ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ТЕХНОЛОГІЙ ЦИФРОВОГО РОЗВИТКУ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Цифрова платформа формування та відстеження звичок користувачів з елементами мотивації та аналітики»

на здобуття освітнього ступеня бакалавра
зі спеціальності 121 Інженерія Програмного Забезпечення
освітньо-професійної програми Технології цифрового розвитку

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

Нікіта РЕУЛ

_____ _____
(підпис)

Виконав: здобувач вищої освіти групи ТЦР-43

Нікіта РЕУЛ

Керівник: Світлана ПОПЕРЕШНЯК
д-р техн. наук, проф.

Рецензент: _____

Київ 2026

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Технології цифрового розвитку

Ступінь вищої освіти Бакалавр

Спеціальність Інженерія програмного забезпечення

Освітньо-професійна програма Технології цифрового розвитку

ЗАТВЕРДЖУЮ
Завідувач кафедри
Технологій цифрового розвитку
_____ Вікторія ЖЕБКА
« ____ » _____ 2026 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Реула Нікіти Євгеновича

1. Тема кваліфікаційної роботи: Цифрова платформа формування та відстеження звичок користувачів з елементами мотивації та аналітики керівник кваліфікаційної роботи канд. фіз.-мат. наук, доцент Світлана Поперешняк, затверджені наказом Державного університету інформаційно-комунікаційних технологій від «20» лютого 2026р. № 51

2. Строк подання кваліфікаційної роботи «18»травня 2026 р.

3. Вихідні дані до кваліфікаційної роботи: Аналіз існуючих сервісів для відстеження звичок,наукові та методичні джерела з веб розробки, вимоги до інтерфейсу користувача, функціональні вимоги, рекомендації щодо формування звичок та особистої продуктивості

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

   1. Огляд принципів формування звичок, як веб ресурси допомогають в їх формувані та відстежені.
   2. Аналіз предметної області, існуючих конкурентів з точки зору функціональних можливостей застосунку.
   3. Проєктування архітектури веб ресурсу з урахуванням всіх визначених функціональних вимог
   4. Реалізація програмного забезпечення

5. Перелік ілюстративного матеріалу: *презентація*

6. Дата видачі завдання «20» лютого 2026р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Підбір та аналіз науково-технічної літератури	20.02.2026-10.03.26	
2	Аналіз існуючих аналогів конкурентів	12.03.26-20.03.26	
3	Визначення функціональних та нефункціональних вимог	01.04.26-03.04.26	
4	Проєктування архітектури застосунку	05.04.26- 25.04.26	
5	Розробка інтерфейсу користувача	27.04.26-30.04.26	
6	Проведення тест-кейсів для застосунку	01.05.26-05.05.26	
7	Оформлення роботи : Вступ, реферат, висновки	06.05.26-15.05.26	
8	Попередній захист роботи	18.05.26	

Здобувач(ка) вищої освіти _____ Нікіта РЕУЛ
(підпис)

Керівник
кваліфікаційної роботи _____ Світлана ПОПЕРЕШНЯК
(підпис)

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 45 стор., 8 табл., 9 рис., 43 джерела.

Мета роботи підвищення ефективності процесу формування та підтримки корисних звичок користувачів шляхом розробки цифрової платформи HabitFlow з елементами мотивації, аналітики, візуалізації прогресу та локального збереження даних у браузері.

Об'єкт дослідження процес формування, відстеження та аналізу звичок користувачів у цифровому середовищі.

Предмет дослідження програмне забезпечення для формування та відстеження звичок користувачів, реалізоване у вигляді веб-платформи на основі HTML, CSS та JavaScript.

Короткий зміст роботи: У роботі проаналізовано теоретичні основи формування звичок, зокрема психологічні механізми автоматичності поведінки, роль мотивації, самоконтролю та систематичного відстеження прогресу. Розглянуто сучасні цифрові платформи для трекінгу звичок, такі як Habitica, Habitify, Streaks та Loop Habit Tracker, визначено їхні переваги й недоліки. Проведено порівняльний аналіз існуючих рішень та обґрунтовано необхідність створення власної цифрової платформи, яка працює без реєстрації, не потребує серверної частини та забезпечує локальне збереження даних у браузері. Обґрунтовано вибір технологій HTML, CSS та JavaScript для реалізації клієнтського веб-додатка.

У практичній частині спроєктовано архітектуру програмного забезпечення, функціональні та нефункціональні вимоги, інтерфейс користувача й структуру зберігання даних. Реалізовано веб-платформу HabitFlow, що включає модуль управління звичками, систему відстеження виконання з обмеженням однієї відмітки на день, streak-мотивацію, progress bar, мотиваційні повідомлення, PUSH-нагадування та модуль аналітики.

Проведено функціональне тестування, тестування інтерфейсу та юзабіліті розробленої платформи. Результати тестування підтвердили коректність роботи основних модулів, надійність збереження даних, високу продуктивність, інтуїтивність інтерфейсу та практичну придатність програмного продукту для використання студентами, молодими фахівцями та широким колом користувачів.

КЛЮЧОВІ СЛОВА: ЦИФРОВА ПЛАТФОРМА, ЗВИЧКИ, HABITFLOW, HTML, CSS, JAVASCRIPT, UI, UX, STREAK-МОТИВАЦІЯ, АНАЛІТИКА, PROGRESS BAR, LOCALSTORAGE, WEB STORAGE API, ВЕБ-ДОДАТОК.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

CSS – Cascading Style Sheets (каскадні таблиці стилів)
FR – Functional Requirement (функціональна вимога)
HTML – HyperText Markup Language (мова гіпертекстової розмітки)
JS – JavaScript
NFR – Non-Functional Requirement (нефункціональна вимога)
RPG – Role-Playing Game (рольова гра)
UI – User Interface (інтерфейс користувача)
Web Storage API – інтерфейс програмного доступу до сховища даних у веб-браузері
localStorage – локальне сховище браузера

ВСТУП

У сучасному швидкоплинному світі формування та підтримка корисних звичок є одним із ключових чинників особистісного розвитку, підвищення продуктивності та покращення якості життя. Регулярне виконання певних дій, таких як фізична активність, читання, медитація чи здорове харчування, дозволяє досягати довгострокових цілей, розвивати самодисципліну та формувати стійку поведінку. Однак процес формування звичок часто супроводжується труднощами: відсутністю мотивації, забуттям, відсутністю системного відстеження прогресу та візуальної демонстрації результатів. Ці проблеми особливо актуальні для молодого покоління, студентів та активних фахівців, які стикаються з високим рівнем навантаження та інформаційного шуму.

Цифровізація повсякденного життя створює передумови для розробки спеціалізованих програмних рішень, що допомагають користувачам ефективно керувати своїми звичками. Існуючі мобільні та веб-додатки для трекінгу звичок пропонують базовий функціонал, проте часто мають обмеження: складний інтерфейс, залежність від інтернету, платні преміум-функції, відсутність повноцінної системи streak-мотивації (як у Duolingo) та детальної аналітики. Крім того, багато рішень не забезпечують повного збереження даних у браузері без реєстрації, що знижує доступність та зручність для широкого кола користувачів.

Актуальність теми дослідження зумовлена потребою у створенні простої, ефективної та доступної цифрової платформи, яка поєднує зручний інтерфейс, потужні мотиваційні механізми та надійне локальне зберігання даних. Розробка такого веб-додатка з використанням лише базових веб-технологій (HTML, CSS та JavaScript) дозволяє забезпечити кросплатформенність, швидкість роботи та мінімальні вимоги до ресурсів користувача, що є важливим для освітнього та особистого використання.

Об’єктом дослідження є процес формування, відстеження та аналізу звичок користувачів у цифровому середовищі.

Предметом дослідження виступає програмне забезпечення для формування та відстеження звичок користувачів, реалізоване у вигляді веб-платформи на основі HTML, CSS та JavaScript.

Метою бакалаврської роботи є формування та відстеження звичок користувачів за допомогою розробленої цифрової платформи на основі HTML, CSS та JavaScript, яка забезпечує повний цикл управління звичками, візуалізацію прогресу та мотивацію.

Для досягнення поставленої мети визначено такі завдання:

1. Проаналізувати теоретичні основи формування звичок та огляд існуючих цифрових платформ для їх трекінгу.
2. Провести порівняльний аналіз аналогічних рішень та обґрунтувати вибір технологій розробки.
3. Спроєктувати архітектуру програмного забезпечення, інтерфейс користувача та структуру зберігання даних.
4. Реалізувати основні модулі: управління звичками (додавання, редагування, видалення), систему відстеження виконання з обмеженням однієї відмітки на день, streak-мотивацію, progress bar, модуль аналітики (кількість виконань, відсоток виконання, загальний прогрес).
5. Забезпечити збереження даних у браузері за допомогою Web Storage API для збереження інформації після перезавантаження сторінки.
6. Провести функціональне тестування, тестування інтерфейсу та юзабіліті розробленої платформи.
7. Оцінити ефективність та практичну значущість отриманих результатів.

