МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЧЕРКАСЬКИЙ ДЕРЖАВНИЙ ФАХОВИЙ БІЗНЕС-КОЛЕДЖ
Циклова комісія комп’ютерної інженерії та інформаційних технологій

КВАЛІФІКАЦІЙНА РОБОТА

на тему

ЗАСТОСУНОК БЛОКУВАННЯ ВИБІРКОВИХ ДАНИХ ДЛЯ БАТЬКІВСЬКОГО КОНТРОЛЮ

Виконала: студентка групи 1П-22
Спеціальності
121 Інженерія програмного забезпечення
Олександр ЧЕМЕРИС
Керівник:
Майя ЛЮТА

Черкаси 2026

ЧЕРКАСЬКИЙ ДЕРЖАВНИЙ ФАХОВИЙ БІЗНЕС-КОЛЕДЖ

Циклова комісія комп'ютерної інженерії та інформаційних технологій

Спеціальність 121 «Інженерія програмного забезпечення»

Освітня програма Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ

Завідувачка циклової комісії КІ та ІТ

_____ Наталя ФАЛЬЧЕНКО

«_____» _____ 2025 р.

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Чемерису Олександру Васильовичу

1. Тема кваліфікаційної роботи Застосунок блокування вибіркових даних для батьківського контролю

Керівник роботи Люта Майя В'ячеславівна, викладач вищої категорії затверджені наказом закладу фахової передвищої освіти від «06» жовтня 2025 року № 84/1-у.

2. Строк подання студентом кваліфікаційної роботи 02.06.2026

3. Вихідні дані до кваліфікаційної роботи: Мови розмітки та програмування HTML5, CSS3, JavaScript, технологія локального зберігання даних Local Storage, середовище розробки Visual Studio Code, система контролю версій Git, веббраузери Google Chrome, Mozilla Firefox, Microsoft Edge, засоби моделювання UML (діаграми прецедентів, класів та діяльності).

4. Зміст кваліфікаційної роботи: Вступ. Аналіз предметної області та огляд наявних рішень. Проєктування та програмна реалізація системи. Тестування та впровадження програмного продукту. Висновки

5. Дата видачі завдання 15.09.2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Терміни виконання етапів	Примітка про виконання
1	Вступ	14.10.2025	
2	Розділ 1 Аналіз предметної області та огляд наявних рішень	09.12.2025	
3	Розділ 2 Проєктування та програмна реалізація системи	10.03.2026	
4	Розділ 3 Тестування та впровадження програмного продукту	28.04.2026	
6	Висновки	12.05.2026	
7	Оформлення кваліфікаційної роботи (чистовий варіант)	26.05.2026	
8	Перевірка кваліфікаційної роботи на наявність ознак плагіату (за 10 днів до захисту)	02.06.2026	
9	Подання кваліфікаційної роботи на затвердження завідувачці циклової комісії (за 7 днів до захисту)	10.06.2026	

Студент _____ Олександр ЧЕМЕРИС

Керівник роботи _____ Майя ЛЮТА

АНОТАЦІЯ

У роботі проведено аналіз сучасних засобів батьківського контролю, досліджено існуючі програмні рішення та визначено їх основні переваги й недоліки. Розглянуто принципи проєктування вебзастосунків, методи організації локального зберігання даних та особливості реалізації клієнтських інформаційних систем.

У процесі виконання роботи спроєктовано структуру даних системи, визначено основні сутності та зв’язки між ними, розроблено архітектуру програмного забезпечення і реалізовано механізми керування даними користувача. Для створення програмного продукту використано технології HTML5, CSS3 та JavaScript, а для зберігання інформації застосовано Local Storage.

Розроблений застосунок забезпечує можливість перегляду списку програм, налаштування параметрів доступу до мережевих ресурсів, керування станом блокування та збереження налаштувань користувача. Інтерфейс системи реалізовано відповідно до сучасних вимог зручності використання та забезпечення швидкого доступу до основних функцій.

Для перевірки працездатності програмного продукту проведено функціональне, кросбраузерне та продуктивне тестування. Результати тестування підтвердили коректність роботи застосунку, стабільність зберігання даних та відповідність реалізованого функціоналу поставленим вимогам.

Практичне значення роботи полягає у створенні програмного засобу, який може бути використаний батьками для контролю доступу дітей до окремих застосунків та мережевих ресурсів, сприяючи підвищенню безпеки та ефективності використання цифрових пристроїв.

Ключові слова: БАТЬКІВСЬКИЙ КОНТРОЛЬ, ВЕБЗАСТОСУНОК, БЛОКУВАННЯ ДАНИХ, МЕРЕЖЕВИЙ ДОСТУП, HTML5, CSS3, JAVASCRIPT, LOCAL STORAGE, ІНФОРМАЦІЙНА СИСТЕМА, ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ.

ANNOTATION

The work analyzes modern parental control tools, investigates existing software solutions, and identifies their main advantages and disadvantages. The principles of web application design, methods of local data storage organization, and the features of implementing client-side information systems are considered.

During the implementation of the project, the data structure of the system was designed, the main entities and relationships between them were defined, the software architecture was developed, and mechanisms for user data management were implemented. HTML5, CSS3, and JavaScript technologies were used to develop the software product, while Local Storage was employed for data persistence.

The developed application provides the ability to view the list of applications, configure network access parameters, manage blocking settings, and store user preferences. The system interface was designed according to modern usability requirements and ensures quick access to the main functions.

To verify the functionality of the software product, functional, cross-browser, and performance testing were carried out. The testing results confirmed the correct operation of the application, the stability of data storage, and the compliance of the implemented functionality with the specified requirements.

The practical significance of the work lies in the creation of a software tool that can be used by parents to control children's access to specific applications and network resources, thereby improving safety and efficiency in the use of digital devices.

Keywords: PARENTAL CONTROL, WEB APPLICATION, DATA BLOCKING, NETWORK ACCESS, HTML5, CSS3, JAVASCRIPT, LOCAL STORAGE, INFORMATION SYSTEM, SOFTWARE TESTING.

ЗМІСТ

ВСТУП

У сучасному суспільстві цифрові технології стали невід'ємною складовою повсякденного життя. Мобільні пристрої, комп'ютери та мережа Інтернет використовуються для навчання, комунікації, отримання інформації, розваг та професійної діяльності. Особливо активно цифрові сервіси використовуються дітьми та підлітками, які починають взаємодіяти з електронними пристроями з раннього віку. За результатами сучасних досліджень, більшість підлітків мають постійний доступ до Інтернету та щоденно користуються соціальними мережами, онлайн-сервісами й мобільними застосунками [1].

Разом із перевагами цифрового середовища виникають нові виклики щодо безпеки дітей в Інтернеті. Неконтрольований доступ до онлайн-ресурсів може призводити до ознайомлення з небажаним контентом, кібербулінгу, шахрайства, надмірного використання соціальних мереж та комп’ютерних ігор. Також актуальними залишаються проблеми залежності від цифрових пристроїв, порушення режиму сну, зниження фізичної активності та концентрації уваги.

Для зменшення впливу зазначених ризиків використовуються системи батьківського контролю, які дозволяють контролювати доступ дітей до мережевих ресурсів, обмежувати використання окремих застосунків, встановлювати часові ліміти роботи з пристроєм та здійснювати моніторинг цифрової активності. Сучасні програмні рішення у сфері батьківського контролю надають широкий спектр можливостей для забезпечення безпечного використання цифрових технологій дітьми та підлітками [8–11].

Однією з найпоширеніших мобільних платформ у світі є операційна система Android. Завдяки відкритій архітектурі, широкому спектру пристроїв та великій кількості користувачів вона займає провідні позиції на ринку мобільних операційних систем [2]. Популярність платформи Android робить її перспективним середовищем для розроблення застосунків батьківського контролю та систем моніторингу цифрової активності.

Актуальність теми полягає у необхідності створення програмних засобів, які дозволяють підвищити рівень цифрової безпеки дітей, забезпечити контроль

використання мережевих ресурсів та допомогти батькам у формуванні безпечного цифрового середовища. Розробка застосунку блокування вибіркових даних для батьківського контролю сприятиме обмеженню доступу до небажаних ресурсів і підвищенню ефективності контролю використання мобільних пристроїв.

Метою кваліфікаційної роботи є проєктування та розробка застосунку блокування вибіркових даних для батьківського контролю, що забезпечує обмеження доступу до мережевих ресурсів та контроль використання застосунків на мобільних пристроях.

Об’єктом дослідження є процес забезпечення безпечного використання мобільних пристроїв та мережі Інтернет дітьми.

Предметом дослідження є методи, засоби та технології розроблення застосунків батьківського контролю для платформи Android.

Для досягнення поставленої мети необхідно виконати такі завдання:

– провести аналіз предметної області та дослідити сучасні проблеми цифрової безпеки дітей;

– виконати огляд існуючих програмних засобів батьківського контролю та визначити їх переваги і недоліки;

– дослідити технології та інструменти розроблення мобільних застосунків для операційної системи Android;

– сформувати функціональні та нефункціональні вимоги до програмного продукту;

– спроєктувати архітектуру застосунку та структуру зберігання даних;

– реалізувати механізми керування доступом до мережевих ресурсів і налаштуваннями блокування;

– виконати тестування програмного продукту та проаналізувати результати його роботи;

– оцінити можливості подальшого розвитку та вдосконалення системи.

Практичне значення роботи полягає у створенні програмного засобу, який може використовуватися батьками для контролю доступу дітей до окремих

застосунків та мережевих ресурсів, що сприятиме підвищенню рівня інформаційної безпеки та формуванню безпечної цифрової поведінки.

У процесі виконання роботи використовувалися методи аналізу та синтезу інформації, методи проєктування програмного забезпечення, технології розроблення мобільних застосунків, а також методи функціонального та продуктивного тестування програмних систем.

РОЗДІЛ 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ОГЛЯД НАЯВНИХ РІШЕНЬ

1.1 Розвиток мобільної ОС Android

Android – це мобільна операційна система з відкритим вихідним кодом, розроблена компанією Google на основі ядра Linux. Платформа була створена з метою забезпечення гнучкого та доступного середовища для мобільних пристроїв, що дозволило виробникам адаптувати систему під різні типи обладнання. Історія Android розпочалася у 2003 році зі створення компанії Android Inc., яка займалася розробкою операційної системи для мобільних пристроїв. У 2005 році компанія була придбана Google, після чого почався активний розвиток платформи. Перший комерційний смартфон на базі Android – HTC Dream (T-Mobile G1) – був представлений у жовтні 2008 року, що стало початком широкого поширення Android серед користувачів [3].

Одним із ключових факторів успіху Android стала відкрита модель розповсюдження. Google надала виробникам мобільних пристроїв можливість безкоштовного використання операційної системи, що сприяло її швидкому впровадженню серед великої кількості OEM-виробників у всьому світі. На відміну від закритої екосистеми Apple, Android дозволив створювати широкий спектр пристроїв різних цінових категорій – від бюджетних смартфонів до флагманських моделей.

Станом на 2026 рік Android залишається провідною мобільною операційною системою у світі. Її частка на глобальному ринку мобільних ОС становить понад 70%, що забезпечує платформі найбільшу кількість активних користувачів серед усіх мобільних систем [2]. Основними причинами такого поширення є відкритість платформи, підтримка великої кількості виробників, різноманітність апаратних конфігурацій та доступність пристроїв для різних категорій користувачів. Основним конкурентом Android є операційна система iOS компанії Apple, яка використовується виключно на пристроях власного виробництва.

Протягом свого розвитку Android пройшов значну еволюцію — від простої мобільної платформи до повноцінної екосистеми, яка підтримує смартфони, планшети, телевізори, автомобільні системи та інші пристрої. Перші версії Android (1.0–2.3, 2008–2010) сформували основні компоненти екосистеми: магазин застосунків Android Market, інтеграцію із сервісами Google, підтримку мультитач-жестів та бездротових технологій.

Версії Android 4.x (2011–2013) стали важливим етапом розвитку платформи, оскільки об’єднали інтерфейс смартфонів і планшетів та представили дизайн-концепцію Holo. У 2014 році з виходом Android 5.0 Lollipop було впроваджено Material Design – новий підхід до створення інтерфейсів, який став основою подальшого розвитку дизайну застосунків.

Версії Android 6.0–9.0 (2015–2018) були спрямовані на підвищення продуктивності, оптимізацію роботи системи та посилення безпеки. У цей період було вдосконалено систему дозволів застосунків, що стало важливим елементом захисту персональних даних користувачів. Android 10 (2019) представив загальносистемний темний режим, нову систему жестової навігації та подальше посилення контролю конфіденційності.

Сучасні версії Android орієнтовані на підвищення рівня безпеки, конфіденційності та інтеграцію нових технологій. Android 12 (2021) представив концепцію Material You, яка дозволила користувачам персоналізувати зовнішній вигляд системи. Android 13–14 (2022–2023) розширили можливості керування дозволами застосунків, покращили захист даних та оптимізували роботу системи. Android 15 (2024) додав функцію Private Space, яка дозволяє ізолювати конфіденційні застосунки та інформацію користувача.

Android 16 (2025) продовжив розвиток напрямів безпеки, продуктивності та інтеграції сучасних сервісів. Оновлення включило покращені механізми захисту пристрою, нові можливості керування сповіщеннями та оптимізацію роботи застосунків. Подальший розвиток Android спрямований на підтримку штучного інтелекту, складних форм-факторів пристроїв та підвищення рівня контролю користувача над персональними даними.

Однією з головних проблем Android залишається фрагментація операційної системи. Велика кількість виробників використовує власні версії Android із різними інтерфейсами та налаштуваннями, а швидкість отримання оновлень залежить від конкретного виробника пристрою. Це створює додаткові труднощі для розробників, які повинні забезпечувати стабільну роботу застосунків на великій кількості моделей смартфонів із різними характеристиками.

Для часткового вирішення проблеми фрагментації Google впровадила технології Project Treble та Project Mainline, які дозволяють швидше оновлювати окремі компоненти системи без повного оновлення операційної системи. Також провідні виробники мобільних пристроїв збільшують терміни підтримки програмного забезпечення, що позитивно впливає на безпеку користувачів.

Відкритість Android та можливість взаємодії із системними API роблять цю платформу перспективною для створення застосунків батьківського контролю. На відміну від iOS, Android надає розробникам ширші можливості доступу до системних функцій, зокрема роботи з дозволами, геолокацією, керуванням застосунками та аналізом активності користувача.

Саме ці можливості є важливими для розробки систем батьківського контролю, оскільки дозволяють реалізувати функції моніторингу використання пристрою, обмеження доступу до певних ресурсів, контролю встановлення застосунків та забезпечення безпечного цифрового середовища для дітей.

