

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

бакалавр

(ступінь вищої освіти)

на тему РОЗРОБКА CHROME-РОЗШИРЕННЯ ДЛЯ
РОБОТИ ЗІ СПИСКАМИ
DEVELOPMENT OF A CHROME EXTENSION
TO WORK WITH LISTS

Виконав(ла): студент(ка) 4 курсу, групи КНТ-211

Спеціальності 122 Комп'ютерні науки

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Комп'ютерні науки

ПАСЬКО С.В.

(ПРИЗВИЩЕ та ініціали)

Керівник ГЛАДКОВА О.М.

(ПРИЗВИЩЕ та ініціали)

Рецензент ДЬЯЧУК Т.С.

(ПРИЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
(повне найменування закладу вищої освіти)

Факультет КНТ
Кафедра програмних засобів
Ступінь вищої освіти бакалавр
Спеціальність 122 Комп'ютерні науки
(код і найменування)
Освітня програма (спеціалізація) Комп'ютерні науки
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
Сергій СУББОТІН
“ ” 2025 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

ПАСЬКА Станіслава Володимировича

(ПРІЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Розробка Chrome-розширення для роботи зі списками.
Development of a Chrome Extension to Work with Lists

керівник проєкту (роботи) к.т.н., ГЛАДКОВА Ольга Миколаївна,
(науковий ступінь, вчене звання, ПРІЗВИЩЕ, ім'я, по батькові)

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

2. Строк подання студентом проєкту (роботи) 03 червня 2025 року

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

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Вибір та обґрунтування інструментарію розробки. 3. Розробка Chrome-розширення. 4. Інструкції з експлуатації, встановлення та тестування.

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

Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРИЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	ГЛАДКОВА О.М., доцент		
Нормоконтроль	БЄЛОВА А.В., асистент		

7. Дата видачі завдання « 14 » квітня 2025 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	1 тиждень	Розділ 1
3	Вибір мови програмування та інших технологій розробки.	2 тиждень	Розділ 2
4	Розробка архітектури програми.	2 тиждень	Розділ 3
5	Розробка програми.	3-4 тижні	Розділ 3, 4
6	Тестування та експериментальне дослідження програмного забезпечення.	5 тиждень	Розділ 4
7	Оформлення пояснювальної записки та документів до неї.	6 тиждень	Додатки
8	Нормоконтроль та рецензування.	7 тиждень	
9	Захист роботи.	8 тиждень	

Студент(ка)

(підпис) Станіслав ПАСЬКО
(Ім'я ПРИЗВИЩЕ)

Керівник проєкту (роботи)

(підпис) Ольга ГЛАДКОВА
(Ім'я ПРИЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра:
90 с., 6 табл., 45 рис., 2 дод., 23 джерел.

CHROME-РОЗШИРЕННЯ, СПИСОК СПРАВ, ВЕБ-ДОДАТОК,
JAVASCRIPT.

Об'єкт дослідження – процес розробки Chrome-розширення для роботи зі списками.

Предмет дослідження – програмне забезпечення для керування списками справ у вигляді Chrome-розширення.

Мета роботи – розробка Chrome-розширення для ефективного управління та підвищення особистої продуктивності за допомогою списку справ.

Матеріали, методи та технічні засоби: мова програмування JavaScript, мова гіпертекстової розмітки HTML та мова каскадних таблиць стилів CSS, технології API Chrome Extensions, середовище розробки Visual Studio Code, персональний комп'ютер з процесором AMD Ryzen 5 3600 під управлінням операційної системи Windows 11 Version 24H2.

Результати. Розроблено Chrome-розширення списку справ для підвищення особистої продуктивності та ефективного використання часу.

Висновки. Результатом розробки Chrome-розширення є можливість забезпечити зручне управління особистими завданнями та відстеження результату у середовищі браузера.

Галузь використання – підвищення продуктивності праці, організація особистих чи робочих завдань користувачів у середовищі веб-браузера Google Chrome.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 90 pages, 6 tables, 45 figures, 2 appendixes, 23 sources.

CHROME EXTENSION, TO-DO LIST, WEB APPLICATION, JAVASCRIPT.

The object of research is the process of developing a Chrome extension for working with lists.

The subject of the research is software for managing to-do lists in the form of a Chrome extension.

Purpose - to develop a Chrome extension for effective management and personal productivity with a to-do list.

Materials, methods and tools: JavaScript programming language, HTML hypertext markup language and CSS cascading style sheets language, Chrome Extensions API technologies, Visual Studio Code development environment, personal computer with AMD Ryzen 5 3600 processor running Windows 11 Version 24H2.

Results. A Chrome extension for the to-do list has been developed to increase personal productivity and efficient use of time.

Conclusions. The result of the development of the Chrome extension is the ability to provide convenient management of personal tasks and tracking of the result in the browser environment.

The field of application is to increase productivity, organize personal or work tasks of users in the Google Chrome web browser.

ЗМІСТ

	С.
Перелік скорочень та умовних познач.....	8
Вступ.....	9
1 Аналіз предметної області.....	11
1.1 Актуальність теми.....	11
1.1.1 Огляд основних методик підвищення продуктивності.....	11
1.2 Огляд існуючих рішень	18
1.3 Постановка задачі розробки.....	25
1.4 Висновки за розділом	25
2 Вибір та обґрунтування інструментарію розробки	26
2.1 Вибір платформи розробки	26
2.2 Вибір мов та технологій	29
2.2.1 Теоритичні відомості про HTML, CSS та JavaScript.....	31
2.3 Інструменти розробки та налагодження.....	32
2.3.1 Використання Chrome DevTools.....	32
2.3.2 Вибір редактора коду та середовища розробки	33
2.4 Висновки за розділом	35
3 Розробка Chrome-розширення	36
3.1 Архітектура розширення	36
3.1.1 Зберігання даних	36
3.2 Основний функціонал.....	38
3.3 Інтерфейс користувача та дизайн	42
3.4 Висновки за розділом	45
4 Інструкції з експлуатації, встановлення та тестування.....	46
4.1 Інструкція з встановлення та використання для користувачів	46
4.2 Тестування функціоналу та аналіз результатів.....	48
4.3 Порівняння з аналогами	53
4.4 Аналіз реалізованого функціоналу.....	54

	7
4.5 Пропозиції для подальшого розвитку	55
4.6 Висновки за розділом	57
Висновки	58
Перелік джерел посилання	59
Додаток А Текст програми.....	61
Додаток Б Слайди презентації	83

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

API	– Application Programming Interface;
CSS	– Cascading Style Sheets;
DOM	– Document Object Model;
GTD	– Getting Things Done;
HTML	– HyperText Markup Language;
IDE	– Integrated Development Environment;
JS	– JavaScript;
JSON	– JavaScript Object Notation;
WIP	– Work In Progress;
OC	– Операційна система.

ВСТУП

Сучасний ритм життя багатьох людей вимагає ефективного управління часом та організації повсякденних завдань. В умовах інформаційного суспільства кількість завдань, які потрібно виконувати щодня, постійно зростає, що створює потребу в якісних інструментах для їх систематизації та контролю.

Веб-браузери стали невід'ємною частиною робочого процесу більшості користувачів, і саме тому розширення браузерів набувають все більшої популярності як зручний спосіб інтеграції додаткових функцій безпосередньо в робоче середовище користувача.

Традиційні додатки для управління завданнями часто вимагають переключення між різними програмами, що може відволікати від основної роботи. Chrome-розширення для роботи зі списками дозволяє користувачам швидко додавати, редагувати та відслідковувати свої завдання безпосередньо в браузері, не відриваючись від поточної діяльності. Ефективне управління списками може суттєво підвищити продуктивність користувачів, особливо тих, хто активно працює з веб-ресурсами.

Саме тому об'єктом дослідження стала розробка Chrome-розширення для ефективної роботи зі списками. Розробка такого розширення дозволить дослідити повний цикл створення браузерних додатків та надасть користувачам зручний інструмент для організації їх повсякденних справ.

Метою даної дипломної роботи є розробка функціонального Chrome-розширення для створення, управління та організації списків завдань.

Для досягнення поставленої мети у дипломній кваліфікаційній роботі були вирішені такі завдання:

- провести аналіз існуючих рішень та методів для управління списками завдань;
- розробити архітектуру та функціональні можливості Chrome-розширення для роботи зі списками;
- обрати відповідні технології та інструменти розробки;

- реалізувати розроблене розширення з усіма необхідними функціями;
- провести тестування та налагодження створеного продукту.

Результатом цієї роботи буде створення функціонального Chrome-розширення, яке забезпечить користувачам зручні інструменти для створення, редагування, організації та відслідковування списків завдань безпосередньо в браузері.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Актуальність теми

У сучасному світі продуктивність є одним із ключових факторів особистого та професійного успіху. В умовах багатозадачності, високої динаміки робочих процесів та великої кількості цифрових каналів взаємодії, зростає потреба у зручних інструментах для організації повсякденних справ. Одним з таких інструментів є списки задач або як їх ще називають “To-Do List”, які дозволяють ефективно керувати задачами, контролювати дедлайни, а також розвантажувати пам'ять людини.

Бути продуктивним означає вміти ефективно та вчасно виконувати якомога більше важливих справ. Це неможливо без системного підходу. Список справ слід складати послідовно, з урахуванням пріоритетності та обсягу завдань. Правильно складений, він служить жорсткими рамками для вашої продуктивної діяльності протягом дня, тижня або місяця, не дозволяючи розсіюватися на другорядні справи і прокрастинувати. Для цього існують певні методи та інструменти, які допоможуть чітко структурувати всі завдання і послідовно розташувати їх у списку.

Списки справ (або списки завдань, англ. To-do list) – це інструмент для організації, планування та контролю виконання різноманітних завдань, метою якого є спрощення та підвищення ефективності управління завданнями [1]. Список може бути як паперовий, так і цифровий, він допомагає людині структурувати свої обов'язки та відстежувати прогрес їх виконання. Прикладами списків можуть бути списки справ, завдань, покупок, бажань тощо.

1.1.1 Огляд основних методик підвищення продуктивності

Існує багато різних методів і інструментів для підвищення продуктивності. Майже кожен список справ “To-do List” створюється з використанням хоча б одного з цих методів.

GTD (англ. Getting Things Done) – це система управління часом що пропонує вирішення проблем за допомогою структурованої системи, яка допоможе вам тримати все під контролем, визначати пріоритети та ефективно розподіляти свій час для підвищення продуктивності та балансу. Ця система розроблена Девідом Алленом і опублікована в однойменній книзі “Getting Things Done: The Art of Stress-Free Productivity” [2].

Чим більше інформації крутиться у вашій голові, тим важче вирішити, що робити в першу чергу. Аллен зауважив, що наш мозок набагато краще обробляє інформацію, ніж зберігає її “ваша голова – жахливий офіс” [3]. Метод GTD створює робочий процес, де ви можете скинути весь цей розумовий безлад у зовнішню систему і організувати його так, щоб у будь-який момент ви могли впевнено відповісти на питання “Над чим я повинен працювати?”, не хвилюючись, що можете забути щось важливе, що вам потрібно зробити пізніше [3].

Обробка вхідної інформації передбачає такі кроки.

Крок 1. Фіксуйте завдання та ідеї – зафіксуйте все, що привернуло вашу увагу, будь-які завдання, ідеї або обов'язки.

Крок 2. Уточнюємо подальші дії – перетворіть усе що було додано до вашого списку і вирішіть чи це проєкт, наступна дія або просто думки.

Крок 3. Організація – після ретельного аналізу у вас, буде довгий список завдань та ідей, які тепер необхідно організувати за певними категоріями (календар, довідки, делегувати, кошик і так далі).

Крок 4. Перегляд – постійно переглядайте, оновлюйте та перевіряйте на можливість вдосконалення.

Крок 5. Виконання – цей крок є останнім і він відповідає за важливість та послідовність виконання завдань, зрозуміти що можна виконати сьогодні, а що краще відкласти і виконати потім [3]. Вигляд алгоритму GTD наведено на рисунку 1.1.

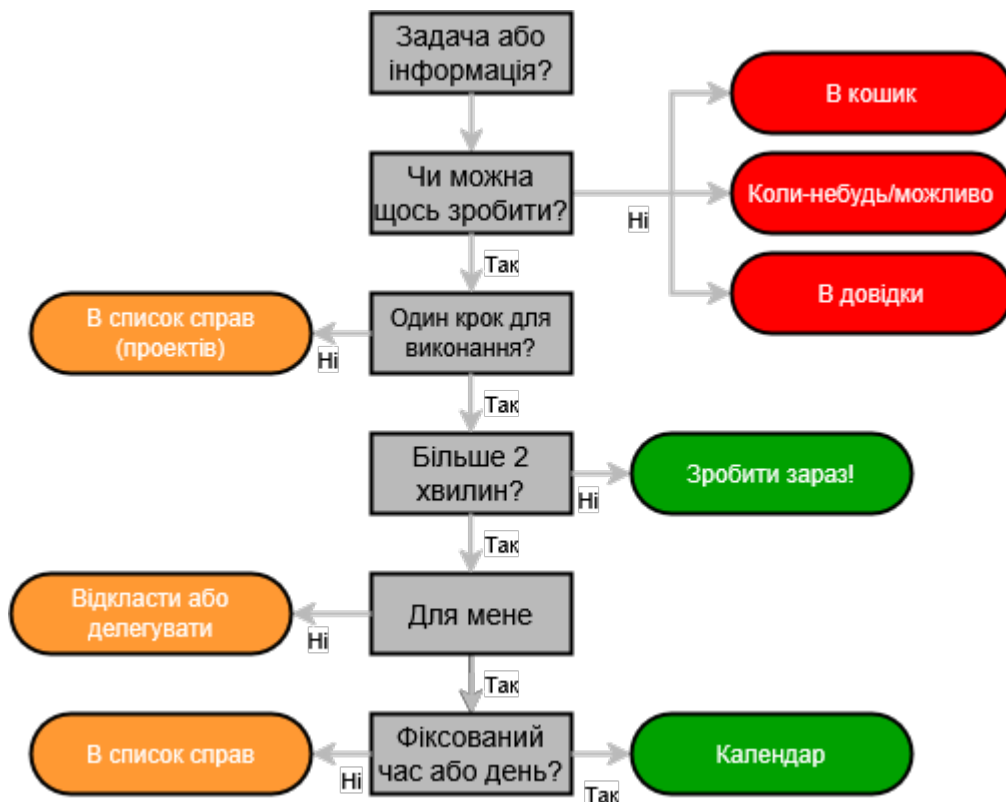


Рисунок 1.1 – Алгоритм обробки вхідних повідомлень згідно з методикою GTD

Метод GTD є ефективним інструментом для довгострокового планування, особливо у випадках, коли необхідно структурувати велику кількість інформації. Допомагає розвантажити мозок, за допомогою перенесення всіх задач в організований список. Складні справи розбивати на дрібні кроки, та робити їх більш керованими. Регулярний перегляд та аналіз завдань дозволить тримати фокус на важливих та уникнути накопичення незавершених справ.

Pomodoro – це метод керування часом, який розбиває роботу на певні інтервали зазвичай 25 хвилин, які чергуються з короткими перервами в 5 хвилин. Після чотирьох таких циклів рекомендується довша перерва, яка складає 15–30 хвилин [4]. У разі відволікання або збою зробити п'ятихвилинну перерву і почати знову. Легше присвятити себе 25 хвилинам роботи за раз, ніж цілий день безперервної роботи.

Ключові етапи виконання методу Pomodoro:

- обрати завдання, яке потрібно виконати;
- встановити таймер на 25 хвилин (один "помідор");
- працювати без відволікань до сигналу таймера;
- зробити коротку перерву приблизно 5 хвилин;
- повторити другий та четвертий пункт ще тричі;
- після чотирьох циклів довга перерва 15-30 хвилин [4].

На рисунку 1.2 буде наведена техніка управління часом Pomodoro.

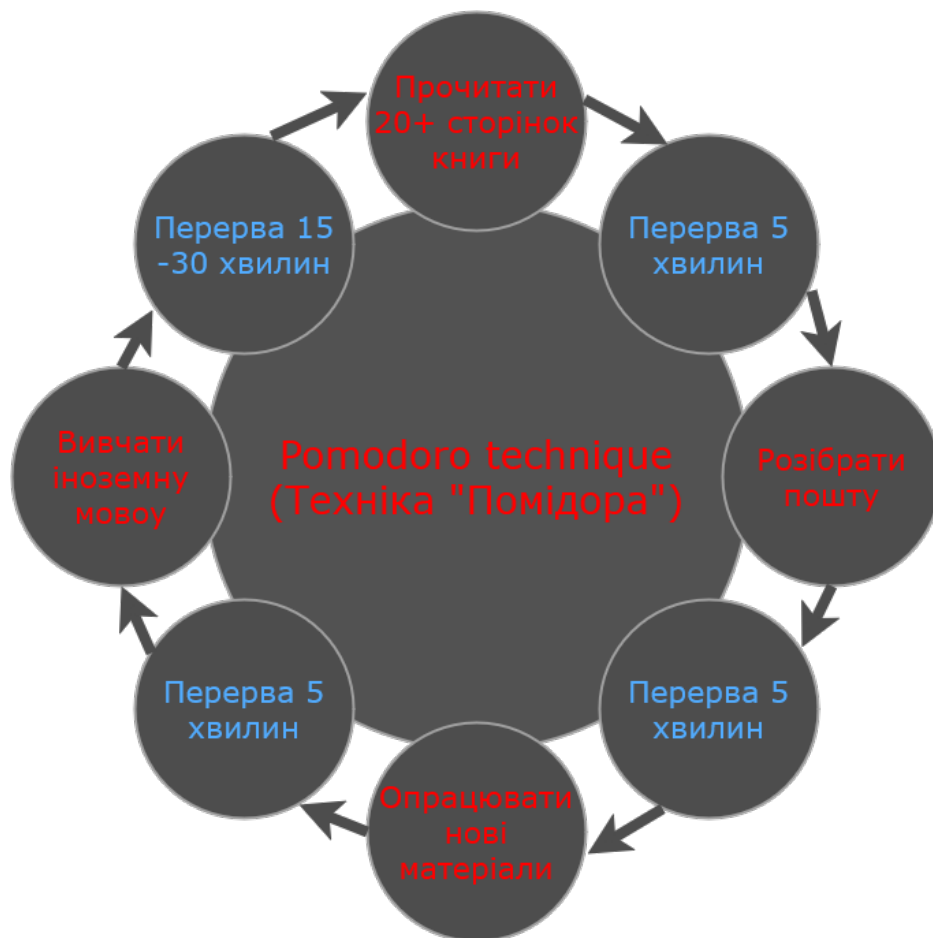


Рисунок 1.2 – Техніка керування часом Pomodoro

Техніка Pomodoro добре підходить людям, які схильні до частих відволікань. Робота короткими інтервалами з подальшою перервою допомагає зберегти концентрацію та запобігає вигоранню. Дозволяє ефективно чергувати продуктивність і відпочинок, що особливо корисно при виконанні монотонних та об'ємних завдань.