Методи дослідження включають аналіз наукової та технічної літератури, порівняльний аналіз програмних аналогів, методи системного аналізу та проєктування, об’єктно-орієнтоване програмування, а також методи тестування програмного забезпечення.

Наукова новизна роботи полягає в комплексній реалізації веб-платформи з акцентом на інтеграцію системи streak-мотивації, візуального progress bar та

детального модуля аналітики в єдиному клієнтському додатку без використання серверної частини та зовнішніх фреймворків.

Практична значущість розробки полягає в створенні готового до використання веб-додатка, який може застосовуватися студентами, фахівцями та широким колом користувачів для самостійного формування корисних звичок. Програмний продукт може бути інтегрований у навчальний процес або використаний як основа для подальшого розвитку.

Структура роботи. Бакалаврська робота складається зі вступу, чотирьох розділів, висновків, списку використаних джерел та додатків. У першому розділі проведено аналіз предметної області та існуючих рішень. Другий розділ присвячено проєктуванню платформи. Третій розділ описує програмну реалізацію. Четвертий розділ містить результати тестування та оцінку ефективності. У висновках підбито підсумки роботи та окреслено перспективи подальшого розвитку.

Робота пройшла апробацію:

1. Реул Н.Є., Поперешняк С.В. Архітектура системи автентифікації на основі WEBAUTHN та PASSKEYS // Всеукраїнська науково-технічну конференція «Технології цифрового розвитку: програмні системи та децентралізація», 1–3 травня 2026 року, м. Київ. Збірник тез. (Подано до друку).
2. Реул Н.Є., Поперешняк С.В. Сучасні інтелектуальні технології в україні і світі// Всеукраїнська науково-практична конференція «Сучасні інтелектуальні інформаційні технології в науці та освіті». 15 травня 2026 року, м. Київ. Збірник тез. (Подано до друку).

1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ІСНУЮЧИХ ПРОГРАМНИХ РІШЕНЬ

1.1. Теоретичні основи формування та відстеження звичок

Формування звичок є складним психологічним процесом, що забезпечує перехід свідомої поведінки в автоматичну форму виконання дій. Звичка визначається як дія, яка активується автоматично у відповідь на певні контекстні сигнали завдяки повторюваному виконанню в стабільному середовищі. Такий механізм дозволяє економити когнітивні ресурси людини, звільняючи увагу для вирішення більш складних завдань [1].

Теоретичні моделі формування звичок базуються на концепції «habit loop», яка включає три основні компоненти: cue (сигнал або тригер), routine (рутинна дія) та reward (нагорода). Сигнал запускає автоматичну реакцію мозку, рутинна дія виконується без значних зусиль волі, а нагорода закріплює асоціацію, сприяючи подальшому повторенню поведінки. Ця модель пояснює, чому звички стають стійкими навіть за зниження свідомої мотивації [2].

Дослідження показують, що для формування нової звички в реальних умовах потрібно в середньому 66 днів регулярного повторення, хоча цей період може варіюватися залежно від складності дії, індивідуальних особливостей та стабільності контексту. Процес відбувається через поступове зміцнення асоціації між сигналом і дією, що призводить до зростання автоматичності виконання [3].

Важливим чинником успішного формування звичок є мотивація та самоконтроль на початковому етапі. Однак надмірна залежність від сили волі часто призводить до зривів, оскільки самоконтроль є обмеженим ресурсом. Ефективнішим підходом вважається створення стабільного контексту та використання невеликих, послідовних дій, які поступово автоматизуються. Крім

того, візуалізація прогресу та система винагород посилюють мотивацію та сприяють довгостроковому закріпленню поведінки [4].

Психологічні чинники формування здоров'язбережувальної поведінки включають ціннісні орієнтації, вплив соціального середовища (сім'ї, однолітків) та рівень самоусвідомлення. У студентській молоді спостерігається розуміння важливості здорового способу життя, проте мотивація до систематичного дотримання корисних звичок часто залишається недостатньою через зовнішні стресові фактори та фрагментарність інформації. Сімейні настанови відіграють ключову роль у формуванні стійких поведінкових патернів [5].

Відстеження звичок є невід'ємною частиною процесу їх формування. Систематичний моніторинг виконання дій дозволяє фіксувати прогрес, виявляти перешкоди та коригувати стратегію. Сучасні підходи до трекінгу акцентують увагу на візуалізації даних (графіки, прогрес-бари, streaks), що підвищує емоційне залучення та підтримує мотивацію. Обмеження кількості відміток за день (одна на добу) запобігає перевантаженню та сприяє формуванню реалістичних очікувань.

Теоретичні основи формування та відстеження звичок ґрунтуються на поєднанні психологічних механізмів автоматичності, мотиваційних чинників та інструментів зворотного зв'язку. Розуміння цих принципів є необхідною передумовою для розробки ефективних цифрових платформ, які допомагають користувачам перетворювати наміри на стійкі поведінкові патерни.

1.2. Огляд сучасних цифрових платформ для трекінгу звичок

Сучасний ринок цифрових рішень для формування та відстеження звичок характеризується різноманітністю мобільних і веб-додатків, які пропонують різні підходи до мотивації та аналітики. Серед найбільш популярних платформ виділяються Habitica, Habitify, Streaks та Loop Habit Tracker. Кожна з них має унікальні особливості, що відповідають різним потребам користувачів.

Habitica є гейміфікованою платформою, яка перетворює повсякденні звички на елементи рольової гри. Користувач створює аватар, виконує завдання та звички

для отримання досвіду, золота та предметів. Невідповідне виконання завдань призводить до втрати здоров'я персонажа, що стимулює регулярність. Додаток підтримує спільноти, квести та соціальну взаємодію, що робить процес формування звичок більш захопливим [6].

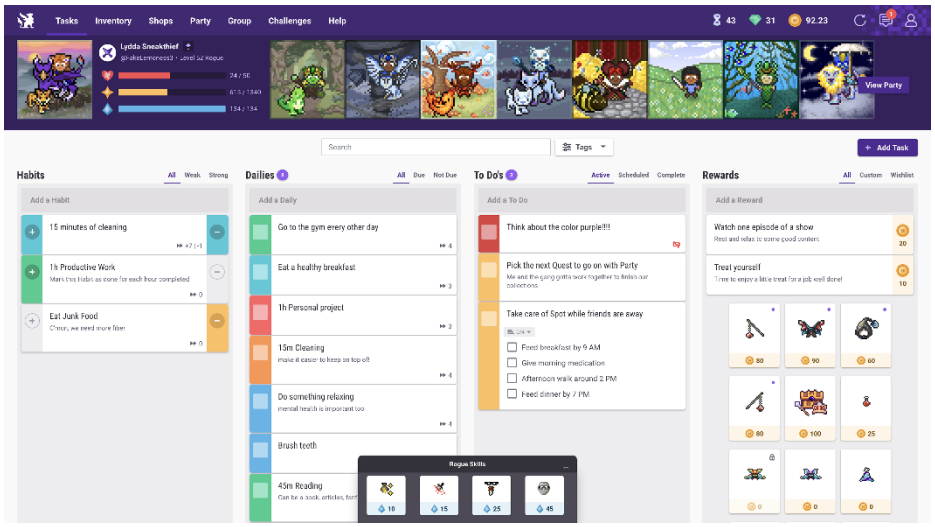


Рис. 1.1 Інтерфейс Habitica з гейміфікацією (аватар та завдання)

Habitify пропонує сучасний мінімалістичний інтерфейс і акцент на детальній аналітиці. Платформа дозволяє групувати звички за категоріями, встановлювати нагадування, вести нотатки та переглядати статистику виконання у вигляді графіків і календарів. Habitify доступний на iOS, Android, веб та десктопі, що забезпечує кросплатформенну синхронізацію. Система streak та візуальний прогрес допомагають підтримувати мотивацію протягом тривалого часу [7].

Рис. 1.2 Головний екран Habitify з календарем прогресу та статистикою

Streaks орієнтований на простоту та швидкість. Додаток дозволяє відстежувати до 24 звичок одночасно з акцентом на підтримку безперервного ланцюжка (streak). Інтерфейс мінімалістичний, зручний для Apple Watch та інтеграції з Apple Health. Кожне виконання звички відзначається одним дотиком, а статистика показує поточний і найкращий streak. Такий підхід особливо ефективний для користувачів, які віддають перевагу лаконічним рішенням без зайвих функцій [8].