1.2 Огляд існуючих рішень

З розвитком мобільних технологій та активним використанням смартфонів дітьми зросла потреба у створенні програмних засобів батьківського контролю. Такі системи дозволяють забезпечити безпечне використання цифрових пристроїв, контролювати активність у мережі, обмежувати доступ до небажаного контенту та формувати безпечне цифрове середовище.

Сучасні застосунки батьківського контролю для Android зазвичай реалізують такі основні можливості:

– фільтрація вебконтенту та блокування небезпечних сайтів;

– контроль встановлення та використання мобільних застосунків;

– обмеження часу використання смартфона та окремих програм;

– відстеження місцезнаходження пристрою;

– створення геозон із повідомленнями про переміщення дитини;

– аналіз цифрової активності та формування звітів;

– контроль покупок у магазинах застосунків;

– надсилання екстрених повідомлень у разі небезпечної ситуації.

Більшість сучасних сервісів працюють за моделлю підписки. Безкоштовні версії зазвичай мають базові функції, тоді як розширені можливості, такі як детальна аналітика, GPS-моніторинг або контроль соціальних мереж, доступні у платних тарифах.

Найбільш поширеними платформами для таких застосунків залишаються Android та iOS. Проте функціональні можливості можуть відрізнятися через різницю в політиках доступу до системних компонентів. Android завдяки відкритішій архітектурі дозволяє реалізувати більше функцій контролю, оскільки розробники мають ширший доступ до системних API, дозволів та служб пристрою.

Одним із найбільш поширених рішень є Google Family Link (рис. 1.1), розроблений компанією Google. Застосунок інтегрований з екосистемою Android та призначений для керування дитячими обліковими записами Google.

Основні можливості Google Family Link:

– контроль часу використання пристрою;

– перегляд встановлених застосунків;

– підтвердження або блокування встановлення програм;

– налаштування вікових обмежень;

– перегляд приблизного місцезнаходження пристрою;

– керування параметрами цифрової безпеки.

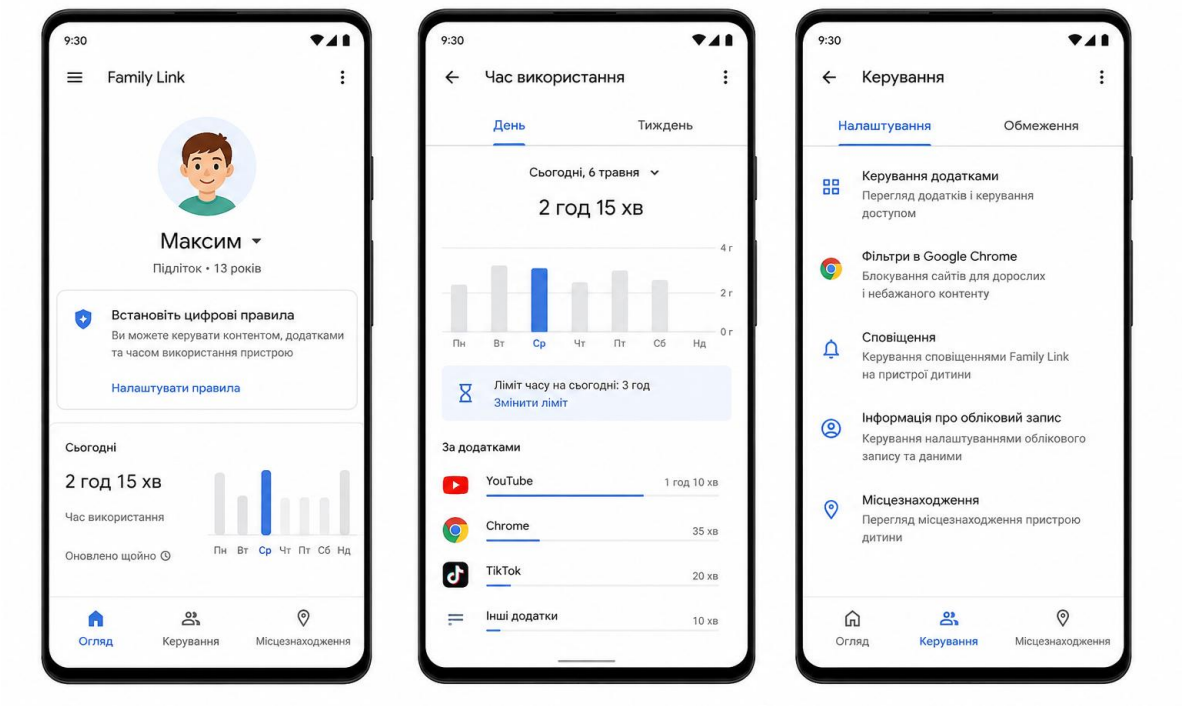

Рисунок 1.1 – Інтерфейс застосунку Google Family Link

Перевагою Google Family Link є безкоштовність та глибока інтеграція з Android. Недоліком є обмежений рівень розширеного моніторингу активності, оскільки система орієнтована переважно на базовий контроль часу, застосунків та контенту.

Наступним сучасним рішенням є Bark (рис. 1.2) – платформа батьківського контролю, яка використовує технології аналізу даних та штучного інтелекту для виявлення потенційних цифрових загроз. На відміну від традиційних систем, які переважно блокують контент, Bark робить акцент на попередженні батьків про небезпечні ситуації. Система аналізує активність у повідомленнях, соціальних мережах, пошуку та інших цифрових сервісах і надсилає сповіщення лише у випадку виявлення потенційної проблеми.

Основні можливості Bark:

– аналіз текстових повідомлень, електронної пошти та активності у підтримуваних сервісах;

– виявлення ознак кібербулінгу, небезпечного спілкування, загроз та неприйнятного контенту;

– контроль часу використання пристрою;

– блокування вебсайтів та застосунків;

– GPS-моніторинг та повідомлення про зміну місцезнаходження;

– формування звітів для батьків.

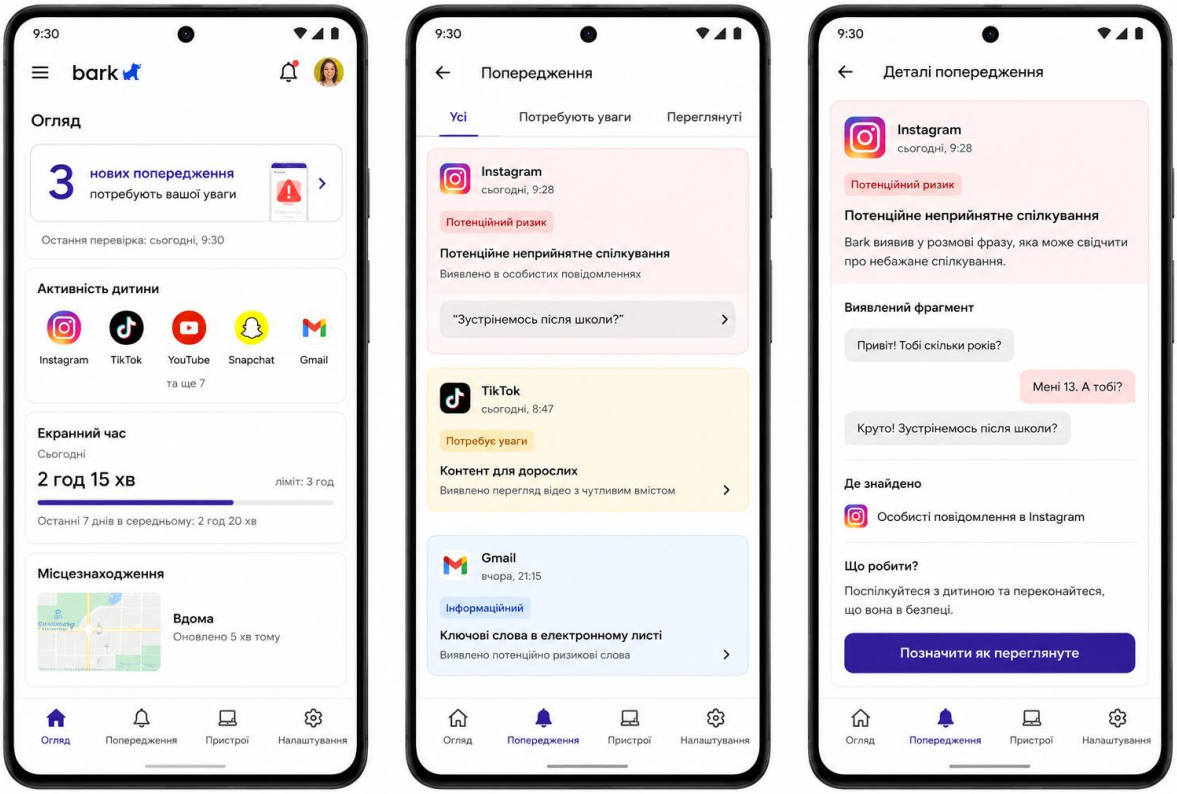

Рисунок 1.2 – Приклад системи попереджень у застосунку Bark

Особливістю Bark є використання моделі «розумного моніторингу». Програма не намагається показувати батькам усю активність дитини, а аналізує інформацію та повідомляє лише про ситуації, які можуть становити ризик. Такий підхід дозволяє зменшити надмірний контроль та зосередитися саме на питаннях безпеки.

Перевагами Bark є орієнтація на сучасні цифрові загрози, підтримка Android та можливість аналізу великої кількості типів онлайн-активності. Недоліком є залежність функціональності від доступних дозволів системи та необхідність використання платної версії для отримання повного набору можливостей.

Ще одним популярним рішенням є Norton Family (рис. 1.3), яке орієнтоване на контроль вебактивності та формування звітів використання пристрою.

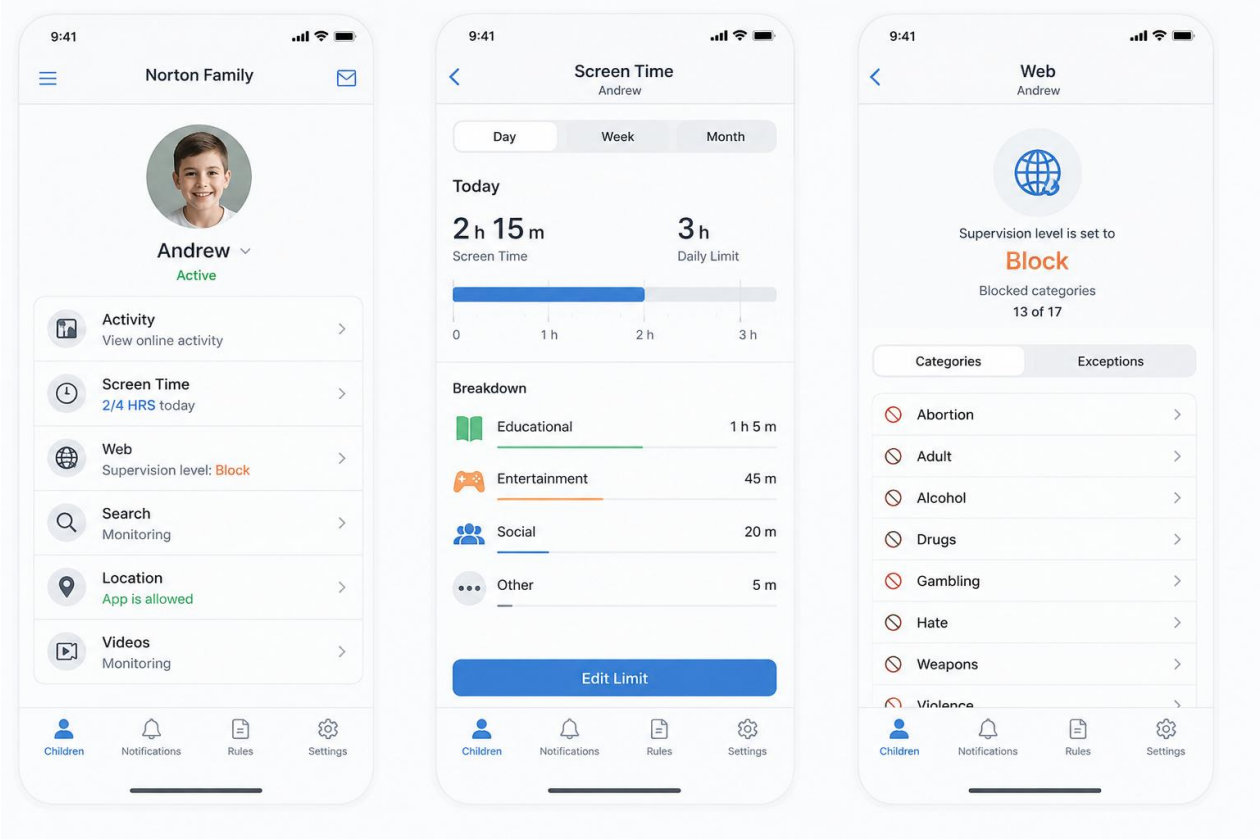

Рисунок 1.3 – Інтерфейс застосунку батьківського контролю Norton Family

Основні функції Norton Family:

– контроль відвіданих вебсайтів;

– фільтрація небажаного контенту;

– створення звітів активності;

– встановлення часових обмежень;

– повідомлення батькам про порушення встановлених правил.

Перевагою Norton Family є стабільна робота та інтеграція з іншими продуктами кібербезпеки. Недоліком є менша кількість функцій аналізу соціальної активності порівняно з сучасними AI-орієнтованими рішеннями.

Одним із найбільш функціональних застосунків є Qustodio (рис. 1.4), який поєднує традиційні можливості батьківського контролю із розширеною аналітикою.