Kanban – це метод управління проєктами, який допомагає візуалізувати певний об’єм роботи та скорочувати число незавершених завдань [5]. Особливо цей метод підходить для роботи з невеликою командою.

Kanban дошка – це інструмент для управління проєктами. Ця система планування полягає у візуалізації робочого процесу, тобто використанні спеціальних дошок та карток, на яких відображаються всі завдання, над якими ведеться робота. Вона дозволяє відстежувати прогрес виконання завдань шляхом поділу їх на етапи. Кожна задача переміщується між колонками на дошці, що відображає її поточний статус. Наприклад, перша колонка – До виконання, друга – В процесі виконання, третя – Виконано [5]-[6]. Приклад наведено на рисунку 1.3.

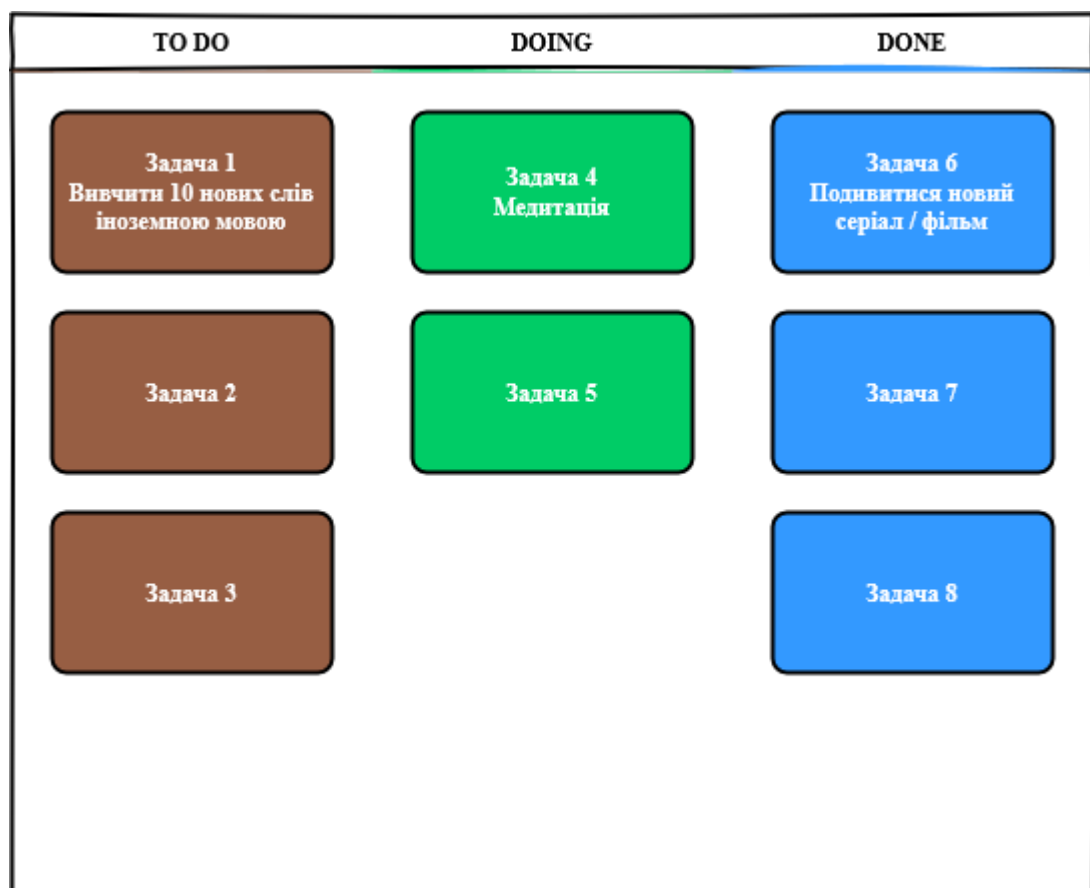


Рисунок 1.3 – Інструмент управління проєктами Kanban дошки

Такий підхід забезпечує прозорість процесу, як для однієї людини, так і для команди, оскільки одразу видно, над чим потрібно працювати, які завдання в роботі, а які вже завершені.

Важливі елементи процесу Канбан:

- візуалізація – це візуалізація особистого або командного робочого процесу на дошці Канбан. Дошка повинна відображати кожен етап, від початку завдання до його завершення;
- незавершена робота (WIP) – обмеження кількості завдань, які виконуються одночасно, дозволяє зосередитись на поточних задачах та уникати перевантаження;
- покращення навичок – робота над помилками, регулярний перегляд та вдосконалення завдань;
- система “витягування” – нові завдання не призначаються активно. Вони “витягуються” членами команди, коли вони мають можливість і готові взятися за нову задачу [5]-[6].

Метод Канбан дошки підійде як для роботи в команді, так і особистому плануванні, допоможе гнучко налаштувати свою дошку для виконання поставлених завдань. Гарно себе продемонструє з довготривалим плануванням списку завдань.

Матриця Ейзенхауера (матриця термінових та важливих завдань) – один з найбільш відомих інструментів для управління часом. Суть методу полягає в простому визначенні пріоритетів завдань та управління робочим навантаженням. Названа на честь Дуайта Д. Ейзенхауера, 34-го президента Сполучених Штатів Америки [7].

За деякими джерелами, автором цього вислову є Дуайт Ейзенхауер: “У мене є два типи проблем: термінові та важливі. Термінові не завжди важливі, а важливі не завжди термінові” [7].

Матриця поділяє всі завдання на чотири квадранти:

- термінові та важливі (Виконати) – це пріоритетні завдання які потрібно вирішити якомога швидше виконати, часто пов'язані з дедлайнами або життєво важливими питаннями;
- не термінові, але важливі (Запланувати) – довгострокові завдання або цілі які є важливими, але не мають чіткого терміну і тому можуть бути заплановані і виконані пізніше;
- термінові, але не важливі (Делегувати) – завдання які потрібно виконати, але не є настільки важливими що потребують конкретно вашої уваги і можуть бути виконані кимось іншим;
- не термінові та не важливі (Видалити) – це завдання та дії, що забирають багато часу та від яких слід безжально відмовитися, заважають досягненню цілі та займають багато часу [7].

Приклад наведено на рисунку 1.4.

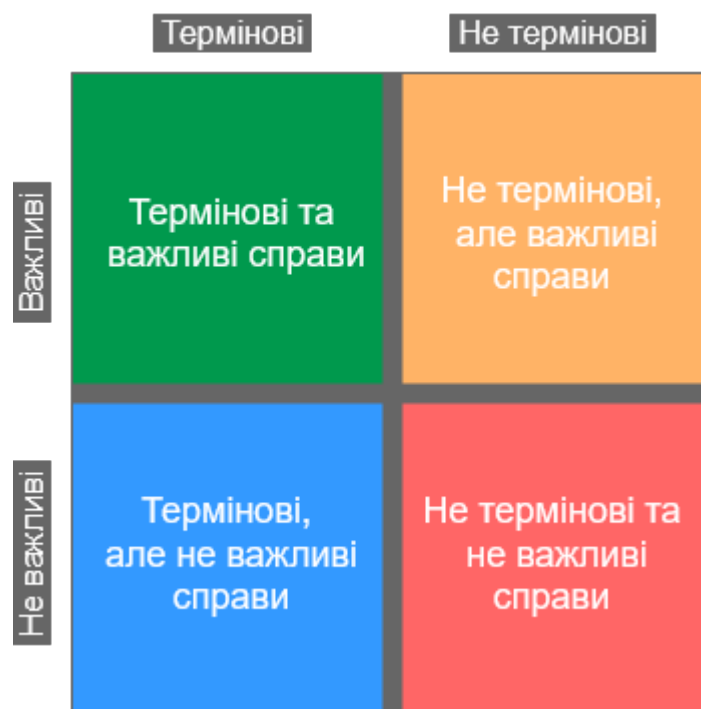


Рисунок 1.4 – Інструмент управління часом Матриця Ейзенхауера

Цей метод допоможе зрозуміти чи взагалі потрібні вам задачі які ви написали та допоможе відсіяти непотрібні, допоможе розставити пріоритети

завдань. Хороший варіант для короткострокового планування, ідеально підходить для тих, хто постійно відчуває, що “все термінове” або “нічого не встигає”. Але для довгострокового планування краще комбінувати з іншими інструментами.

1.2 Огляд існуючих рішень

У попередньому розділі були розглянуті актуальні методи, що лежать в основі розробки більшості сучасних застосунків для ведення списків справ і управління завданнями. У цьому розділі будуть розглянуті популярні варіанти, такі як Todoist, Google Tasks та інші.

Todoist – це популярний веб-сервіс для керування завданнями та управління проектами, який допомагає організовувати особисті та командні справи. Розвантажте свій розум, вибудуйте звички та підвищте свою продуктивність. Цей інструмент відповідає всім вимогам базового планування та управління завданнями. Можна створювати завдання і визначати пріоритети, додавати терміни виконання і встановлювати нагадування, щоб нічого не залишилося поза нашою увагою [8] (рис. 1.5).

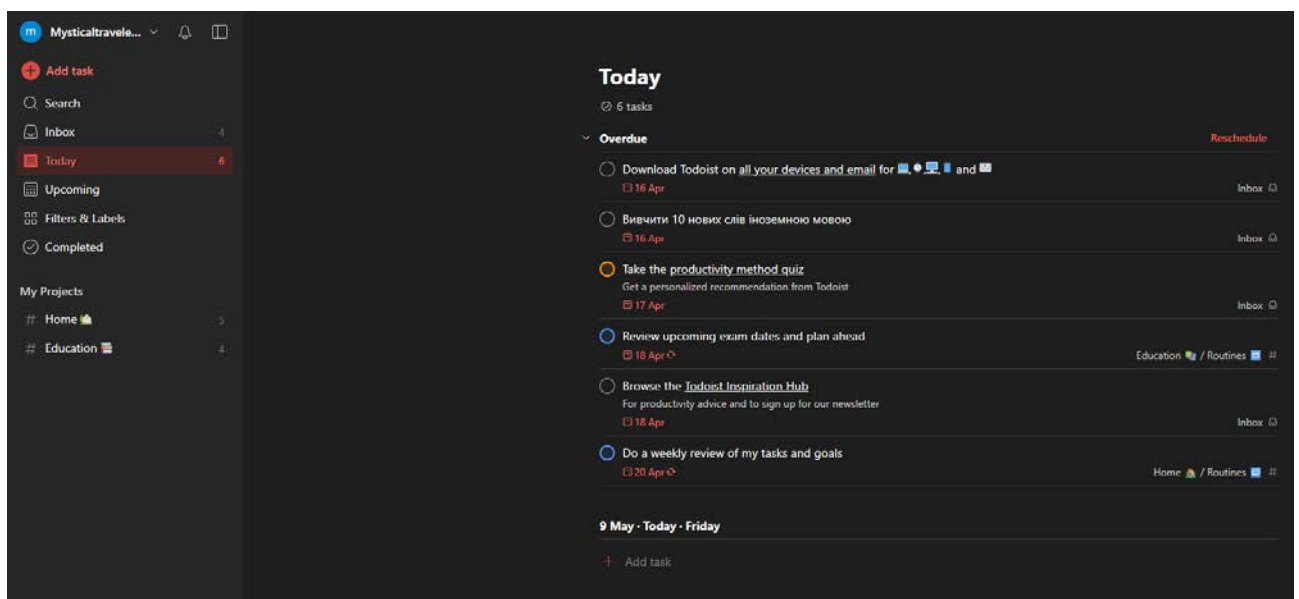


Рисунок 1.5 – Онлайн платформа для управління завданнями Todoist

Переваги:

- ефективне управління щоденними завданнями;
- зручне відстеження результатів;
- базова підтримка командної співпраці;
- кросплатформна функціональність;
- інтеграція з різними програмами для підвищення продуктивності та ефективного відстеження часу;
- система Karma для підвищення мотивації.

Недоліки:

- обмежена функціональність у безкоштовній версії (до 5 активних проєктів, відсутність міток та фільтрів);
- доступ до більшості функцій та методів лише за підпискою.

Google Tasks – це безкоштовний інструмент керування завданнями від Google, який дозволяє користувачам ефективно організовувати та відстежувати свої списки справ. Дозволяє створювати списки, підзадачі, встановлювати дедлайни, нагадування, працювати з кількох пристроїв та додавати файли [9] (рис. 1.6).

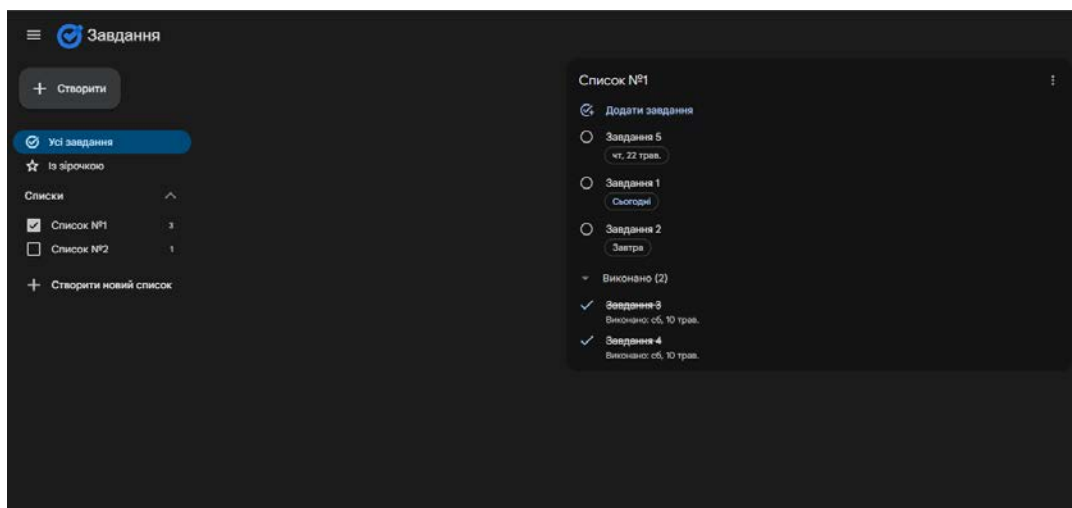


Рисунок 1.6 – Інструмент для керування завданнями Google Tasks

Переваги:

- повністю безкоштовний;

- простий та інтуїтивно зрозумілий інтерфейс;
- синхронізація між усіма пристроями без додаткових налаштувань;
- мінімалістичний дизайн без зайвих елементів;
- глибока інтеграція в екосистему Google (Gmail, Google Calendar, Google Assistant);
- доступність на різних платформах.

Недоліки:

- обмежений функціонал в порівнянні з іншими рішеннями;
- відсутність системи міток та категорій;
- немає статистики та аналітики продуктивності;
- обмежені можливості для командної роботи (спільний доступ лише через сторонні сервіси).

Trello – це гнучкий інструмент для управління завданнями та проєктами, який дозволяє переміщувати завдання, такі як наліпки на дошці Kanban з функцією перетягування для легшого відстеження статусу. Користувачі можуть використовувати його як для створення персоналізованих списків завдань, так і створення командних списків де члени команди можуть покладатися на розширені контрольні списки, щоб додавати контекст до завдань, додаючи терміни виконання та виконавців [10] (рис. 1.7).

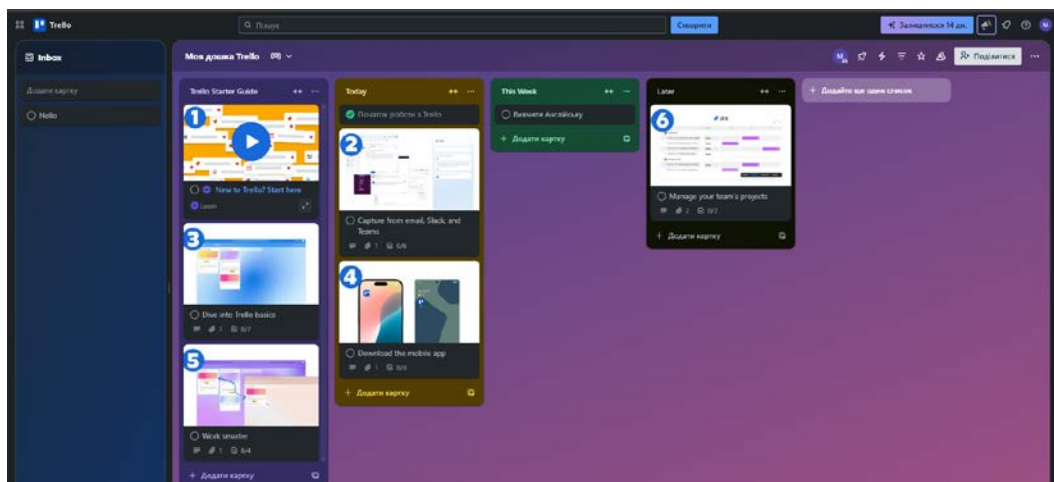


Рисунок 1.7 – Онлайн платформа для управління завданнями Trello

Переваги:

- відстеження завдань на основі Kanban дошки;
- гнучка система організації проєктів;
- виділені різним кольором комірки допомагають розставляти пріоритети завдань;
- розширена функціональність через Power-Ups (інтеграції та додаткові можливості) ;
- підходить для командної роботи та спільного доступу;
- шаблони для швидкого створення проєктів.

Недоліки:

- чим більше кількість користувачів, тим більша вартість;
- орієнтований більше на проєкти, ніж на персональні завдання;
- може бути складним для простих завдань та списків;
- не підтримує складні управління проєктами;
- обмежені можливості роботи в автономному режимі.

Ticktick – це крос-платформенний додаток, простий і зручний інструмент для управління завданнями та часом, який допомагає зберігати всі поточні справи, а також синхронізувати їх за допомогою хмарного сховища. Його можна використовувати для організації робочих і особистих завдань, а також для досягнення цілей у коротші терміни [11]. Представлено у вигляді розширення на рисунку 1.8.

Основна особливість Ticktick це те що в ньому безкоштовно додані декілька різних методів для підвищення особистої продуктивності:

- списки справ та календар;
- трекер звичок;
- матриця Ейзенхауера;
- метод Pomodoro.

Переваги:

- розширені інструменти для планування;
- функції для особистої продуктивності;

- можливість автономної роботи;
- можливість створення завдань голосом;
- статистика та аналітика продуктивності.