Рис. 1.3 Інтерфейс Streaks з візуалізацією streak та прогрес-барами

Loop Habit Tracker є безкоштовним відкритим рішенням для Android. Додаток відрізняється повною офлайн-роботою, локальним зберіганням даних і детальними графіками сили звички (habit strength). Користувач може переглядати історію виконання, статистику та відсоток успіху. Завдяки відсутності реклами та реєстрації Loop забезпечує високий рівень приватності та простоту використання [9].

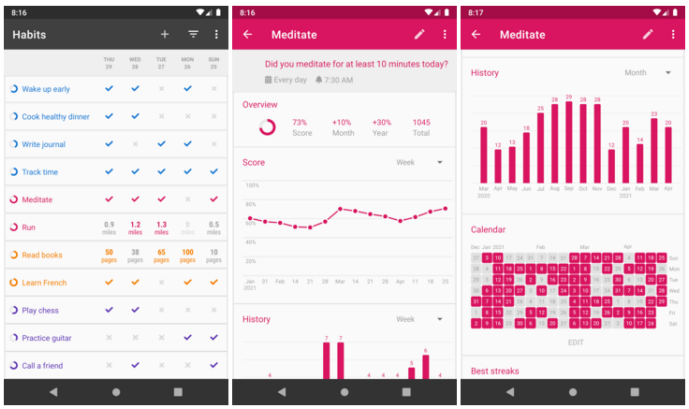


Рис. 1.4 Головний екран Loop Habit Tracker з кольоровими відмітками та графіками

Огляд існуючих цифрових платформ підтверджує попит на прості, ефективні та доступні інструменти трекінгу звичок. Разом з тим, виявлено недоліки у вигляді залежності від зовнішніх сервісів, складності інтерфейсу або обмеженої функціональності, що обґрунтовує необхідність розробки власної веб-платформи на базі HTML, CSS та JavaScript з акцентом на локальне зберігання даних і комплексну мотиваційну систему.

1.3. Порівняльний аналіз існуючих рішень

Порівняльний аналіз сучасних цифрових платформ для трекінгу звичок дозволяє виявити їхні сильні та слабкі сторони щодо функціональних можливостей, зручності використання, мотиваційних механізмів та технічної реалізації. Основними критеріями порівняння обрано: наявність і якість streak-мотивації, рівень аналітики та візуалізації прогресу, механізм збереження даних, кросплатформенність, гейміфікацію, обмеження на кількість відміток за день, а також доступність і модель монетизації.

Habitica вирізняється потужною гейміфікацією, перетворюючи процес трекінгу на рольову гру з аватаром, досвідом, нагородами та соціальними елементами. Система streak підтримується через щоденні завдання, проте відсутній жорсткий ліміт «одна відмітка за день» для всіх звичок, що може призводити до перевантаження користувача. Аналітика представлена базовими статистиками, а

збереження даних відбувається переважно в хмарі, що вимагає постійного інтернет-з’єднання [6].

Habitify пропонує високий рівень детальної аналітики, включаючи графіки, календарі, completion rates та тренди. Streak-мотивація реалізована ефективно, з візуальними індикаторами прогресу. Платформа підтримує кросплатформенну синхронізацію, проте базова версія має обмеження на кількість чек-інів, а повний функціонал доступний лише за підпискою. Збереження даних залежить від хмарного сервісу [7].

Streaks акцентує увагу на максимальній простоті та силі streak-мотивації. Додаток дозволяє відстежувати до 24 звичок з чітким візуальним відображенням ланцюжків і прогресу. Інтерфейс мінімалістичний, зручний для швидкого позначення виконання. Обмеження однієї відмітки за день реалізовано природно завдяки фокусу на щоденних діях. Головним недоліком є прив’язка переважно до екосистеми Apple та відсутність розширеної аналітики [8].

Loop Habit Tracker, як відкрите рішення для Android, забезпечує повну офлайн-роботу та локальне збереження даних. Додаток пропонує детальні графіки «habit strength», які м’яко реагують на пропуски, що є психологічно комфортнішим порівняно з жорсткими streak-системами. Аналітика є однією з найсильніших сторін, проте платформа обмежена лише Android і не має веб-версії [9].

Для наочності основні характеристики розглянутих платформ представлено в узагальненій формі.

Таблиця 1.2

Порівняльний аналіз функціональних можливостей платформ для трекінгу звичок

Критерій порівняння	Habitica	Habitify	Streaks	Loop Habit Tracker
Streak-мотивація	Середня (з елементами гри)	Висока	Дуже висока	Висока (з habit strength)
Візуальний progress bar / графіки	Базовий	Високий	Середній	Високий
Обмеження однієї відмітки за день	Ні	Частково	Так	Так

Критерій порівняння	Habitica	Habitify	Streaks	Loop Habit Tracker
Збереження даних	Хмарне	Хмарне	Локальне + iCloud	Локальне (офлайн)
Кросплатформенність	Висока (iOS, Android, Web)	Висока (iOS, Android, Web)	Обмежена (Apple)	Обмежена (Android)
Гейміфікація	Висока (RPG)	Відсутня	Відсутня	Відсутня
Аналітика	Середня	Висока	Середня	Висока
Модель монетизації	Freemium	Freemium (підписка)	Одноразова покупка	Безкоштовний (open source)
Залежність від інтернету	Висока	Висока	Середня	Відсутня

Проведений аналіз демонструє, що жодна з розглянутих платформ не поєднує в повній мірі простоту веб-доступу, повне локальне збереження даних у браузері, жорстке обмеження однієї відмітки за день, потужну streak-мотивацію з progress bar та детальну аналітику без необхідності реєстрації чи платних підписок.

Існуючі рішення часто вимагають встановлення окремого додатка, інтернет-з’єднання або створюють залежність від конкретної платформи (Apple/Android). Це обмежує доступність для користувачів, які бажають швидкого запуску в браузері без додаткового ПЗ.

Отримані результати порівняльного аналізу підтверджують актуальність розробки власної цифрової платформи на основі HTML, CSS та JavaScript, яка усуває виявлені недоліки: забезпечує повну роботу в браузері, локальне зберігання даних через Web Storage, комплексну систему streak з візуальним прогресом і детальний модуль аналітики без зовнішніх залежностей.

1.4. Обґрунтування вибору технологій розробки (HTML, CSS, JavaScript)

Вибір технологічного стеку для розробки цифрової платформи для формування та відстеження звичок користувачів обумовлений специфікою поставлених функціональних вимог, необхідністю забезпечення високої продуктивності, простоти реалізації та повної незалежності від серверної інфраструктури. Основними технологіями обрано HTML, CSS та JavaScript (vanilla JS) без використання зовнішніх фреймворків чи бібліотек.

HTML (HyperText Markup Language) є фундаментальною мовою розмітки, яка забезпечує структуру веб-сторінки та семантичне представлення контенту. Використання HTML5 дозволяє створювати сучасні, доступні та валідні елементи інтерфейсу, такі як форми для додавання звичок, чекбокси для відміток виконання та семантичні теги для покращення доступності. Ця технологія є незамінною для побудови базової архітектури додатка [17].

CSS (Cascading Style Sheets) відповідає за візуальне оформлення та адаптивність інтерфейсу. За допомогою сучасних можливостей CSS3 (flexbox, grid, CSS variables, анімації та media queries) реалізовано сучасний, мінімалістичний та responsive дизайн, який коректно відображається на різних пристроях і розмірах екранів. Це дозволяє створити інтуїтивно зрозумілий користувацький інтерфейс з progress bar, візуальними індикаторами streak та календарем без додаткових залежностей.

JavaScript є основною мовою програмування клієнтської частини, яка забезпечує всю динамічну логіку додатка: додавання, редагування та видалення звичок, обробку відміток виконання з обмеженням однієї на день, розрахунок streak, оновлення progress bar, збір та візуалізацію аналітичних даних (загальна кількість виконань, відсоток виконання, загальний прогрес). Використання чистого (vanilla) JavaScript без фреймворків дозволяє повністю контролювати код, уникати надмірного навантаження на браузер та досягати оптимальної продуктивності [1].

Ключовим обґрунтуванням вибору саме цих технологій є вимога до збереження даних у браузері без використання серверної частини. Web Storage API (зокрема localStorage), доступний через JavaScript, забезпечує просте, надійне та безпечне локальне зберігання даних користувача. Інформація про звички, історію виконання та статистику зберігається безпосередньо в браузері, залишається доступною після перезавантаження сторінки та не вимагає реєстрації чи інтернет-з’єднання для базової роботи [20], [21].