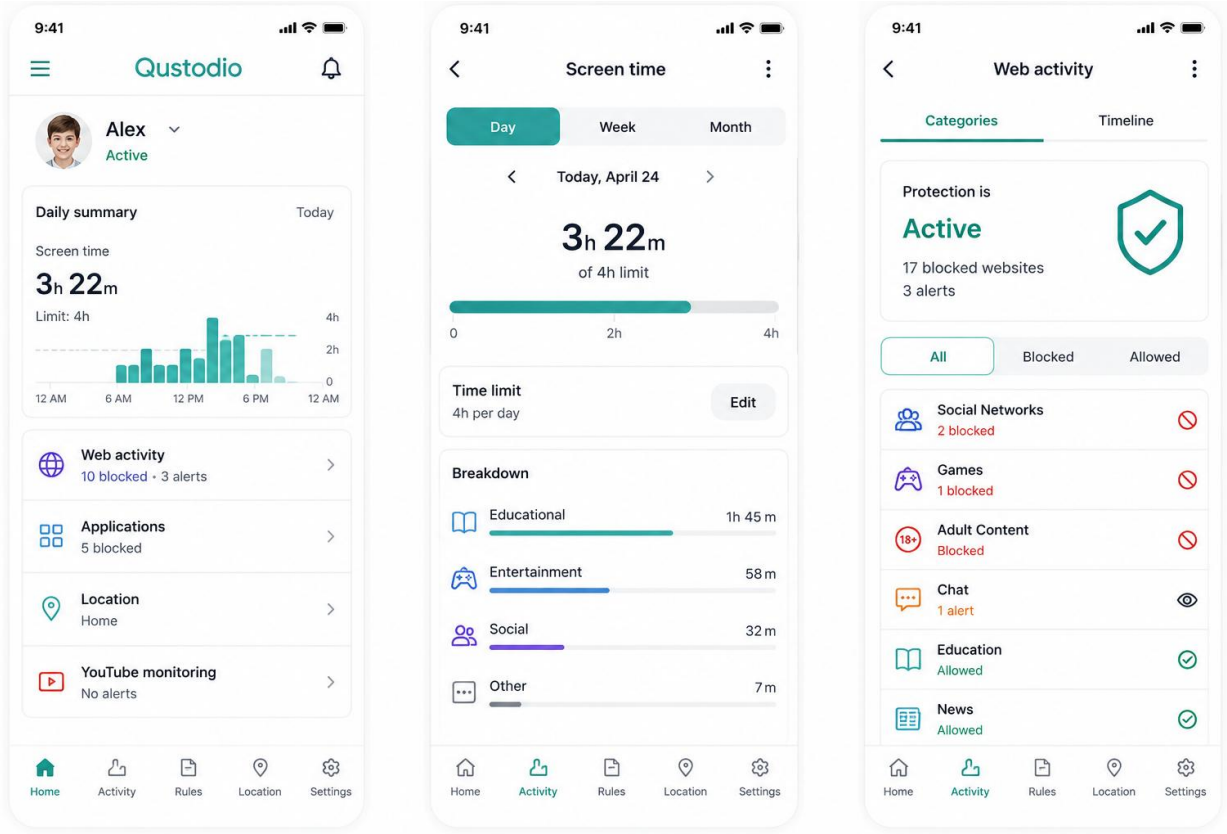

Рисунок 1.4 – Інтерфейс застосунку батьківського контролю Qustodio

Основні можливості Qustodio:

– контроль часу використання пристрою;

– блокування застосунків;

– GPS-відстеження;

– перегляд історії активності;

– вебфільтрація;

– функція SOS для екстреного зв’язку.

Перевагою Qustodio є широкий функціонал та зручна система звітів. Недоліком є те, що більшість розширених можливостей доступна лише у платній версії.

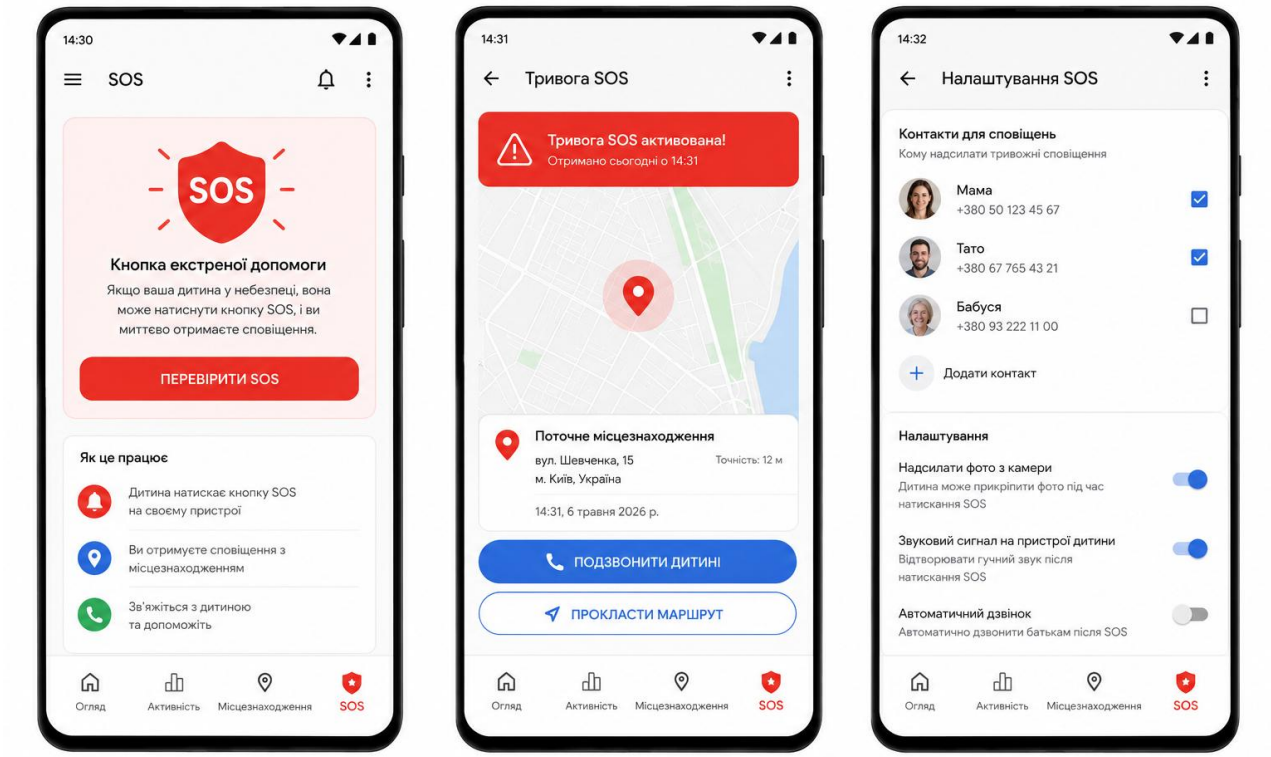

Рисунок 1.5 – Приклад реалізації функції SOS у застосунках батьківського контролю

Порівняння основних сучасних рішень наведено у таблиці 1.1.

Таблиця 1.1 – Порівняння застосунків батьківського контролю

Застосунок	Основні можливості	Недоліки
Google Family Link	контроль часу, застосунків, базова геолокація	обмежений моніторинг активності
Bark	AI-аналіз ризиків, соціальні мережі, попередження про загрози	частина функцій платна
Norton Family	вебконтроль, звіти, обмеження часу	менше можливостей аналізу контенту
Qustodio	GPS, SOS, контроль застосунків, аналітика	висока вартість повної версії

Проведений аналіз показує, що сучасні рішення батьківського контролю поступово переходять від простого блокування контенту до комплексного аналізу цифрової поведінки користувача. Нові системи використовують штучний інтелект для виявлення потенційних загроз, що дозволяє підвищити рівень безпеки дітей в Інтернеті.

Водночас існуючі рішення мають певні обмеження: залежність від політик мобільних платформ, платний доступ до розширених функцій та недостатню гнучкість налаштувань. Саме тому розробка власного Android-застосунку батьківського контролю є актуальним завданням, яке дозволить поєднати необхідні функції безпеки та адаптувати систему під конкретні потреби користувачів.

1.3 Постановка задачі

У межах даної кваліфікаційної роботи планується розроблення мобільного застосунку батьківського контролю для платформи Android, призначеного для забезпечення безпечного використання цифрових пристроїв дітьми. Основною метою розробки є створення програмного рішення, яке дозволить батькам контролювати цифрову активність дитини, обмежувати доступ до небажаного контенту та формувати безпечне інформаційне середовище.

Під час виконання роботи необхідно вирішити такі основні завдання:

– провести аналіз предметної області та дослідити сучасні проблеми безпечного використання мобільних пристроїв дітьми;

– виконати аналіз існуючих програмних рішень батьківського контролю, визначити їх переваги та недоліки;

– дослідити сучасні технології та інструменти розробки мобільних застосунків для операційної системи Android;

– визначити функціональні та нефункціональні вимоги до системи батьківського контролю;

– розробити архітектуру застосунку та визначити основні компоненти програмної системи;

– реалізувати функціональні можливості застосунку, зокрема:

– блокування небажаного вебконтенту та обмеження доступу до небезпечних ресурсів;

– контроль часу використання пристрою та окремих застосунків;

– перегляд інформації про активність користувача;

– реалізацію механізмів керування доступом до програм;

– створення зручного та зрозумілого графічного інтерфейсу для комфортної взаємодії користувача із системою;

– забезпечити захист персональних даних та безпечну роботу застосунку;

– провести тестування розробленого програмного рішення, перевірити його стабільність та відповідність визначеним вимогам;

– виконати аналіз продуктивності програмного коду, визначити можливі проблемні місця та провести оптимізацію у разі необхідності.

Для виконання поставлених завдань необхідно провести:

– аналіз сучасних інструментів і технологій розробки Android-застосунків;

– аналіз існуючих систем батьківського контролю та їх функціональних можливостей;

– вибір оптимальних програмних засобів та підходів для реалізації системи;

– проєктування структури застосунку та його основних модулів;

– розроблення та тестування програмного продукту.

Результатом виконання роботи має стати програмний застосунок батьківського контролю, який забезпечує базові функції контролю цифрової активності дитини та може бути використаний як основа для подальшого розвитку системи.

РОЗДІЛ 2 ПРОЄКТУВАННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ

2.1 Розробка архітектури додатку

Розробка архітектури мобільного застосунку передбачає визначення основних компонентів системи, способів їх взаємодії, структури зберігання даних та принципів обміну інформацією між окремими модулями. Від правильно обраної архітектури залежить масштабованість, продуктивність, безпека та можливість подальшого розвитку програмного продукту.

Для мобільних застосунків існує декілька основних архітектурних підходів: файл-серверна, клієнт-серверна, багаторівнева, веборієнтована та безсерверна архітектури.

Файл-серверна архітектура є одним із найпростіших варіантів побудови програмних систем. У такому підході клієнт безпосередньо працює з файлами або локальною базою даних. Основними недоліками цього підходу є:

– високе навантаження на мережу;

– складність масштабування при збільшенні кількості користувачів;

– недостатній рівень безпеки;

– складність контролю цілісності даних.

Через зазначені недоліки файл-серверна архітектура не є оптимальним рішенням для сучасних мобільних застосунків, які потребують стабільного обміну даними та захисту інформації.

Більш поширеною є клієнт-серверна архітектура, у якій система розподіляється між клієнтською частиною та серверною частиною. Клієнт відповідає за взаємодію з користувачем, а сервер виконує обробку запитів, зберігання даних та виконання основної бізнес-логіки.

Клієнт-серверна архітектура включає такі компоненти:

– клієнтський мобільний застосунок Android;

– серверну частину для обробки запитів;

– базу даних для збереження інформації;

– мережевий рівень взаємодії між компонентами.

Перевагами такого підходу є:

– можливість підтримки великої кількості користувачів;

– централізоване керування даними;

– підвищена безпека;

– можливість масштабування системи.

Ще одним сучасним підходом є безсерверна архітектура (Serverless), у якій розробник використовує готові хмарні сервіси для автентифікації, зберігання даних та виконання серверних функцій. Такий підхід дозволяє зменшити витрати на підтримку інфраструктури, однак залежність від сторонніх сервісів може бути недоліком при розробці автономних мобільних систем.

Після аналізу існуючих архітектур було обрано клієнт-серверну модель із використанням сучасного підходу MVVM (Model-View-ViewModel). Дана архітектура дозволяє розділити логіку застосунку на незалежні компоненти та спростити подальше тестування і розширення функціоналу.

Для розробки мобільного застосунку батьківського контролю «NetBlock» було обрано такі технології:

– Android Studio як основне середовище розробки;

– мова програмування Kotlin;

– Android SDK та Jetpack-компоненти;

– архітектура MVVM;

– локальна база даних Room Database;

– механізм Android VPN Service для контролю мережевого доступу.

Android Studio є офіційним інтегрованим середовищем розробки Android-застосунків. Воно містить інструменти для створення інтерфейсу, компіляції, тестування та аналізу продуктивності програмного забезпечення.

Для перевірки роботи застосунку використовується Android Emulator або фізичний пристрій. Середовище дозволяє моделювати різні версії Android, перевіряти сумісність та виконувати тестування функціоналу.

Основною функцією розроблюваного застосунку є контроль доступу встановлених програм до мережі Інтернет. Для реалізації цього механізму використовується Android VPN Service, який створює локальний VPN-канал та дозволяє аналізувати мережеві запити програм без необхідності отримання root-доступу.

Логіка роботи системи полягає у наступному:

– користувач запускає застосунок;

– система запитує дозвіл на створення VPN-з'єднання;

– після підтвердження запускається служба контролю мережевого трафіку;

– застосунок отримує список встановлених програм;

– користувач може визначити, яким програмам дозволити або заборонити доступ до Інтернету;

– налаштування застосовуються в режимі реального часу.

Структура роботи застосунку представлена на рисунку 2.1.