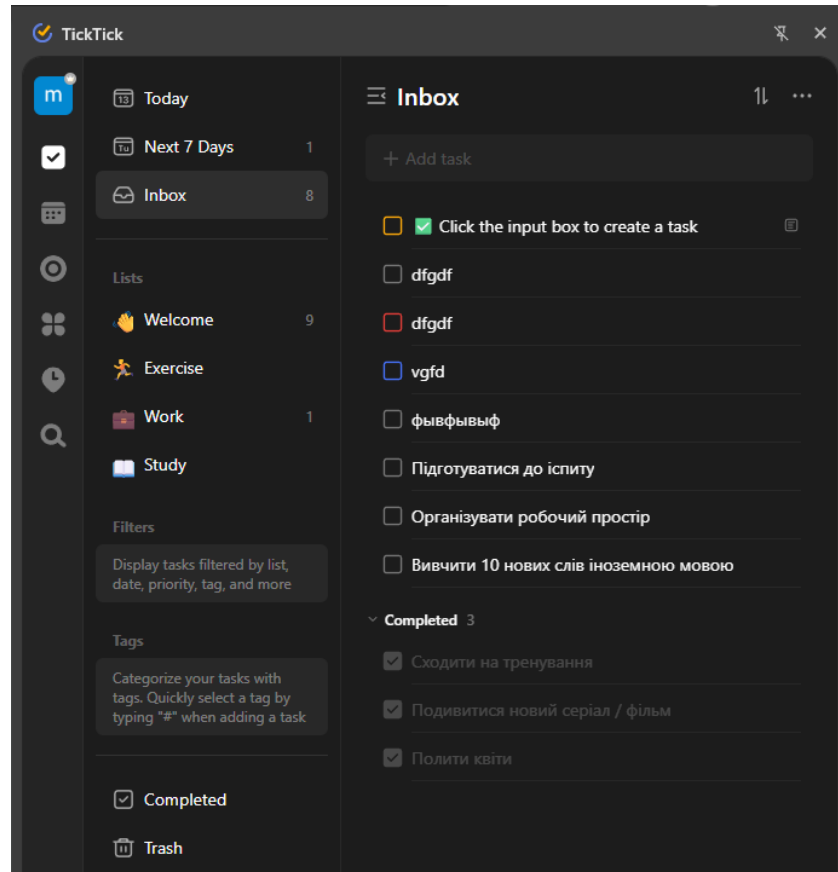


Рисунок 1.8 – Інструмент для управління завданнями TickTick

Недоліки:

- обмежена безкоштовна версія;
- обмежена інтеграція з зовнішніми сервісами;
- деякі функції доступні лише в платній версії;
- інтерфейс трохи перевантажений для новачків;
- обмежені можливості командної роботи порівняно з іншими варіантами.

Порівняльний аналіз різних онлайн платформ для ведення списків справ наведено у таблиці 1.1.

Таблиця 1.1 – Порівняння платформ для ведення списків справ

Категорії	Google Tasks	Todoist	Trello	TickTick
Безкоштовна версія	+	+—	+—	+—
Кросплатформність	+	+	+	+
Працює в автономному режимі	+—	+	+—	+
Інтеграції з іншими сервісами	+	+	+	+—
Підтримка командної роботи	—	+	+	+—
Проста ієрархія завдань	+	+	+	+

Після проведеного аналізу невеликої частини існуючих рішень видно що кожен варіант використовує різні методи та техніки, щоб надати користувачу можливість для відстеження свого прогресу. Проте в більшості варіантів пропонують обмежено функціональну безкоштовну версію списків справ, або безкоштовний пробний період, а якщо потрібен повний доступ до функціоналу, то потрібно придбати підписку.

Також було проаналізовано, який відсоток користувачів використовує браузер Chrome як основний. На персональних комп'ютерах його середня частка у світі за період з травня 2024 по травень 2025 року склала приблизно 65% серед усіх інших браузерів [12]. В Україні за той самий період цей показник становив близько 57% [13]. На рисунках 1.9 – 1.10 більш детально зображено статистику за період з травня 2024 по травень 2025 року.

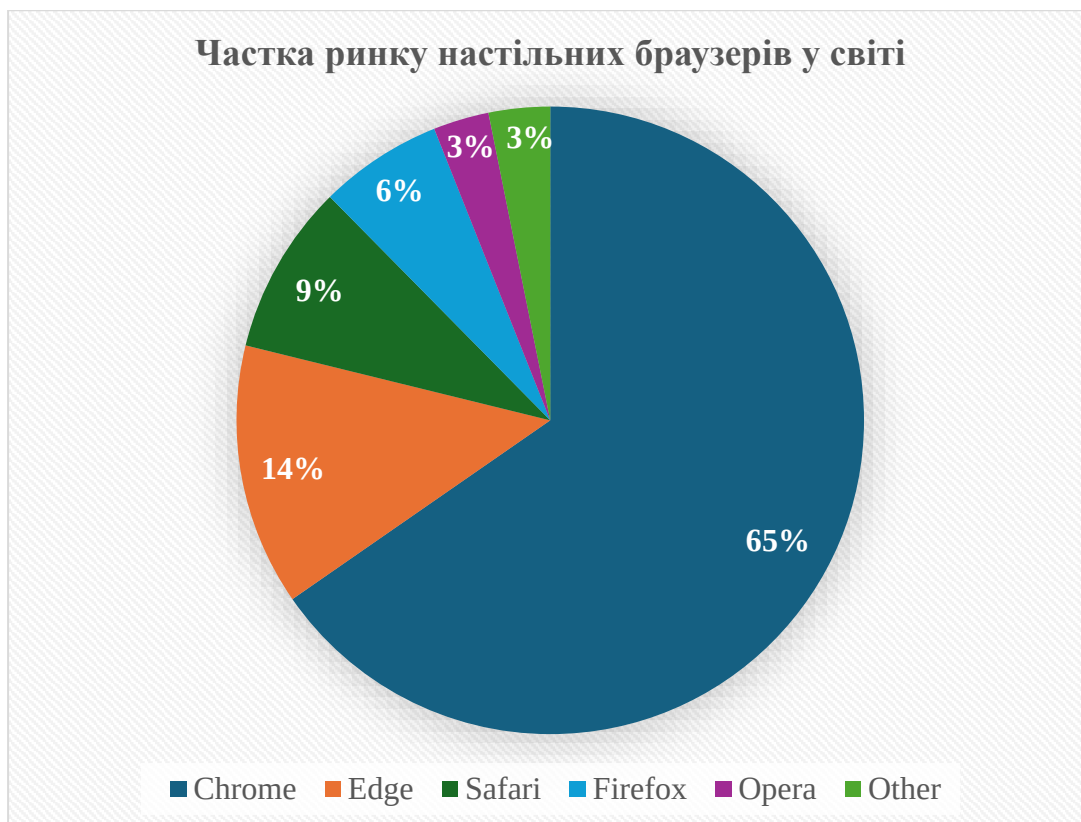


Рисунок 1.9 – Діаграма частки ринку настільних браузерів у світі

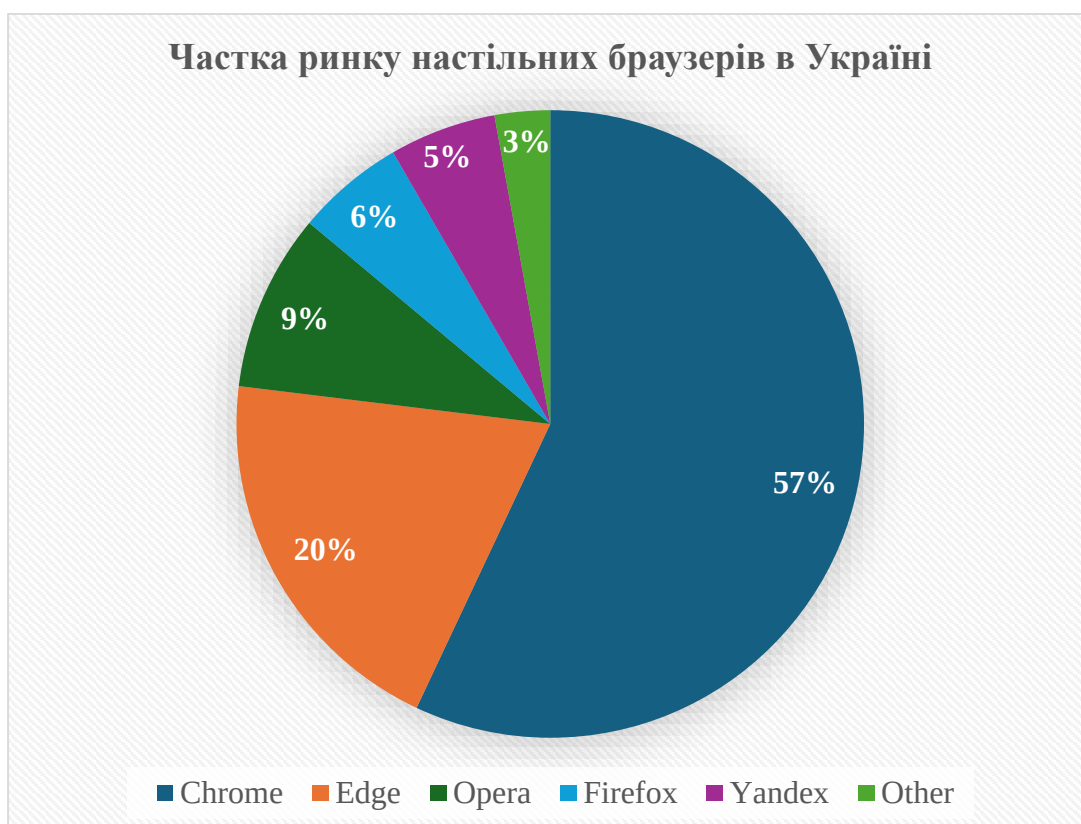


Рисунок 1.10 – Діаграма частки ринку настільних браузерів в Україні

Таким чином на основі цієї статистики було обрано саме браузер Google Chrome як основна платформа де буде реалізовано розширення, а також воно повинно працювати і в інших браузерах на основі Chromium (Opera, Microsoft Edge).

1.3 Постановка задачі розробки

На основі аналізу існуючих рішень для роботи зі списками справ, було вирішено створити мінімалістичну програму, яка містила б основний та необхідний для користувача функціонал, а саме:

- можливість додавання завдань;
- вибір для фільтрації на який саме день тижня;
- редагування та видалення завдань;
- можливість перегляду та помічати справи як виконані;
- відстеження загальної кількості завдань.

1.4 Висновки за розділом

У даному розділі було проведено аналіз популярних методів управління завданнями, які забезпечують різні підходи до планування, структурування та виконання справ, орієнтуючись на різні потреби користувачів. Їх аналіз дозволяє краще зрозуміти функціональні вимоги до інструментів типу. Розглянуто популярні вже наявні рішення на різних платформах виявлені їх переваги та недоліки і на основі всіх рішень було обрано що саме повинно бути в додатку.

2 ВИБІР ТА ОБҐРУНТУВАННЯ ІНСТРУМЕНТАРІЮ РОЗРОБКИ

2.1 Вибір платформи розробки

Вибір оптимальної платформи для розробки програмного забезпечення є критично важливим етапом, оскільки він безпосередньо впливає на функціональність, продуктивність, масштабованість та зручність використання застосунку. У даному розділі проведено аналіз основних типів платформ розробки та обґрунтовано вибір оптимального рішення для реалізації поставлених завдань.

У ході аналізу було розглянуто чотири основні варіанти:

- десктопні застосунки;
- мобільні застосунки;
- веб-застосунки;
- розширення для браузерів.

Кожен із цих варіантів має свої переваги та недоліки, які враховувались при прийнятті остаточного рішення.

Десктоп застосунки – це програми, які встановлюються на локальний комп'ютер або ноутбук та безпосередньо працюють на ОС користувача (Windows, macOS, Linux), що забезпечує доступ до файлової системи, апаратних ресурсів, системних бібліотек тощо. Їх можна використовувати без підключення до Інтернету. Для виробництва десктопних застосунків використовують різні мови програмування, такі як Java, C++/Qt, C#, Electron.js, Python, та інші [14].

Переваги:

- повний доступ до ресурсів операційної системи;
- можливість роботи без підключення до Інтернету;
- підвищена безпека та приватність даних;
- висока продуктивність та можливість оптимізації під певне залізо.

Недоліки:

- висока вартість підтримки та складність масштабування;

- складність оновлення та розгортання;
- обмежена кросплатформність (потреба розробки окремих версій для різних ОС).

Для простого To-Do застосунку десктопна платформа є надмірною. Основна логіка не потребує системного доступу чи складної інтеграції з ОС. Крім того, розробка та підтримка десктопних додатків для кількох ОС вимагає значних ресурсів.

Мобільні застосунки – це програмні забезпечення, розроблені для використання на мобільних пристроях, таких як смартфони та планшети з операційними системами iOS або Android. Зазвичай встановлюються через магазини додатків, які відповідають ОС пристрою. Такі застосунки забезпечують зручний доступ до функціоналу у будь-якому місці, можуть використовувати апаратні можливості пристрою камеру або сенсор, а також підтримують push-сповіщення. Для розробки мобільних додатків застосовують нативні (Swift, Kotlin, Java) та кросплатформні (Flutter, React Native) технології [15].

Переваги:

- висока мобільність та доступність;
- інтеграція з апаратними функціями пристрою;
- можливість роботи офлайн (залежно від реалізації);
- зручний інтерфейс.

Недоліки:

- необхідність розробки окремих версій для iOS та Android;
- обмежений екранний простір;
- складніша система публікації (Play Market, App Store);
- складність інтеграції з браузерним середовищем та сторонніми вебсервісами.

Мобільна платформа доцільна для задач, де критичною є мобільність та доступ до апаратних функцій. Для організації списків у браузері вона менш актуальна.

Веб-застосунки або веб-додатоки – це програмні забезпечення, що працюють у браузері та доступні з будь-якого пристрою з інтернетом. Вони не потребують встановлення, забезпечують кросплатформеність, централізоване оновлення та можливість колективної роботи. Для розробки використовують HTML, CSS, JavaScript, а також серверні мови й фреймворки Python/Django, Ruby on Rails, Angular, React, Vue.js, Node.js та інші [16].

Переваги:

- доступність з будь-якого пристрою та ОС;
- відсутність необхідності встановлення;
- автоматичні оновлення;
- можливість інтеграції з іншими веб-сервісами.

Недоліки:

- залежність від стабільного Інтернет-з'єднання (у більшості випадків) ;
- обмежений доступ до апаратних ресурсів пристрою;
- питання безпеки та приватності даних;
- можливі обмеження продуктивності для складних задач.

Веб-застосунок є універсальним рішенням для організації списків, але його ключовим недоліком є обмежена інтеграція з інтерфейсом браузера. На відміну від розширень або десктопних застосунків.

Розширення для браузерів – це невеликі програмні засоби, які інтегруються у браузер (Google Chrome, Firefox тощо) та розширюють його стандартний функціонал. Вони дозволяють взаємодіяти з вебсторінками, додавати нові можливості, автоматизувати дії користувача, виконувати фонові задачі та надавати користувачеві швидкий доступ до потрібних інструментів без відволікання від основної роботи [17].

Переваги:

- безпосередня інтеграція у браузер і зручний доступ до функціоналу у будь-який момент;
- простота встановлення та оновлення через магазини розширень;

- відсутність необхідності у власному сервері чи хостингу;
- мінімальні вимоги до ресурсів пристрою;
- висока безпека та ізольованість середовища виконання.

Недоліки:

- обмежений доступ до апаратних ресурсів пристрою;
- залежність від політик та обмежень браузера;
- потреба у дотриманні стандартів безпеки та конфіденційності даних;
- можливі обмеження продуктивності для складних задач.

Розширення для браузера є оптимальним вибором для реалізації інструменту організації списків, оскільки дозволяє вбудувати функціонал безпосередньо у робочий процес користувача, забезпечує швидкий доступ та простоту використання без необхідності перемикання між додатками чи вкладками. Даний підхід ідеально підходить для задач, які не вимагають глибокої інтеграції з ОС або апаратними ресурсами, але потребують тісної взаємодії з веб сторінками та браузерним середовищем

2.2 Вибір мов та технологій

Від вибору мови буде залежати подальша розробка та вибір технологій які будуть використовуватись у розробці програми. В розділі буде розглянуто популярні мови програмування такі як Python, C++ та JavaScript.

Головна мета списку справ полягала у тому, щоб він був компактним, не заважав як додаткове вікно та працював у браузері. Саме це стало причиною виключення інших мов програмування (табл. 2.1).

Оскільки лише JavaScript дозволяє розробляти повноцінні фронтенд-розширення, які працюють в межах Chrome без додаткових інструментів чи серверів.

Таблиця 2.1 – Вибір мови програмування

Критерій	JavaScript	Python	C++
Підтримка у браузері	Вбудована	Потребує інтерпретатора	Не підтримується напряду
Придатність для розширень	Пряма підтримка (Chrome API)	Можлива лише як бекенд	Неможливо, тільки через обхідні шляхи (WebAssembly)
Рівень складності синтаксису	Низький (простий, гнучкий)	Низький (інтуїтивний)	Високий
Інтеграція з Chrome API	Повний доступ (для розширень)	Відсутня (лише через бекенд)	Відсутня
Типізація	Динамічна	Динамічна	Статична
Швидкість розробки та навчання	Максимальна, Велика кількість безкоштовних курсів	Висока, Простий вступ, багато підручників	Повільна, більш академічна база, складніше для новачків
Підтримка спільноти	Дуже активна; найбільша спільнота в галузі вебробки	Велика	Велика, але більше орієнтована на системне програмування
Переваги	Нативність, Простий синтаксис	Легко вивчити, Читабельний код	Висока продуктивність
Недоліки	Обмежена багатопоточність, Обмежена типізація	Не виконується в браузері	Складність, довга розробка

Для розробки розширення в Google Chrome було обрано JavaScript у поєднанні з HTML та CSS. Це рішення ґрунтується на кількох ключових факторах:

- JavaScript є основною мовою для створення розширень у Chrome, що забезпечує повну сумісність з API браузера;
- JavaScript має велику спільноту розробників, численні бібліотеки та фреймворки, що полегшує розробку та підтримку коду;
- JavaScript у поєднанні з HTML та CSS дозволяє створювати динамічні та привабливі інтерфейси для користувачів.

2.2.1 Теоритичні відомості про HTML, CSS та JavaScript

HTML (HyperText Markup Language) або мова розмітки гіпертексту – це стандартна мова розмітки, яка є основним будівельним матеріалом для структури веб-сторінок. Забезпечує основу для додавання заголовків, тексту, посилань, зображень, списків та іншого контенту на веб-сторінку [18].

CSS (Cascading Style Sheets) або каскадні таблиці стилів – це мова таблиць стилів, яка використовується для стилізації і оформлення веб-сторінок, роблячи їх більш привабливими і естетично приємними для користувачів. Визначає, як повинні виглядати елементи на веб-сторінці, включаючи такі атрибути, як кольори, шрифти, інтервали та макет. Відокремлюючи контент HTML від представлення CSS, розробники можуть створювати візуально привабливі та добре структуровані веб-сайти [19].

