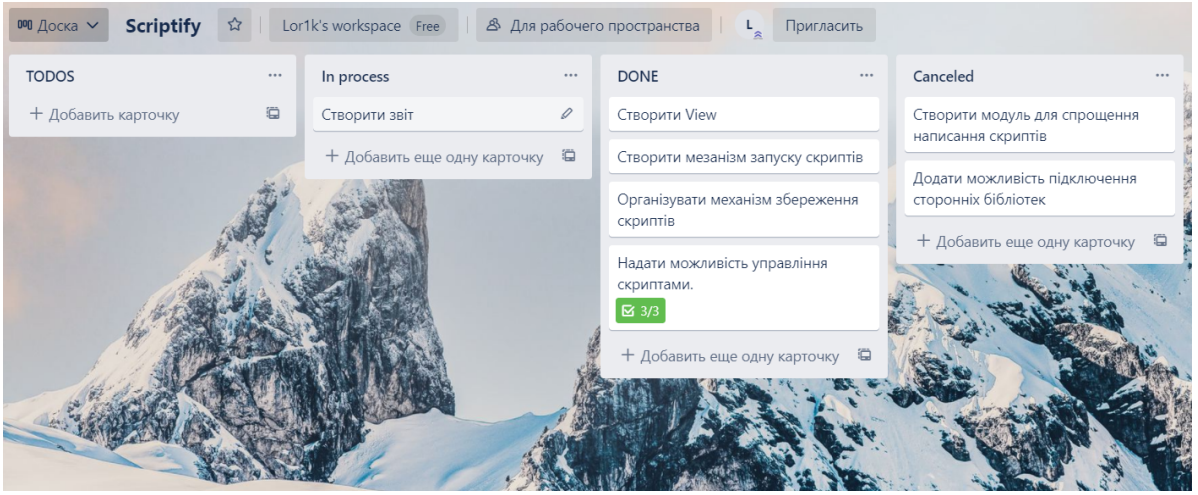

Рисунок 3.2 — Вигляд дошки на момент написання звіту з виконання практичного завдання

Хоча таблиця на рис. 3.2. містить невиконані додаткові завдання, фінальний результат є повним, функціональним розширенням. Додаток може

бути встановлений та містить всі основні функції. В таблиці на рис. 3.2. видно, що додаток має можливість запускати сторонні скрипти, що є основною функцією розширення. Всі скрипти зберігаються до наступного запуску браузеру та додатку. Тобто користувач повинен додати сценарій тільки один раз, і його буде збережено на тривалий час, але обране рішення має ряд нюансів та недоліків, які будуть описані далі, також буде описано альтернативні варіанти вирішення поставленої задачі, котрі не були використані через ряд причин, про котрі також буде згадано у відповідному розділі пояснювальної записки. Додатково було створенні можливості організації створених скриптів, які наведені в картці «Надати можливість управління скриптами.» у виді чек-листу, що є однією з найважливіших функцій безплатної багатоплатформної системи управління проектами Trello. Запланований та, як видно з таблиці, створений функціонал можна побачити на рис 3.3. нижче.

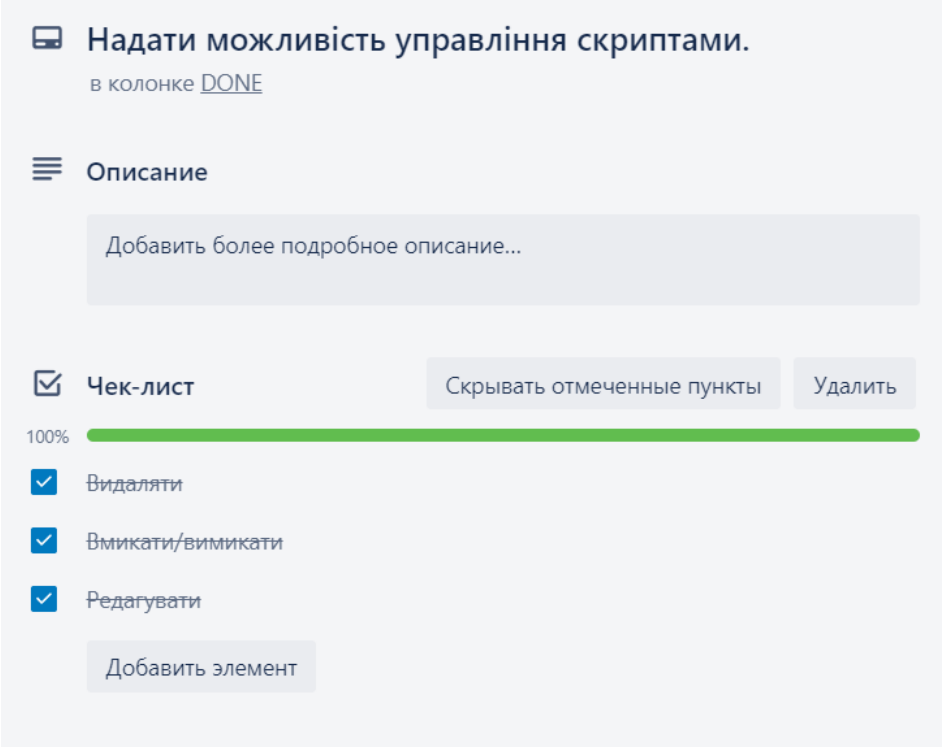

Рисунок 3.3 —Чек-лист створених можливостей організації створених у додатку сценаріїв

Виконавши основну задачу – планування проекту, створивши список задач по створенню проекту, провівши аналіз поставлених задач, обравши методику розробки програмного забезпечення, створивши дошку в Trello, було розпочато розробку застосунку. Адже при наявності конкретного плану роботи, процес розробки буде найбільш ефективним, менше часу буде затрачено на вибір та постановку задачі в процесі виконання роботи. Також це допомагає не відволікатися на конкретизацію та формалізацію завдання під час роботи над проектом. Також управління проектами (або проектний менеджмент) як раз і допомагає швидко і ефективно досягати поставлених цілей. Крім того, в процесі цього формується ціла система комплексів, які можуть бути використані для досягнення загальних цілей компанії, а також розробляється схема розумного розподілу ресурсів.[6]

3.2 Розробка розширення для браузеру

Аналіз поставлених показав, що основною задачею застосунку є власне запуску скриптів, на певних сторінках. Google Chrome має вбудовані інструменти для розробників, які серед яких є консоль, котра надає можливості введення команд мовою JavaScript.

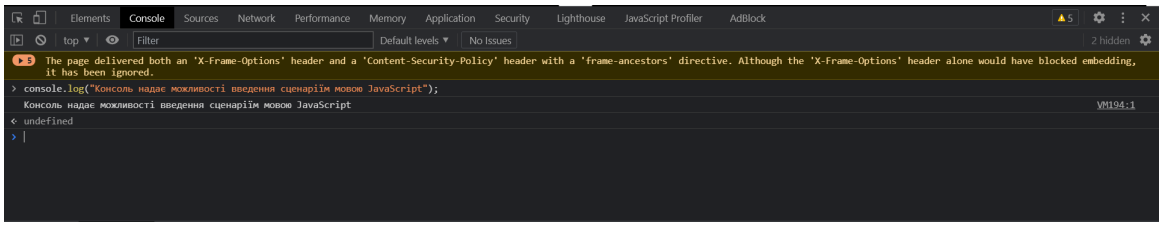

Рисунок 3.4 — Приклад введення команд в консоль Google Chrome

Але консоль розробника браузеру Google Chrome має можливість виконувати не тільки поодинокі комади, а й цілі блоки команд обмежених символом «;». Також вона спроможна опрацьовувати всі кодові структури

мови JavaScript, такі як: for{…}, if(…){…}else{…}, while(…){…}, do{…}while(…), try{…}catch(…){…} та інші існуючі команди мови JavaScript.