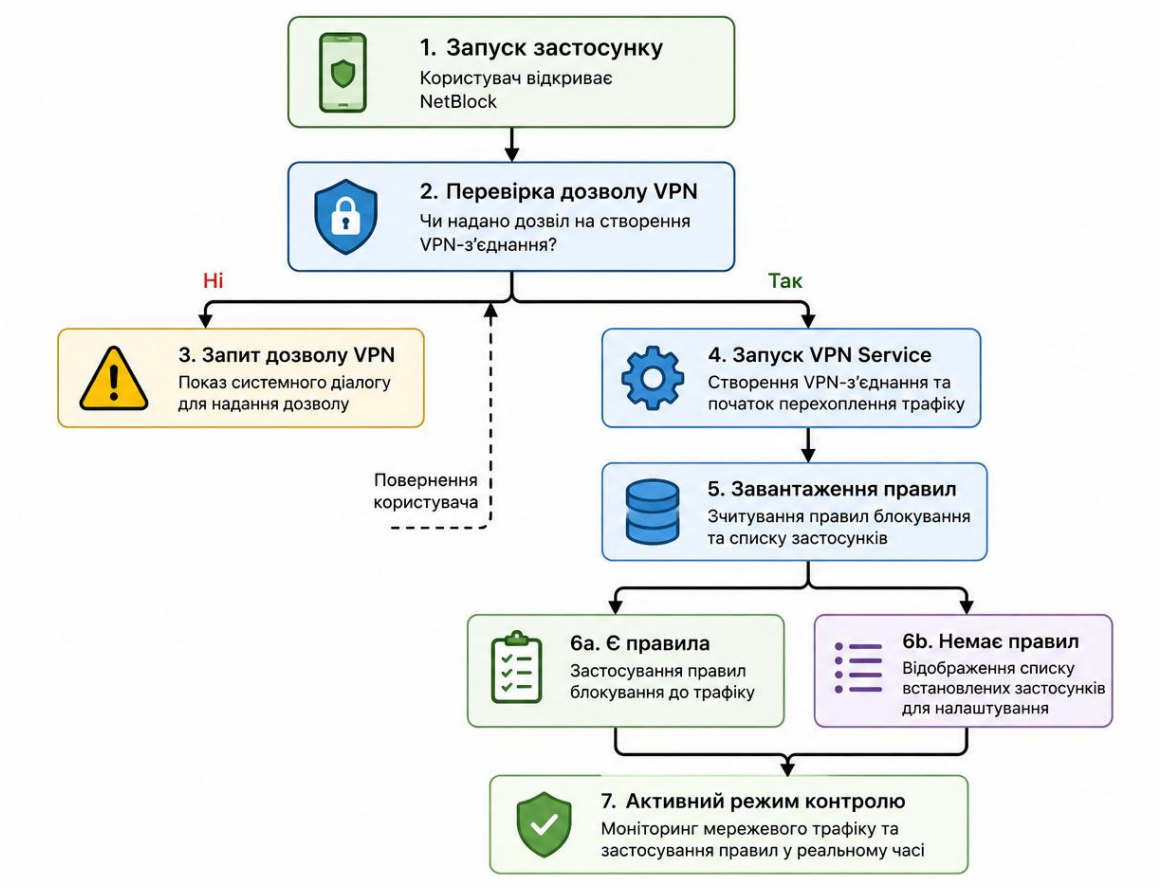

Рисунок 2.1 – Діаграма станів процесу запуску застосунку NetBlock

Після успішного запуску VPN-служби система переходить у режим активного контролю. Якщо користувач попередньо налаштував правила доступу, вони автоматично застосовуються. Якщо правила відсутні, користувач отримує список встановлених програм для подальшого налаштування.

Обрана архітектура забезпечує необхідний рівень продуктивності, безпеки та можливість подальшого розширення функціоналу застосунку батьківського контролю.

2.2 Проєктування структури бази даних

Під час розробки вебзастосунку важливим етапом є проєктування структури бази даних, яка забезпечує надійне зберігання інформації та можливість виконання операцій над нею. На цьому етапі визначаються сутності предметної області, їх атрибути та взаємозв’язки. Правильно спроєктована структура даних забезпечує цілісність інформації, мінімізує дублювання даних та спрощує подальшу підтримку програмного продукту.

Для взаємодії із сучасними системами керування базами даних найчастіше використовується мова SQL (Structured Query Language), яка є міжнародним стандартом для роботи з реляційними базами даних. SQL дозволяє створювати, модифікувати та видаляти структури даних, а також виконувати операції пошуку, оновлення та видалення записів. Сучасні стандарти SQL постійно розвиваються та підтримуються більшістю популярних СУБД.

У розроблюваному застосунку для зберігання даних використовується вбудоване браузерне сховище Local Storage API. Дана технологія забезпечує локальне збереження інформації у вигляді пар «ключ–значення» без необхідності використання окремого сервера баз даних. Дані, збережені в Local Storage, залишаються доступними після закриття браузера або перезавантаження сторінки, що дозволяє забезпечити довготривале зберігання налаштувань та службової інформації користувача.

Використання Local Storage має ряд переваг:

– зберігання даних без підключення до сервера;

– відсутність передачі збережених даних у кожному HTTP-запиті;

– простий програмний інтерфейс доступу через JavaScript;

– підтримка всіма сучасними веббраузерами;

– можливість зберігати значно більший обсяг даних порівняно з Cookie.

На відміну від файлів Cookie, які автоматично передаються серверу під час кожного HTTP-запиту та мають обмежений обсяг зберігання, Local Storage працює виключно на стороні клієнта, що підвищує продуктивність вебзастосунку та зменшує мережевий трафік.

Для організації даних використовується логічна реляційна модель. Незважаючи на те, що Local Storage фізично не є реляційною базою даних, дані можуть бути представлені у вигляді структурованих JSON-об’єктів, що дозволяє реалізувати логіку, подібну до таблиць реляційної БД.

Рисунок 2.2 демонструє архітектуру зберігання даних у вебзастосунку. Взаємодія користувача із системою здійснюється через вебінтерфейс, реалізований за допомогою HTML та CSS. Обробка введених даних, виконання бізнес-логіки та формування запитів до сховища покладаються на JavaScript-модуль. Для збереження інформації використовується Local Storage, що забезпечує довготривале зберігання даних без необхідності використання серверної бази даних. Такий підхід дозволяє підвищити швидкодію застосунку та зменшити навантаження на мережу.

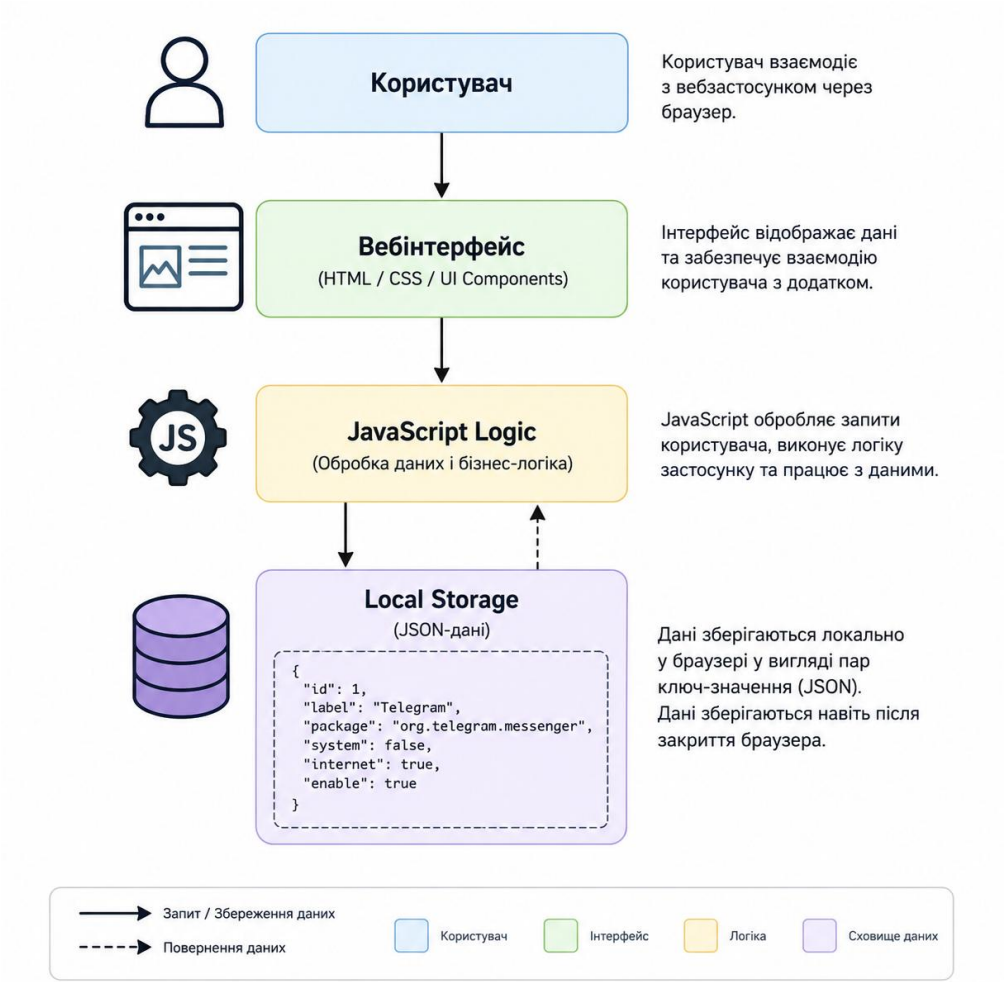

Рисунок 2.2 – Архітектура зберігання даних у вебзастосунку

У межах системи формується колекція даних про застосунки, для яких здійснюється контроль використання мережевих ресурсів. Структура запису наведена в таблиці 2.1.

Таблиця 2.1 – Структура об'єкта застосунку

Об'єкт	Властивість	Тип даних	Ідентифікатор
Додаток	первинний ключ	Number	id
	назва	String	label
	пакет застосунку	String	package
	системний застосунок	Boolean	system
	використовує Інтернет	Boolean	internet
	статус блокування	Boolean	enable

Після запуску застосунку виконується перевірка наявних програм, які використовують мережеве підключення. Отримана інформація зберігається у локальному сховищі браузера та використовується для подальшого аналізу і керування доступом до мережі. Користувач може обрати застосунки, для яких необхідно обмежити доступ до Інтернету. Після вибору відповідні записи оновлюються у сховищі, а інтерфейс користувача відображає актуальний перелік заблокованих програм.

На рисунку 2.3 представлена схема взаємодії застосунку з Local Storage. Після виконання дій користувачем інтерфейс передає запит до програмної логіки, реалізованої засобами JavaScript API. За допомогою методів getItem(), setItem() та removeItem() здійснюється читання, збереження та видалення даних у локальному сховищі браузера. Отримані дані обробляються та повертаються до інтерфейсу для подальшого відображення користувачу. У сховищі інформація зберігається у форматі JSON, що забезпечує простоту структурування та швидкий доступ до записів.

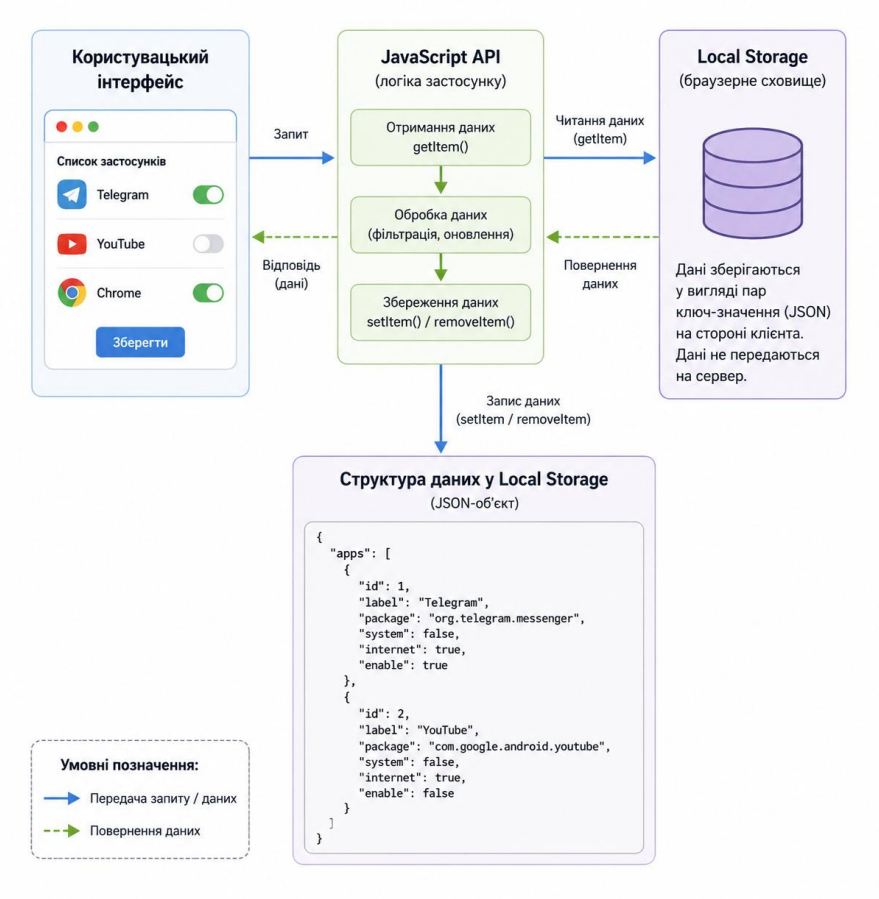

Рисунок 2.3 – Схема взаємодії застосунку з Local Storage

Рисунок 2.4 відображає ER-діаграму структури даних вебзастосунку. Основною сутністю системи є об'єкт «Додаток», який містить інформацію про програму, її ідентифікатор, назву, пакет, а також параметри доступу до мережі. Первинний ключ id забезпечує унікальну ідентифікацію кожного запису. Поля system, internet та enable мають логічний тип даних і використовуються для зберігання службових характеристик застосунку. Запропонована структура є достатньою для реалізації механізму обліку та контролю доступу програм до мережевих ресурсів у межах розроблюваної системи.

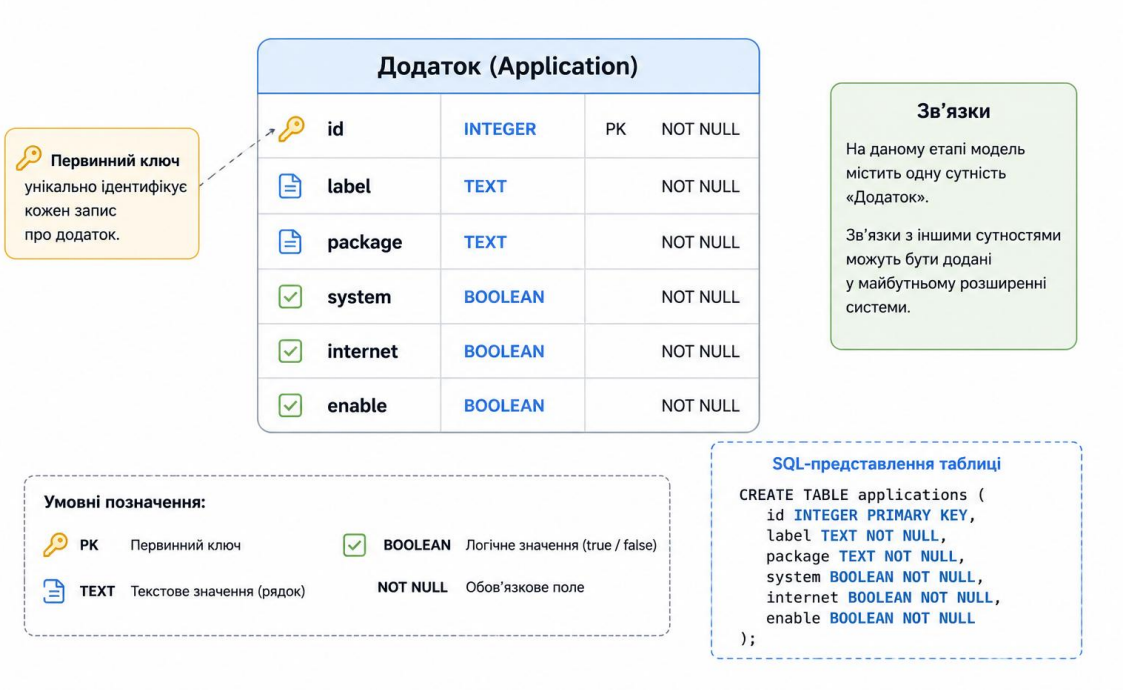

Рисунок 2.4 – ER-діаграма структури об'єкта «Додаток»

Таким чином, використання Local Storage дозволяє реалізувати простий та ефективний механізм локального зберігання даних без необхідності розгортання окремого сервера баз даних, що є доцільним для невеликих вебзастосунків та навчальних проєктів.