JavaScript – це високорівнева, інтерпретована мова програмування, яку використовують для створення інтерактивних та динамічних веб-додатків. JavaScript став незамінною мовою у світі веб-розробки. Дозволяє встановлювати пряму взаємодію між користувачами та веб-сторінками. Динамічно змінюючи зовнішній вигляд сторінок та реагуючи на дії користувача. Він є універсальним і зручним для початківців [20].

Основні функціональні особливості JavaScript перелічені нижче.

Інтерпретованість: дозволяє вносити зміни в JavaScript код і відразу бачити результат.

Велика кількість бібліотек і фреймворків JavaScript: готові рішення які дозволяють економити час.

Вбудована підтримка DOM: за допомогою JavaScript можна отримати доступ та маніпулювати структурою веб-сторінок.

Слабка типізація: змінні можуть змінювати свій тип у процесі виконання, що може стати причиною помилок.

Об'єктно-орієнтована мова програмування: ідеально підходить для створення масштабованих та адаптивних застосунків.

Кросплатформеність: виконується у всіх сучасних браузерах що робить його універсальним інструментом для розробки веб-додатків [20].

2.3 Інструменти розробки та налагодження

У процесі розробки програмного забезпечення від правильного вибору інструментів розробки буде залежати ефективність, швидкість та якість кінцевого результату. У даному розділі розглядаються основні інструменти, які можуть бути використані для створення, налагодження та тестування розширення для браузера Google Chrome.

2.3.1 Використання Chrome DevTools

Chrome DevTools – це потужний набір інструментів, вбудованих безпосередньо в браузер Google Chrome, які дозволяють розробникам налагоджувати та оптимізувати вебсторінки та розширення [21].

Основні можливості включають:

- перегляд і редагування HTML, CSS у реальному часі;
- налагодження JavaScript коду з використанням брейкпоінтів, перегляду змінних та викликів функцій;

- аналіз продуктивності (performance), мережевих запитів (network), безпеки (security);
- вкладка application для роботи з локальним сховищем, cookie, IndexedDB, а також перегляд і тестування background/service worker-скриптів у розширеннях.

2.3.2 Вибір редактора коду та середовища розробки

Від вибору інтегрованого середовища розробки (IDE) або редактору коду буде залежати ефективність та продуктивність розробника. Правильний вибір інструмента може значною мірою спростити процес написання, тестування та налагодження коду.

При розробці Chrome-розширення важливо обирати IDE або редактор коду, який підтримує JavaScript, HTML, CSS та має можливості для роботи з JSON файлами. Загальна характеристика представлена у таблиці 2.2.

Таблиця 2.2 – Загальна характеристика порівняння середовищ розробки

IDE	Основні особливості	Платформи	Вартість
Visual Studio Code	Легкий, швидкий та масштабований редактор коду з великою кількістю плагінів	Windows, macOS, Linux	Безкоштовно
WebStorm	Повнофункціональне потужне середовище	Windows, macOS, Linux	Платний (Безкоштовна пробна версія)
Atom	Легкий редактор коду з відкритим вихідним кодом та зручним інтерфейсом з великою кількістю плагінів	Windows, macOS, Linux	Безкоштовно
Sublime Text	Легкий та швидкий редактор коду з мінімалістичним інтерфейсом і великою кількістю плагінів	Windows, macOS, Linux	Платний (Безкоштовна пробна версія)

В таблиці 2.3 розглянуті переваги та недоліки обраних IDE та редакторів коду.

Таблиця 2.3 – Переваги та недоліки середовищ розробки

IDE	Переваги	Недоліки
Visual Studio Code	Безкоштовний; відкритий вихідний код; велика кількість плагінів; часті оновлення; наявність офіційних плагінів та найпопулярніших фреймворків;	Побудований на Electron (може бути повільним при великій кількості розширень); деякі функції потребують додаткових розширень.
WebStorm	Потужні інструменти для роботи фреймворками; швидкодія; вбудована підтримка інструментів ШІ; професійні інструменти налагодження; повноцінна IDE.	Платний; вимагає більше ресурсів, ніж звичайні редактори; деякі користувачі відзначають перевантажений інтерфейс та складність налаштування; довгий час запуску.
Atom	Повністю безкоштовний; відкритий вихідний код; висока кастомізація; активна спільнота розробників; інтеграція з GitHub; велика кількість тем та плагінів.	Повільна робота з великими файлами; високе споживання пам'яті; припинено активну розробку у 2022 році; нестабільність деяких плагінів; побудований на Electron
Sublime Text	Легкий та швидкий навіть на слабких ПК; Простий, мінімалістичний інтерфейс; велика кількість плагінів; мінімальне споживання ресурсів; стабільна робота.	Платний; гірша інтеграція з фреймворками та VCS, порівняно з конкурентами; обмежене автодоповнення без плагінів; відсутність вбудованого налагодження; рідші оновлення.

Серед проаналізованих середовищ розробки найкращим вибором є Visual Studio Code, оскільки це безкоштовний інструмент із широкими можливостями налаштування та масштабування. Крім того, він має велику активну спільноту та хорошу підтримку.

2.4 Висновки за розділом

У даному розділі було проведено аналіз та обґрунтування вибору технологічних рішень для створення веб-додатку. Залежно від порівняльного аналізу різних платформ розробки та мов програмування було прийнято рішення використовувати JavaScript як основну мову програмування для реалізації проєкту. Для розробки було вибрано інтегроване середовище розробки Visual Studio Code, яке надає потужні можливості для роботи з JavaScript, підтримкою різних плагінів та ефективними засобами налагодження.

Обрані технологічні рішення створюють надійну основу для ефективної розробки веб-додатку, забезпечуючи можливість подальшого розвитку та масштабування відповідно до потреб проєкту.

3 РОЗРОБКА CHROME-РОЗШИРЕННЯ

3.1 Архітектура розширення

Загальна структура розширення:

- `manifest.json`: основний конфігураційний файл, де знаходиться інформація про версію, назву, дозволи, файли інтерфейсу та скрипти;
- `popup.html`: HTML-сторінка, яка відображається при натисканні на іконку розширення та містить розмітку всіх елементів інтерфейсу;
- `styles.css`: таблиця стилів, яка відповідає за візуальне оформлення інтерфейсу;
- `script.js`: основна логіка роботи розширення (фільтрація, додавання, редагування задач тощо);
- `icons/`: іконки різного розміру для відображення розширення у браузері.

Файл `manifest.json` – це єдиний файл, який обов'язково має бути в кожному розширенні він містить важливу інформацію про структуру та поведінку цього розширення, що використовує WebExtension API [22].

Використовуючи `manifest.json`, ви визначаєте базові метадані про розширення, такі як ім'я та версія. Також можна визначити деякі аспекти функціональності, як фонові скрипти, контент-скрипти та дії браузера [22].

3.1.1 Зберігання даних

Зберігання даних є важливим аспектом розробки будь-яких програм. Ефективне зберігання забезпечує швидкий доступ до інформації, надійну та безпечну роботу веб-додатку. Було прийнято рішення не використовувати бази даних, а реалізувати розширення через локальне збереження даних.

Основними причинами були:

- простота інтеграції: API спеціально розроблено для розширень Chrome і не потребує додаткових бібліотек;

- асинхронна робота: не блокує інтерфейс користувача під час операцій;
- достатня продуктивність: обсягу до 10 МБ буде цілком достатньо для зберігання списків завдань;
- безпека: дані зберігаються локально на ПК в ізольованому середовищі розширення [23].

Алгоритм зберігання даних можна побачити на рисунку 3.1.

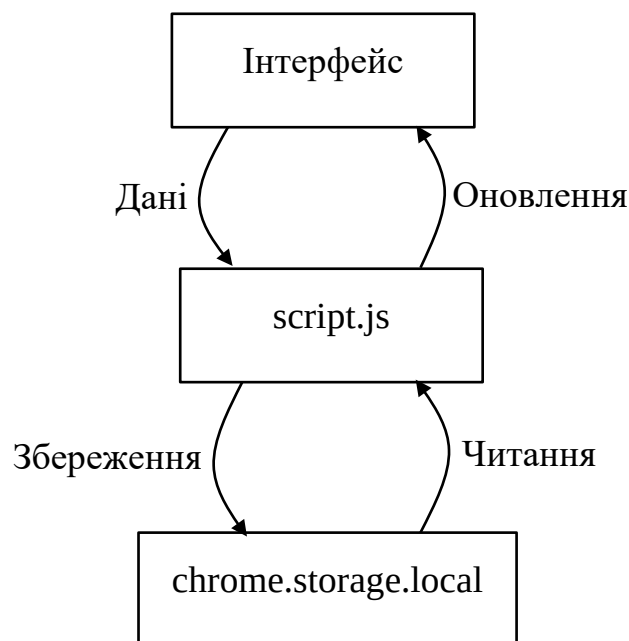


Рисунок 3.1 – Алгоритм зберігання даних

Ця схема демонструє взаємозв'язок між інтерфейсом, логікою програми та сховищем. Усі зміни, які ініціює користувач через інтерфейс, обробляються у `script.js` і зберігаються у `chrome.storage.local`. У свою чергу, при відкритті розширення, інформація зчитується зі сховища, передається у `script.js` та оновлюється в HTML. Такий підхід забезпечує чітку структуру, зручність у тестуванні та можливість розширення функціоналу без суттєвих змін у базових компонентах.

Для перегляду збережених даних можна скористатися інструментами розробника браузера Google Chrome (DevTools). Для цього необхідно відкрити консоль розробника, перейти до вкладки application, знайти розділ extension storage, а далі підрозділ local, де відображається уся збережена інформація з розширення (рис. 3.2).

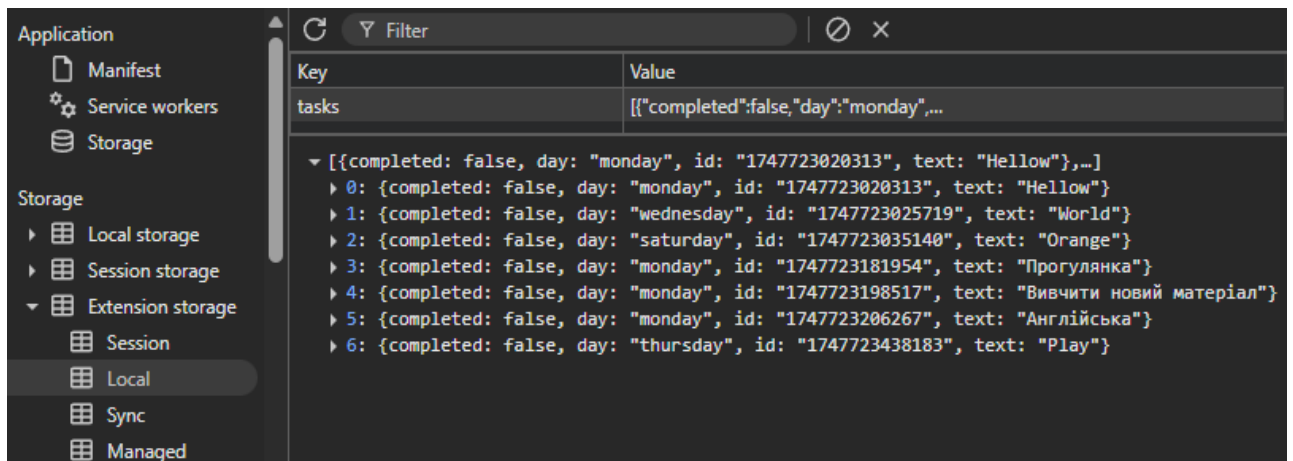


Рисунок 3.2 – Зберігання даних в extension storage

Також розглядався варіант використання `chrome.storage.sync`, який забезпечує синхронізацію даних між пристроями через Google акаунт. Однак цей спосіб має обмеження, максимальний обсяг даних становить лише 100 кілобайт, а також бувають певні затримки з їх отриманням, що робить його менш привабливим варіантом.

3.2 Основний функціонал

У цьому підрозділі розглянуто реалізацію основного функціоналу Chrome-розширення. Основна мета якого надати користувачу інтуїтивно зрозумілий інструмент для керування задачами, завдяки якому можна створювати, редагувати, видаляти, фільтрувати за днями тижня. Де усі дані будуть зберігаються локально, що гарантує їхню доступність та навіть після перезапуску браузера.

Процес додавання нової задачі `addTask()` реалізований через систему валідації вводу та асинхронного збереження даних. При спробі додати задачу система перевіряє, чи не є поле вводу порожнім, створює новий об'єкт задачі з унікальним ідентифікатором на основі поточної мітки часу та додає його до існуючого масиву задач у Chrome Storage.

Унікальність ідентифікатора забезпечується використанням методу `Date.now()`, який повертає поточну мітку часу в мілісекундах. Такий підхід гарантує унікальність навіть при швидкому додаванні декількох задач підряд.

Після успішного збереження в локальному сховищі система автоматично оновлює інтерфейс користувача, додаючи нову задачу до списку без необхідності перезавантаження. Більш детально описано на рисунку 3.3.

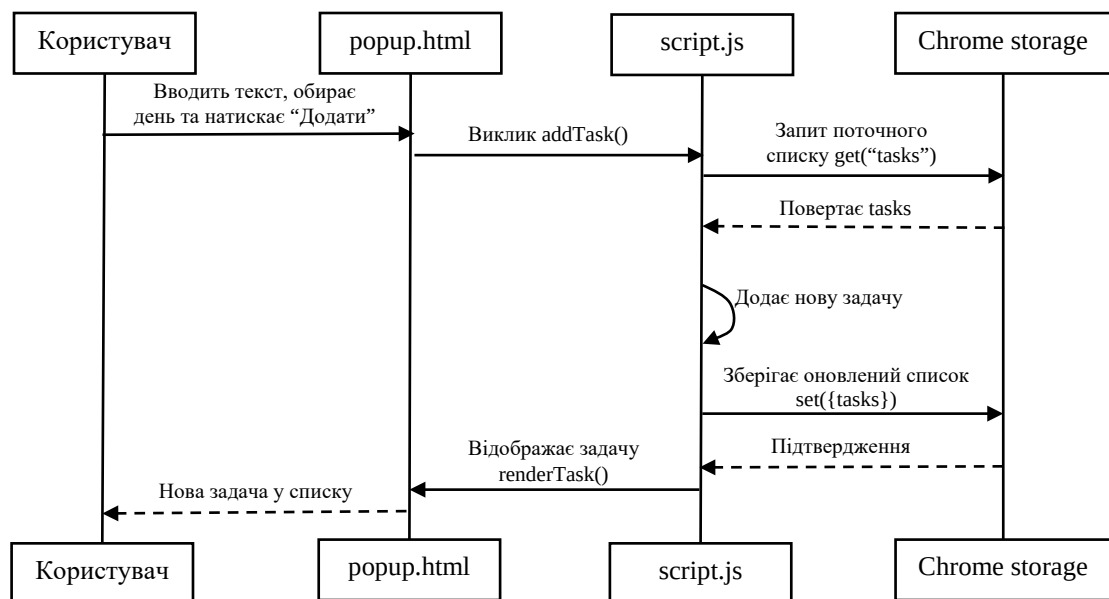


Рисунок 3.3 – Діаграма послідовності додавання нової задачі

Система редагування задач `enterEditMode()` яка дозволяє користувачам редагувати текст задачі безпосередньо в списку без відкриття додаткових діалогових вікон. При активації режиму редагування система замінює звичайний текстовий елемент на поле вводу з попередньо заповненим

значенням. Це забезпечує плавний перехід між режимами перегляду та редагування (рис. 3.4).

Збереження змін `saveEditedTask()` відбувається через натискання клавіші `enter`, втрати фокусу полем вводу або натискання кнопки збереження. Такий підхід забезпечує інтуїтивність використання та мінімізує ризик втрати даних (рис. 3.4).

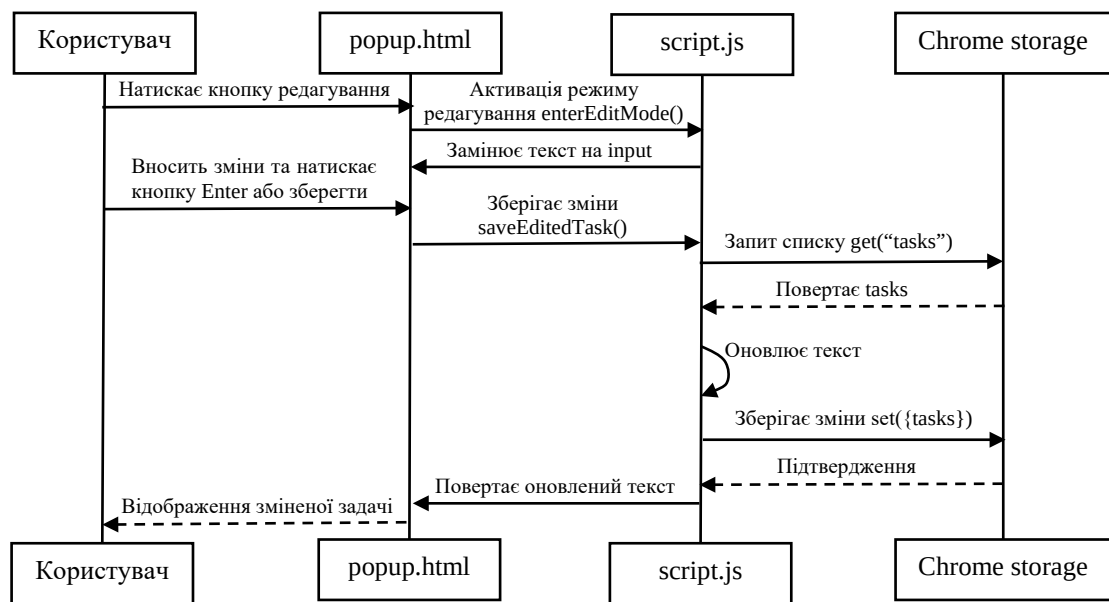


Рисунок 3.4 – Діаграма послідовності редагування задач

Видалення задачі реалізовано через кнопку “Видалити” біля кожного елемента списку. При натисканні система знаходить задачу за унікальним ідентифікатором, видаляє її з масиву задач у сховищі та миттєво прибирає відповідний елемент з DOM. Це дозволяє користувачу швидко очищати список від непотрібних задач без затримок (рис. 3.5).

Також є функція масового видалення всіх задач `clearAllTasks()`, яка включає обов'язкове підтвердження через браузерний діалог `confirm()`. При підтвердженні система очищає як локальне сховище, так і DOM представлення списку задач. Ця функціональність особливо корисна для користувачів, які регулярно очищують свій список задач та починають планування знову.