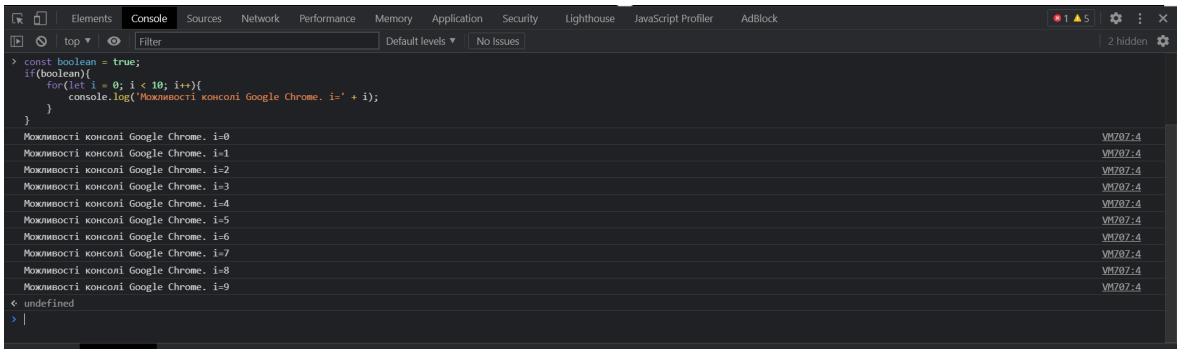

Рисунок 3.5 —Демонстрація можливостей консолі розробника у браузері Google Chrome

Хоча консоль розробника розробника у браузері Google Chrome надає широкі можливості для виконання скриптів в межах відкритої Web-сторінки, вона спроможна виконувати сценарії лише при умові ручного введення скриптів, що не вирішує поставлену задачу. Тому було прийнято рішення шукати іншу можливість виконання коду в браузері.

Провівши аналіз можливостей мови JavaScript, було знайдено, як здалося на перший погляд, ідеальне рішення проблеми. [7]

Вбудована функція eval дозволяє виконувати рядок коду (рис. 3.6).

```
var x = 10;
var y = 20;
var a = eval("x * y") + "<br>";
var b = eval("2 + 2") + "<br>";
var c = eval("x + 17") + "<br>";

var res = a + b + c;
```

Рисунок 3.6 — Вбудована функція eval виконує код JavaScript наданий у вигляді радку

Рядок коду може бути великий, містити переклади рядків, оголошення функцій, змінні та ін. Аргумент функції eval() - рядок. eval () виконує що міститься в рядку вираз, один або кілька операторів JavaScript. Не варто викликати eval () для визначення значення арифметичного вираження; JavaScript обчислює їх автоматично.

eval () можна використовувати для обчислення значення арифметичного виразу, записаного в строковому вигляді, на більш пізній стадії виконання. Припустимо, існує змінна x. Можна відкласти обчислення виразу, в якому міститься x, якщо привласнити змінної цей вислів у вигляді рядка (припустимо, "3 * x + 2"), а потім викликати eval () в пізнішій точці коду.

Якщо аргумент, переданий eval(), не є рядком, eval () повертає його незмінним. У наступному прикладі визначений конструктор String, і eval () не вирахував значення виразу, записаного в строковому вигляді, а повертає об'єкт типу String.

eval() - небезпечна функція, яка виконує код, що проходить з усіма привілеями визиватель. Якщо ви запускаєте eval() з рядком, на яку можуть впливати зловмисники, то ви можете запустити шкідливий код на пристрій користувача з правами вашої веб-сторінки / розширення. Найбільш важливо, код третьої сторони може бачити область видимості, в якій був викликаний eval(), що може може привести до атак, схожим на Function.

Також eval(), як правило, повільніше альтернатив, тому що викликає інтерпретатор JS, тоді як багато інших конструкції оптимізовані сучасними JS двигунами. [8]

Є безпечні (і швидкі!) альтернативи eval() для загальних випадків використання.

Отже, eval () - це глобальна функція Javascript, яка оцінює рядок як команду. Основною причиною, через яку слід уникати його використання, є безпека. Оцінка коду JavaScript із рядка небезпечна. Рядок може складатися зі шкідливого коду, який буде запущений на машині користувача, а область дії,

куди було викликано eval(), буде піддана можливим атакам. Друга причина - продуктивність. Виклик eval() буде повільнішим, ніж використання альтернативних варіантів, оскільки він повинен викликати інтерпретатор JavaScript, який перетворить обчислюваний код на машинну мову. Це означає, що якщо ви запускаєте код більше одного разу, браузеру доведеться інтерпретувати той самий код ще раз, що є дуже неефективним.

Більше того, зміна типу змінної через eval() змусить браузер повторно запустити код. А використання змінних, які не входять до області застосування eval(), вимагатиме від браузера дорогого пошуку, щоб перевірити, чи існує змінна.

Найпростіша альтернатива - це використання windows.Function(). Це створює глобальну функцію області з рядка. Таким чином, ви можете написати власний синтаксичний аналізатор для оцінки коду із рядка. Менше ймовірно, що можливі атаки завдають шкоди порівняно з eval(). [9]

Функція eval() рідко використовується в сучасному JavaScript з-за високої вразливості та ефективності. Неправильне використання функції може призвести до запуску шкідливого коду на машині користувача та втрати даних. Існують сторонні альтернативи, які захистять від обчислення рядків JavaScript, і для простого використання ви можете використовувати глобальну функцію () для написання власної функції оцінки. Через ряд наведених вище причин, у використанні функції eval() було відмовлено, адже наражати користувачів на небезпеку виконання шкідливого коду є не найкращим вибором у розробці програмного забезпечення.

Одним з альтернативних рішень запуску JavaScript коду в браузері стало запуск коду через адресний рядок. При додаванні ко адреси сайту javascript:alert("hi"); буде виконано вказану команду. Насправді цей метод був доданий до спільного списку лише через його фактичне існування. Насправді цей спосіб має ряд недоліків та жодної переваги.

- Запуск скрипту можливо лише при завантаженні сторінки.

- Наявність завантаженого коду можливо помітити в адресному рядку браузеру.
- Довгий код нівечить посилання на сайт.
- Код неможливо запустити через додаток до браузеру.
- Він не працює в Google Chrome версії 80.0.3987.132 (офіційна збірка) (64-розрядної версії). Chrome автоматично видаляє javascript: префікс з адресного рядка.

Тому використання цього способу не просто є не бажаним, а й не можливим в першу чергу. Тому було продовжено пошуки підходящого способу інтеграції користувацького коду на сторінці. На даному етапі було відвідано немало посібників та форумів.

За допомогою глобальної мережі Інтернет та певного рівня кмітливості було знайдено ідеальний метод ін’єкції JavaScript коду. Хоча рішення може здатися таким, що приходить до згадки ледь не найпершим, знадобилось чимало часу, щоб виявити його ефективність та доречність для вирішення задачі. Спочатку було створено гіпотезу створювати <script></script> тег через код розширення, та поміщати в нього введений користувачем та збережений код.

Тег <script> використовується для вбудовування скрипту на стороні клієнта (JavaScript). Елемент <script> містить або оператори сценаріїв, або вказує на зовнішній файл сценарію через атрибут src.

Існує кілька способів виконання зовнішнього сценарію.

- Якщо async = "async": сценарій виконується асинхронно з рештою сторінки (скрипт буде виконуватися, поки сторінка продовжує розбір)
- Якщо async відсутній і defer = "defer": Сценарій виконується, коли сторінка закінчила розбір
- Якщо немає асинхронізації або відстрочки: сценарій отримується та виконується негайно, перш ніж браузер продовжить аналіз сторінки

Будь-який тип скрипта на стороні клієнта може бути записаний всередині тегу <script> у html. Тег сценарію ідентифікує блок коду сценарію на сторінці html. Він також завантажує файл сценарію з атрибутом src. Html 4.x вимагає атрибут type у тезі сценарію. Атрибут type використовується для ідентифікації мови коду сценарію, вбудованого в тег скрипта. Це вказано як тип MIME, наприклад text / javascript, text / ecmascript, text / vbscript тощо. Отже, для коду JavaScript вкажіть type = "text / javascript" у тезі сценарію на сторінці html 4.x. Сторінка HTML5 не вимагає атрибуту type у тегу <script>, оскільки в HTML 5 мовою сценарію за замовчуванням є JavaScript. Якщо ви не хочете писати JavaScript між тегом сценарію на веб-сторінці, ви також можете написати код JavaScript в окремому файлі з розширенням .js та включити його на веб-сторінку за допомогою тегу <script> і посилатися на файл за допомогою атрибута src.[10]

Браузер завантажує всі сценарії в тег head перед завантаженням та рендерингом HTML html. Рекомендується включати сценарії до закінчення тегу body, якщо сценарії не потрібні під час завантаження вікна. Сценарії також можна додати в кінці тегу body. Таким чином ви можете бути впевнені, що всі ресурси веб-сторінки завантажені і безпечно взаємодіяти з DOM.