2.3 Програмна реалізація проєкту

Для реалізації проєкту було обрано сучасний стек вебтехнологій, що включає HTML5, CSS3 та JavaScript ES6. Такий підхід дозволяє створити кросплатформний застосунок, який може працювати в будь-якому сучасному

веббраузері без необхідності встановлення додаткового програмного забезпечення.

Основою інтерфейсу користувача є мова розмітки HTML5, яка забезпечує структурування елементів сторінки. Для оформлення зовнішнього вигляду застосунку використовується CSS3, що дозволяє реалізувати адаптивний дизайн та забезпечити коректне відображення інтерфейсу на різних типах пристроїв. Логіка роботи застосунку реалізована засобами JavaScript, який відповідає за обробку дій користувача, управління даними та взаємодію з локальним сховищем браузера.

Розробка програмного продукту виконувалася у середовищі Visual Studio Code. Дане середовище розробки підтримує сучасні вебтехнології, забезпечує зручне редагування коду, автоматичне форматування, налагодження програм та інтеграцію із системами контролю версій.

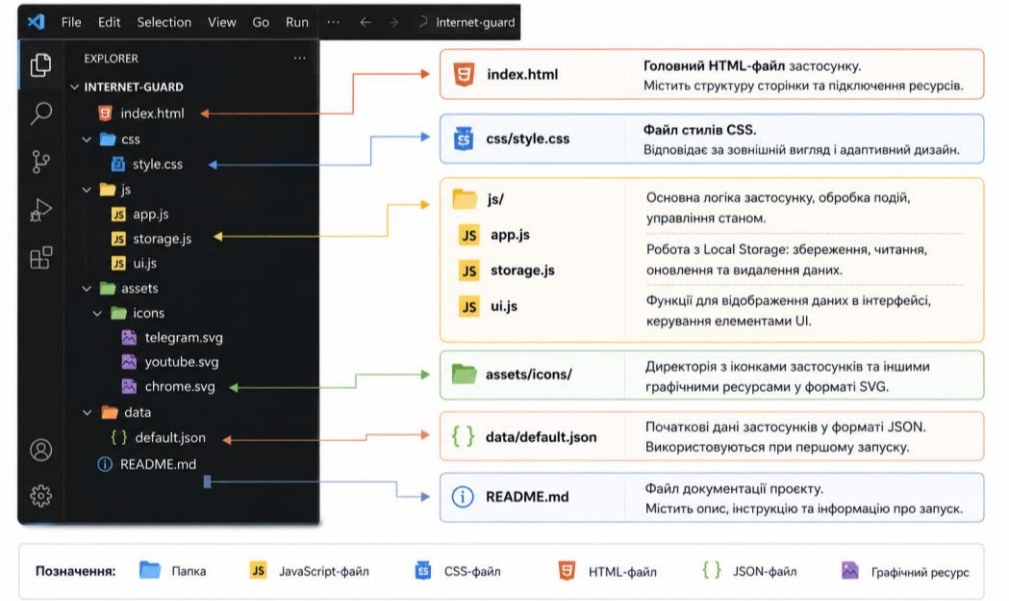

Рисунок 2.5 – Структура проєкту вебзастосунку

Як показано на рисунку 2.5, програмний код розділено на окремі модулі, що відповідають за відображення інтерфейсу, обробку даних та роботу з локальним сховищем браузера.

Архітектура застосунку побудована за принципом розділення інтерфейсу користувача та бізнес-логіки. Інтерфейс відповідає за відображення даних та

взаємодію з користувачем, тоді як JavaScript-модулі реалізують обробку подій, перевірку введених даних та роботу з локальним сховищем.

Для зберігання інформації використовується Local Storage API. Дана технологія дозволяє зберігати дані у браузері користувача у вигляді пар «ключ–значення». Збережена інформація не втрачається після закриття браузера або перезавантаження сторінки, що забезпечує зручну роботу із налаштуваннями та даними застосунку.

Для зберігання інформації використовується Local Storage API. Дана технологія дозволяє зберігати дані у браузері користувача у вигляді пар «ключ–значення».

Загальний алгоритм взаємодії програмної логіки із локальним сховищем наведено на рисунку 2.6. Показано процес читання, запису та оновлення даних у Local Storage за допомогою JavaScript-функцій.

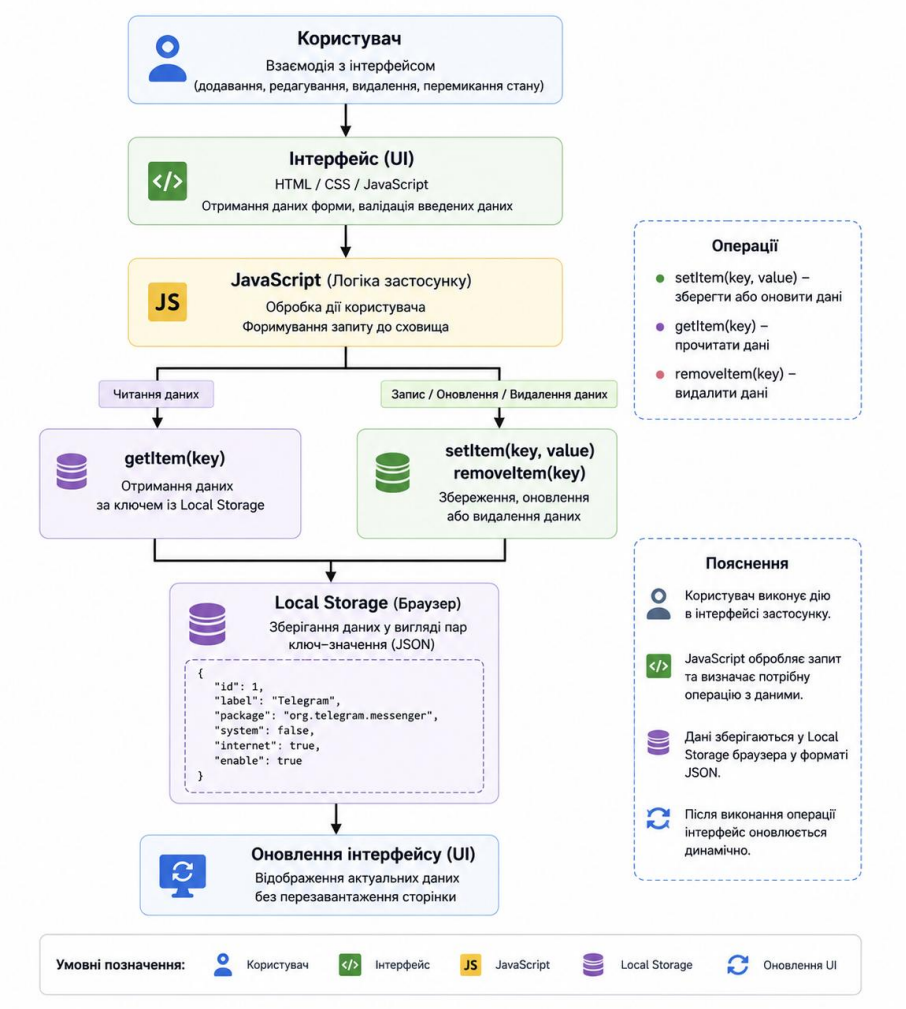

Рисунок 2.6 – Алгоритм роботи застосунку з Local Storage

Нижче наведено приклад JavaScript-коду для збереження інформації про застосунок у локальному сховищі браузера.

Лістинг 3.1 – Збереження даних у Local Storage

```
const app = {
   id: 1,
   label: "Telegram",
   package: "org.telegram.messenger",
   system: false,
   internet: true,
   enable: true
};

localStorage.setItem("app_1", JSON.stringify(app));
```

Для отримання раніше збережених даних використовується метод getItem(), який повертає значення за вказаним ключем.

Лістинг 3.2 – Отримання даних із Local Storage

```
const appData = JSON.parse(
   localStorage.getItem("app_1")
);

console.log(appData);
```

Інтерфейс користувача реалізований у вигляді односторінкового вебзастосунку. Всі зміни стану елементів відображаються динамічно без перезавантаження сторінки, що підвищує швидкодію та зручність використання системи. Загальний вигляд інтерфейсу розробленого програмного засобу наведено на рисунку 2.7.

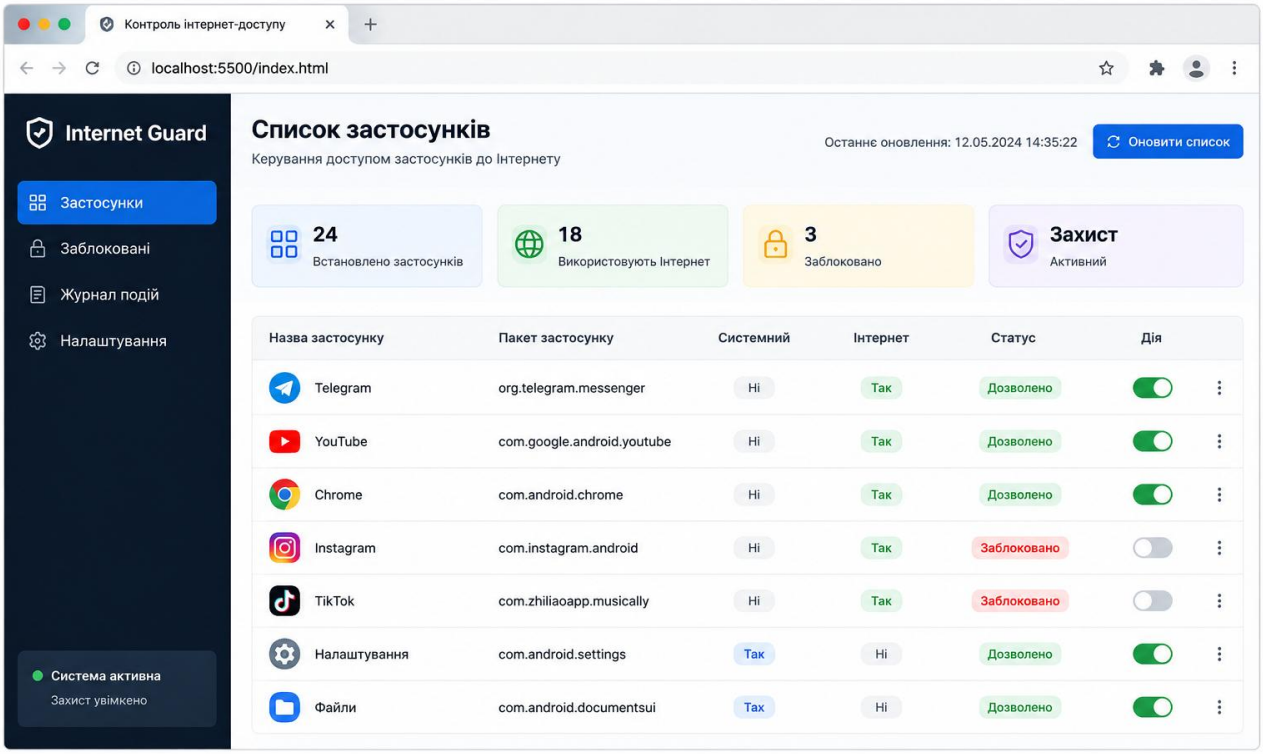

Рисунок 2.7 – Головне вікно вебзастосунку

На рисунку 2.7 представлено головне вікно вебзастосунку, яке забезпечує перегляд списку застосунків, зміну їхнього статусу та керування доступом до мережевих ресурсів.

Для відображення списку застосунків використовується динамічне формування HTML-елементів засобами JavaScript.

Лістинг 3.3 – Формування списку застосунків

```
const container =
    document.getElementById("apps");

apps.forEach(app => {
    const item =
        document.createElement("div");

    item.textContent = app.label;

    container.appendChild(item);
```

});

Послідовність виконання основних операцій із даними в розробленому вебзастосунку представлена на рисунку 2.8.