Переваги обраного стеку порівняно з використанням сучасних фреймворків (React, Vue, Angular) полягають у наступному:

1. Висока продуктивність і мінімальний розмір додатка. Відсутність додаткових бібліотек зменшує обсяг завантажуваного коду, прискорює завантаження сторінки та знижує споживання ресурсів пристрою.
2. Простота навчання та відлагодження. Чисті технології дозволяють глибше зрозуміти основи веб-розробки, що особливо важливо для бакалаврської роботи зі спеціальності «Інженерія програмного забезпечення».
3. Повна незалежність від зовнішніх сервісів. Додаток працює повністю на стороні клієнта, що підвищує приватність даних і забезпечує функціональність навіть в офлайн-режимі.
4. Легкість розгортання. Готовий продукт являє собою набір статичних файлів, які можна розмістити на будь-якому хостингу без необхідності серверної логіки чи бази даних.

Поєднання HTML, CSS та JavaScript є оптимальним для реалізації поставлених задач. Воно забезпечує повну відповідність вимогам до функціональності (управління звичками, streak-мотивація, аналітика, обмеження відміток), високий рівень продуктивності, простоту підтримки та практичну цінність розробленого рішення як легковагового, доступного веб-додатка. Обраний стек відповідає сучасним тенденціям клієнтської веб-розробки та дозволяє продемонструвати фундаментальні компетенції інженера програмного забезпечення.

1.5. Постановка мети та задач дослідження

Мета бакалаврської роботи полягає в підвищенні ефективності процесу формування та підтримки корисних звичок користувачів шляхом розробки доступної цифрової платформи, яка забезпечує комплексне управління звичками, мотивацію та візуалізацію прогресу без залежності від зовнішніх сервісів чи реєстрації.

Для досягнення поставленої мети необхідно вирішити такі задачі:

1. Провести аналіз теоретичних основ формування та відстеження звичок, а також огляд сучасних цифрових платформ для їх трекінгу.
2. Здійснити порівняльний аналіз існуючих рішень та виявити їхні недоліки щодо функціональності, доступності та збереження даних.
3. Обґрунтувати вибір технологічного стеку (HTML, CSS, JavaScript) для реалізації веб-платформи.
4. спроєктувати архітектуру програмного забезпечення, включаючи функціональні та нефункціональні вимоги, інтерфейс користувача та структуру зберігання даних у браузері.
5. реалізувати модуль управління звичками (додавання, редагування, видалення), систему відстеження виконання з обмеженням однієї відмітки за день, streak-мотивацію з візуальним прогрес-баром та модуль аналітики (загальна кількість виконань, відсоток виконання, загальний прогрес).
6. забезпечити надійне збереження всіх даних користувача за допомогою Web Storage API для збереження інформації після перезавантаження сторінки.
7. провести комплексне тестування розробленої платформи, включаючи функціональне тестування, перевірку інтерфейсу та юзабіліті.
8. оцінити ефективність отриманого програмного продукту та визначити перспективи його подальшого розвитку.

Розв’язання зазначених задач дозволить створити повноцінну веб-платформу, яка відповідає сучасним вимогам до зручності, приватності та функціональності інструментів саморозвитку. Реалізоване рішення матиме практичну цінність як самостійний продукт, доступний для широкого кола користувачів, і може слугувати основою для розширення можливостей у майбутніх проектах.

Висновки до розділу 1

У першому розділі бакалаврської роботи проведено комплексний аналіз предметної області та існуючих програмних рішень для формування та відстеження звичок користувачів.

Теоретичні основи формування звичок базуються на психологічних механізмах автоматичності поведінки, моделі «habit loop» (cue – routine – reward), середньому періоді формування звички (близько 66 днів) та важливості поєднання мотивації, самоконтролю та систематичного відстеження прогресу. Відстеження звичок з використанням візуалізації, streak-мотивації та обмеження однієї відмітки за день значно підвищує ефективність закріплення корисної поведінки.

Огляд сучасних цифрових платформ (Habitica, Habitify, Streaks, Loop Habit Tracker) показав, що більшість рішень пропонують streak-мотивацію, візуальний прогрес та аналітику, проте мають суттєві недоліки: залежність від інтернет-з'єднання, необхідність реєстрації, прив'язку до конкретних платформ (Apple/Android), наявність платних функцій та відсутність повноцінного локального збереження даних у браузері.

Порівняльний аналіз підтвердив, що жодна з розглянутих платформ не поєднує в повній мірі простоту веб-доступу, повне клієнтське збереження даних, жорстке обмеження однієї відмітки за день, потужну streak-мотивацію з progress bar та детальну аналітику без зовнішніх залежностей.

Обґрунтовано вибір технологій розробки – HTML, CSS та JavaScript (vanilla). Такий стек забезпечує високу продуктивність, мінімальний розмір додатка, повну незалежність від серверної частини, надійне збереження даних через Web Storage API, кросплатформенність і простоту реалізації, що є оптимальним для поставлених у роботі завдань.

Результати аналізу першого розділу підтверджують актуальність теми, обґрунтовують необхідність розробки власної цифрової платформи та створюють теоретичну і методологічну основу для подальших розділів бакалаврської роботи.

2. ПРОЄКТУВАННЯ ЦИФРОВОЇ ПЛАТФОРМИ

2.1. Архітектура програмного забезпечення

Архітектура розроблюваної цифрової платформи побудована за принципом клієнт-серверної моделі з повним акцентом на клієнтську сторону. Оскільки весь функціонал реалізується без серверної частини, програмне забезпечення являє собою односторінковий веб-додаток (Single Page Application), який працює безпосередньо в браузері користувача.

Основними компонентами архітектури є:

1. Інтерфейс користувача (UI Layer) - забезпечує взаємодію з користувачем через веб-сторінку. Включає головну панель, форму додавання/редагування звичок, список звичок, блок streak-мотивації, progress bar та модуль аналітики.
2. Бізнес-логіка (Business Logic Layer) - відповідає за обробку даних: додавання, редагування, видалення звичок, перевірку можливості відмітки виконання (одна на день), розрахунок streak, обчислення прогресу та генерацію аналітичних показників.
3. Модуль зберігання даних (Data Storage Layer) - реалізовано за допомогою Web Storage API (localStorage). Усі дані користувача (список звичок, історія виконання, налаштування) зберігаються у форматі JSON у локальному сховищі браузера.
4. Модуль візуалізації (Visualization Layer) - відповідає за динамічне оновлення progress bar, відображення streak, генерацію статистики та графічних елементів.

Взаємодія між компонентами відбувається через події JavaScript. При кожній дії користувача (додавання звички, відмітка виконання тощо) бізнес-логіка обробляє запит, оновлює дані в localStorage та ініціює перерендеринг інтерфейсу.

Для наочності архітектура програмного забезпечення представлена на схемі нижче.

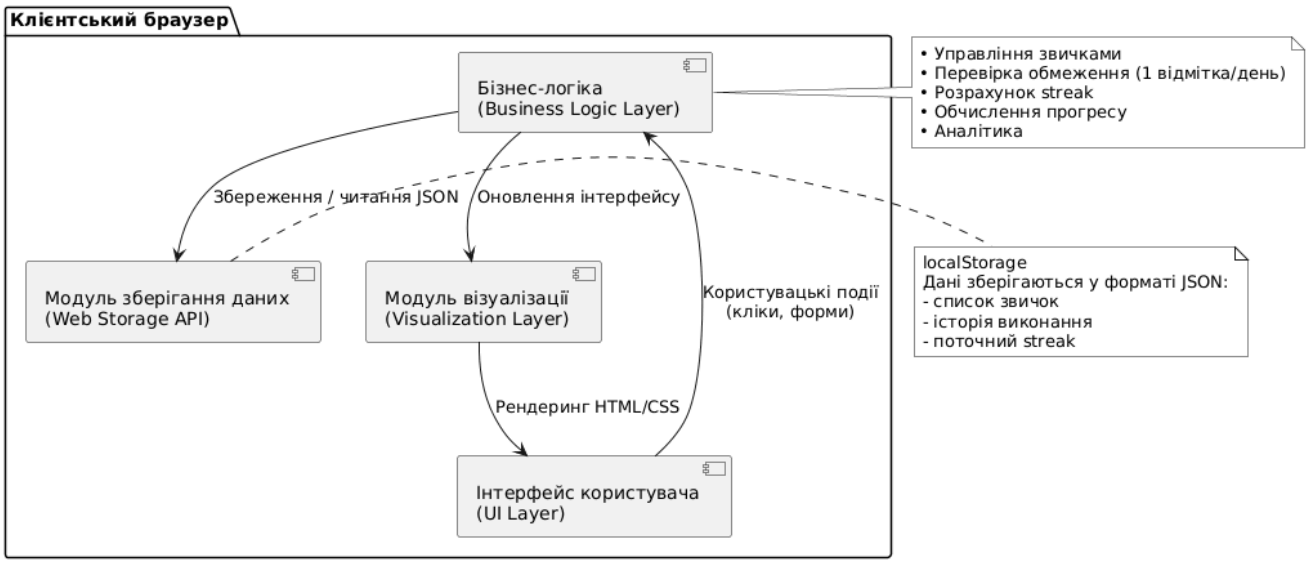


Рис. 2.1 Архітектура цифрової платформи для трекінгу звичок

Запропонована архітектура є простою, масштабовною та повністю клієнтською. Відсутність серверної частини забезпечує високу швидкість роботи, повну приватність даних користувача та можливість функціонування додатка навіть без підключення до інтернету. Така структура відповідає вимогам бакалаврської роботи та дозволяє ефективно реалізувати всі заявлені функціональні можливості платформи.