Важливі моменти використання script тегу.

1. Код JavaScript повинен бути записаний з тегом <script>.
2. На зовнішній файл JavaScript (.js) можна посилатися за допомогою <script src = "/ PathToScriptFile.js"> </script>, де атрибут src використовується для вказівки повного шляху до файлу .js.
3. Стандарт Html5 не вимагає атрибуту type = "text / javascript", тоді як попередні стандарти HTML вимагають атрибута type.
4. Тег <script> можна додати до тегу <head> або <body>.
5. Сценарій, включений до тегу <head>, може не мати доступу до елементів DOM, оскільки <head> завантажується до <body>.

Напишіть сценарій перед закінченням тегу </body>, якщо коду сценарію потрібно отримати доступ до елементів DOM.

Провівши аналіз та ряд досліджень, було виявлено, що код, що знаходиться в середини script тегу виконується одразу після додавання тегу до сторінки і не має різниці, був тег з початку завантаження сторінки, або ж був доданий в результату виконання коду. Тому для вирішення поставленої задачі було обрано саме цей спосіб, через легкість його реалізації та доречність використання. Крім того, інтеграція окремого скриптового тегу дає доступ до всієї DOM структури сторінки для маніпуляції html. Також це надає можливість користувачам створювати <style> теги для додавання власних CSS правил до сторінки.

Тег <style> використовується для визначення інформації про стиль (CSS) для документа.

Всередині елемента <style> ви вказуєте, як елементи HTML повинні відображатись у браузері.

Примітка. Коли браузер читає таблицю стилів, він відформатує документ HTML відповідно до інформації в таблиці стилів. Якщо деякі властивості були визначені для одного і того ж селектора (елемента) в різних таблицях стилів, буде використано значення з останньої прочитаної таблиці стилів.[11]

Щоб зробити посилання на зовнішню таблицю стилів, використовується тег <link>.

Починаючи з HTML 5.2, елемент <style> тепер дозволений в тілі документа. Однак специфікація HTML радить наступне: елемент стилю бажано використовувати в заголовку документа. Використання стилю в тілі документа може спричинити рестайлінг, макет тригера та / або перефарбування, а отже, слід використовувати з обережністю.

Елемент <style> має досить багато історії щодо його розміщення всередині елемента body. До HTML 5.2 елемент <style> дозволялося

розміщувати лише в голові документа (хоча розміщення його в тілі стало широко поширеною практикою серед розробників CSS). Було багато суперечок щодо того, чи слід дозволяти елемент <style> в тілі документа чи ні (з різних причин).[12]

Незважаючи на це, більшість браузерів підтримували елемент <style>, розміщений в елементі body. Чорнові версії специфікації HTML 5.1 включали атрибут масштабу, який дозволяв елементу <style> відображатися в тілі документа.

Атрибут дозволив би авторам визначати стилі лише для підрозділу документа (тобто вони не впливали б на решту документа). Однак атрибут scoped був вилучений із специфікації HTML 5.1 на початку 2016 року.

До елемента HTML можна додати атрибути, щоб надати більше інформації про те, як елемент повинен виглядати або поводитися.

CSS — це спеціальна мова стилю сторінок, що використовується для опису їхнього зовнішнього вигляду. Самі ж сторінки написані мовами розмітки даних.

CSS є основною технологією всесвітньої павутини, поряд із HTML та JavaScript.[13]

Найчастіше CSS використовують для візуальної презентації сторінок, написаних HTML та XHTML, але формат CSS може застосовуватися до інших видів XML-документів.

Специфікації CSS були створені та розвиваються Консорціумом Всесвітньої мережі.

CSS має різні рівні та профілі. Наступний рівень CSS створюється на основі попередніх, додаючи нову функціональність або розширюючи вже наявні функції. Рівні позначаються як CSS1, CSS2 та CSS3. Профілі — сукупність правил CSS одного або більше рівнів, створені для окремих типів пристроїв або інтерфейсів. Наприклад, існують профілі CSS для принтерів, мобільних пристроїв тощо.

CSS (каскадна або блочна верстка) прийшла на заміну табличній верстці веб-сторінок. Головна перевага блочної верстки — розділення змісту сторінки (даних) та їхньої візуальної презентації.

Обравши метод запуску коду та реалізувавши функцію його запуску, яку можна побачити на рисунку, було поставлено задачу про збереження скриптів для подальшого їхнього використання.[13]

```
function runScript () {
    let url = window.location.href.split('?')[0];

    chrome.storage.local.get(url, function (res) {
        console.log(url);
        if (res[url] && res[url].enabled) {
            let scriptTag = document.createElement('script');
            scriptTag.setAttribute('defer', '');
            scriptTag.innerHTML = innerHTML = res[url].script;
            setTimeout(()=>{
                document.querySelector('body').append(scriptTag);
            }, 2000);
        }
    });
}
```

Рисунок 3.7 — Функція runScript, що виконує код на сторінці.

Оскільки JavaScript, як мова на якій розробляється застосунок, не має доступу до сесій сайтів, було розпочато пошук оптимального рішення задачі.

Найбільш передбачуване рішення – використання бази даних. Через неможливість використовувати такі популярні БД, як MySQL або SQLite, за допомогою JavaScript, було розглянуто інші бази даних, що надають можливості маніпуляції даним через API та запити. Серед таких БД є наступні: Firebase; MongoDB; Cassandra; Couchbase; Redis; LevelDB; Neo4j.[14]

База даних Firebase Realtime - це база даних, розміщена в хмарі. Дані зберігаються в форматі JSON і синхронізуються в реальному часі з кожним підключеним клієнтом.

MongoDB - це база даних документів, що означає, що вона зберігає дані у схожих на JSON документах. MongoDB Atlas - це багатохмарний сервіс баз даних для MongoDB, доступний на AWS, Google Cloud та Azure. Найкраща у своєму класі автоматизація та вбудовані перевірені практики забезпечують постійну доступність, еластичну масштабованість та підтримку з дотриманням нормативних вимог.[15]

Apache Cassandra - це нереляційна відказостійка розподілена СУБД, розроблена при створенні високомасштабних та надмірних сховищ величезних масивів даних, представлених у відео хеша. Проект був розроблений на мові Java у корпораціях Facebook у 2008 році, а переданий фонд Apache Software Foundation у 2009. Ця СУБД відноситься до гібридного рішення NoSQL, оскільки вона збирає модель зберігання даних на базі сімейств таблиць (ColumnFamily) з концепцією ключ-значення (ключ-значення).

Couchbase Server - це розподілена, гнучка база даних документів JSON, орієнтована на використання в пам'яті, яка строго узгоджена з локальним кластером. Couchbase Server також підтримує перекрестну реплікацію центрів обробки даних з можливістю узгодженості між кластерами.

Redis - сховище структури даних в пам'яті, що використовується як база даних, кеш-пам’ятки та посередник повідомлень з відкритим кодом. Redis надає такі структури даних, як рядки, хеші, списки, набори, відсортовані набори із запитами діапазону, растровими зображеннями, гіперлогічними журналами, геопросторовими індексами та потоками. Redis має вбудовану реплікацію, сценарії Lua, виселення LRU, транзакції та різні рівні збереження на диску, а також забезпечує високу доступність через Redis Sentinel та автоматичне розділення за допомогою кластера Redis.

LevelDB - це високопродуктивна NoSQL система для зберігання даних в форматі ключ / значення, розроблена компанією Google. Сховище LevelDB написано на мові C++ і підключається до додатків у вигляді вашої бібліотеки (як SQLite і BerkeleyDB), забезпечуючи можливість зберігання впорядкованих наборів даних, в яких рядкові ключі зіставленні із строковими значеннями. [16]

Neo4j - провідна у світі база даних графіків. Це високопродуктивний сховище графіків з усіма функціями, які очікуються від зрілої та надійної бази даних, наприклад, дружньої мови запитів та транзакцій ACID. Програміст працює з гнучкою мережевою структурою вузлів та взаємозв’язків, а не зі статичними таблицями, але при цьому користується всіма перевагами бази даних корпоративної якості. Для багатьох додатків Neo4j пропонує переваги продуктивності на порядок порівняно з реляційними базами даних.