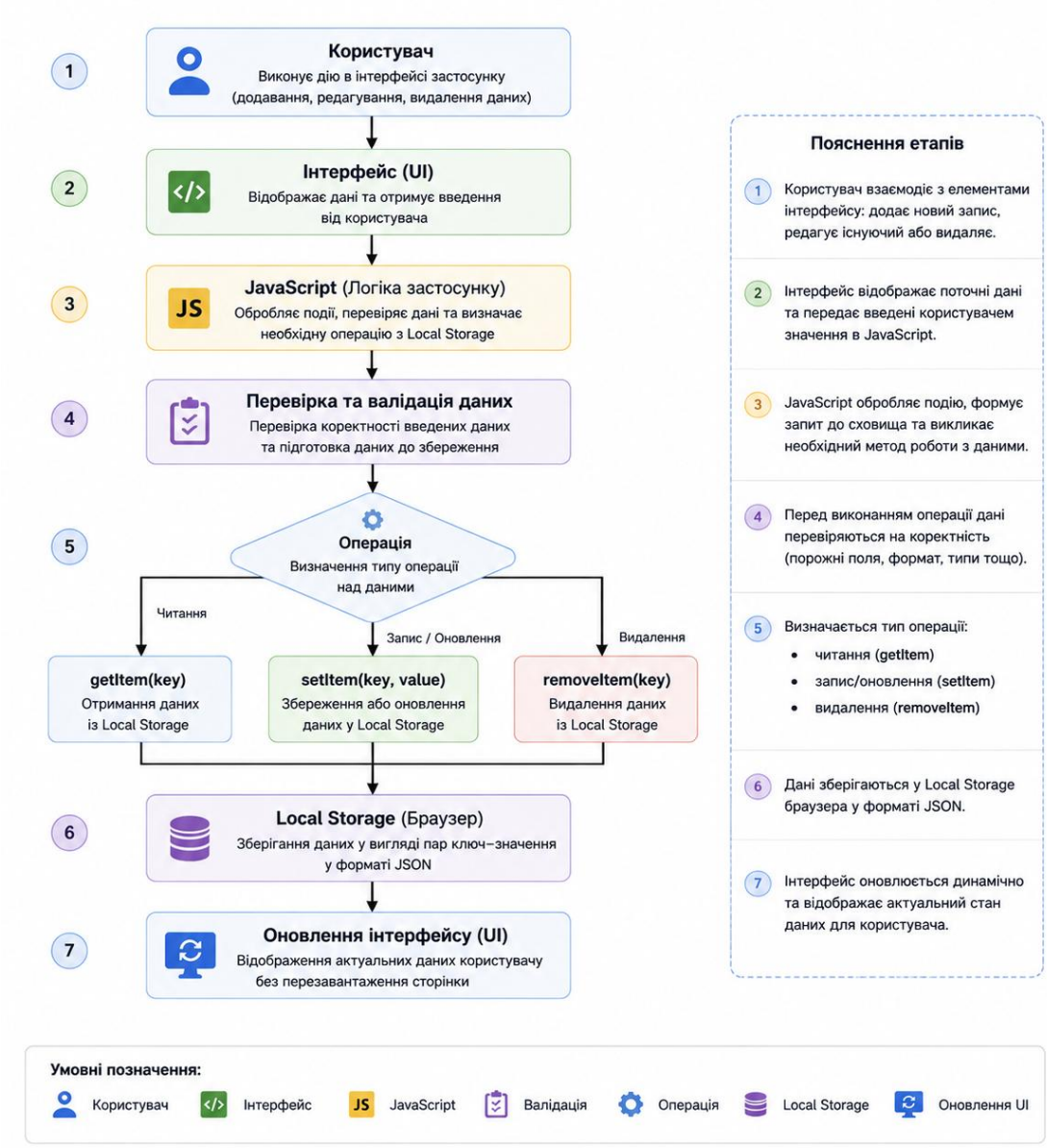

Рисунок 2.8 – Послідовність обробки даних у вебзастосунку

З рисунка 2.8 видно, що після взаємодії користувача з інтерфейсом дані проходять етап перевірки, обробки та зберігання у Local Storage, після чого виконується автоматичне оновлення інтерфейсу без перезавантаження сторінки.

Таким чином, використання HTML5, CSS3, JavaScript та Local Storage дозволило реалізувати сучасний вебзастосунок із зручним інтерфейсом

користувача, високою швидкодією та можливістю локального зберігання даних без використання серверної інфраструктури.

2.4 Програмна реалізація бази даних

Для реалізації механізму зберігання даних у розробленому вебзастосунку використовується технологія Local Storage API. На відміну від традиційних систем керування базами даних, таких як Microsoft SQL Server або MySQL, Local Storage дозволяє зберігати інформацію безпосередньо у браузері користувача без необхідності встановлення серверного програмного забезпечення та налаштування мережевої взаємодії.

Local Storage є частиною Web Storage API та підтримується всіма сучасними браузерами. Технологія забезпечує довготривале зберігання даних у вигляді пар «ключ–значення». Дані залишаються доступними після закриття браузера та повторного відкриття вебзастосунку.

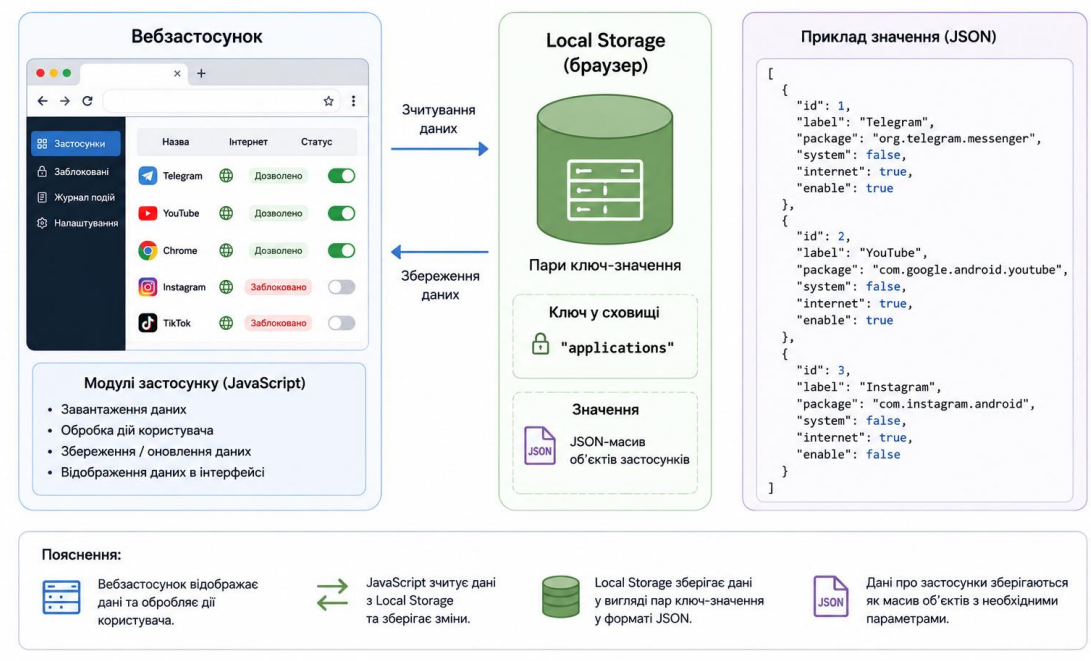

Рисунок 2.9 – Структура зберігання даних у Local Storage

Для зберігання інформації про застосунки використовується структура об'єктів формату JSON. Кожен запис містить ідентифікатор застосунку, його

назву, пакет, ознаку системного застосунку, параметр використання мережі та статус блокування доступу до Інтернету.

Приклад структури об'єкта наведено у лістингу 3.4.

Лістинг 3.4 – Структура запису застосунку

```
const app = {
   id: 1,
   label: "Telegram",
   package: "org.telegram.messenger",
   system: false,
   internet: true,
   enable: true
};
```

Для збереження даних використовується метод setItem(), який записує інформацію до локального сховища браузера.

Лістинг 3.5 – Збереження даних у Local Storage

```
const apps = [app];

localStorage.setItem(
   "applications",
   JSON.stringify(apps)
);
```

Для отримання збережених даних використовується метод getItem(), який повертає значення за вказаним ключем.

Лістинг 3.6 – Отримання даних із Local Storage

```
const applications =
   JSON.parse(
      localStorage.getItem("applications")
   ) || [];
```

Для оновлення інформації про застосунок використовується пошук запису за його ідентифікатором та подальше збереження змінених даних.

Лістинг 3.7 – Оновлення запису

```
function updateApplication(id, status) {

    const applications =
        JSON.parse(
            localStorage.getItem("applications")
        ) || [];

    const app = applications.find(
        item => item.id === id
    );

    if (app) {
        app.enable = status;

        localStorage.setItem(
            "applications",
            JSON.stringify(applications)
        );
    }
}
```

Для видалення запису використовується метод фільтрації масиву та повторне збереження оновленого набору даних.

Лістинг 3.8 – Видалення запису

```
function deleteApplication(id) {

    const applications =
        JSON.parse(
            localStorage.getItem("applications")
        ) || [];
```

```
    const filteredApps =
        applications.filter(
            item => item.id !== id
        );

    localStorage.setItem(
        "applications",
        JSON.stringify(filteredApps)
    );
}
```

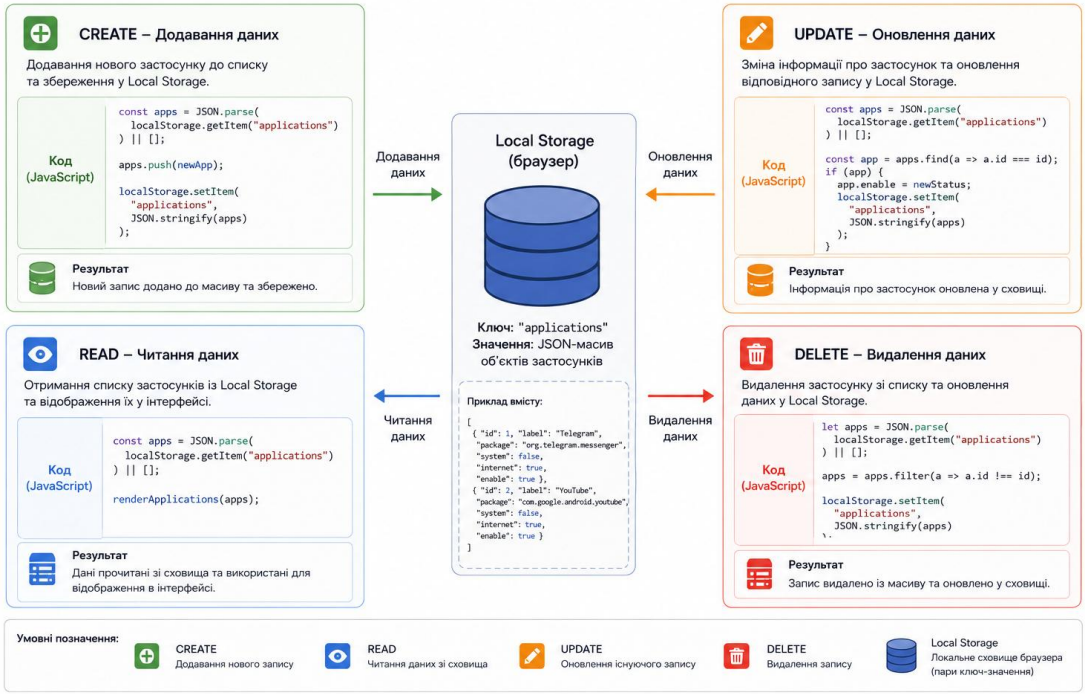

Рисунок 2.10 – CRUD-операції над даними в Local Storage

Сховище даних функціонує за принципом клієнтської бази даних. Після запуску застосунку JavaScript-модуль зчитує дані з Local Storage, виконує їх обробку та формує інтерфейс користувача. У разі внесення змін користувачем інформація автоматично оновлюється у локальному сховищі та відображається в інтерфейсі без перезавантаження сторінки.

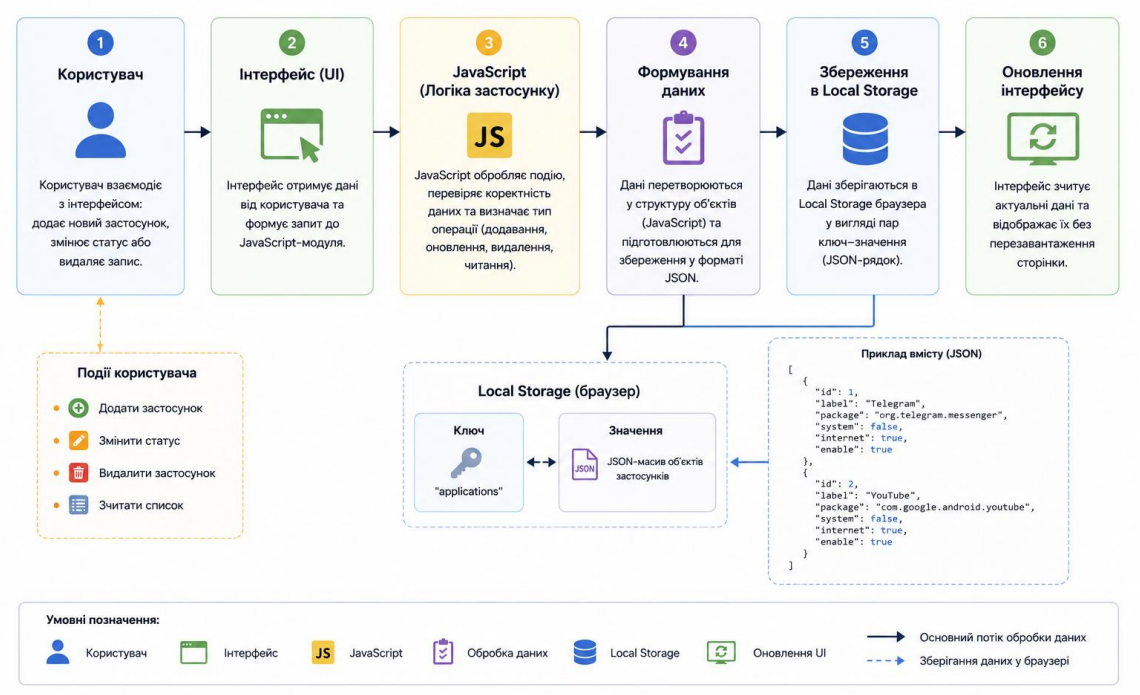

Рисунок 2.11 – Процес обробки та збереження даних у вебзастосунку

Таким чином, використання Local Storage дозволило реалізувати простий та ефективний механізм зберігання даних без використання серверної інфраструктури. Такий підхід забезпечує високу швидкодію застосунку, спрощує його розгортання та відповідає вимогам розроблюваної інформаційної системи.

РОЗДІЛ 3 ТЕСТУВАННЯ ТА ВПРОВАДЖЕННЯ ПРОГРАМНОГО ПРОДУКТУ

3.1 Тестування додатку

Тестування є одним із найважливіших етапів життєвого циклу розробки програмного забезпечення, оскільки дозволяє виявити помилки, перевірити коректність роботи функціоналу та оцінити відповідність програмного продукту поставленим вимогам. Основною метою тестування є забезпечення стабільної роботи застосунку та підвищення якості користувацького досвіду.

У процесі розробки вебзастосунку було проведено комплексне тестування його функціональних можливостей. Перевірка виконувалася шляхом ручного тестування інтерфейсу користувача та основних сценаріїв роботи із даними, що зберігаються у Local Storage.

Для перевірки сумісності застосунку було проведено тестування у сучасних веббраузерах:

– Google Chrome;

– Mozilla Firefox;

– Microsoft Edge;

– Opera.

Під час тестування було перевірено коректність відображення інтерфейсу, роботу елементів керування та функціонування локального сховища браузера.

Було виконано такі види тестування:

1. Функціональне тестування

У ході тестування перевірено основні функції вебзастосунку:

– додавання нового запису;

– редагування існуючих даних;

– видалення записів;

– відображення списку застосунків;

– збереження даних у Local Storage;

– зчитування даних із Local Storage після перезавантаження сторінки.

Усі функції виконувалися коректно та відповідали поставленим вимогам.

2. Тестування інтерфейсу користувача

Було перевірено:

– коректність відображення елементів інтерфейсу;

– роботу кнопок та форм введення;

– адаптацію інтерфейсу до різних розмірів екрана;

– правильність повідомлень про результати виконання операцій.