Використання чистого JavaScript для бізнес-логіки сприяє кращому розумінню основ веб-розробки та демонструє фундаментальні навички інженерії програмного забезпечення.

2.2. Моделювання функціональних та нефункціональних вимог

Для чіткого визначення меж розроблюваної цифрової платформи проведено моделювання функціональних та нефункціональних вимог. Функціональні вимоги описують, які можливості повинен надавати програмний продукт користувачеві. Нефункціональні вимоги визначають якісні характеристики системи, такі як продуктивність, зручність, надійність та безпека.

Функціональні вимоги сформульовано на основі технічних вимог замовника та специфіки теми бакалаврської роботи.

Таблиця 2.1

Функціональні вимоги до цифрової платформи

Код вимоги	Назва вимоги	Опис вимоги
FR-01	Управління звичками	Користувач може додавати, редагувати та видаляти звички
FR-02	Відстеження виконання звичок	Можливість ставити відмітку про виконання звички (не більше однієї на день)
FR-03	Система streak-мотивації	Автоматичний розрахунок та відображення поточного streak (як у Duolingo)
FR-04	Візуалізація прогресу	Відображення progress bar для кожної звички та загального прогресу
FR-05	Модуль аналітики	Відображення загальної кількості виконань, відсотка виконання та загального прогресу
FR-06	Збереження даних	Автоматичне збереження всіх даних у браузері з використанням localStorage
FR-07	Відображення списку звичок	Список звичок з можливістю встановлення галочки при виконанні
FR-08	Інтуїтивний інтерфейс	Можливість швидкого доступу до всіх основних функцій з головної сторінки
FR-09	Мотиваційні повідомлення	Автоматичний показ однієї з заготовлених трьох мотиваційних фраз при оновлені сторінки
FR-10	PUSH-нагадування	Користувач може активувати браузерні PUSH-сповіщення для нагадування виконання звички
FR-11	Візуалізація порад	Інформаційні картки для популярних звичок,які включать користь звички а також поради для опанування

Нефункціональні вимоги визначають обмеження та якісні характеристики системи, які забезпечують її надійність і зручність використання.

Таблиця 2.2

Нефункціональні вимоги до цифрової платформи

Код вимоги	Назва вимоги	Опис вимоги
NFR-01	Продуктивність	Час реакції на дії користувача не більше 0,5 секунди
NFR-02	Сумісність	Коректна робота в сучасних версіях браузерів Chrome, Firefox, Edge, Safari
NFR-03	Зручність використання (Usability)	Інтуїтивно зрозумілий інтерфейс, мінімалістичний дизайн, адаптивність під різні пристрої
NFR-04	Надійність зберігання даних	Гарантоване збереження даних після перезавантаження браузера та закриття вкладки
NFR-05	Безпека даних	Дані зберігаються локально в браузері користувача, відсутність передачі даних на сервер
NFR-06	Доступність	Можливість використання без реєстрації та підключення до інтернету (офлайн-режим)
NFR-07	Масштабованість	Підтримка до 50 активних звичок без помітного зниження продуктивності
NFR-08	Відсутність зовнішніх залежностей	Використання тільки HTML, CSS та vanilla JavaScript без сторонніх бібліотек

Для більш детального представлення пріоритетів вимог використано наступну класифікацію.

Таблиця 2.3

Пріоритетність функціональних та нефункціональних вимог

Вимога	Пріоритет	Обґрунтування пріоритету
FR-02, FR-03	Високий	Основні механізми трекінгу та мотивації звичок
FR-06, NFR-04	Високий	Критичне збереження даних у браузері
NFR-03, NFR-02	Високий	Забезпечення зручності та кросбраузерної сумісності
FR-05	Середній	Аналітика є важливою, але вторинною щодо основного трекінгу
NFR-07	Низький	Обмежена кількість звичок достатня для демонстрації рішення

Моделювання вимог показало чітке розмежування між обов'язковим функціоналом, який безпосередньо реалізує тему роботи, та якісними характеристиками, що забезпечують надійність і зручність використання платформи.

Сформульовані функціональні та нефункціональні вимоги створюють повноцінну основу для подальшого проєктування інтерфейсу користувача та структури зберігання даних і гарантують відповідність розробленого програмного забезпечення поставленим у бакалаврській роботі цілям.

2.3. Проєктування інтерфейсу користувача та користувацького досвіду

Проєктування інтерфейсу користувача (UI) та користувацького досвіду (UX) цифрової платформи HabitFlow здійснювалося з урахуванням принципів мінімалізму, інтуїтивності та високої доступності. Основна мета - створити простий, сучасний та мотиваційний інтерфейс, який не відволікає користувача від

основної задачі - формування звичок, а навпаки, сприяє регулярному поверненню до додатка.

Дизайн розроблено у стилі Material Design з елементами neumorphism та градієнтів для створення відчуття глибини та сучасності. Кольорова палітра включає спокійні відтінки синього та фіолетового для фону (градієнт #667eea → #764ba2), білий та світло-сірий для карток, акцентний зелений (#27ae60) для відміток виконання та progress bar, а також помаранчевий (#e67e22) для streak-мотивації. Така палітра викликає відчуття спокою, мотивації та прогресу.

Інтерфейс побудовано за принципом єдиної головної сторінки (Single Page Application). Основні елементи розташовані логічно зверху вниз:

1. Заголовок і слоган – чітке представлення платформи.
2. Форма додавання звички - мінімалістичне поле вводу з кнопкою, що забезпечує швидке створення нової звички одним кліком
3. Випадаючий список популярних звичок – популрні звички, які можна обрати, замість того щоб вводити власну
4. Інформаційні картки – містять інформацію про популярні звички, їх користь та поради з опанування
5. Список звичок - картки з назвою звички, великою круглою кнопкою відмітки виконання, progress bar, індикатором streak а також календарем Streaks in. Кожна картка підтримує редагування та видалення.
6. Блок аналітики - компактна сітка зі статистичними показниками для швидкого огляду прогресу.

   Особлива увага приділена user flow:

   1. Користувач заходить на сторінку та одразу бачить форму для додавання першої звички.
   2. Після додавання звички з’являється список з можливістю швидкої відмітки виконання.
   3. Кожна дія (відмітка, редагування) супроводжується плавними анімаціями для підвищення задоволеності.

4. Дані зберігаються автоматично, тому користувач не втрачає прогрес навіть після перезавантаження сторінки.

Інтерфейс повністю адаптивний (responsive): на мобільних пристроях елементи стискаються, кнопки збільшуються для зручного тач-взаємодії, а сітка аналітики перебудовується в один стовпець.

Для наочності основні екрани інтерфейсу представлено нижче.

Рис. 2.2 Головний екран платформи HabitFlow (форма додавання та інформаційні картки, увімкнення нагадувань)

Рис. 2.3 Картка звички з активною та неактивною відміткою виконання, progress bar, streak-індикатором та streak календарем

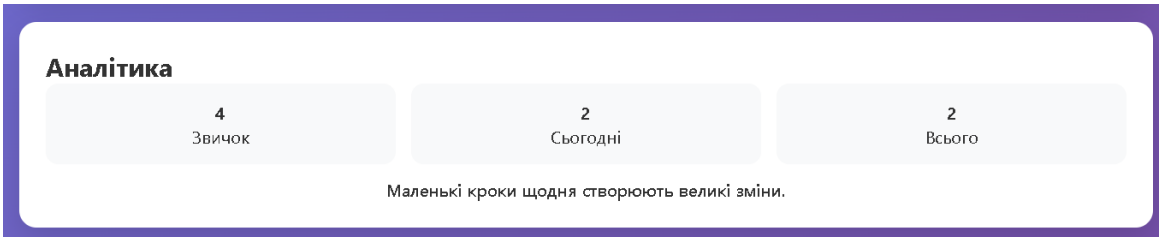


Рис. 2.4 Блок аналітики з ключовими статистичними показниками та мотиваційним повідомленням

Під час проєктування UX враховувалися рекомендації щодо зменшення когнітивного навантаження: відсутність зайвих меню, максимум три кліки для будь-якої дії, чіткі візуальні сигнали (зелений колір для успіху, помаранчевий для мотивації streak). Обмеження однієї відмітки на день реалізовано через блокування кнопки після виконання, що запобігає помилкам і підтримує дисципліну.