Оскільки робота з наведеними вище базами даних потребує використання додаткових модулів, та більш масштабної розробки, було прийнято рішення шукати інший спосіб збереження користувацького коду.

Наступним потенційним рішенням стало використання Cookies.

Cookies - це невеликі текстові файли у нас на комп'ютерах, в яких зберігається інформація про наших попередніх діях на сайтах. Крім входів в акаунти вони вміють запам'ятовувати:

- переваги користувачів, наприклад, мова, валюту або розмір шрифту.
- товари, які ми переглядали або додали в кошик;
- текст, який ми вводили на сайті раніше;
- IP-адреса і місце розташування користувача;
- дату і час відвідин сайту;
- версію ОС і браузера;
- кліки та переходи.

Коли ми робимо на сайті якісь дії, наприклад, додаємо товар до кошика або вводимо деталі входу в обліковий запис, сервер записує цю інформацію в Cookies і відправляє браузеру разом зі сторінкою. Коли ми переходимо на іншу сторінку сайту або заходимо на нього через час, браузер відправляє Cookies назад.[17]

Cookies бувають тимчасовими і постійними. Постійні Cookies залишаються на комп'ютері, коли ми закриваємо вкладку з сайтом, а тимчасові видаляються. Які саме Cookies використовувати на конкретному сайті - тимчасові або постійні - вирішує його розробник. Саме тому на одних сайтах ми не виходимо з облікових записів, навіть коли заходимо на них раз через кілька днів, а на інших вводимо пароль заново, хоча відійшли від комп'ютера на п'ять хвилин.

Хоча рішення з використанням Cookies ідеально підходить для збереження користувацького коду, цей спосіб не є зручним для подальших маніпуляцій над збереженими даними такими, як видалення або редагування. Наприклад, щоб відредагувати вже існуючий код, потрібно відвідати сайт для зчитування його Cookies та переписувати зміст коду там. Тому пошуки було продовжено.[18]

Варто згадати, що збереження даних у файлах не є можливим, адже подібний функціонал потребує створення серверної частини застосунку, а це вже виходить за рамки розробки проекту, та не відповідає наявним ресурсам.

Альтернативним та вдалим рішенням стало використання Google Chrome Storage, пам'ять браузеру Google Chrome.

API Google Chrome Storage оптимізовано для задоволення конкретних потреб сховищ розширень. Він надає ті ж можливості зберігання, що і localStorage API, з такими ключовими відмінностями:

- Дані користувача можна автоматично синхронізувати із синхронізацією Chrome (за допомогою storage.sync).

- Сценарії вмісту вашого розширення можуть безпосередньо отримувати доступ до даних користувача, не потребуючи фонової сторінки.
- Налаштування розширення користувача можна зберегти, навіть якщо використовується поведінка в режимі інкогніто.
- Це асинхронно з масовими операціями читання та запису, і, отже, швидше, ніж блокуючий та послідовний API LocalStorage.
- Дані користувача можна зберігати як об'єкти (localStorage API зберігає дані у рядках).
- Політики підприємства, налаштовані адміністратором для розширення, можна читати (використовуючи storage.managed зі схемою).

Для використання Google Chrome Storage потрібно вказати відповідні дозволи додатку.

```
{
  "permissions": [
    "storage"
  ],
}
```

Рисунок 3.8 — Дозволи в manifest файлі для використання Google Chrome Storage додатком

chrome.storage - це не велике сховище. Це багато малих сховищ. Для більшої ясності це сховище можна заповнити, і якщо воно заповняться, коли додаються нові дані, вони потрапляє в чергу, і ця затримка контролюється тим, хто вкладає величезну кількість матеріалу в сховище.[19]

Коли Chrome не має підключення до мережі Інтернет, Chrome зберігає дані локально. Наступного разу, коли браузер буде в Інтернеті, Chrome

синхронізує дані. Навіть якщо користувач вимкне синхронізацію, storage.sync все одно працюватиме. У цьому випадку він буде поводитися ідентично storage.local.

```
chrome.storage.local.set({key: value}, function() {
  console.log('Value is set to ' + value);
});

chrome.storage.local.get(['key'], function(result) {
  console.log('Value currently is ' + result.key);
});
```

Рисунок 3.9 — Приклад використання storage.local

Можливе питання, що можна використовувати localstorage, що може використовуватися різними браузерами, а не тільки в Google Chrome. На це можна дати просту відповідь.

Властивість localStorage лише для читання інтерфейсу вікна дозволяє отримати доступ до об'єкта Storage для походження документа; збережені дані зберігаються протягом сеансів браузера.

localStorage подібний до sessionStorage, за винятком того, що хоча дані localStorage не мають часу закінчення, дані sessionStorage очищаються, коли сеанс сторінки закінчується - тобто, коли сторінку закрито. (дані localStorage для документа, завантаженого в сеансі "приватного перегляду" або "анонімного перегляду", очищаються, коли закрита остання вкладка "приватно.")[20]

Ключі та значення, що зберігаються у localStorage, завжди мають формат DOMString UTF-16, який використовує два байти на символ. Як і у випадку з об'єктами, цілочисельні ключі автоматично перетворюються на рядки.

Для документів, завантажених із файлу: URL-адреси (тобто файли, що відкриваються у браузері безпосередньо з локальної файлової системи користувача, а не подаються з веб-сервера), вимоги до поведінки localStorage невизначені та можуть відрізнятися в різних браузерах.

У всіх поточних браузерах localStorage, здається, повертає інший об'єкт для кожного файлу: URL. Іншими словами, кожен файл: URL, схоже, має свою унікальну локальну область зберігання. Але немає жодних гарантій щодо такої поведінки, тому вам не слід на неї покладатися, оскільки, як зазначалося вище, вимоги до файлу: URL-адреси залишаються невизначеними. Тож можливо, що браузери можуть будь-коли змінити свій файл: обробка URL-адрес для localStorage. Насправді деякі браузери з часом змінили спосіб обробки.

localStorage зберігає дані у вигляді: ім'я = значення (значення - строка), тому неможливо зберегти складні об'єкти. При чому, типи вроде true, false будуть зберігатися як "true" і "false".

chrome.storage.local - зберігання даних розширення. Дозволяє зберігати об'єкти, відтворюючи від localStorage. Звернення до даних відбувається асинхронно. Позволяє "послухати" зміни змінних.

chrome.storage.sync - працює як chrome.storage.local, а також зберігає дані на серверах, синхронізовані за Google-аккаунтом між усіма авторизованими браузерами.[21]

Тому згідно з наведених вище аргументів, проаналізувавши всі існуючі методи збереження даних, було обрано, описаний останнім, спосіб використання Google Chrome Storage, адже він задовольняє всі критерії, які необхідно задовільнити для виконання поставленої задачі.

Наступним під етапом стала організація даних, що потрібно зберігати. Хоча chrome.storage і зберігає дані у вигляді об’єктів, але структуру об’єкта, що зберігається була наведена в форматі json на рис. 3.2.7.

```
{
    "https://youtube.com": {
        "enabled": true,
        "script": "console.log('This is YouTube script');"
    }
}
```

Рисунок 3.10 — Формат збереження скриптів у chrome.storage

Як можна помітити, скрипти зберігаються не у виді масиву, а у вигляді об’єкту, а ключами кожного з об’єктів, що зберігаються є посилання на сайт. Таке рішення було обрано для легшого пошуку по всіх скриптах, що збережені в сховищі, адже отримавши посилання сайту, який було відвідано, при завантаженні сторінки, можна отримати потрібний скрипт моментально, звернувшись до відповідного скрипту використавши отримане посилання як ключ. Також на рис. 3.6 можна помітити крім поля script, що містить власне код, який виконується на сторінці, поле enabled, що зберігає логічне значення true/false. Його було додано для можливості користувачам “вмикати або вимикати” виконання коду, якщо користувач хоче зберегти написаний код, але не виконувати на сторінці, то йому достатньо зняти галочку біля скрипту. Вміст поля script додається в середину тегу script, який вставляється на відповідну сторінку. Як вже було згадано раніше, код, який міститься всередині тегу script, виконується відразу після додавання його на сторінку. Так отримується вже готова функціональність, яка вирішує поставлену задачу.