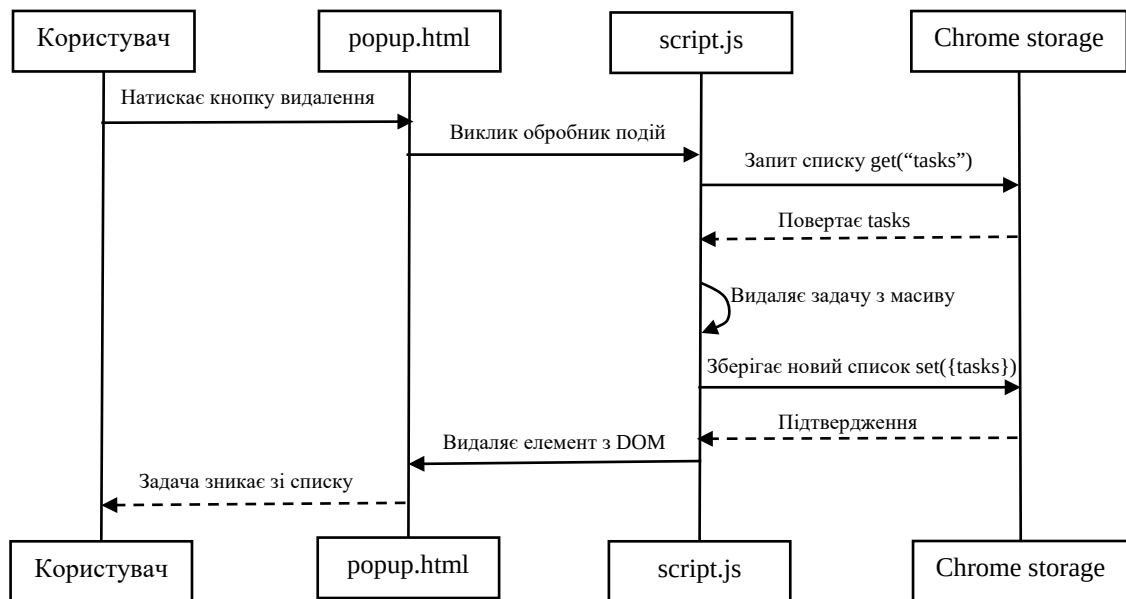


Рисунок 3.5 – Діаграма послідовності видалення задач

Фільтрація `applyFilter()` реалізована через динамічне управління видимістю елементів інтерфейсу без необхідності повторного завантаження або перерендерингу всього списку задач. При виборі дня система перебирає всі задачі в DOM та приховує ті, які не відповідають обраному фільтру. Такий підхід забезпечує високу швидкість та плавність взаємодії. Дозволяє користувачам фокусуватися на задачах конкретного дня або отримувати загальний огляд всіх запланованих активностей (рис. 3.6).

Також є і інші функції які допомагають роботі додатку. `updateItemsCount()` – адаптує інформацію відповідно до активного фільтра, відображаючи як кількість задач, що відповідають поточному критерію, так і загальну кількість задач у системі. `getDayName(dayValue)` – перетворює значення дня у зрозумілу для користувача аббревіатуру для відображення в списку, `loadTasks()` – завантажує задачі з Chrome Storage та відображає їх у списку.

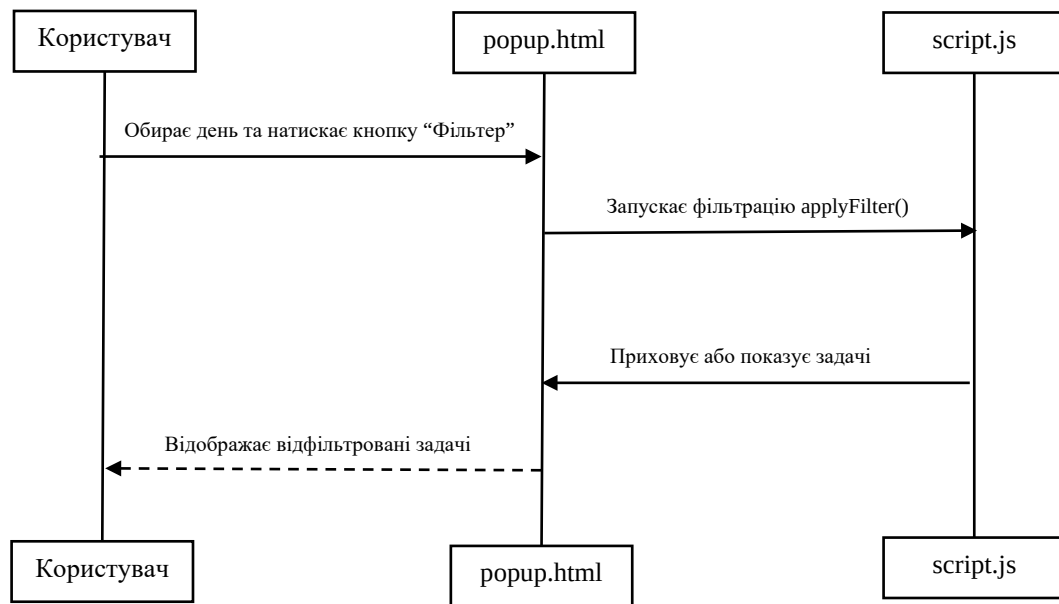


Рисунок 3.6 – Діаграма послідовності фільтрування задач

3.3 Інтерфейс користувача та дизайн

Під час розробки інтерфейсу розширення для роботи зі списками справ було прийнято рішення дотримуватися мінімалістичного дизайну. Щоб забезпечити простий та зрозумілий у використанні інтерфейс, який не відволікає користувача зайвими елементами й зосереджує увагу лише на основному, на управлінні завданнями.

На початковому етапі створення проєкту було зроблено паперовий макет, що дозволило візуалізувати ключові елементи інтерфейсу: розміщення заголовка, поля для введення задачі, кнопку додавання, фільтра днів тижня тощо. Такий підхід дав змогу продумати логіку взаємодії користувача з розширенням ще до написання коду та дозволив уникнути надлишкових елементів (рис. 3.7).

Щоб краще проілюструвати логіку роботи інтерфейсу, розширення можна розбити на декілька станів.

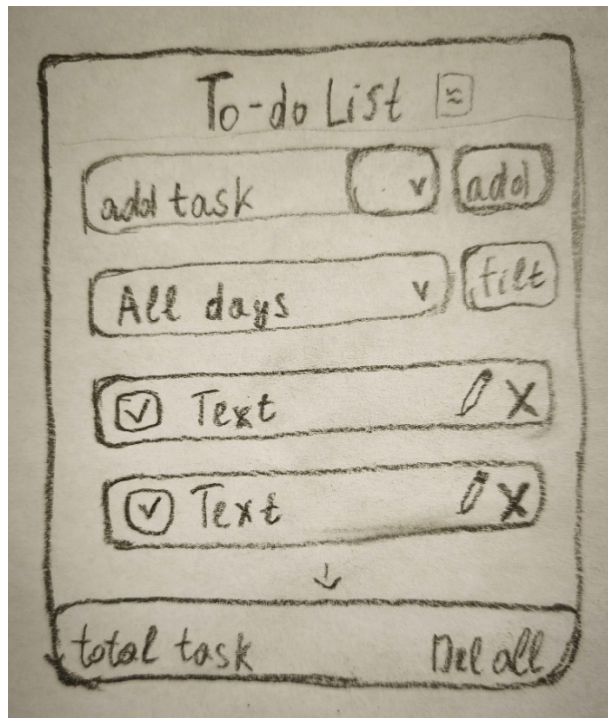


Рисунок 3.7 – Паперовий макет розширення для ведення списків

Стан 1. Порожній список справ:

- інтерфейс не перевантажений;
- фільтр днів тижня доступний, але не впливає на відображення задач;
- лічильник задач показує 0.

Цей стан мотивує користувача почати роботу та додати першу задачу (рис. 3.8).

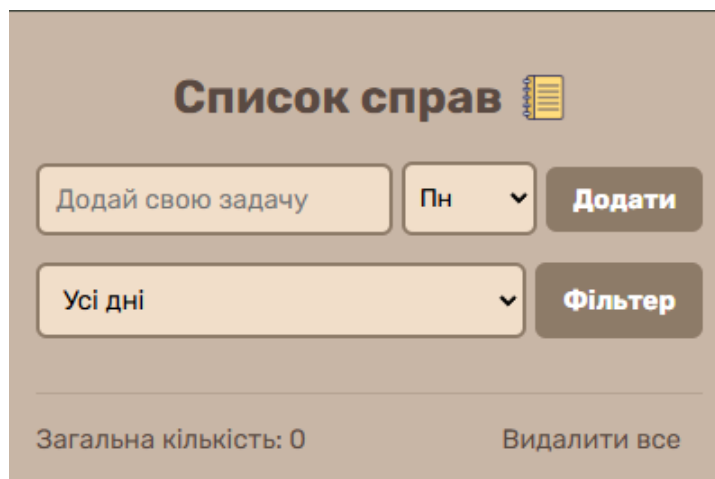


Рисунок 3.8 – Порожній стан списку справ

Стан 2. Задачі додані до списку:

- у списку відображаються всі створені задачі, незалежно від дня тижня;
- кожна задача містить: текст задачі, чекбокс виконання, кнопку редагування, кнопку видалення та позначку дня тижня;
- лічильник показує загальну кількість завдань (рис. 3.9).

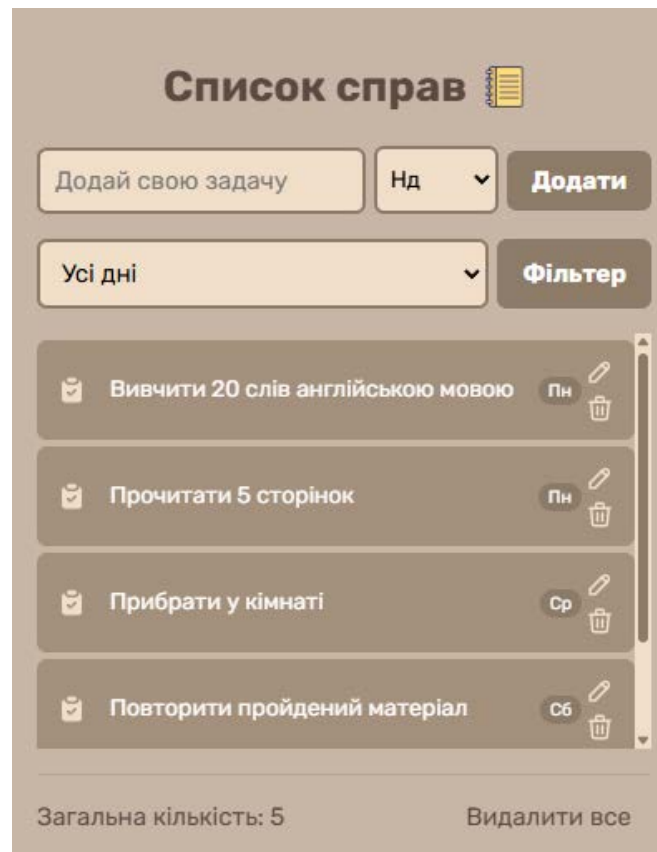


Рисунок 3.9 – Стан списку справ з доданими завданнями

Стан 3. Задачі присутні та відфільтровані за днем тижня:

- у списку відображаються лише задачі, що заплановані на понеділок;
- всі інші задачі тимчасово приховані;
- зберігає повну функціональність: редагування, виконання, видалення, додавання задач;
- лічильник показує відфільтровану та загальну кількість задач (рис.3.10).

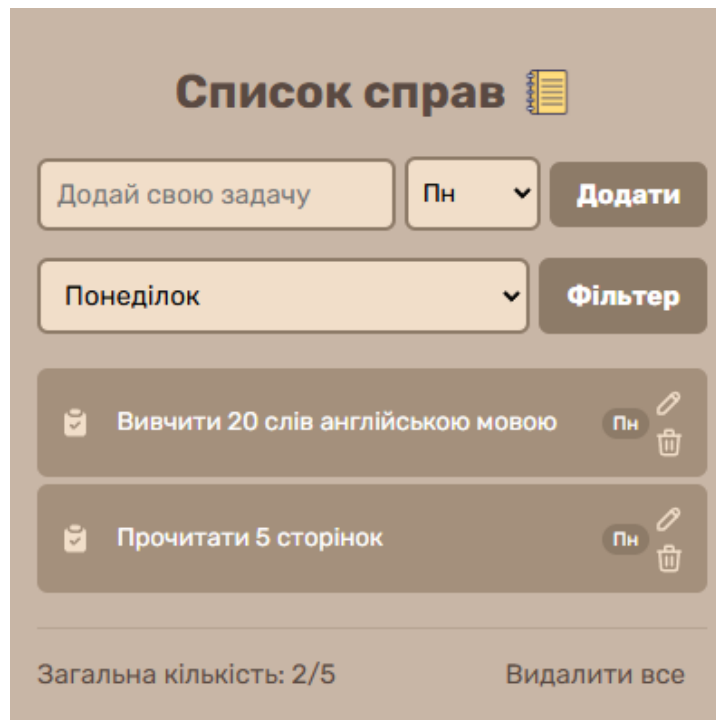


Рисунок 3.10 – Відфільтрований стан списку справ

3.4 Висновки за розділом

У даному розділі було детально розглянуто архітектуру, логіку роботи, основні функції та інтерфейс користувача розробленого Chrome-розширення для управління списками справ. Показано структуру розширення з чітким розмежуванням між візуальною частиною, логікою обробки подій та зберіганням даних. Проаналізовано основний функціонал розширення та приділено увагу способам взаємодії з користувачем через мінімалістичний дизайн інтерфейсу, що забезпечує простоту використання навіть для недосвідчених користувачів.

4 ІНСТРУКЦІЇ З ЕКСПЛУАТАЦІЇ, ВСТАНОВЛЕННЯ ТА ТЕСТУВАННЯ

4.1 Інструкція з встановлення та використання для користувачів

Для початку використання розширення користувачу необхідно мати браузер Google Chrome, оскільки розширення було розроблено для цієї платформи. Використання альтернативних браузерів (наприклад Mozilla Firefox), може призвести до виникнення помилок або некоректної роботи функціоналу. З метою уникнення можливих помилок рекомендується чітко дотримуватись наведеної нижче інструкції.

Для початку необхідно завантажити архів з файлами розширення. Там знаходяться всі необхідні файли такі як popup.html, styles.css, script.js, manifest.json та папка icons.

Перемістити файл у зручне та безпечне місце на комп'ютері (наприклад, у папку "Документи"), щоб уникнути випадкового видалення або переміщення, яке може порушити роботу розширення.

Відкрити браузер Google Chrome та ввести в адресний рядок: chrome://extensions або самому зайти через іконку "Пазлу" та керувати розширеннями і натиснути (рис. 4.1).

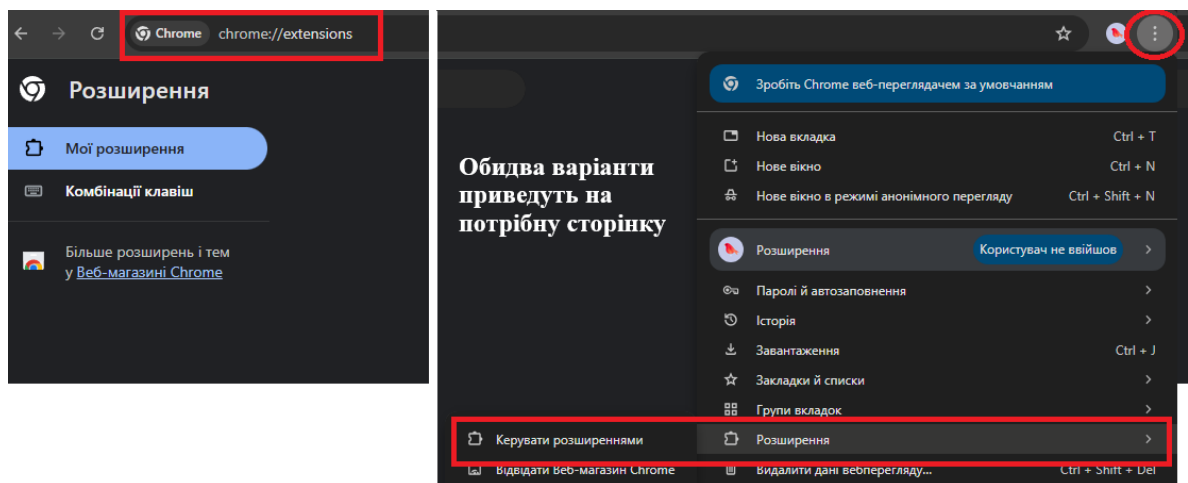


Рисунок 4.1 – Меню керування розширеннями

У верхньому правому куті сторінки необхідно активувати "Режим розробника" (рис. 4.2).

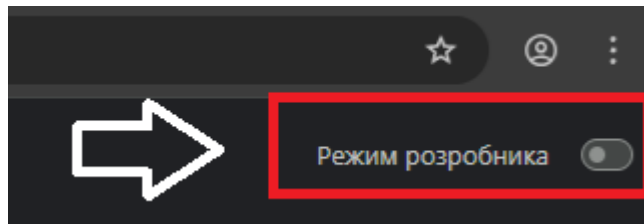


Рисунок 4.2 – Ввімкнення режиму розробника

Після активації режиму з'явиться додаткове меню з кнопкою «Завантажити розпаковане розширення», яку необхідно натиснути. І у вікні файлового провідника слід обрати ту папку, що містить файли розширення, та підтвердити свій вибір (рис. 4.3).

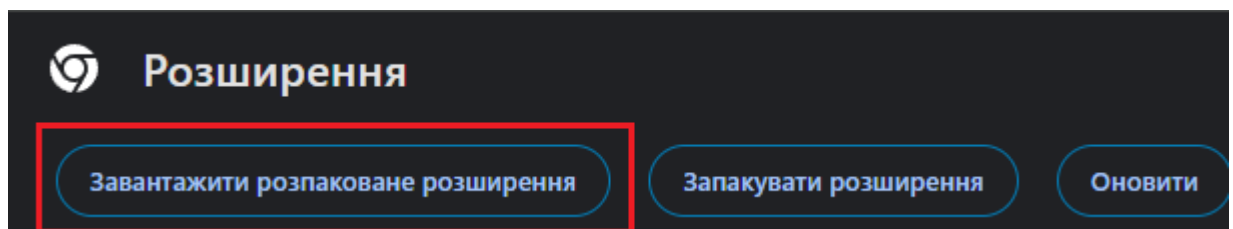


Рисунок 4.3 – Меню розробника та завантаження розширення

Після завершення завантаження розширення буде автоматично додано до браузера. У правому верхньому куті, біля адресного рядка, з'явиться іконка у вигляді "Пазлу" – це меню керування розширеннями. Натиснувши на неї, користувач може переглянути список встановлених розширень. Рекомендується закріпити розширення для зручного доступу натиснувши "закріпити" схожий на значок кнопки (рис.4.4).