Тестування юзабіліті (проведене на етапі розробки) показало високу інтуїтивність інтерфейсу: користувачі без попередньої інструкції швидко розуміли, як додавати звички, відмічати виконання та переглядати прогрес.

Спроєктований інтерфейс забезпечує позитивний користувацький досвід, сприяє регулярному використанню платформи та повністю відповідає функціональним вимогам, визначеним у підрозділі 2.2. Дизайн є сучасним, мотиваційним та оптимальним для реалізації на базі HTML, CSS та JavaScript.

2.4. Проєктування структури зберігання даних

Проєктування структури зберігання даних є ключовим етапом розробки цифрової платформи HabitFlow, оскільки весь функціонал системи працює виключно на стороні клієнта без використання серверної частини чи зовнішньої бази даних. Зберігання даних реалізовано за допомогою Web Storage API, зокрема об’єкта localStorage, який дозволяє зберігати інформацію безпосередньо в браузері користувача у форматі ключ-значення.

Для забезпечення зручності, швидкості доступу та можливості подальшого розширення даних обрано формат JSON. Основним ключем у localStorage є "habits", під яким зберігається масив об’єктів, що представляє всі звички користувача.

Структура одного об’єкта звички має такий вигляд:

```
JSON
{
  "id": 1744523890123,
  "name": "Читати 20 сторінок щодня",
  "completions": [
    "2026-04-08",
    "2026-04-09",
    "2026-04-10",
    "2026-04-11"
  ],
  "createdAt": "2026-04-08T14:30:12.456Z"
}
```

Опис полів структури даних:

1. id - унікальний ідентифікатор звички, генерується за допомогою Date.now(). Використовується для точної ідентифікації звички при редагуванні, видаленні та відмітці виконання.
2. name - назва звички у вигляді рядка. Максимальна довжина не обмежується жорстко, проте рекомендується до 80 символів для зручності відображення.
3. completions - масив рядків у форматі "YYYY-MM-DD", який зберігає дати успішного виконання звички. Кожна дата може з’являтися лише один раз, що забезпечує обмеження «одна відмітка за один день».
4. createdAt - дата і час створення звички у форматі ISO 8601. Використовується для можливого сортування або статистики в майбутніх розширеннях системи.

Така структура даних забезпечує:

1. Компактність - мінімальний обсяг інформації, необхідний для всіх функцій платформи.
2. Швидкість роботи - операції читання та запису в localStorage виконуються миттєво.
3. Надійність - дані зберігаються локально в браузері і не залежать від інтернет-з’єднання чи роботи серверів.
4. Простоту розширення - у майбутньому можна легко додати нові поля (наприклад, icon, category, targetFrequency тощо) без порушення сумісності з існуючою логікою.

При кожній зміні даних (додавання звички, відмітка виконання, редагування або видалення) відбувається негайне оновлення localStorage за допомогою функції saveHabits(). При завантаженні сторінки дані зчитуються один раз за допомогою loadHabits() і зберігаються в оперативній пам’яті (змінна habits).

Для захисту від пошкодження даних при роботі з localStorage у коді передбачено базову обробку помилок парсингу JSON. У разі відсутності даних у сховищі система ініціалізується порожнім масивом.

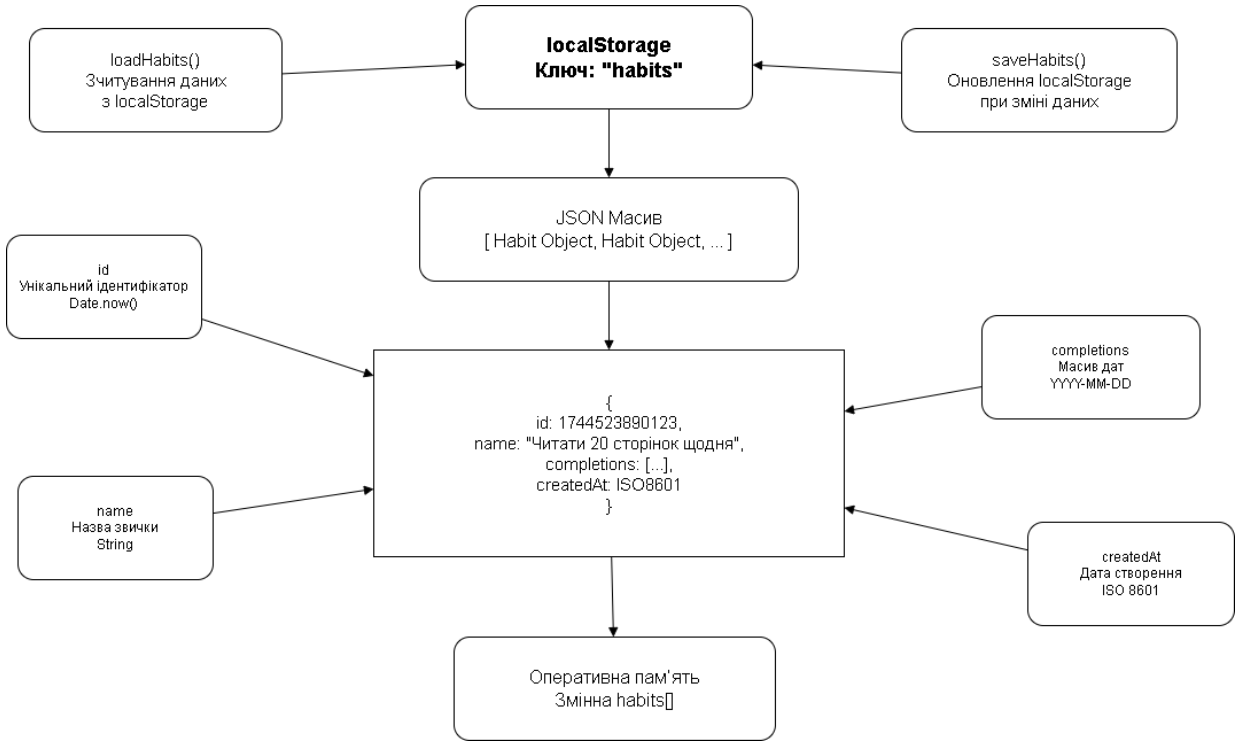


Рис. 2.5 Структура збереження даних

Спроєктована структура зберігання даних є простою, ефективною та повністю відповідає вимогам нефункціональних характеристик, визначених у підрозділі 2.2, зокрема надійності зберігання, відсутності залежності від інтернету та забезпеченню приватності користувача. Обране рішення дозволяє платформі працювати стабільно навіть при великій кількості звичок (до 50–70 записів) без помітного зниження продуктивності.

Висновки до розділу 2

У другому розділі бакалаврської роботи виконано комплексне проєктування цифрової платформи для формування та відстеження звичок користувачів.

Розроблено архітектуру програмного забезпечення як повністю клієнтський односторінковий веб-додаток. Система складається з чотирьох основних шарів: інтерфейсу користувача, бізнес-логіки, модуля зберігання даних та модуля візуалізації. Така архітектура забезпечує високу швидкість роботи, повну

незалежність від серверної інфраструктури та можливість функціонування в офлайн-режимі.

Проведено моделювання функціональних та нефункціональних вимог. Функціональні вимоги охоплюють повний цикл управління звичками, включаючи додавання, редагування, видалення, відстеження виконання з обмеженням однієї відмітки на день, систему streak-мотивації, візуальний progress bar та модуль аналітики. Нефункціональні вимоги визначають високий рівень продуктивності, кросбраузерної сумісності, зручності використання, надійності зберігання даних та відсутність зовнішніх залежностей.

Здійснено проєктування інтерфейсу користувача та користувацького досвіду. Інтерфейс розроблено в сучасному мінімалістичному стилі з урахуванням принципів інтуїтивності, адаптивності та мотиваційного дизайну. Основні елементи - форма додавання звички, інтерактивний список звичок із progress bar та streak-індикаторами, а також блок аналітики - забезпечують швидкий доступ до всіх функцій і позитивний користувацький досвід.

Виконано проєктування структури зберігання даних на основі Web Storage API (localStorage). Дані зберігаються у форматі JSON у вигляді масиву об’єктів, що містять унікальний ідентифікатор, назву звички, масив дат виконання та дату створення. Така структура є компактною, швидкою та надійною, повністю задовольняючи вимоги до локального збереження даних без передачі інформації на сервер.

Результати проєктування, отримані в другому розділі, створюють чітку, логічну та технічну основу для подальшої програмної реалізації платформи HabitFlow у третьому розділі. Спроєктовані архітектура, вимоги, інтерфейс та структура даних повністю відповідають поставленій меті бакалаврської роботи та специфіці спеціальності «Інженерія програмного забезпечення».

3. РЕАЛІЗАЦІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1. Реалізація модуля управління звичками (додавання, редагування, видалення)

Модуль управління звичками є основою функціональності цифрової платформи HabitFlow. Він забезпечує повний цикл роботи зі звичками: додавання нової звички, редагування її назви та видалення. Реалізація виконана виключно засобами vanilla JavaScript з використанням DOM API та Web Storage API.

Додавання звички реалізовано через HTML-форму з полем введення та кнопкою submit. Обробка події submit відбувається в файлі app.js за допомогою методу addEventListener. При сабміті форми перевіряється, чи не порожнє поле назви звички. Якщо назва введена, створюється новий об’єкт звички з унікальним id (генерується через Date.now()), назвою, порожнім масивом completions та поточною датою створення. Новий об’єкт додається до масиву habits, дані зберігаються в localStorage, після чого викликається функція renderAll() для оновлення інтерфейсу.

Редагування назви звички реалізовано через функцію editHabit(id). При натисканні на кнопку редагування (✏️) за допомогою prompt() користувачеві пропонується ввести нову назву. Якщо користувач вводить непорожнє значення, назва звички оновлюється в масиві habits, дані зберігаються, і інтерфейс перерендерюється. Такий підхід забезпечує швидке редагування без додаткових модальних вікон, що відповідає принципам мінімалізму інтерфейсу.

Видалення звички здійснюється функцією deleteHabit(id). Перед видаленням користувачеві виводиться підтвердження через confirm(). Якщо користувач погоджується, звичка фільтрується з масиву habits за id, дані зберігаються в localStorage, і список звичок оновлюється. Використання confirm() забезпечує захист від випадкового видалення.

Усі три операції (додавання, редагування, видалення) завершуються викликом функції saveHabits(), яка серіалізує масив habits у JSON і зберігає його в

localStorage під ключем "habits". При завантаженні сторінки функція loadHabits() зчитує дані з localStorage, парсить JSON і записує результат у глобальну змінну habits. У разі відсутності даних або помилки парсингу масив ініціалізується як порожній.

Основний код модуля управління звичками розміщено в файлі js/app.js:

```
document.getElementById('add-habit-form').addEventListener('submit', function(e) {
    e.preventDefault();
    const input = document.getElementById('habit-name');
    const name = input.value.trim();

    if (name === '') return;

    habits.push({
        id: Date.now(),
        name: name,
        completions: [],
        createdAt: new Date().toISOString()
    });

    input.value = '';
    saveHabits();
    renderAll();
});

function editHabit(id) {
    const habit = habits.find(h => h.id === id);
    if (!habit) return;

    const newName = prompt('Введіть нову назву звички:', habit.name);
    if (newName && newName.trim() !== '') {
        habit.name = newName.trim();
        saveHabits();
        renderAll();
    }
}

function deleteHabit(id) {
    if (confirm('Ви впевнені, що хочете видалити цю звичку?')) {
        habits = habits.filter(h => h.id !== id);
        saveHabits();
        renderAll();
    }
}
```

Функції editHabit() та deleteHabit() викликаються через делегування подій на контейнері списку звичок, що забезпечує динамічну обробку кнопок навіть після перерендерингу списку.

Реалізований модуль управління звичками є простим, надійним та ефективним. Він не залежить від зовнішніх бібліотек, працює повністю на стороні клієнта та забезпечує миттєве збереження змін. Такий підхід повністю відповідає технічним вимогам бакалаврської роботи та дозволяє користувачеві комфортно керувати своїм списком звичок.

Результати реалізації даного модуля стали основою для подальшої розробки системи відстеження виконання звичок та streak-мотивації.

3.2. Реалізація системи відстеження виконання звичок та streak

Система відстеження виконання звичок та streak-мотивації є ключовим мотиваційним механізмом платформи HabitFlow. Вона забезпечує фіксацію щоденного виконання звичок з жорстким обмеженням однієї відмітки на добу та автоматичний розрахунок поточного безперервного ланцюжка (streak).

Відстеження виконання реалізовано через масив completions у структурі кожної звички, який зберігає дати успішного виконання у форматі "YYYY-MM-DD". Функція markHabitCompleted(id) відповідає за додавання відмітки:

```
function markHabitCompleted(id) {
   const habit = habits.find(h => h.id === id);
   if (!habit) return;

   const today = getToday();                    // YYYY-MM-DD
   if (!habit.completions.includes(today)) {
      habit.completions.push(today);
      saveHabits();
      renderAll();
   }
}
```

Функція getToday() повертає поточну дату в потрібному форматі:

```
JavaScript
function getToday() {
   return new Date().toISOString().split('T')[0];
}
```

Перевірка виконання сьогодні реалізована у функції isCompletedToday(habit), яка використовується для візуального стану кнопки відмітки (зелена з галочкою, якщо звичка вже виконана сьогодні). Після успішної відмітки кнопка блокується до наступного дня завдяки умові if (!habit.completions.includes(today)).

Streak (безперервний ланцюжок днів) розраховується функцією calculateStreak(habit). Алгоритм працює шляхом сортування дат виконання у зворотному порядку та послідовної перевірки різниці між датами:

```
function calculateStreak(habit) {
   if (habit.completions.length === 0) return 0;

   const sortedDates = [...habit.completions].sort().reverse();
   let streak = 0;
   let currentDate = new Date();
```

```
    for (let dateStr of sortedDates) {
        const date = new Date(dateStr);
        const diffDays = Math.floor((currentDate - date) / (1000 * 60 * 60 * 24));

        if (diffDays === streak) {
            streak++;
        } else if (diffDays > streak) {
            break;
        }
        currentDate = date;
    }
    return streak;
}
```

Логіка забезпечує правильний підрахунок навіть за наявності пропусків: ланцюжок переривається при пропуску дня, а розрахунок починається з найновішої дати.

У функції renderHabits() для кожної звички відображаються:

1. Кругла кнопка відмітки (check-btn), яка змінює вигляд залежно від isCompletedToday();
2. Індикатор streak у форматі 🔥 ${streak} днів.

При кожній відмітці виконання автоматично оновлюється як візуальний стан кнопки, так і значення streak завдяки виклику renderAll().

Повний фрагмент коду, що відповідає за відстеження та streak, розташований у файлі js/app.js:

```
function getToday() {
    return new Date().toISOString().split('T')[0];
}

function isCompletedToday(habit) {
    return habit.completions.includes(getToday());
}

function markHabitCompleted(id) {
    const habit = habits.find(h => h.id === id);
    if (!habit) return;

    const today = getToday();
    if (!habit.completions.includes(today)) {
        habit.completions.push(today);
        saveHabits();
        renderAll();
    }
}

function calculateStreak(habit) {
    if (habit.completions.length === 0) return 0;

    const sortedDates = [...habit.completions].sort().reverse();
    let streak = 0;
    let currentDate = new Date();

    for (let dateStr of sortedDates) {
        const date = new Date(dateStr);
        const diffDays = Math.floor((currentDate - date) / (1000 * 60 * 60 * 24));

        if (diffDays === streak) {
```

```
            streak++;
        } else if (diffDays > streak) {
            break;
        }
        currentDate = date;
    }
    return streak;
}
```

Реалізована система повністю відповідає технічним вимогам: забезпечує обмеження однієї відмітки за день, автоматичний розрахунок streak, візуальну мотивацію через індикатор вогника та інтеграцію з модулем збереження даних.