Вирішивши практичні проблеми, розібравшись з технічними деталями, було розпочато роботу над наступним етапом розробки – створення інтерфейсу. Насправді через відсутність досвіду розробки користувацьких інтерфейсів, було прийнято рішення зробити інтерфейс максимально простим.

Всього розширення має три екрани: головний, що містить список збережених скриптів та кнопку “Add script to current page”, що веде на екран додавання нового скрипту, екран редагування існуючого коду та екран для додавання нових сценаріїв.

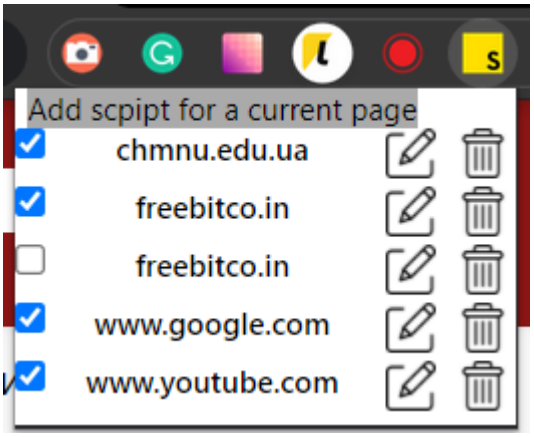

Рисунок 3.11 —Головний екран розширення

На головному екрані (рис 3.11) знаходиться список збережених скриптів з кнопками редагування та видалення. Також можна помітити кнопку додвавання нового сценарію, при натисканні якої перед користувачем з’являється екран з полем введення тексту та кнопкою збереження. Новому скрипту буде присвоєно поле enabled зі значенням true, та посилання поточної сторінки з видаленням GET-запитів.

GET-запит - метод передачі даних від клієнта до сервера з метою отримання інформації, зазначеної за допомогою конкретних GET-параметрів.

Це публічні дані, доступні при повторному перегляді посилання в історії. Такий запит актуально використовувати при незмінних даних в адресному рядку. Тобто при кожному зверненні до сторінки з заданими параметрами її адресу залишається постійним.

GET-запит складається з домену, адреси сторінки і параметрів, які слідують після знаку «?». Формат одного параметра виглядає так: «ключ = пояснення».

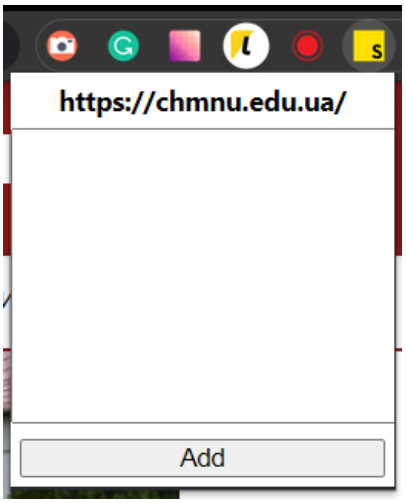

Рисунок 3.12 — Екран додавання нового сценарію

На рис. 3.12. можна побачити елементи екрану створення нового скрипту. В верхній частині вікна видно посилання на сайт, на якому буде виконуватися введений у полі нижче код. Та кнопку «Додати», яка виконує збереження скрипту до chrome.storage. Код збереження нових сценаріїв наведено на рисунку 3.13.

```
function saveScript () {
    let script = document.querySelector('textarea').value;

    if (!script) {
        console.error('Error: No script specified');
        return;
    }
    chrome.storage.local.set({ [url.split('?')[0]]: {enabled: true, script: script} }, function () {
        // Notify that we saved.
        console.log('Settings saved');
        window.location.href = '/popup.html';
    });
}

```

Рисунок 3.13 —. Функція saveScript, що зберігає введений код до локального сховища

Також на рисунку 3.13. видно роботу з chrome.storage API та перевірку на те, чи було введено в поле скрипт, чи ні. Рядок під номером 27 перенаправляє користувача на головний екран, де вже буде виведено список скриптів разом зі щойно доданим. Нижче, на рисунку 3.15 наведено код виведення списку збережених сценаріїв, елементів управління ними та код присвоєння обробників подій для проведення таких маніпуляцій над елементами, як: редагування, видалення та вмикання або вимикання.

Рисунок 3.15 — Функція updateList, що отримує зі локального сховища скрипти та будує з них список для виведення на головному екрані додатку

Як видно на рисунку 3.15, деякий код було приховано. В схованому коді знаходяться шаблони розмітки, HTML-код. Цей код в кінці виконання інструкцій буде виведено на головний екран розширення в виді списку збережених скриптів. Також на рядку двадцять п’ять додається обробник події «change» на елемент «checkbox» біля назви скрипту, яка є посиланням на сторінку, на якій виконується певний код. Цей обробник змінює значення поля enabled відповідно до того, в якому стані знаходиться checkbox. Ці маніпуляції вирішують, буде код виконуватися на відповідній сторінці, чи ні. Крім вище наведеного, додаються обробники події «click» на іконки

«редагувати» та «видалити». Логіку, та функціонал, заради якого було додано ці кнопки, передбачити не важко, але є ряд тонкощів, про які варто згадати. Необхідно передавати дані про обраний скрипт для редагування на іншому екрані. Це стало черговою проблемою в процесі розробки. Але проблему було вирішено нехитрим чином.

Як вже було згадано раніше всі елементи зберігаються в єдиному об’єкті в chrome.storage. Для вирішення задачі було прийнято зберігати посилання на обраний скрипт по унікальному ключу «current_url» (рис. 3.16) та за ним вибирати зі загального списку збереженого коду.

```
45  function edit (key) {
46      chrome.storage.local.set({ current_url: key }, function () {
47          console.log('Settings saved');
48      });
49      window.location.href = '/edit_script.html';
50  }
```

Рисунок 3.16 — Функція edit, що зберігає обраний ключ в поле «current_url»

Також на рисунку 3.16. видно перехід но екран редагування коду, де вже на момент його завантаження, в поле ведення буде виведено збережений код, який після натискання на кнопку «Зберегти», буде змінено у сховищі браузеру. Після завантаження прив’язаної до обраного коду сторінки буде виконано уже змінений скрипт.

Щодо видалення скриптів, цей процес значно простіший, адже видалення запису зі chrome.storage відбувається на тому ж екрані, да знаходиться іконка з корзиною, і не потребує додаткових маніпуляцій зі збереженням обраного скрипту до сховища. Єдине що варто зробити при натисканні на кнопку видалення та після завершення стирання запису – оновити список скриптів на головному екрані. Зробити це можна декількома способами: перезавантажити сторінку розширення, котра в самому початку

викликає описано вище функцію updateList, або ж викликати її власноруч, що видалить не актуальний список та замінить його на оновлений. Було обрано останній з перелічених варіантів, адже це, виходячи з всіх базових правил розробки веб-застосунків, є більш ефективним а елегантним рішенням, яке, додатково, виконується швидше, адже оновити потрібно лише невелику частину сторінки, а не завантажувати всю DOM-структуру та всі підключені до документу файли: js файли, css файли та інші.

3.3 Розробка сайту-бібліотеки користувацьких сценаріїв

Знання мови програмування JavaScript є невід'ємною частиною використання розширення Chrome Scriptify, оскільки дозволяє користувачам створювати та редагувати сценарії. Такі умови сильно зменшують кількість користувачів, що можуть користуватися додатком і поліплупавити свій користувацький досвід. Тому було поставлено задачу спростити користування додатком для користувачів, що не знайомі з програмуванням. Публічна бібліотека сценаріїв надасть користувачам рішення цієї проблеми, оскільки містить сценарії, створені іншими користувачами.

На сторінці каталогу загальнодоступної бібліотеки сценаріїв є панель пошуку, фільтри та список сценаріїв. Панель пошуку дозволяє користувачам шукати певні сценарії, а фільтри можуть допомогти звузити вибір. Список сценаріїв містить назву, опис, рейтинг і код для кожного окремого сценарію.

Одна сторінка сценарію має назву, опис сценарію, оцінки та код сценарію. Назва сценарію відображається у верхній частині сторінки, а опис містить більш детальне пояснення сценарію та його призначення. Оцінки дають користувачі та допомагають зрозуміти, наскільки ефективним є сценарій. Нарешті, код сценарію відображається внизу сторінки, показуючи користувачам фактичний код, який їм потрібно використовувати для запуску сценарію.