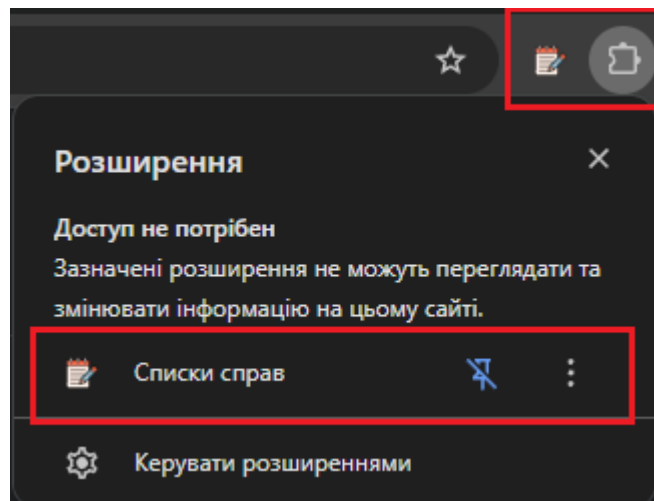


Рисунок 4.4 – Меню активованих розширень

Після завершення інсталяції розширення готове до використання без необхідності повторного встановлення або авторизації (рис. 4.5).

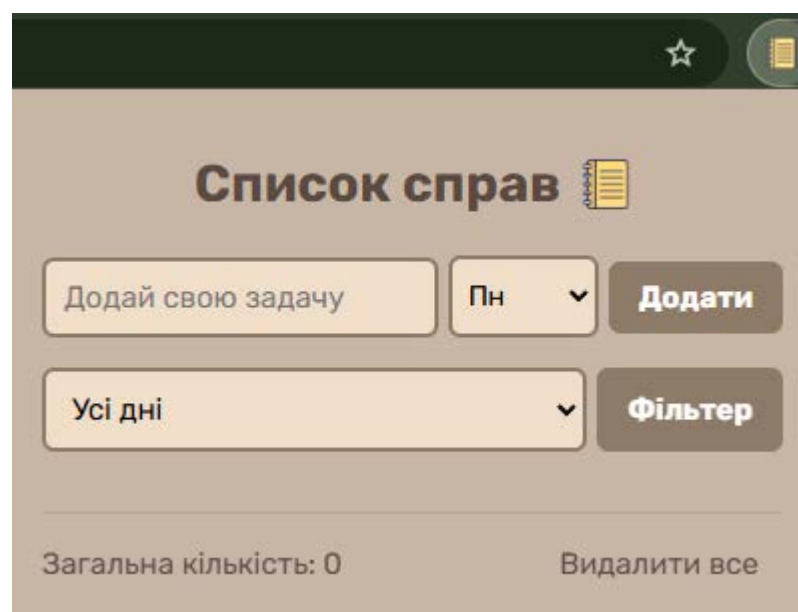


Рисунок 4.5 – Розширення “Список справ”

4.2 Тестування функціоналу та аналіз результатів

У процесі створення програмного забезпечення важливим етапом є перевірка його функціональності та зручності використання. У межах даної роботи було проведено два основні типи тестування: функціональне та

юзабіліті-тестування. Вони дозволяють оцінити не лише правильність роботи розширення, але й зручність взаємодії користувача з інтерфейсом.

Функціональне тестування полягає у перевірці того, чи виконує програмне забезпечення усі заявлені функції відповідно до технічного завдання. Його мета це переконатися, що кожна кнопка, кожен елемент інтерфейсу та кожна дія призводить до очікуваного результату. У межах функціонального тестування були перевірені ключові сценарії.

Створення задачі – після введення тексту у відповідне поле "Додай свою задачу", обрати день тижня на який запланована задача та натиснути кнопку "Додати". Після чого нова задача з'явиться в розширенні (рис. 4.6).

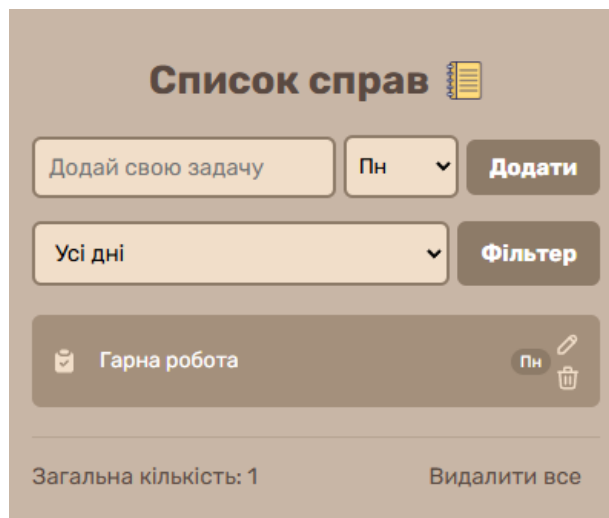


Рисунок 4.6 – Додана нова задача

Позначення виконання задачі – завершені задачі можна відзначати як виконані для цього необхідно натиснути на чек бокс ліворуч від тексту. Це дозволяє візуально відділити виконані задачі від тих які ще потребують уваги (рис. 4.7).



Рисунок 4.7 – Задача позначена як виконана

Редагування задачі – при натисканні кнопки редагування “Олівець” користувач має змогу змінити текст задачі та зберегти оновлену версію (рис. 4.8).

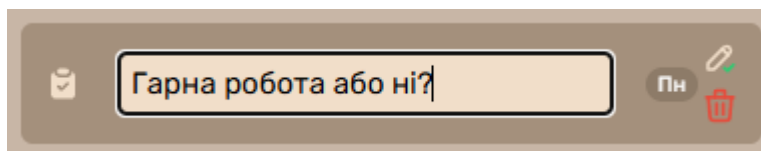


Рисунок 4.8 – Редагування задачі

Видалення задачі – при натисканні кнопки видалити “Кошик” користувач зможе видалити конкретну задачу.

Фільтрація за днями тижня – при виборі конкретного дня (наприклад “Понеділок”) та відфільтрувати натиснувши кнопку “Фільтер”, у списку мають відображатися лише ті задачі, що прив’язані до нього (рис. 4.9).

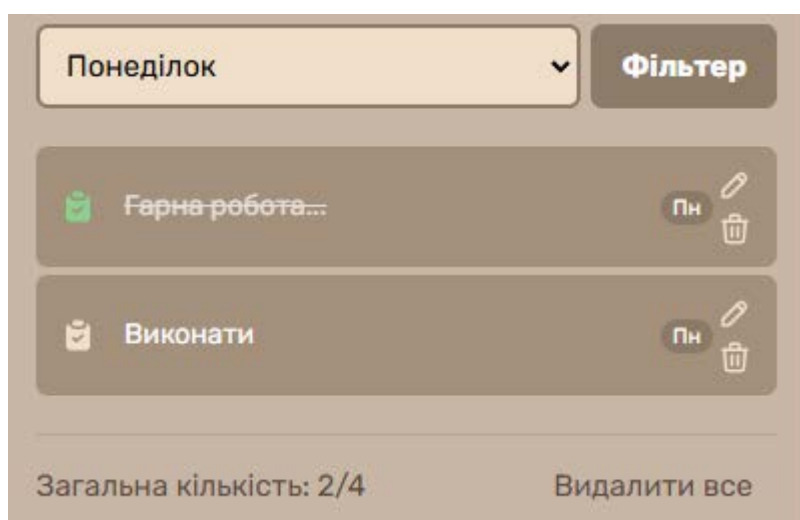


Рисунок 4.9 – Задачі відфільтровані за Понеділком

Для того, щоб видалити всі задачі необхідно буде натиснути на кнопку “Видалити все”, яка розташована у нижньому правому куті інтерфейсу (рис. 4.9). При натисканні на яку відкриється вікно підтвердження з повідомленням “Ви впевнені, що хочете видалити всі завдання?” натиснувши “ОК”, усі поточні задачі буде видалено. (рис. 4.10).

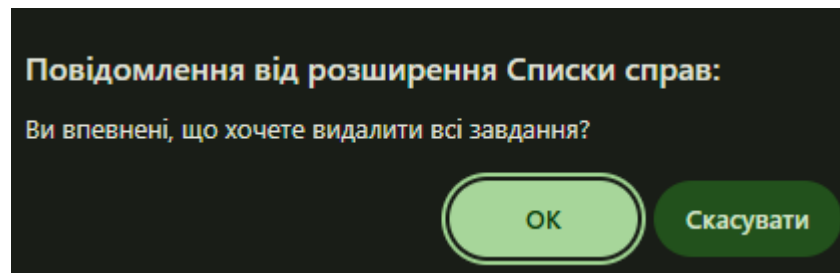


Рисунок 4.10 – Вікно з підтвердженням про остаточне видалення задач

Збереження даних у локальному сховищі – після перезавантаження сторінки або закриття та відкриття браузера список задач повинен залишатися незмінним, якщо користувач сам не видалив задачі.

Усі перелічені дії були протестовані вручну в середовищі браузера Google Chrome з використанням режиму розробника, інструментів Chrome DevTools, а також простого взаємодіючого сценарію користувача. Результати тестування підтвердили, що всі ключові функції додатку працюють коректно, без збоїв та помилок.

Юзабіліті-тестування (або тестування зручності використання) спрямоване на вивчення досвіду користувача при взаємодії з програмним забезпеченням. Це тестування є особливо важливим для інтерфейсів, орієнтованих на широку аудиторію, оскільки навіть технічно бездоганна система може виявитися незручною у використанні, що призводить до зниження ефективності та задоволеності користувачів.

Для оцінки юзабіліті було проаналізовано наступні аспекти:

- зрозумілість інтерфейсу – усі основні елементи (поле вводу, кнопка додавання, список задач, кнопки редагування та видалення) розміщені інтуїтивно зрозуміло;
- логічність взаємодії – після виконання дії користувач одразу бачить її результат (наприклад, задача одразу додається до списку без оновлення сторінки);
- зовнішній вигляд – обрано мінімалістичний стиль з приємною кольоровою палітрою, кнопки мають достатній розмір і добре видимі, а шрифт є читабельним;

– реакція на дії користувача – елементи керування реагують миттєво без затримок, а візуальні ефекти покращують загальне враження від використання.

Окрім особистої перевірки, до тестування було залучено декілька знайомих добровольців, яким було запропоновано:

- додати задачу ;
- відфільтрувати задачі за днем тижня;
- перевірити зберігаються дані після оновлення сторінки;
- редагувати та видалити задачу.

Результати тестування показали, що всі користувачі успішно виконали поставлені завдання без необхідності інструкцій або додаткових пояснень. Це свідчить про високий рівень інтуїтивності інтерфейсу та відсутність значних труднощів (табл. 4.1).

Таблиця 4.1 – Тест-кейс очікуваний і фактичний результат

№	Назва тесту	Очікуваний результат	Фактичний результат	Статус
1	Додавання задачі	Задача з'являється у списку	Задача з'являється	Успішно
2	Візуальне виділення виконаної задачі	Задача повинна якось виділятися від невиконаної	Задача позначається як виконана, стає більш прозорою і закреслюється	Успішно
3	Видалення задачі	Задача зникає зі списку	Зникає	Успішно
4	Фільтрація по днях	Показуються задачі лише для вибраного дня	Все працює правильно	Успішно
5	Збереження після оновлення сторінки	Задача не зникає після перезавантаження сторінки та перезапуску	Не зникає	Успішно
6	Зручний і зрозумілий інтерфейс	Може без інструкції додати/видалити задачу	Може	Успішно

4.3 Порівняння з аналогами

Для оцінки доцільності розробки власного розширення було проведено порівняння з популярними аналогами, зокрема Google Tasks, Todoist та Microsoft To Do. У таблиці нижче наведено основні характеристики, що впливають на зручність використання, гнучкість функціоналу, автономність та локалізацію (табл. 4.2).

Таблиця 4.2 – Порівняння розширення з існуючими аналогами

Характеристика	Google Tasks	Todoist	TickTick	Власне розширення
Додавання та видалення задач	Так	Так	Так	Так
Редагування задач	Так	Так	Так	Так
Повторювані задачі	Так	Так	Так	Ні
Фільтрація за днями тижня	Ні	Частково	Ні	Так
Вибір дати з календаря	Так	Так	Так	Ні
Збереження даних без акаунта	Ні	Ні	Ні	Так
Офлайн-робота	Частково	Ні	Ні	Так
Локалізація українською мовою	Так	Ні	Так	Так
Простота інтерфейсу	Так	Ні	Ні	Так
Кросплатформність	Так	Так	Так	Ні
Аналітика та звіти	Ні	Так	Так	Ні

У поточній версії розширення деякі функції ще поступаються аналогам на ринку (To-Do List). Однак основний функціонал вже реалізовано, а такі

елементи, як календар, перебувають на етапі розробки. Тому поки що розширення для списків справ дещо відстає, але з часом воно матиме аналогічний або подібний рівень функціональності.

4.4 Аналіз реалізованого функціоналу

У межах реалізації проєкту було успішно реалізовано основну частину функціоналу:

- створення, редагування та видалення задач;
- прив'язка задач до конкретного дня тижня;
- фільтрація списку задач за днями;
- локальне збереження задач через `chrome.storage.local`;
- інтерфейс українською мовою;
- можливість роботи офлайн без доступу до інтернету;
- повна автономність (не потребує акаунтів).

Це дозволяє користувачу швидко взаємодіяти з додатком у звичному середовищі браузера, не витрачаючи час на реєстрацію або вивчення складного інтерфейсу.

Під час розробки було визначено кілька моментів, які поки залишаються не реалізованими, але важливими для покращення досвіду користувача:

- відсутність підтримки повторюваних задач;
- немає можливості вибору точної дати через календар;
- задачі не розподіляються по категоріях (робота, дім, навчання тощо);
- не реалізовано пошук по задачах;
- не підтримується сортування задач вручну або за пріоритетом.

Ці функції не є критично необхідними, але їх реалізація дозволила б розширити аудиторію користувачів та надати більше інструментів для управління завданнями.

4.5 Пропозиції для подальшого розвитку

Під час реалізації цього проєкту виникло багато ідей щодо вдосконалення, але й певні труднощі. Вони стосувалися не лише технічних навичок, а й організації інтерфейсу, оскільки хотілося зробити розширення компактним, зручним та інтуїтивно зрозумілим. Саме тому деякі функції, такі як календар, були тимчасово виключені - попри те, що базова реалізація працювала, її присутність ускладнювала загальний інтерфейс. Для повноцінного впровадження цих можливостей потрібні додатковий час та чітке проєктування структури розширення.

З огляду на результати експлуатації, було сформульовано низку ідей щодо вдосконалення розширення в майбутньому:

- додавання календарного планування – реалізація можливості вибору конкретної дати через календар;
- категорії задач – створення фільтрів за типами задач: “робота”, “особисте”, “навчання” тощо;
- система повторюваних задач – впровадження можливості автоматичного повторення задач (щодня, щотижня, щомісяця);
- зберігання даних через бази даних або синхронізація з Google-акаунтом – зберігання задач у хмарі з доступом з будь-якого пристрою;
- підтримка кількох мов інтерфейсу як мінімум Українська та Англійська мова;
- розширена статистика задач – додавання модуля аналітики: кількість виконаних задач, середнє навантаження на день, динаміка виконання.

Інтеграція таймера за методом Pomodoro – додати можливість користувачу запускати таймер фокусованої роботи (25 хвилин) із перервами, як це реалізовано в методі Pomodoro. Такий інструмент дозволить не лише створювати задачі, а й ефективно працювати над їх виконанням без відволікань. Приклад як він буде виглядати наведено на рисунку 4.11.

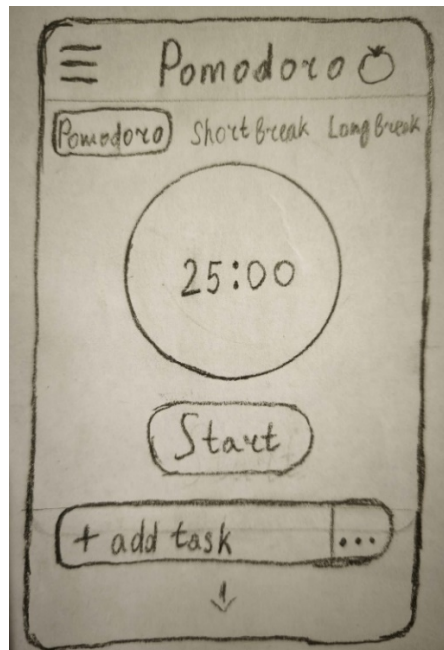


Рисунок 4.11 – Макет Pomodoro таймера

Було створено макет як буде виглядати фінальний варіант розширення після всіх доопрацювань та втілення пропозицій (рис. 4.12).

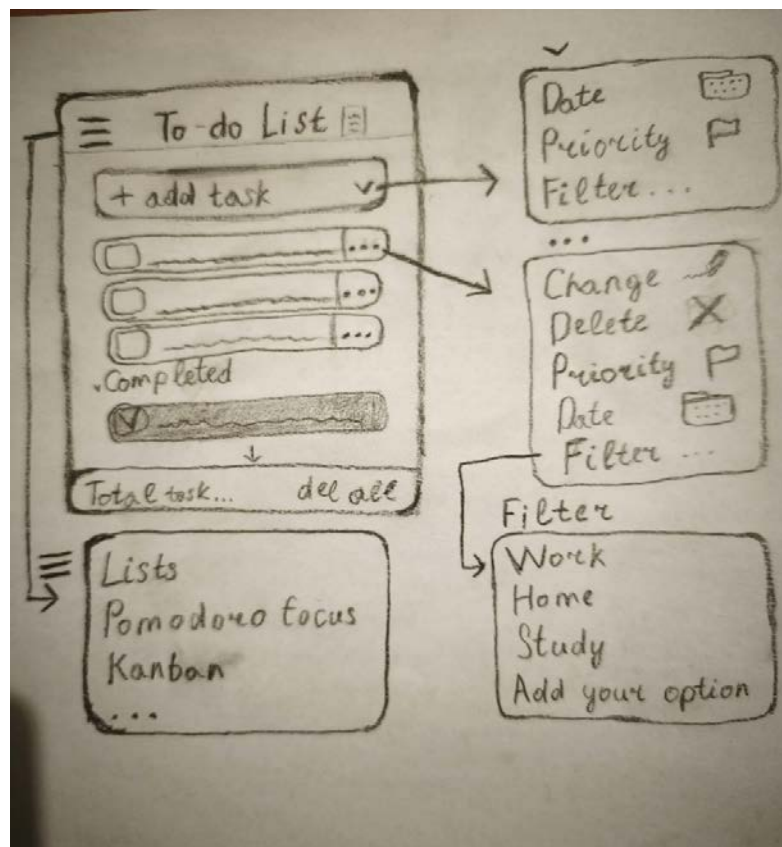


Рисунок 4.12 – Макет вигляду майбутнього розширення

4.6 Висновки за розділом

У даному розділі було розглянуто процес встановлення та налаштування розробленого розширення, а також проведено аналіз його функціональності. Для оцінки ефективності було виконано тестування основних сценаріїв використання, що підтвердило коректність роботи основних функцій, а також проведено порівняльний аналіз з аналогами.