Під час тестування помилок відображення не виявлено.

3. Тестування локального сховища

Було перевірено:

– створення записів у Local Storage;

– оновлення існуючих записів;

– видалення даних;

– збереження інформації після закриття браузера;

– коректність обробки JSON-даних.

Результати тестування підтвердили правильність реалізації механізму зберігання даних.

4. Тестування продуктивності

Для оцінки продуктивності виконувалося тестування за таких умов:

– багаторазове додавання записів до сховища;

– одночасне виконання декількох операцій над даними;

– робота застосунку при значному обсязі збереженої інформації.

Застосунок продемонстрував стабільну роботу без помітного зниження швидкодії.

5. Кросбраузерне тестування

Було перевірено працездатність застосунку у різних браузерах та на різних пристроях. У всіх випадках функціональні можливості працювали однаково коректно, а інтерфейс відображався відповідно до проєктних вимог.

Результати перевірки основних функціональних можливостей розробленого вебзастосунку наведено в таблиці 3.1.

Таблиця 3.1 – Результати тестування функціональних можливостей вебзастосунку

№	Тестовий сценарій	Очікуваний результат	Фактичний результат	Статус
1	Додавання нового запису	Новий запис додається до списку та зберігається у Local Storage	Запис успішно додано та збережено	Виконано
2	Редагування запису	Дані запису змінюються та оновлюються у сховищі	Дані оновлено коректно	Виконано
3	Видалення запису	Обраний запис видаляється зі списку та сховища	Запис видалено успішно	Виконано
4	Відображення списку даних	На сторінці відображається актуальний перелік записів	Дані відображаються коректно	Виконано
5	Збереження після перезавантаження сторінки	Дані залишаються доступними після оновлення сторінки	Дані збережено	Виконано
6	Робота з Local Storage	Дані записуються та зчитуються без помилок	Помилок не виявлено	Виконано
7	Перевірка адаптивності інтерфейсу	Інтерфейс коректно відображається на різних екранах	Відображення коректне	Виконано
8	Перевірка пошуку записів	Відображаються лише записи, що відповідають запиту	Пошук працює коректно	Виконано
9	Обробка порожніх полів форми	Відображається повідомлення про помилку введення	Валідація працює коректно	Виконано
10	Кросбраузерне тестування	Функціонал працює однаково в усіх браузерах	Відхилень не виявлено	Виконано

Як видно з таблиці 3.1, усі основні функції вебзастосунку працюють відповідно до поставлених вимог. Під час тестування не було виявлено критичних помилок, що можуть вплинути на коректність роботи системи.

Результати проведеного тестування показали, що вебзастосунок коректно реалізує всі передбачені функції, забезпечує надійне зберігання даних у Local Storage та має стабільну роботу в сучасних браузерах. Критичних помилок або

функціональних недоліків під час тестування виявлено не було, тому розроблений програмний продукт може бути рекомендований до подальшої експлуатації.

Для наочного відображення результатів функціонального тестування побудовано узагальнену схему, наведену на рисунку 3.1.

№	Функція / Сценарій	Опис перевірки	Очікуваний результат	Фактичний результат	Статус	Коментар
1	Додавання застосунку	Додати новий застосунок через форму (назва, пакет, системний, інтернет)	Запис додається до списку та зберігається у Local Storage	Запис успішно додано та збережено	Успішно	Дані відображаються відразу після додавання
2	Редагування застосунку	Змінити параметри існуючого застосунку та зберегти зміни	Зміни зберігаються у Local Storage та відображаються у списку	Зміни збережено та відображено	Успішно	Оновлення відбувається без перезавантаження сторінки
3	Видалення застосунку	Видалити застосунок зі списку	Запис видаляється зі списку та з Local Storage	Запис успішно видалено зі списку та сховища	Успішно	Після видалення запис не відновлюється
4	Відображення списку застосунків	Відкрити головну сторінку та переглянути список	Відображається актуальний список застосунків	Список відображено коректно	Успішно	Кількість записів відповідає даним у сховищі
5	Зміна статусу доступу	Увімкнути/вимкнути доступ до Інтернету для застосунку	Статус змінюється та зберігається у Local Storage	Статус змінено та збережено	Успішно	Іконка та текст статусу оновлюються коректно
6	Пошук застосунку	Ввести текст у поле пошуку та перевірити фільтрацію	Відображаються лише записи, що відповідають запиту	Фільтрація працює коректно	Успішно	Пошук нечутливий до регістру
7	Збереження після перезавантаження	Додати/змінити дані, перезавантажити сторінку	Дані зберігаються у Local Storage та відновлюються після перезавантаження	Дані відновлено коректно	Успішно	Стан додатків зберігається після перезавантаження браузера
8	Адаптивність інтерфейсу	Відкрити застосунок на різних розмірах екрану	Інтерфейс коректно адаптується до ширини екрану	Адаптація працює на всіх перевірених розмірах	Успішно	Перевірено на мобільних та десктопних екранах
9	Робота Local Storage	Перевірити наявність ключа "applications" у сховищі через DevTools	Ключ існує, значення має формат JSON-масиву	Ключ існує, дані у форматі JSON-масиву	Успішно	Дані зберігаються у вигляді пар ключ-значення
10	Обробка некоректних даних	Спробувати додати запис із порожніми полями	Відображаються повідомлення про помилку, запис не додається	Помилка відображається, запис не створюється	Успішно	Валідація працює коректно

Загальний висновок:
Усі основні функції вебзастосунку працюють коректно та відповідають вимогам.
Критичних помилок під час тестування не виявлено.

Умовні позначення: Успішно Неуспішно

Рисунок 3.1 – Результати функціонального тестування вебзастосунку

Додатково було виконано перевірку коректності збереження даних у Local Storage за допомогою інструментів розробника браузера. Результати перевірки наведено на рисунку 3.2.

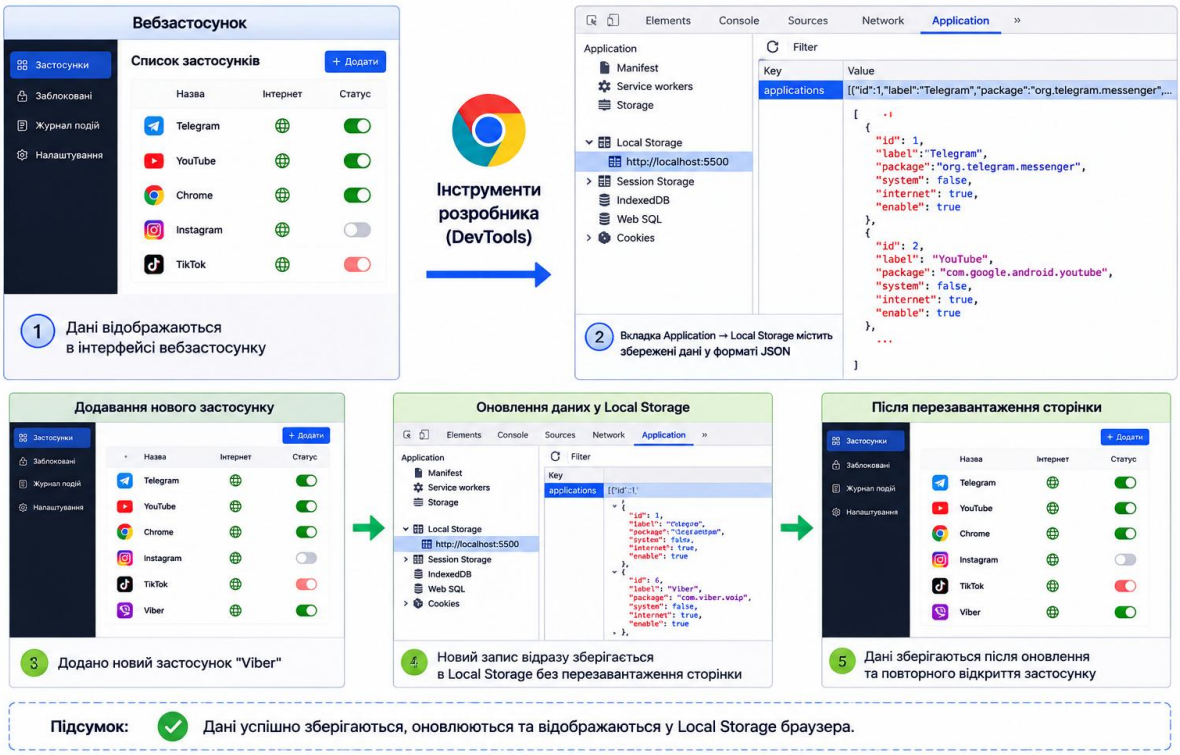

Рисунок 3.2 – Перевірка даних у Local Storage через інструменти розробника браузера

Отримані результати підтверджують коректність роботи механізму зберігання та обробки даних. Всі операції додавання, редагування, видалення та зчитування інформації виконуються без помилок, а дані успішно зберігаються у локальному сховищі браузера. Таким чином, вебзастосунок готовий до подальшої експлуатації та відповідає поставленим функціональним вимогам.

3.2 Впровадження мобільного застосунку

Для початку роботи із розробленим вебзастосунком користувачу необхідно відкрити його у будь-якому сучасному браузері, що підтримує технології HTML5, CSS3 та JavaScript. Встановлення додаткового програмного забезпечення не потрібне, оскільки всі функціональні можливості реалізовані засобами вебтехнологій та виконуються безпосередньо на стороні клієнта.

Після запуску вебзастосунку користувачу відображається головна сторінка, що містить список наявних записів та елементи керування даними. Загальний вигляд головного вікна застосунку наведено на рисунку 3.3.

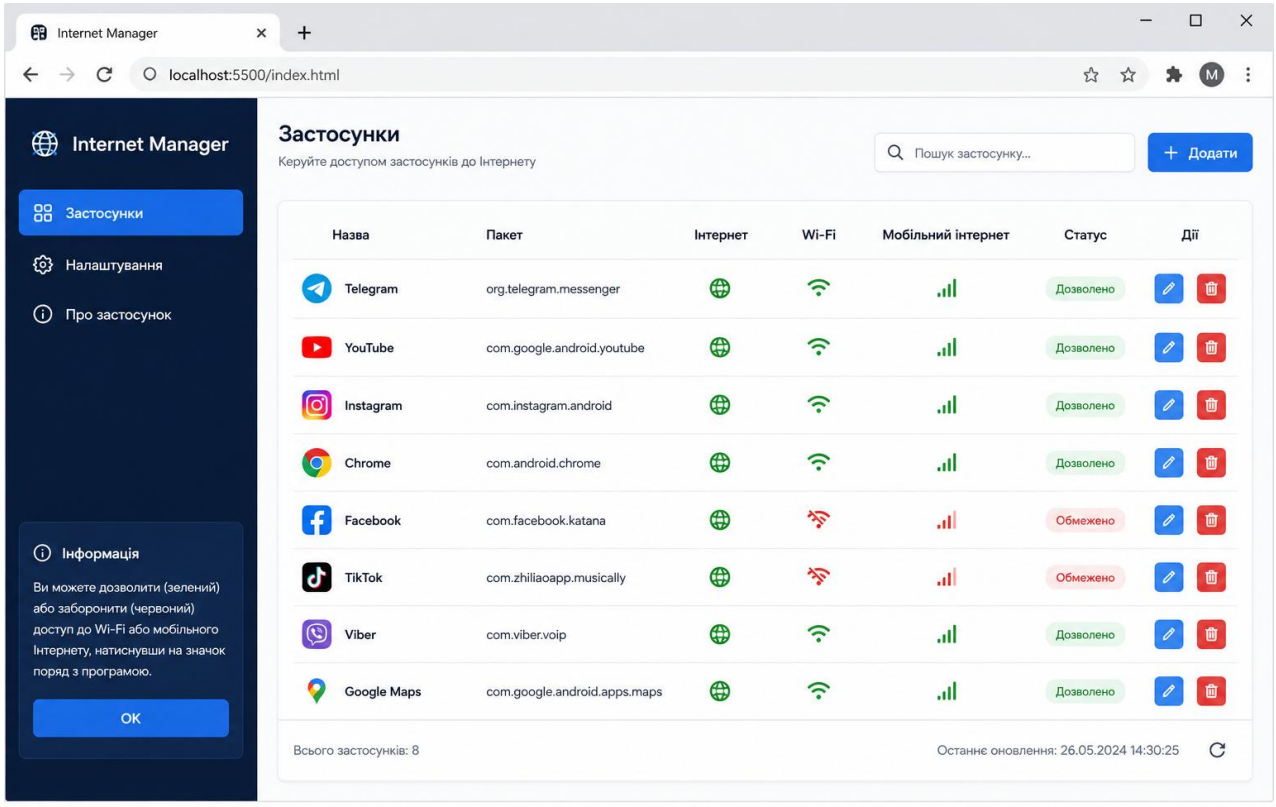

Рисунок 3.3 – Головне вікно вебзастосунку

На рисунку 3.3 представлено головне вікно вебзастосунку, яке забезпечує перегляд списку застосунків, пошук інформації та виконання основних операцій над даними.

Користувач має можливість переглядати наявні записи, додавати нові елементи, редагувати існуючу інформацію та видаляти непотрібні записи. Усі зміни виконуються без перезавантаження сторінки завдяки використанню JavaScript.

Для додавання нового запису необхідно натиснути кнопку «Додати» та заповнити відповідні поля форми. Після підтвердження введених даних інформація автоматично зберігається до Local Storage браузера. Процес додавання нового запису наведено на рисунку 3.4.