Спеціальні сценарії та розширення Chrome, яке їх запускає, можуть бути важливим способом надання користувачам можливості визначати поведінку своїх улюблених сайтів. Scriptify надає користувачам простий спосіб створювати, керувати та розгортати користувальницькі сценарії, щоб вони могли налаштувати свій досвід роботи в Інтернеті. Scriptify також дозволяє користувачам швидко та легко ділитися своїми користувальницькими сценаріями з іншими, що полегшує, ніж будь-коли, створення персоналізованого онлайнового досвіду.

Для створення візуальної оболочки сайту-бібліотеки було обрано популярний JavaScript фреймворк від компанії Google Angular.

Переваги використання Angular для розробки інтерфейсів включають:

- підвищена продуктивність і читабельність коду;
- простота використання з інтуїтивно зрозумілим і зручним синтаксисом;
- оптимізована продуктивність коду;
- можливість використовувати шаблони HTML для швидкого створення динамічних переглядів;
- інтеграція з існуючими кодовими базами;
- підтримка сучасних фреймворків і бібліотек, таких як TypeScript і Node.js;
- модульний підхід до створення додатків;
- чистіша та гнучкіша архітектура;
- підтримка односторінкових програм (SPA);
- можливості зв’язування даних, щоб підтримувати інтерфейс користувача в актуальному стані відповідно до вашої моделі даних;
- можливості реактивного програмування, щоб зробити роботу з асинхронними даними більш природною та ефективною;
- вбудована перевірка форм і підтримка власних валідаторів;
- підтримка відтворення на стороні сервера за допомогою Universal;

- простота тестування та налагодження.

Також не менш важливим у процесі розробки стало використання TypeScript.

Typescript — це мова програмування з відкритим вихідним кодом, розроблена та підтримувана Microsoft. Це надмножина JavaScript, яка забезпечує статичний тип, класи, інтерфейси та інші функції. Переваги використання Typescript включають покращену швидкість кодування, менше помилок, кращу читабельність коду та більше функцій, які можна використовувати для створення потужніших програм.

Для розробки бібліотеки вже не вдалося обійтись без використання баз даних. Перед створенням на налагодженням роботи проекту було детально досліджено методи та інструменти організації даних та способи їх збереження і отримання.Найпопулярнішими базами даних є MySQL, Oracle Database, Microsoft SQL Server і PostgreSQL.

MySQL - система управління реляційними базами даних з відкритим кодом; Плюси: безкоштовно, швидко, надійно; Мінуси: обмежена масштабованість, складна структура бази даних.

Oracle Database - власна система управління базами даних; Плюси: висока продуктивність бази даних, гнучкість; Мінуси: дорого, складна установка.

Microsoft SQL Server - реляційна система управління базами даних; Плюси: простий у встановленні та використанні, чудова масштабованість; Мінуси: обмежені функції бази даних, дорого.

PostgreSQL - об'єктно-реляційна система управління базами даних з відкритим кодом; Плюси: потужний і надійний, широкий вибір сумісних інструментів; Мінуси: труднощі з налаштуванням і налаштуванням, обмежена масштабованість.

Та більшість з них потребують окремого сервера для роботи, що не зовсім підходить для досягнення мети проекту. Тому було прийнято рішення розглянути хмарні NoSQL бази даних.

Бази даних NoSQL — це нереляційні бази даних, які зберігають і керують даними у форматах, відмінних від табличних зв'язків, які використовуються в реляційних базах даних. Приклади баз даних NoSQL включають MongoDB, Cassandra, Amazon DynamoDB і CouchDB.

MongoDB — це документно-орієнтована база даних NoSQL з відкритим вихідним кодом, яка зберігає дані в JSON-подібних документах. Він використовує динамічні схеми, що дозволяє зберігати дані більш гнучким способом, ніж реляційні бази даних.

Cassandra — це розподілена база даних NoSQL з відкритим кодом. Він дуже масштабований і забезпечує відмовостійкість. Він ідеально підходить для програм, які потребують високої доступності та масштабованості.

Amazon DynamoDB — це повністю керована служба бази даних NoSQL, яку надає Amazon Web Services. Це високодоступна, масштабована служба бази даних із низькою затримкою, яка використовується для швидкої розробки додатків Інтернет-масштабу.

CouchDB — це база даних NoSQL з відкритим кодом, яка використовує JSON для зберігання даних. Він розроблений для високої доступності та розподіленості, а також дозволяє легко тиражувати та синхронізувати дані в кількох екземплярах.

Та найкращим вибором для створення бази даних стала платформа Firebase, яка з 2014 року також належить Google.

Firebase – це платформа розробки мобільних і веб-додатків, розроблена Firebase, Inc. е 2011 році. Firebase надає розробникам низку функцій і переваг, як-от база даних у реальному часі, автентифікація користувачів, хостинг, зберігання , і більше. Firebase також надає інструменти аналітики, звіти про збої та можливості тестування A/B. Він розроблений, щоб допомогти

розробникам швидко створювати та масштабувати програми з мінімальними зусиллями.

Firebase — це платформа Backend-as-a-Service (BaaS), яка надає розробникам широкий спектр функцій і служб для створення веб-додатків, програм для мобільних пристроїв і планшетів. Функції Firebase включають синхронізацію даних у реальному часі, аутентифікацію користувачів, аналітику, звітування про збої, обмін повідомленнями в хмарі тощо. Firebase також надає набір інструментів розробки, включаючи інтегроване середовище розробки (IDE) та інтерфейс командного рядка (CLI). Firebase також пропонує низку клієнтських бібліотек, таких як Firebase JavaScript SDK, Node.js SDK та iOS SDK. Ці бібліотеки надають розробникам зручний спосіб доступу та керування службами Firebase, такими як автентифікація, зберігання та синхронізація даних у реальному часі.

Після того, як було обрано основні технології для розробки сайту-бібліотеки, процес створення проекту було розпочато. Було визначено список функціоналу для мінімально-працюючого проекту, він набув такого вигляду:

- головна сторінка зі списком доступних скриптів;
- можливість автентифікації користувача;
- додатково: надати можливість використовувати Google аккаунт доля логіну;
- сторінка конкретного сценарію з детальною інформацією про скрипт:
    - автор скрипта;
    - його назва;
    - опис скрипта;
    - статистика використання скрипта іншими користувачами;
    - код скрипта для продвинутих користувачів, які хочуть бути впевненими, що сценарій не наробить шкоди для них, адже

в код можна помістити що-завгодно і без його перегляду, дізнатись про його шкідливість неможливо.

Найголовнішим викликом у створенні проекту стало підключення Firebase до фронтенду, але для цього потрібно було пройти ряд кроків.

1. Встановити бібліотеки Firebase і AngularFire2 за допомогою менеджера пакетів npm.
2. Створити проект Firebase і додати конфігурацію Firebase до Angular проекту.
3. Імпортувати AngularFireModule і додати його до масиву імпортів у файлі app.module.ts.
4. Створити службу Firebase для обробки підключення до бази даних Firebase.
5. Додати службу Firebase у компонент, де її необхідно використовувати.
6. Використовувати службу Firebase для доступу до бази даних Firebase.

SDK (software development kit) було встановлено за допомогою npm (node package manager). І було створено проект на платформі Firebase.