Окрім того, на основі створеного розширення, було оцінено перспективи розвитку, та подальші вдосконалення, такі як оптимізація продуктивності та розширення функціональних можливостей.

ВИСНОВКИ

У ході виконання дипломної кваліфікаційної роботи було реалізовано розширення яке надає користувачам зручний інструмент для управління списками завдань безпосередньо в браузері Chrome. Проєкт був реалізований з використанням сучасних веб-технологій, таких як HTML5, CSS3, JavaScript, а також API Chrome Extensions для забезпечення інтеграції з браузером.

У першому розділі було проведено аналіз предметної області, проаналізовані основні методи для підвищення продуктивності, розглянуто існуючі рішення для організації списків та визначено їх основні переваги й недоліки. Це дозволило сформулювати вимоги до функціоналу та інтерфейсу майбутнього розширення.

У другому розділі була вибрана платформа на якій буде реалізований веддодаток, обрано методи та технології розробки: JavaScript як основну мову програмування, HTML та CSS для створення користувацького інтерфейсу, Chrome extension API для взаємодії з браузером та збереження даних користувача.

У третьому розділі описано процес розробки Chrome-розширення, включаючи створення архітектури додатку, реалізацію основних функцій: створення, редагування, видалення списків та створення мінімалістичного інтерфейсу користувача.

У четвертому розділі подано інструкцію з встановлення та використання розширення, наведено результати тестування функціональності, проаналізовано його працездатність та зручність використання.

Розроблене Chrome-розширення може бути корисне для широкого кола користувачів які хочуть організувати особисті та робочі завдання, підвищити продуктивність та ефективність роботи. Програмний продукт є зручним у використанні, має простий інтерфейс та може стати основою для подальшого розширення функціоналу.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. To-do list – Wikipedia [Electronic resource]. – Access mode: https://fr.wikipedia.org/wiki/To-do_list.
2. The power of getting things done: the GTD method explained [Electronic resource]. – Access mode: <https://www.float.com/resources/getting-things-done-method>.
3. Getting Things Done (GTD) – Todoist [Electronic resource]. – Access mode: <https://www.todoist.com/productivity-methods/getting-things-done>.
4. Метод Pomodoro – Вікіпедія [Електрон. ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/%D0%9C%D0%B5%D1%82%D0%BE%D0%B4_Pomodoro.
5. Kanban board – for agile project management [Electronic resource]. – Access mode: https://otrs.com/blog/using-otrs/kanban-board/#pll_switcher.
6. Kanban – Todoist [Electronic resource]. – Access mode: <https://www.todoist.com/productivity-methods/kanban>.
7. Що таке матриця Ейзенхауера? – Experience Dropbox [Електрон. ресурс]. – Режим доступу: <https://experience.dropbox.com/uk-ua/resources/eisenhower-matrix>.
8. Todoist Review: Features, Pros & Cons [Electronic resource]. – Access mode: <https://www.forbes.com/advisor/business/software/todoist-review/>.
9. Learn about Google Tasks [Electronic resource]. – Access mode: <https://support.google.com/tasks/answer/7675772?hl=en>.
10. About Trello. What's behind the boards – trello [Electronic resource]. – Access mode: <https://trello.com/en/about>.
11. Beginner's Guide – ticktick [Electronic resource]. – Access mode: <https://help.ticktick.com/articles/7054286604315131904>.
12. Desktop Browser Market Share Worldwide – statcounter [Electronic resource]. – Access mode: <https://gs.statcounter.com/browser-market-share/desktop/worldwide#monthly-202405-202505-bar>.

13. Desktop Browser Market Share Ukraine – statcounter [Electronic resource]. – Access mode: <https://gs.statcounter.com/browser-market-share/desktop/ukraine#monthly-202405-202505-bar>.
14. Десктоп застосунки. Noname - Digital Agency [Електрон. ресурс]. – Режим доступу: <https://noname.digital/services/desktop-applications/?lang=ua>.
15. Mobile Application – Techopedia [Electronic resource]. – Access mode: <https://www.techopedia.com/definition/2953/mobile-application-mobile-app>.
16. What is web application and its benefits? [Electronic resource]. – Access mode: <https://www.techtarget.com/searchsoftwarequality/definition/Web-application-Web-app>.
17. The Benefits of Using Browser Extensions for Enhanced Productivity – consumersearch [Electronic resource]. – Access mode: <https://www.consumersearch.com/technology/benefits-using-browser-extensions-enhanced-productivity>.
18. HTML: HyperText Markup Language – MDN Web Docs [Electronic resource]. – Access mode: <https://developer.mozilla.org/en-US/docs/Web/HTML>.
19. Mastering CSS: A Comprehensive Guide for Web Developers – linkedin [Electronic resource]. – Access mode: <https://www.linkedin.com/pulse/mastering-css-comprehensive-guide-web-developers-muhammad-mazhar-uksff>.
20. JavaScript Tutorial – geeksforgeeks [Electronic resource]. – Access mode: <https://www.geeksforgeeks.org/javascript/>.
21. Chrome DevTools. Overview – Chrome for Developers [Electronic resource]. – Access mode: <https://developer.chrome.com/docs/devtools/overview>.
22. Browser extensions. manifest.json – MDN Web Docs [Electronic resource]. – Access mode: <https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons/WebExtensions/manifest.json>.
23. Chrome Extensions. Reference. chrome.storage – Chrome for Developers [Electronic resource]. – Access mode: <https://developer.chrome.com/docs/extensions/reference/api/storage>.

ДОДАТОК А
Текст програми

А.1 Файл popup.html

```

<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>To-Do List</title>
  <link rel="preconnect" href="https://fonts.googleapis.com">
  <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
  <link
href="https://fonts.googleapis.com/css2?family=Rubik:ital,wght@0,300..900;1,300.
.900&display=swap" rel="stylesheet">
  <link rel="stylesheet" href="./styles.css">
</head>
<body>
  <div class="container">
    <div class="container-header">
      <h1>Список справ</h1>
      
    </div>
    <div class="input-container">
      <input type="text" id="task-input" placeholder="Додай свою задачу"
autofocus>
      <select id="day-select" class="compact-day">
        <option value="monday">Пн</option>
        <option value="tuesday">Вт</option>
        <option value="wednesday">Ср</option>
        <option value="thursday">Чт</option>
        <option value="friday">Пт</option>
        <option value="saturday">Сб</option>

```

```

    <option value="sunday">Нд</option>
</select>

<button id="add-task">Додати</button>
</div>
<div class="filter-container">
    <select id="filter-day">
        <option value="all">Усі дні</option>
        <option value="monday">Понеділок</option>
        <option value="tuesday">Вівторок</option>
        <option value="wednesday">Середа</option>
        <option value="thursday">Четвер</option>
        <option value="friday">П'ятниця</option>
        <option value="saturday">Субота</option>
        <option value="sunday">Неділя</option>
    </select>
    <button id="apply-filter">Фільтер</button>
</div>
<ul id="task-list"></ul>
<div class="footer">
    <div class="footer-separator"></div>
    <div class="footer-content">
        <span id="items-count">Загальна кількість: 0</span>
        <button id="clear-all">Видалити все</button>
    </div>
</div>
</div>

<svg xmlns="http://www.w3.org/2000/svg" style="display:none;">
    <symbol id="icon-edit" viewBox="0 0 24 24" fill="none" stroke="currentColor"
stroke-width="2" stroke-linecap="round" stroke-linejoin="round">

```

```

<path stroke="none" d="M0 0h24v24H0z" fill="none"/>
<path d="M4 20h4l10.5 -10.5a2.828 2.828 0 1 0 -4 -4l-10.5 10.5v4"/>
<path d="M13.5 6.5l4 4"/>
</symbol>
<symbol id="icon-save" viewBox="0 0 24 24" fill="none" stroke="currentColor"
stroke-width="2" stroke-linecap="round" stroke-linejoin="round">
  <path stroke="none" d="M0 0h24v24H0z" fill="none"/>
  <path d="M4 20h4l10.5 -10.5a2.828 2.828 0 1 0 -4 -4l-10.5 10.5v4"/>
  <path d="M13.5 6.5l4 4"/>
  <path d="M15 19l2 2l4 -4" stroke="#2ecc71"/>
</symbol>
<symbol id="icon-delete" viewBox="0 0 24 24" fill="none" stroke="currentColor"
stroke-width="2" stroke-linecap="round" stroke-linejoin="round">
  <path stroke="none" d="M0 0h24v24H0z" fill="none"/>
  <path d="M4 7l16 0"/>
  <path d="M10 11l0 6"/>
  <path d="M14 11l0 6"/>
  <path d="M5 7l1 12a2 2 0 0 0 2 2h8a2 2 0 0 0 2 -2l1 -12"/>
  <path d="M9 7v-3a1 1 0 0 1 1 -1h4a1 1 0 0 1 1 1v3"/>
</symbol>
<symbol id="icon-check" viewBox="0 0 24 24" fill="currentColor">
  <path stroke="none" d="M0 0h24v24H0z" fill="none"/>
  <path d="M17.997 4.17a3 3 0 0 1 2.003 2.83v12a3 3 0 0 1 -3 3h-10a3 3 0 0 1 -3
-3v-12a3 3 0 0 1 2.003 -2.83a4 4 0 0 0 3.997 3.83h4a4 4 0 0 0 3.98 -3.597zm-3.704
7.123l-3.293 3.292l-1.293 -1.292a1 1 0 1 0 -1.414 1.414l2 2a1 1 0 0 0 1.414 0l4 -4a1
1 0 0 0 -1.414 -1.414m-.293 -9.293a2 2 0 1 1 0 4h-4a2 2 0 1 1 0 -4z" />
</symbol>
</svg>
<script src="./script.js"></script>
</body>

```

</html>

A.2 Файл styles.css

```
body {  
  font-family: "Rubik", sans-serif;  
  color: #8D7B68;  
  height: 100%;  
  margin: 0;  
  width: 400px;  
  background: #C8B6A6;  
  font-size: 15px;  
}
```

```
.container {  
  padding: 15px;  
}
```

```
.container-header {  
  display: flex;  
  text-align: center;  
  justify-content: center;  
}
```

```
h1 {  
  text-align: center;  
  margin-bottom: 20px;  
  color: #5A4A42;  
  font-size: 26px;  
}
```

```
.container-img {  
  width: 25px;  
  height: 25px;  
  padding-top: 20px;  
  padding-left: 10px;  
}
```

```
.input-container {  
  display: flex;  
  margin-bottom: 15px;  
  gap: 5px;  
  align-items: center;  
}
```

```
.input-container input {  
  flex: 1;  
  padding: 9px;  
  border: 2px solid #8D7B68;  
  border-radius: 6px;  
  background: #F1DEC9;  
  outline: none;  
  font-size: 15px;  
  font-family: "Rubik", sans-serif;  
  min-width: 120px;  
}
```

```
.input-container select.compact-day {  
  width: 75px;  
  padding: 9px 4px;
```

```
border: 2px solid #8D7B68;  
border-radius: 6px;  
background: #F1DEC9;  
font-family: "Rubik", sans-serif;  
font-size: 14px;  
}
```

```
.filter-container {  
  display: flex;  
  margin-bottom: 15px;  
  gap: 5px;  
}
```

```
.filter-container select {  
  flex: 1;  
  padding: 9px;  
  border: 2px solid #8D7B68;  
  border-radius: 6px;  
  background: #F1DEC9;  
  font-family: "Rubik", sans-serif;  
  font-size: 15px;  
}
```

```
button {  
  border: none;  
  cursor: pointer;  
  border-radius: 6px;  
  font-size: 15px;  
  font-family: "Rubik", sans-serif;  
}
```

```
#add-task, #apply-filter {  
  padding: 9px 15px;  
  background: #8D7B68;  
  color: white;  
  font-weight: bold;  
  width: auto;  
  min-width: 60px;  
}
```

```
#add-task:hover, #apply-filter:hover {  
  background: #7A6A5A;  
  transition: background 0.2s ease;  
}
```

```
ul {  
  list-style: none;  
  padding: 0;  
  max-height: 250px;  
  overflow-y: auto;  
  margin-bottom: 12px;  
  scrollbar-width: thin;  
  scrollbar-color: #8D7B68 #F1DEC9;  
}
```

```
li {  
  display: flex;  
  align-items: center;  
  background: #A4907C;  
  margin: 4px 0;
```

```
padding: 8px 10px;  
border-radius: 6px;  
color: white;  
font-size: 14px;  
min-height: 36px;  
}
```

```
.task-content {  
  display: flex;  
  align-items: center;  
  flex-grow: 1;  
  gap: 6px;  
}
```

```
.task-text {  
  flex-grow: 1;  
  word-break: break-word;  
  line-height: 1.3;  
  margin: 0 6px;  
}
```

```
.task-day {  
  font-size: 0.75em;  
  background: #8D7B68;  
  color: white;  
  padding: 2px 6px;  
  border-radius: 10px;  
  white-space: nowrap;  
}
```

```
.complete-btn, .edit-btn, .delete-btn, .save-btn {  
  background: none;  
  border: none;  
  cursor: pointer;  
  width: 22px;  
  height: 22px;  
  padding: 0;  
  display: flex;  
  align-items: center;  
  justify-content: center;  
}
```

```
.edit-btn svg,  
.delete-btn svg,  
.save-btn svg {  
  width: 18px;  
  height: 18px;  
  color: #F1DEC9;  
  transition: all 0.3s ease;  
}
```

```
.complete-btn svg {  
  width: 18px;  
  height: 18px;  
  color: #F1DEC9;  
}
```

```
li.completed .task-text {  
  text-decoration: line-through;  
  opacity: 0.7;
```

```
}
```

```
li.completed .complete-btn svg {  
  color: #8ad18c;  
}
```

```
.save-btn {  
  display: none;  
}
```

```
.edit-mode .edit-btn {  
  display: none;  
}
```

```
.edit-mode .save-btn {  
  display: block;  
}
```

```
.edit-input {  
  flex-grow: 1;  
  padding: 6px;  
  border: 1px solid #8D7B68;  
  border-radius: 4px;  
  background: #F1DEC9;  
  margin: 0 10px;  
  font-family: "Rubik", sans-serif;  
  font-size: 15px;  
}
```

```
.footer {
```

```
margin-top: 15px;  
}
```

```
.footer-separator {  
  height: 1px;  
  background-color: #8D7B68;  
  margin-bottom: 10px;  
  opacity: 0.3;  
}
```

```
.footer-content {  
  display: flex;  
  justify-content: space-between;  
  align-items: center;  
  font-size: 15px;  
  color: #5A4A42;  
}
```

```
#clear-all {  
  background: none;  
  border: none;  
  color: #5A4A42;  
  cursor: pointer;  
  padding: 6px 12px;  
  border-radius: 4px;  
  transition: all 0.3s ease;  
}
```

```
#clear-all:hover {  
  color: #e74c3c;
```

```
background: rgba(218, 110, 98, 0.1);
}
```

```
.edit-btn:hover svg {
  color: #81D4FA;
}
```

```
.delete-btn:hover svg {
  color: #e74c3c;
}
```

```
.save-btn svg {
  color: #F1DEC9;
}
```

```
.save-btn:hover svg {
  color: #8ad18c;
}
```

```
.task-actions button:hover svg {
  transform: scale(1.1);
}
```

A.3 Файл script.js

```
document.addEventListener('DOMContentLoaded', function() {
  loadTasks();
```

```
document.getElementById("add-task").addEventListener("click", addTask);
```

```

document.getElementById("task-input").addEventListener("keypress", function(e)
{
    if (e.key === "Enter") addTask();
});
document.getElementById("apply-filter").addEventListener("click", applyFilter);
document.getElementById("clear-all").addEventListener("click", clearAllTasks);
});

```

```

function addTask() {
    const input = document.getElementById("task-input");
    const taskText = input.value.trim();
    const day = document.getElementById("day-select").value;

```

```

    if (taskText === "") return;

```

```

    const task = {
        text: taskText,
        day: day,
        completed: false,
        id: Date.now().toString()
    };

```

```

chrome.storage.local.get(['tasks'], function(result) {
    const tasks = result.tasks || [];
    tasks.push(task);

```

```

chrome.storage.local.set({ tasks: tasks }, function() {
    renderTask(task);
    input.value = "";
    updateItemsCount();

```

```
});
});
}
```

```
function renderTask(task) {
  const li = document.createElement("li");
  li.dataset.id = task.id;
  li.dataset.day = task.day;
  if (task.completed) li.classList.add("completed");

  li.innerHTML = `
    <div class="task-content">
      <button class="complete-btn" title="${task.completed ? 'Unmark' : 'Mark as
complete'}">
        <svg width="20" height="20"><use href="#icon-check"/></svg>
      </button>
      <span class="task-text">${task.text}</span>
      <span class="task-day">${getDayName(task.day)}</span>
    </div>
    <div class="task-actions">
      <button class="edit-btn" title="Edit task">
        <svg width="20" height="20"><use href="#icon-edit"/></svg>
      </button>
      <button class="save-btn" title="Save changes" hidden>
        <svg width="20" height="20"><use href="#icon-save"/></svg>
      </button>
      <button class="delete-btn" title="Delete task">
        <svg width="20" height="20"><use href="#icon-delete"/></svg>
      </button>
    </div>
  `
}
```

```
`;
```

```
document.getElementById("task-list").appendChild(li);
setupTaskEvents(li);
}
```

```
function setupTaskEvents(li) {
  const taskId = li.dataset.id;
```

```
  li.querySelector(".complete-btn").addEventListener("click", function() {
    li.classList.toggle("completed");
```

```

    chrome.storage.local.get(['tasks'], function(result) {
      const tasks = result.tasks.map(task => {
        if (task.id === taskId) {
          return { ...task, completed: !task.completed };
        }
        return task;
      });
    });
```

```
    chrome.storage.local.set({ tasks: tasks }, updateItemsCount);
  });
});
```

```
li.querySelector(".edit-btn").addEventListener("click", function() {
  enterEditMode(li);
});
```

```
li.querySelector(".save-btn").addEventListener("click", function() {
  saveEditedTask(li);
```

```
});
```

```
li.querySelector(".delete-btn").addEventListener("click", function() {
  chrome.storage.local.get(['tasks'], function(result) {
    const tasks = result.tasks.filter(task => task.id !== taskId);
    chrome.storage.local.set({ tasks: tasks }, function() {
      li.remove();
      updateItemsCount();
    });
  });
});
}
```

```
function enterEditMode(li) {
  const taskText = li.querySelector(".task-text");
  const text = taskText.textContent;

  const editInput = document.createElement("input");
  editInput.type = "text";
  editInput.className = "edit-input";
  editInput.value = text;

  taskText.replaceWith(editInput);
  editInput.focus();
  li.classList.add("edit-mode");

  editInput.addEventListener("blur", function() {
    saveEditedTask(li);
  });
}
```

```

editInput.addEventListener("keypress", function(e) {
  if (e.key === "Enter") {
    saveEditedTask(li);
  }
});
}