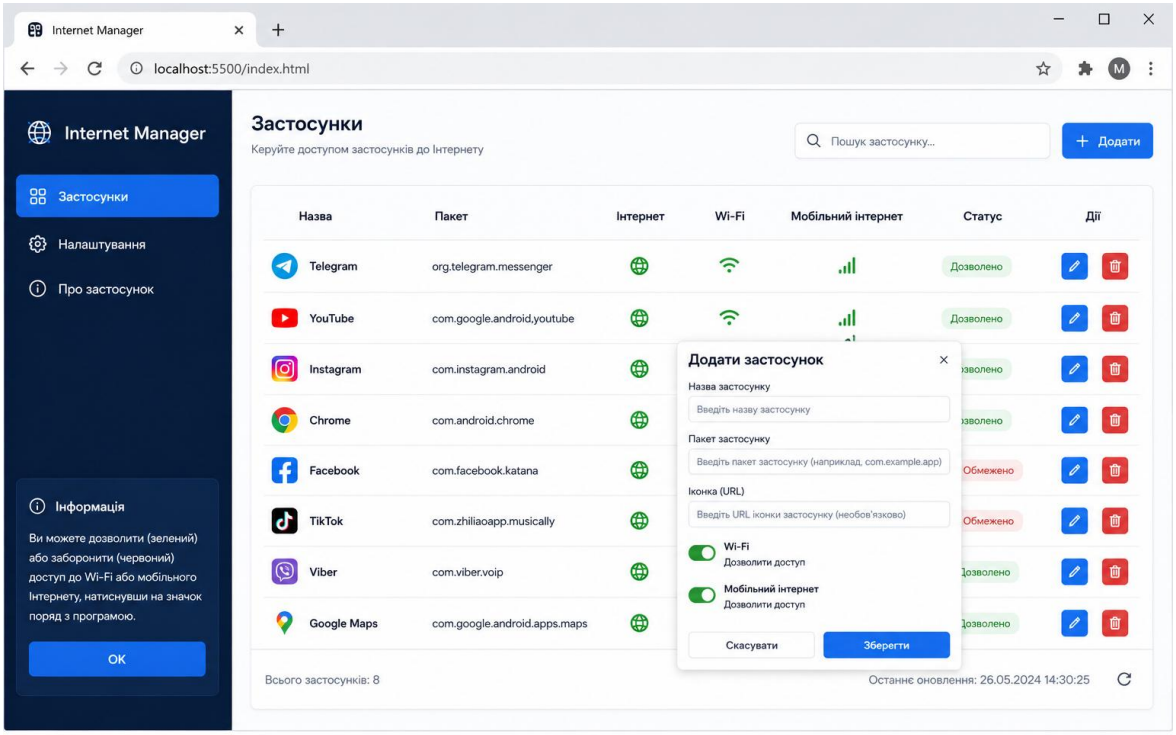

Рисунок 3.4 – Додавання нового запису до системи

Як показано на рисунку 3.4, користувач вводить необхідні дані у форму та зберігає їх у системі, після чого новий запис автоматично додається до списку.

Після збереження інформації новий запис миттєво відображається у списку даних. Усі зміни автоматично фіксуються у локальному сховищі браузера та залишаються доступними навіть після закриття вкладки або перезапуску браузера.

Для перевірки правильності роботи механізму зберігання даних було використано інструменти розробника браузера. Збережені записи відображаються у вкладці Application → Local Storage у форматі JSON. Приклад перевірки даних наведено на рисунку 3.5.

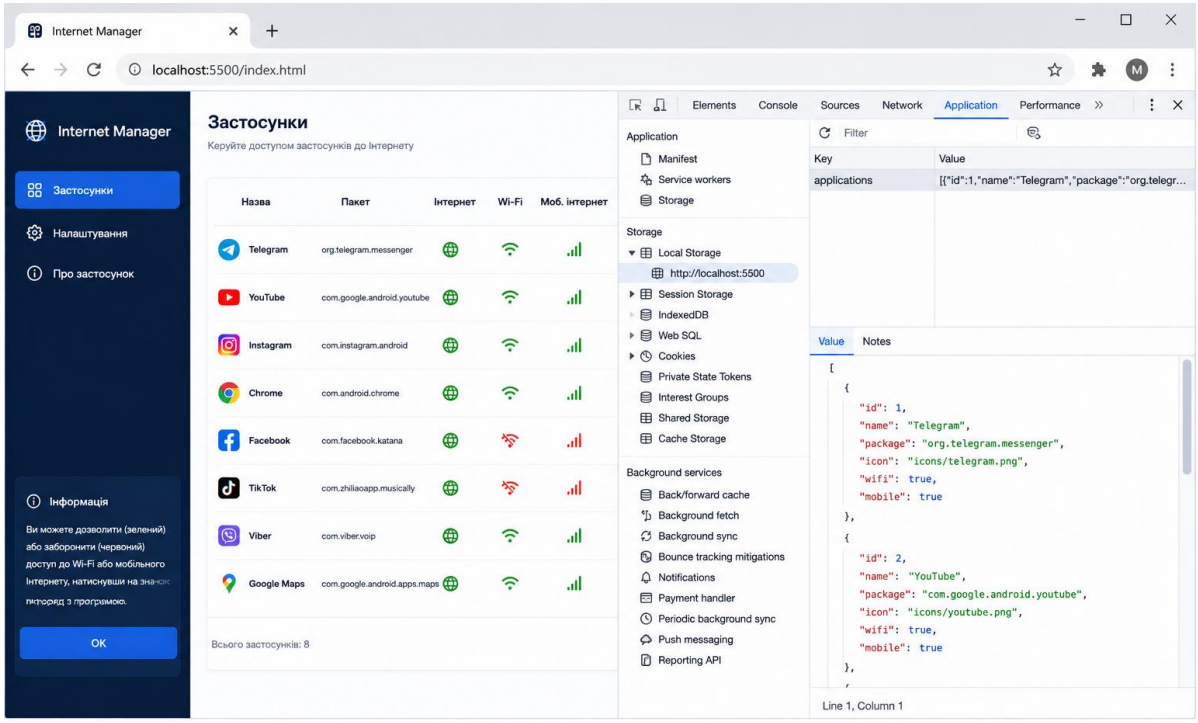

Рисунок 3.5 – Перевірка даних у Local Storage браузера

На рисунку 3.5 показано результати перевірки локального сховища браузера, де відображаються всі записи, створені користувачем під час роботи із вебзастосунком.

Однією з переваг реалізованого підходу є автоматичне відновлення інформації після оновлення сторінки. Після повторного відкриття вебзастосунку всі раніше збережені дані завантажуються з Local Storage та відображаються користувачу без втрати інформації. Результат повторного завантаження даних представлено на рисунку 3.6.

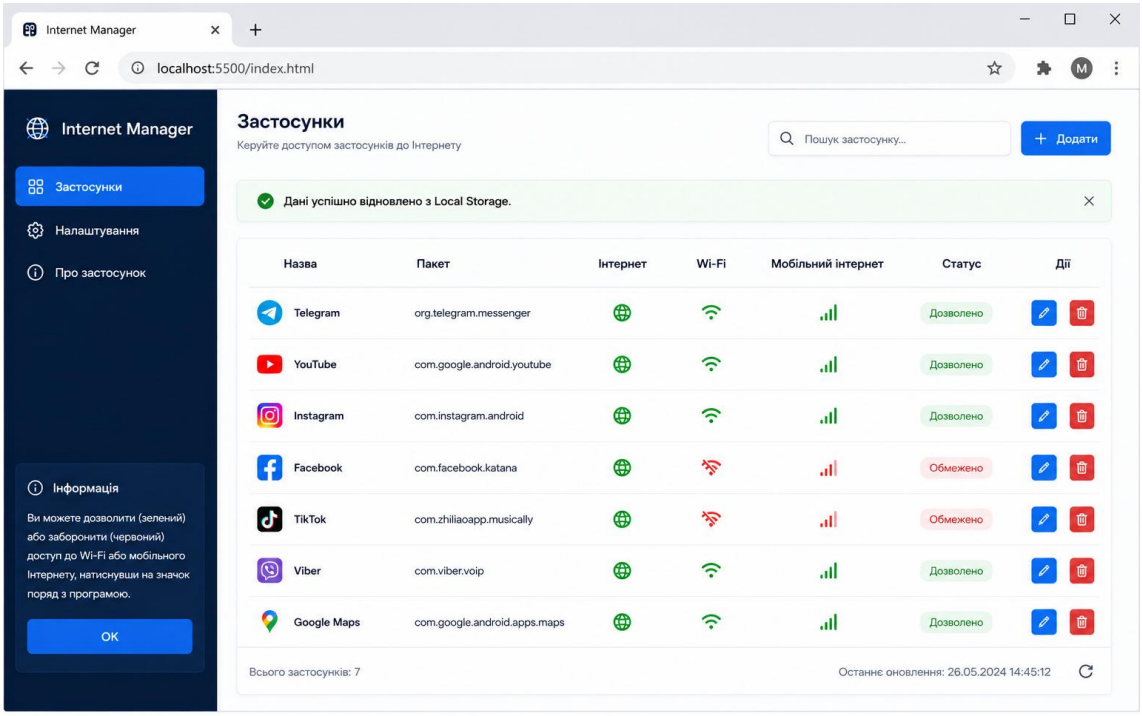

Рисунок 3.6 – Відновлення даних після перезавантаження сторінки

З рисунка 3.6 видно, що після перезавантаження сторінки всі дані були успішно відновлені з Local Storage та повторно відображені в інтерфейсі користувача без втрати інформації.

Проведене тестування показало, що вебзастосунок стабільно працює у сучасних браузерах, коректно виконує операції додавання, редагування, видалення та збереження інформації. Використання Local Storage забезпечує надійне локальне зберігання даних без необхідності застосування серверної бази даних.

Таким чином, розроблений вебзастосунок повністю реалізує поставлені функціональні вимоги, має простий та зрозумілий інтерфейс користувача, а також готовий до практичного використання.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи було проведено аналіз сучасних вебтехнологій та засобів розробки клієнтських застосунків. Розглянуто існуючі рішення для управління даними та локального зберігання інформації, визначено їхні переваги та недоліки, а також обґрунтовано вибір технологій для реалізації програмного продукту.

У результаті виконаного дослідження було спроєктовано структуру даних вебзастосунку, визначено основні сутності та взаємозв’язки між ними. Для зберігання інформації обрано технологію Local Storage, яка забезпечує довготривале локальне зберігання даних без використання серверної бази даних та додаткової інфраструктури.

У процесі розробки програмного продукту використано сучасні вебтехнології HTML5, CSS3 та JavaScript. Інтерфейс користувача реалізовано відповідно до принципів мінімалізму та зручності використання. Застосунок забезпечує виконання основних операцій над даними, зокрема створення, перегляд, редагування та видалення записів, а також автоматичне збереження інформації у локальному сховищі браузера.

Під час роботи було розроблено архітектуру програмної системи, реалізовано механізми взаємодії користувача з даними та створено програмні модулі для обробки інформації. Завдяки використанню JavaScript усі операції виконуються динамічно без перезавантаження сторінок, що підвищує швидкодію та зручність роботи користувача.

Для перевірки працездатності програмного продукту проведено функціональне, кросбраузерне та продуктивне тестування. Результати тестування підтвердили коректність роботи всіх реалізованих функцій, стабільність механізмів збереження даних та сумісність застосунку із сучасними веббраузерами. Критичних помилок під час тестування виявлено не було.

Практичним результатом роботи є створений вебзастосунок для керування та зберігання даних, який характеризується простотою використання, надійністю та незалежністю від серверної інфраструктури. Розроблене програмне

забезпечення може бути використане як основа для подальшого розширення функціональних можливостей, інтеграції із серверними сервісами або переходу до повноцінної клієнт-серверної архітектури.

Таким чином, усі поставлені завдання кваліфікаційної роботи виконано в повному обсязі, а розроблений вебзастосунок відповідає визначеним функціональним вимогам та може бути рекомендований до практичного використання.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Teens, Social Media and Technology 2024. URL: https://www.pewresearch.org/internet/2024/12/12/teens-social-media-and-technology-2024/ (дата звернення: 28.05.2026)

2. Mobile Operating System Market Share Worldwide. URL: https://gs.statcounter.com/os-market-share/mobile/worldwide (дата звернення: 28.05.2026)

3. Android (operating system). URL: https://en.wikipedia.org/wiki/Android_(operating_system) (дата звернення: 28.05.2026)

4. Global Browser Market Share Statistics 2025. URL: https://gs.statcounter.com/browser-market-share (дата звернення: 28.05.2026)

5. HTML5 Specification. URL: https://html.spec.whatwg.org/

6. Cascading Style Sheets (CSS) Overview. URL: https://www.w3.org/Style/CSS/ (дата звернення: 28.05.2026)

7. JavaScript Guide. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide (дата звернення: 28.05.2026)

8. Family Link – Parental Controls. URL: https://families.google/familylink/ (дата звернення: 28.05.2026)

9. Qustodio Parental Control Software. URL: https://www.qustodio.com (дата звернення: 28.05.2026)

10. Bark Technologies – Parental Control Platform. URL: https://www.bark.us (дата звернення: 28.05.2026)

11. Norton Family Parental Control. URL: https://family.norton.com (дата звернення: 28.05.2026)

12. Архітектура та проєктування програмного забезпечення. URL: http://dspace.wunu.edu.ua (дата звернення: 28.05.2026)

13. SQL та сучасні бази даних. URL: https://proglib.io/p/databases-2019 (дата звернення: 28.05.2026)

14. Мартін Р. С. Чиста архітектура. – Харків : Фабула, 2019. – 368 с.

15. Visual Studio Code Documentation. URL: https://code.visualstudio.com/docs (дата звернення: 28.05.2026)

16. Git Documentation. URL: https://git-scm.com/doc (дата звернення: 28.05.2026)

17. Window: localStorage Property. URL: https://developer.mozilla.org/en-US/docs/Web/API/Window/localStorage (дата звернення: 28.05.2026)

18. Фленаган Д. JavaScript. Детальний посібник. – 7-ме вид. Київ : КМ-Букс, 2021. 736 с.

19. Web Storage API. URL: https://developer.mozilla.org/en-US/docs/Web/API/Web_Storage_API (дата звернення: 28.05.2026)

20. JSON Data Interchange Format. URL: https://www.json.org/json-en.html (дата звернення: 28.05.2026)

21. Responsive Web Design Basics. URL: https://web.dev/responsive-web-design-basics/ (дата звернення: 28.05.2026)

22. Mozilla Developer Network Web Docs. URL: https://developer.mozilla.org/ (дата звернення: 28.05.2026)

23. ECMAScript Language Specification. URL: https://tc39.es/ecma262/ (дата звернення: 28.05.2026)

24. Google Chrome DevTools Documentation. URL: https://developer.chrome.com/docs/devtools/ (дата звернення: 28.05.2026)

25. Firefox Developer Tools. URL: https://firefox-source-docs.mozilla.org/devtools-user/ (дата звернення: 28.05.2026)

26. W3C Web Accessibility Initiative (WAI). URL: https://www.w3.org/WAI/ (дата звернення: 28.05.2026)

27. Nielsen J. Usability Engineering. Morgan Kaufmann, 1994. 362 p.

28. Sommerville I. Software Engineering. 10th Edition. Pearson, 2016. 816 p.

29. Pressman R. Software Engineering: A Practitioner's Approach. 9th Edition. McGraw-Hill, 2019. 880 p.

30. ISTQB Foundation Level Syllabus 4.0. URL: https://www.istqb.org (дата звернення: 28.05.2026)

31. Google Web Fundamentals. URL: https://web.dev (дата звернення: 28.05.2026)