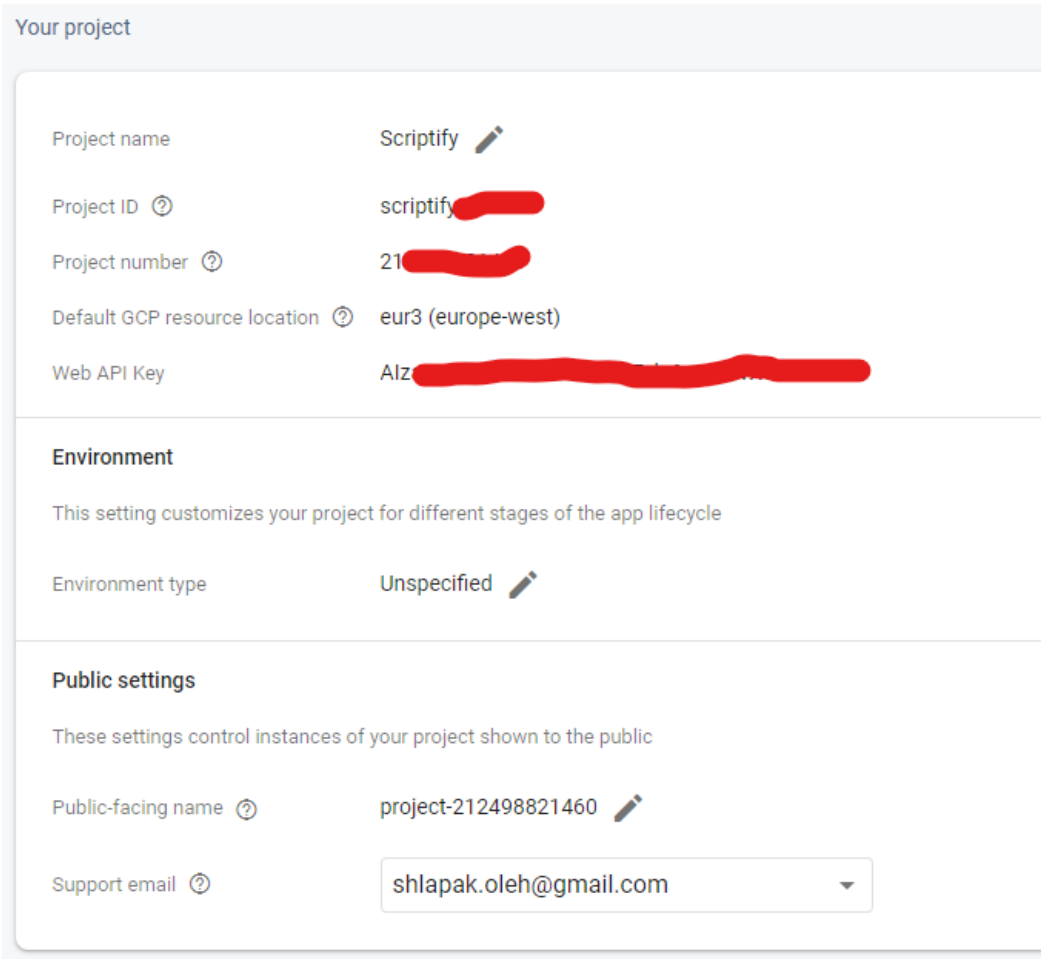

Рисунок 3.17 — Інформація про проект на платформі Firebase (секретну інформацію приховано)

Після чого інформацію для підключення було перенесено в середовище проекту.

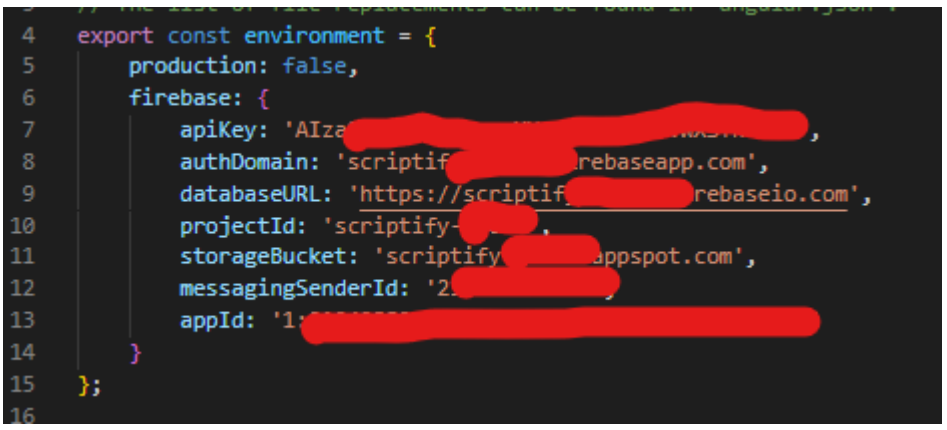

Рисунок 3.18 — Середовище проекту (секретну інформацію приховано)

Використовувати середовище необхідно для зберігання секретної інформації, оскільки це допомагає захистити інформацію від доступу неавторизованих користувачів. Середовища забезпечують додатковий рівень безпеки та можуть допомогти гарантувати, що конфіденційні дані не будуть скомпрометовані.

Після налаштування проекту було створено інтерфейс сайту використовуючи методи отримання та збереження даних за допомогою платформи Firebase.

На головній сторінці сайту розташовано хедер, в якому знаходиться назва сайту “Scriptify” і кнопка авторизації, що викликає службу Firebase, яка відкриває вікно використання Google акаунтів для ідентифікації користувача. Нижче розташовано банер із закликом встановити розширення Chrome, адже без нього користувацькі сценарії працювати не будуть, одразу під яким знаходиться список доступних сценаріїв, відсортованих за кількістю їх запусків, щоб надати розуміння користувачам, щодо їх ефективності та безпечності. Велика кількість запусків скрипту показує, що інші користувачі його використовують і, якщо вони його не видаляють, то він виконує саме ті функції, що зазначено в описі. Також лічильник запусків скрипту було додано для можливої монетизації розробки користувацьких сценаріїв, що б надало розробникам мотивації створювати контент для інших людей. Але монетизацій проєкту виходила за рамки дипломної роботи, через що її не було імплементовано, але було збережено як ідею для подальшого розвитку всієї платформи.

Рисунок 3.19 —Головна сторінка сайту

Справа зверху (див. рис. 3.19.) можна помітити кнопку “LOG IN”, яка відкриває вікно авторизації Google (див. рис. 2.20.)

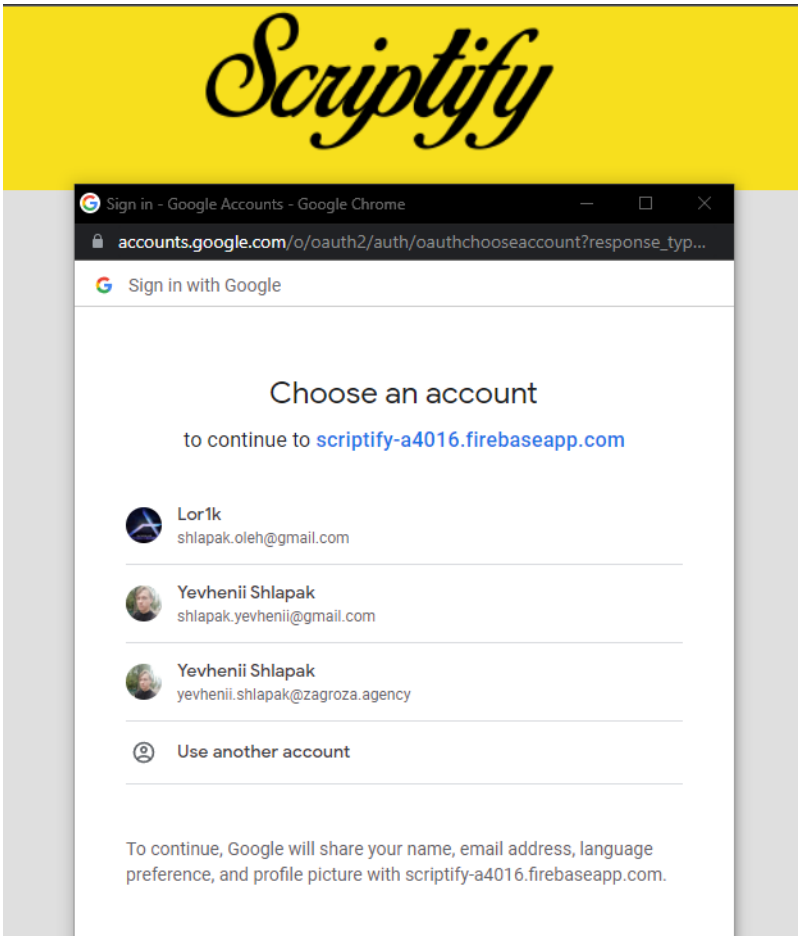

Рисунок 3.20 — Вікно авторизації через Google аккаунт

Таку можливість надає платформа Firebase. Passport.js — це альтернативний варіант бібліотеки JavaScript для автентифікації з різними стратегіями автентифікації, такими як OAuth, OpenID і LDAP, а також надає API високого рівня для аутентифікації та авторизації. Він підтримує кілька стратегій автентифікації та забезпечує розширену структуру, яку можна інтегрувати в будь-який веб-додаток на основі Express. Passport.js розроблено, щоб запропонувати просту, інтуїтивно зрозумілу та безпечну систему аутентифікації для веб-додатків. Але Firebase пропонує кращий спосіб автентифікації користувачів і зберігання пов’язаних із ними даних. Це також допомагає оптимізувати зберігання даних і керування іншими даними, які використовуються в програмі. Для мінімально працюючого проект було створено дві колекції: користувачів і скриптів. Колекція користувачів містить таку інформацію користувача, як ім’я, адреса електронної пошти та інша контактна інформація. Колекція сценаріїв містить сценарії, пов’язані з ідентифікатором користувача в колекції користувачів, так що кожен сценарій можна пов’язати з конкретним користувачем.