```

```

function saveEditedTask(li) {
  const editInput = li.querySelector(".edit-input");
  const taskText = document.createElement("span");
  taskText.className = "task-text";
  taskText.textContent = editInput.value.trim();

```

```

editInput.replaceWith(taskText);
li.classList.remove("edit-mode");

```

```

const taskId = li.dataset.id;
chrome.storage.local.get(['tasks'], function(result) {
  const tasks = result.tasks.map(task => {
    if (task.id === taskId) {
      return { ...task, text: taskText.textContent };
    }
    return task;
  });

  chrome.storage.local.set({ tasks: tasks });
});
}

```

```

function loadTasks() {

```

```

chrome.storage.local.get(['tasks'], function(result) {
  const savedTasks = result.tasks || [];
  savedTasks.forEach(task => {
    const li = document.createElement('li');
    li.dataset.id = task.id;
    li.dataset.day = task.day;
    if (task.completed) li.classList.add('completed');

    li.innerHTML = `
      <div class="task-content">
        <button class="complete-btn" title="${task.completed ? 'Unmark' : 'Mark as
complete'}">
          <svg width="20" height="20"><use href="#icon-check"/></svg>
        </button>
        <span class="task-text">${task.text}</span>
        <span class="task-day">${getDayName(task.day)}</span>
      </div>
      <div class="task-actions">
        <button class="edit-btn" title="Edit task">
          <svg width="20" height="20"><use href="#icon-edit"/></svg>
        </button>
        <button class="save-btn" title="Save changes" hidden>
          <svg width="20" height="20"><use href="#icon-save"/></svg>
        </button>
        <button class="delete-btn" title="Delete task">
          <svg width="20" height="20"><use href="#icon-delete"/></svg>
        </button>
      </div>
    `;
  }
);

```

```

    document.getElementById("task-list").appendChild(li);
    setupTaskEvents(li);
  });
  updateItemsCount();
});
}

```

```

function applyFilter() {
  const selectedDay = document.getElementById("filter-day").value;
  const tasks = document.querySelectorAll("#task-list li");

  tasks.forEach(task => {
    if (selectedDay === "all" || task.dataset.day === selectedDay) {
      task.style.display = "flex";
    } else {
      task.style.display = "none";
    }
  });

  updateItemsCount();
}

```

```

function getDayName(dayValue) {
  const days = {
    monday: "Пн",
    tuesday: "Вт",
    wednesday: "Ср",
    thursday: "Чт",
    friday: "Пт",
    saturday: "Сб",
  };
}

```

```

    sunday: "Нд"
  };
  return days[dayValue] || "";
}

```

```

function clearAllTasks() {
  if (confirm("Ви впевнені, що хочете видалити всі завдання?")) {
    chrome.storage.local.set({ tasks: [] }, function() {
      document.getElementById("task-list").innerHTML = "";
      updateItemsCount();
    });
  }
}

```

```

function updateItemsCount() {
  const allTasks = document.querySelectorAll('#task-list li').length;
  const visibleTasks = document.querySelectorAll('#task-list li:not([style*="display: none"])).length;
  const filterDay = document.getElementById("filter-day").value;

  if (filterDay === "all") {
    document.getElementById('items-count').textContent = `Загальна кількість: ${allTasks}`;
  } else {
    document.getElementById('items-count').textContent = `Загальна кількість: ${visibleTasks}/${allTasks}`;
  }
}

```

А.4 Файл manifest.json

```
{
  "manifest_version": 3,
  "name": "Списки справ",
  "version": "1.0",
  "permissions": ["storage"],
  "description": "Мінімалістичний список справ і задач",
  "action": {
    "default_popup": "./popup.html",
    "default_title": "To-Do List",
    "default_icon": {
      "16": "icons/icon16.png",
      "32": "icons/icon32.png",
      "48": "icons/icon48.png",
      "128": "icons/icon128.png"
    }
  },
  "icons": {
    "16": "icons/icon16.png",
    "32": "icons/icon32.png",
    "48": "icons/icon48.png",
    "128": "icons/icon128.png"
  }
}
```

ДОДАТОК Б
Слайди презентації

Національний університет «Запорізька політехніка»
Кафедра програмних засобів

Дипломна кваліфікаційна робота бакалавра на тему:

Розробка Chrome-розширення для роботи зі списками

Виконав:
студент групи КНТ-211

Станіслав ПАСЬКО

Керівник:
к.т.н., доцент

Ольга ГЛАДКОВА

Рисунок Б.1 – Слайд № 1

Об'єкт, предмет та мета роботи

- Об'єкт дослідження – процес розробки Chrome-розширення для роботи зі списками.
- Предмет дослідження – програмне забезпечення для керування списками справ у вигляді Chrome-розширення.
- Мета роботи – розробка Chrome-розширення для ефективного управління та підвищення особистої продуктивності за допомогою списку справ.

Рисунок Б.2 – Слайд № 2

Аналіз існуючих рішень

Категорії	Google Tasks	Todoist	Trello	TickTick	Власне розширення
Повністю безкоштовний	+	–	–	–	+
Інтеграції з іншими сервісами	+	+	+	+	–
Проста ієрархія завдань	+	+	+	+	+
Залежить від інтернет мережі	+	+–	+–	+–	–
Українська локалізація	+–	–	+	+	+
Додавання завдань	+	+	+	+	+
Фільтрація за днем	–	+	–	–	+
Видалення всього списку	–	–	–	–	+
Підрахунок задач	–	+	–	+	+

3

Рисунок Б.3 – Слайд № 3

Аналіз ринку браузерів

Світовий ринок браузерів		Український ринок браузерів	
Браузер	Відсоток	Браузер	Відсоток
Chrome	65.35%	Chrome	57%
Edge	13.49%	Edge	19.92%
Safari	8.77%	Opera	9.14%
Firefox	6.36%	Firefox	5.58%
Opera	2.87%	Yandex	5.5%
Other	3.16%	Other	2.85%

4

Рисунок Б.4 – Слайд № 4

Вибір мови програмування

Критерій	JavaScript	Python	C++
Зрозумілий синтаксис	+	+	—
Підтримка спільноти	+	+	+
Підтримка IDE	+	+	+
Висока продуктивність	—	—	+
Кросплатформна розробка	+	+	+
Інтеграція з Chrome API	+	—	—
Підтримка у браузері	+	—	—
Придатність для розширень	+	—	—
Швидкість розробки та навчання	+	+	—
Безпека	+—	+—	+

5

Рисунок Б.5 – Слайд № 5

Вибір середовища розробки

IDE	Основні особливості	Платформи	Вартість
Visual Studio Code	Легкий, швидкий та масштабований редактор коду з великою кількістю плагінів	Windows, macOS, Linux	Безкоштовно
WebStorm	Повнофункціональне потужне середовище	Windows, macOS, Linux	Платний (Безкоштовна пробна версія)
Atom	Легкий редактор коду з відкритим вихідним кодом та зручним інтерфейсом з великою кількістю плагінів	Windows, macOS, Linux	Безкоштовно
Sublime Text	Легкий та швидкий редактор коду з мінімалістичним інтерфейсом і великою кількістю плагінів	Windows, macOS, Linux	Платний (Безкоштовна пробна версія)

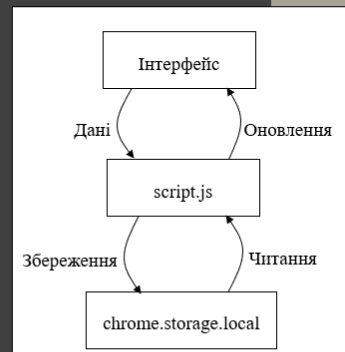
6

Рисунок Б.6 – Слайд № 6

Зберігання даних

Критерій	chrome.storage.local	IndexedDB	chrome.storage.sync	Бази даних
Потреба в сервері	–	–	–	+
Робота офлайн	+	+	+-	+-
Асинхронність	+	+	–	+
API складність	Проста	Складна	Проста	Складна
Швидкість доступу	Швидка	Швидка	Залежить від мережі	Залежить від мережі

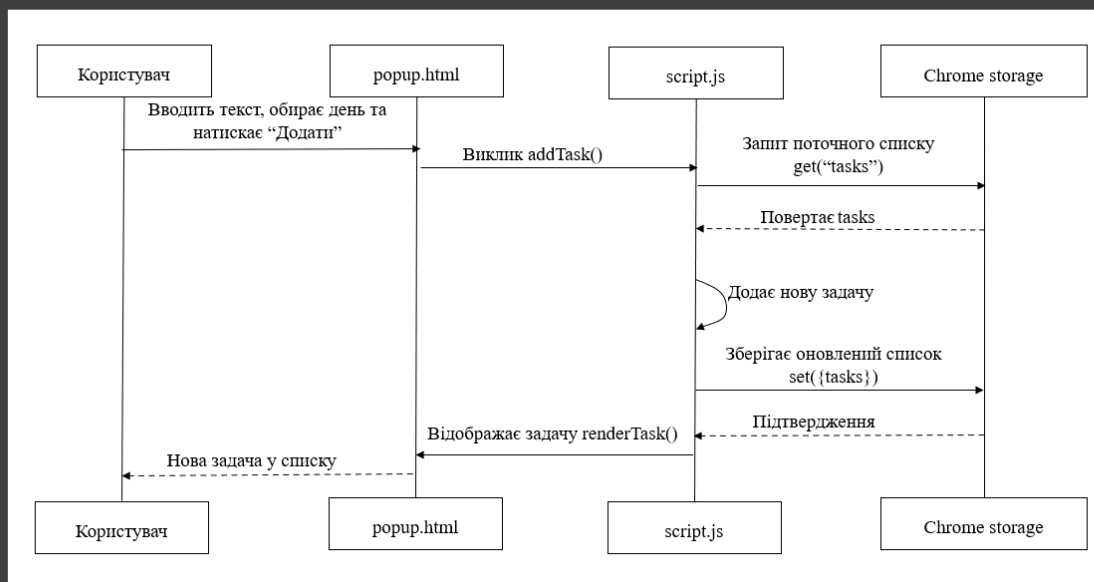
Алгоритм зберігання даних



7

Рисунок Б.7 – Слайд № 7

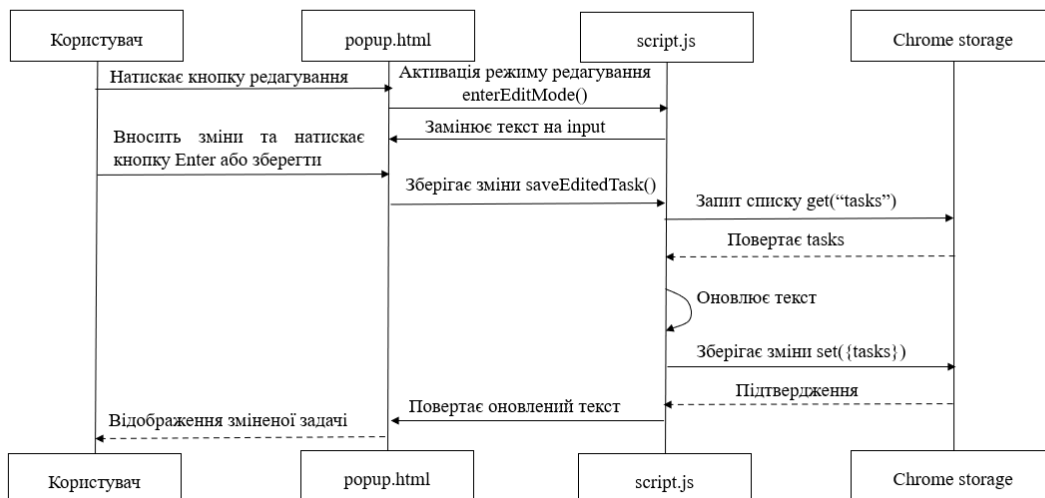
Діаграма послідовності додавання



8

Рисунок Б.8 – Слайд № 8

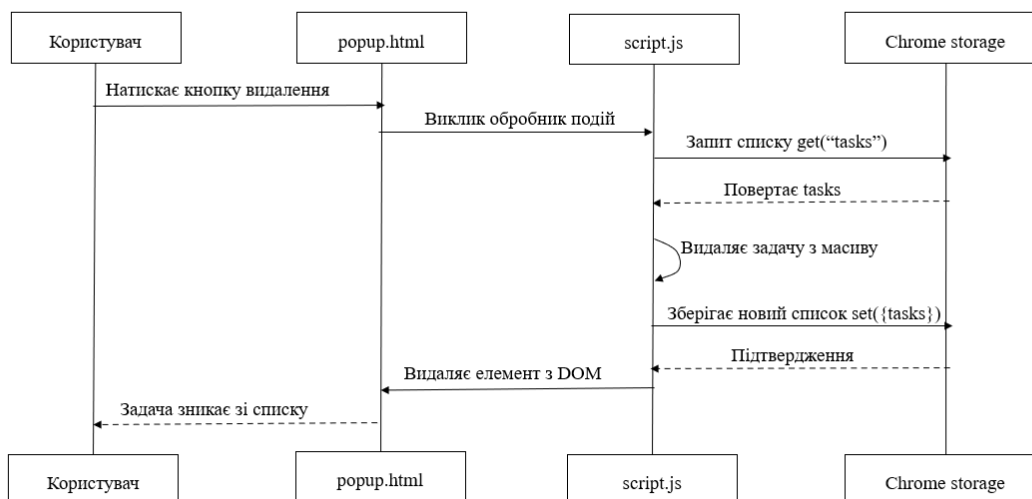
Діаграма послідовності редагування



9

Рисунок Б.9 – Слайд № 9

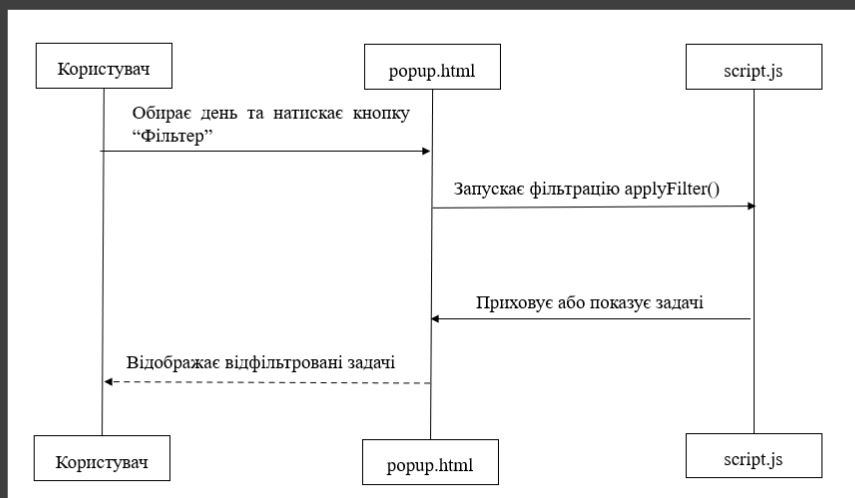
Діаграма послідовності видалення



10

Рисунок Б.10 – Слайд № 10

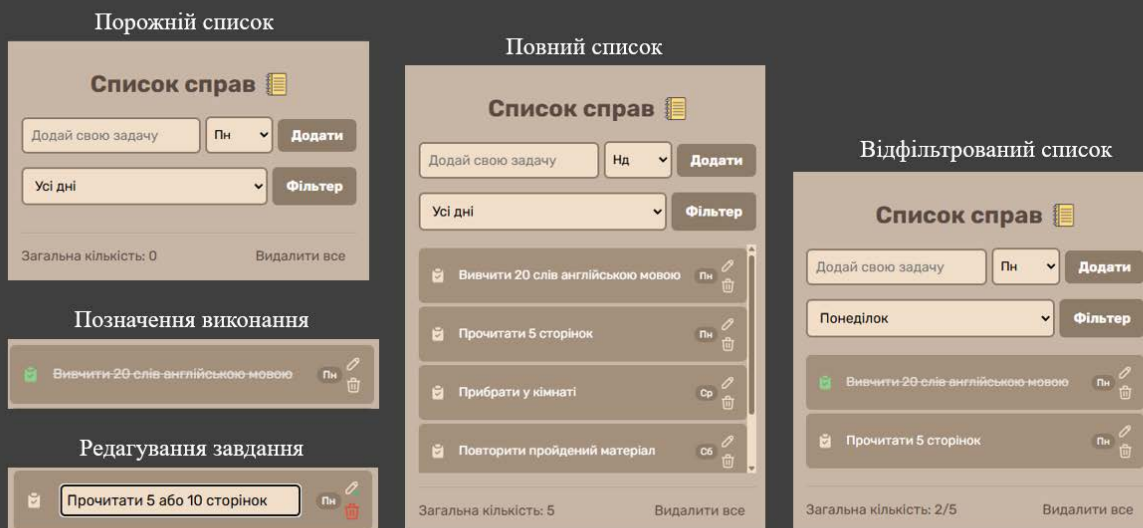
Діаграма послідовності фільтрації



11

Рисунок Б.11 – Слайд № 11

Приклад інтерфейсу



12

Рисунок Б.12 – Слайд № 12

Висновки

- У ході даної дипломної кваліфікаційної роботи було реалізовано браузерне розширення яке надає усі необхідні функції для ведення списку справ, єдине що не вдалось реалізувати на даний момент це прикріплення задачі до конкретної дати календаря. Тестування програми довело, що увесь розроблений функціонал працює коректно.
- Може бути корисним для широкого кола користувачів які хочуть організувати особисті та робочі завдання, підвищити ефективність роботи. Програмний продукт є зручним у використанні, має простий інтерфейс та може стати основою для подальшого розширення функціоналу.
- В подальшому можливо покращити створене розширення за допомогою:
 - Додавання сповіщень про необхідність виконати завдання;
 - Можливість обирати між записом на конкретний день або календарним плануванням;
 - Інтеграція таймера за методом Pomodoro для фокусованої роботи за певним проміжком часу, або інших методів для ефективного управління списками справ.

Рисунок Б.13 – Слайд № 13