Рисунок 3.21 — Сторінка конкретного сценарію.

Сторінка сценарію також має містити посилання на інструкції щодо запуску коду, які інформуватимуть користувача про те, що йому потрібно зробити для виконання сценарію. Крім того, на сторінці також має бути можливість зв’язатися з автором сценарію, якщо у нього виникнуть запитання чи сумніви щодо коду. Сторінка також має містити посилання на сховище вихідного коду, щоб користувач міг переглянути код і переконатися, що його безпечно запускати. Нарешті, сторінка сценарію також має інформувати користувача про потенційні ризики, з якими вони можуть зіткнутися, якщо вирішили виконати код, а також про будь-які додаткові кроки, які вони повинні зробити для захисту своєї системи, наприклад створення резервної копії своїх даних перед запуском сценаріїв.

Люди повинні знати про небезпечний код, який запускається на сторінках, які вони відвідують, оскільки зловмисний код може використовуватися для отримання доступу до особистої інформації або заподіяння шкоди комп’ютерам. Зловмисний код можна використовувати для викрадення облікових даних для входу, розміщення зловмисної реклами та навіть вимкнення комп’ютерів. Щоб захистити себе від цих зловмисних кодів рекомендовано перевіряти кожен скрипт перед його запуском. Крім того, важливо знати типи шкідливого коду, який можна використовувати. Деякі поширені типи включають JavaScript і міжсайтовий сценарій (Cross Site Scipting). JavaScript можна використовувати для відстеження активності користувачів, виконання зловмисних команд або навіть запуску атак типу «відмова в обслуговуванні». Міжсайтовий сценарій (Cross Site Scipting) — це форма зловмисного коду, яка дозволяє впроваджувати шкідливий код на веб-сайт, дозволяючи зловмисникам викрадати інформацію або маніпулювати зовнішнім виглядом або поведінкою веб-сайту. Важливо знати про ці типи коду та вживати заходів для захисту свого комп’ютера та особистої інформації від них.

ВИСНОВКИ

В ході виконання роботи було проведено аналіз наявних інструментів та методів створення та запуску користувацьких скриптів і ринку браузерних розширень, в ході якого було виявлено необхідність створення власної системи кастомізації користувацького досвіду.

Дослідження сформувало бачення та розуміння ринку розширень для браузерів, проблематика користувацького досвіду та загального стану існуючих веб-ресурсів для розваг, бізнесу та сервісів, надає необхідні навички та досвід розробки інформаційних продуктів та сформувала вектор подальшого розвитку проекту.

При виконанні магістерської кваліфікаційної роботи було проаналізувано предметну сферу та розглянуто альтернативні системи організації та запуску користувацьких сценаріїв, досліджено потреби користувачів та актуальність створення системи індивідуальної системи кастомізації користувацького досвіду, проаналізувано та вивчено технології для створення функціонуючого продукту та створено і протестовано систему організації та відтворення користувацьких сценаріїв та сайт-бібліотеку скриптів. Таким чином мету МР було досягнуто і результатом стали покращення індивідуального користувацького досвіду та створення спеціальної бібліотеки сценаріїв і програми, яка її запускає, оскільки вони дозволяють користувачам налаштовувати роботу з певними веб-сторінками. Програма функціонує як розширення, яке можна встановити у веб-браузері та запускає код на певних сторінках, дозволяючи вносити зміни, визначені користувачем. Це може включати зміни у зовнішньому вигляді, відчуттях і функціональності веб-сторінки, що дозволяє користувачеві адаптувати свій досвід у спосіб, який найкраще відповідає його потребам.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Алан Бьюли. Изучаем SQL. – Санкт-Петербург : Вид-во Символ-Плюс, 2007. – 312 с.
2. Брайан Клифтон. Google Analytics для профессионалов, 3 видання. – Львів: Вид-во Діалектика, 2012. – 608 с.
3. Гудман, Д. JavaScript и DHTML. Сборник рецептов. Для профессионалов. - М.: Питер, 2015. - 523 с.
4. Дакетт, Джон HTML и CSS. Разработка и дизайн веб-сайтов . - М.: Эксмо, 2013. - 480 с.
5. Дакетт, Джон Основы веб-программирования с использованием HTML, XHTML и CSS / Джон Дакетт. - М.: Эксмо, 2013. - 768 с.
6. Дронов, В. HTML 5 и CSS 3. Разработка современных Web-сайтов / В. Дронов. - М.: БХВ-Петербург, 2014. - 138 с.
7. Дронов, Владимир HTML 5 и Web 2.0. Разработка современных Web-сайтов / Владимир Дронов. - М.: БХВ-Петербург, 2013. - 416 с.
8. Лазаро, Исси Коэн Полный справочник по HTML, CSS и JavaScript / Лазаро Никсон, Р. Создаем динамические веб-сайты с помощью PHP, MySQL, JavaScript, CSS и HTML5 / Р. Никсон. - Москва: Машиностроение, 2016. - 688 с.
9. Пауэрс, Дэвид Adobe Dreamweaver, CSS, Ajax и PHP / Дэвид Пауэрс. - М.: БХВ-Петербург, 2012. - 829 с.
10. Прохоренок, Н. А. HTML, JavaScript, PHP и MySQL. Джентльменский набор Web-мастера / Н.А. Прохоренок, В.А. Дронов. - Москва: СПб. [и др.] : Питер, 2015. - 768 с.
11. Пфаффенбергер HTML, XHTML и CSS. Библия пользователя / Пфаффенбергер и др. - М.: Вильямс; Издание 3-е, 2015. - 752 с.
12. Рич Боуэн, Даниэль Лопес. Apache. Настольная книга администратора. – Санкт-Петербург : Вид-во ДиаСофтЮП, 2002. – 384 с.

13. Титтел, Эд HTML, XHTML и CSS для чайников / Эд Титтел, Джефф Ноубл. - М.: Диалектика, 2013. - 400 с.

14. Фримен, Элизабет Изучаем HTML, XHTML и CSS / Элизабет Фримен, Эрик Фримен. - М.: Питер, 2016. - 720 с.

15. Шафер, Стивен HTML, XHTML и CSS. Библия пользователя / Стивен Шафер. - Москва: СИНТЕГ, 2013. - 656 с.

16. Энж Эрик, Стефан Спенсер. SEO. Искусство раскрутки сайтов. – Санкт-Петербург : Вид-во БХВ-Петербург, 2014. – 668 с.

17. Гетия И.Г., Леонтьева И.Н., Шумилин В.К. Методические указания по проведению занятия по дисциплине «Безопасность жизнедеятельности» на тему: «Определение интегральной бальной оценки тяжести труда на робочем месте» . – М.: МГАПИ, 2002 . – 22 с.

18. Гігієнічна класифікація праці за показниками шкідливості та небезпечності факторів виробничого середовища, важкості та напруженості трудового процесу //Охорона праці. – 2001. –№ 12. – С. 12-20.

19. Жидецький В.Ц. Основи охорони праці. Підручник. – Львів: УАД, 2006. – 336 с.

20. НПАОП 0.00-1.28-10. Правила охорони праці під час експлуатації електронно-обчислювальних машин.

21. Жидецький В.Ц. Охорона праці користувачів комп’ютерів [Текст]/В.Ц. Жидецький. – Львів: Афіша, 2002. –162 с.