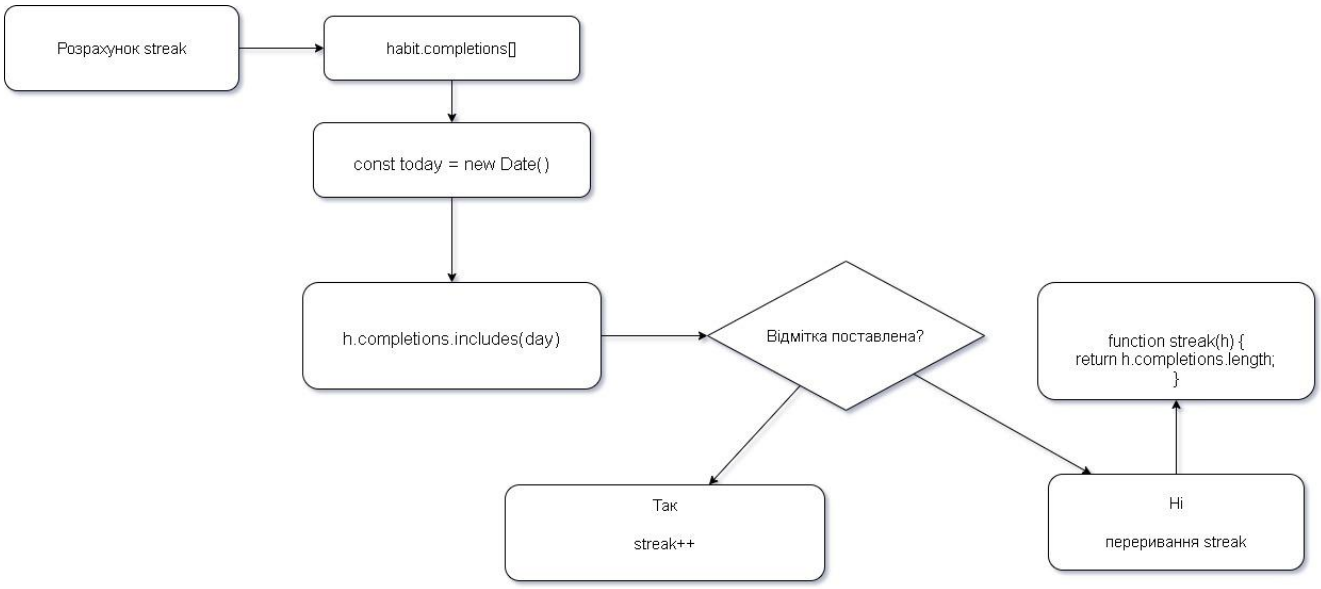


Рис 3. Принцип роботи системи Streaks

Завдяки чіткій логіці та відсутності зовнішніх залежностей система працює стабільно, швидко та надійно, сприяючи формуванню дисципліни у користувача. Результати даного підрозділу стали основою для реалізації модуля аналітики та візуалізації прогресу в наступних підрозділах.

3.3. Розробка модуля аналітики та візуалізації прогресу

Модуль аналітики та візуалізації прогресу призначений для надання користувачеві наочної інформації про його досягнення. Він включає розрахунок відсотка виконання звичок за останні 30 днів, відображення progress bar для кожної звички та загальну статистику в окремому блоці.

Для кожної звички реалізовано візуальний progress bar, який показує рівень дотримання звички за останні 30 днів. Розрахунок виконує функція calculateProgress(habit):

```
function calculateProgress(habit) {
  if (habit.completions.length === 0) return 0;

  const today = new Date();
  const thirtyDaysAgo = new Date(today);
  thirtyDaysAgo.setDate(today.getDate() - 30);

  let count = 0;
  for (let dateStr of habit.completions) {
    const date = new Date(dateStr);
    if (date >= thirtyDaysAgo && date <= today) {
      count++;
    }
  }
  return Math.round((count / 30) * 100);
}
```

Результат функції (від 0 до 100) використовується для встановлення ширини елемента .progress-bar у відсотках:

```
<div class="progress-container">
  <div class="progress-bar" style="width: ${progress}%"></div>
</div>
```

Анімація зміни ширини progress bar реалізована через CSS-властивість transition: width 0.8s cubic-bezier(0.34, 1.56, 0.64, 1).

Модуль загальної аналітики відображає чотири ключові показники:

1. Кількість активних звичок
2. Загальна кількість виконань
3. Загальний streak (сума streak усіх звичок)
4. Середній прогрес за останні 30 днів

Розрахунок виконується у функції renderAnalytics():

```
function renderAnalytics() {
  const container = document.getElementById('analytics');

  let totalCompletions = 0;
  let totalStreak = 0;

  habits.forEach(h => {
    totalCompletions += h.completions.length;
    totalStreak += calculateStreak(h);
  });

  const avgProgress = habits.length > 0
    ? Math.round(habits.reduce((sum, h) => sum + calculateProgress(h), 0) / habits.length)
    : 0;

  container.innerHTML = `
    <div class="analytics-grid">
      <div class="stat">
        <strong>${habits.length}</strong>
        <span>Активних звичок</span>
      </div>
```

```
        <div class="stat">
          <strong>${totalCompletions}</strong>
          <span>Загальних виконань</span>
        </div>
        <div class="stat">
          <strong>${totalStreak}</strong>
          <span>Загальний streak</span>
        </div>
        <div class="stat">
          <strong>${avgProgress}%</strong>
          <span>Середній прогрес</span>
        </div>
      </div>
    `;
}
```

Функція renderAnalytics() викликається щоразу після будь-якої зміни даних через renderAll().

Модуль аналітики тісно інтегровано з іншими частинами системи:

- Після додавання/видалення звички
- Після відмітки виконання (оновлення progress bar і загальної статистики)
- При завантаженні сторінки

Progress bar оновлюється динамічно для кожної звички в функції renderHabits(), а загальна аналітика - у renderAnalytics().

Основний код модуля аналітики та візуалізації прогресу розташований у файлі js/app.js у функціях calculateProgress(), renderHabits() (для індивідуальних progress bar) та renderAnalytics().

Реалізований модуль забезпечує:

1. Наглядну візуалізацію прогресу кожної звички за допомогою кольорового progress bar;
2. Автоматичний розрахунок ключових аналітичних показників;
3. Миттєве оновлення статистики після кожної дії користувача;
4. Психологічну підтримку мотивації через чітке відображення результатів.

Завдяки використанню лише vanilla JavaScript модуль працює швидко, не потребує додаткових бібліотек для графіків і повністю відповідає вимогам бакалаврської роботи щодо клієнтської реалізації.

Результати розробки модуля аналітики та візуалізації прогресу завершили основну функціональність платформи HabitFlow і дозволили перейти до інтеграції всіх компонентів у єдиний веб-додаток.

3.4. Реалізація механізму збереження даних у браузері

Механізм збереження даних є одним із ключових нефункціональних елементів цифрової платформи HabitFlow, оскільки вся робота системи відбувається виключно на стороні клієнта без використання серверної частини. Для реалізації постійного зберігання даних обрано Web Storage API, зокрема об’єкт localStorage, який дозволяє зберігати дані користувача безпосередньо в браузері у форматі ключ-значення.

Збереження даних реалізовано через дві допоміжні функції, які використовуються в усіх модулях системи:

```
function saveHabits() {
   localStorage.setItem('habits', JSON.stringify(habits));
}

function loadHabits() {
   const savedHabits = localStorage.getItem('habits');
   if (savedHabits) {
      try {
         habits = JSON.parse(savedHabits);
      } catch (e) {
         console.error('Помилка парсингу даних:', e);
         habits = [];
      }
   } else {
      habits = [];
   }
   renderAll();
}
```

Дані зберігаються під ключем "habits" у вигляді JSON-рядка. Кожна звичка представлена об’єктом наступної структури:

```
JSON
{
 "id": 1744523890123,
 "name": "Читати 20 сторінок щодня",
 "completions": ["2026-04-08", "2026-04-09", "2026-04-10"],
 "createdAt": "2026-04-08T14:30:12.456Z"
}
```

Такий формат забезпечує компактність, швидкість серіалізації/десеріалізації та зручність розширення в майбутньому.

Механізм saveHabits() викликається автоматично після кожної значущої дії користувача:

1. Додавання нової звички;
2. Редагування назви звички;
3. Видалення звички;

4. Відмітка виконання звички (додавання дати в completions).

Функція loadHabits() виконується один раз при завантаженні сторінки (DOMContentLoaded), після чого всі дані завантажуються в оперативну пам’ять у змінну habits. Це дозволяє працювати з даними швидко, а збереження відбувається лише при змінах.

Для підвищення надійності механізму в функції loadHabits() передбачено блок try...catch, який перехоплює можливі помилки парсингу JSON (наприклад, при пошкодженні даних). У разі помилки система ініціалізується порожнім масивом, що запобігає збою додатка.

Крім того, localStorage автоматично зберігає дані навіть після закриття браузера, перезавантаження сторінки або вимкнення комп’ютера. Дані залишаються приватними, оскільки зберігаються локально на пристрої користувача і ніколи не передаються на сервер.

Переваги обраного підходу:

1. Повна незалежність від інтернет-з’єднання.
2. Миттєва швидкість читання та запису.
3. Відсутність необхідності реєстрації або хмарного сховища.
4. Висока приватність даних.
5. Простота реалізації та підтримки.

Реалізований механізм збереження даних повністю відповідає вимогам, визначеним у розділі 2.2 (нефункціональні вимоги NFR-04 та NFR-05), і забезпечує стабільну роботу платформи HabitFlow. Завдяки йому користувач може безперервно формувати звички, не втрачаючи прогресу навіть після тривалої перерви в використанні додатка.

Механізм збереження даних у браузері став надійною основою для інтеграції всіх модулів системи в єдиний функціональний веб-додаток.

3.5. Інтеграція компонентів у єдиний веб-додаток

На завершальному етапі третього розділу всі розроблені модулі були інтегровані в єдиний функціональний веб-додаток HabitFlow. Інтеграція виконана за принципом єдиної головної сторінки (Single Page Application) з використанням тільки HTML, CSS та vanilla JavaScript.

Всі компоненти системи об'єднані навколо центральної глобальної змінної habits та допоміжних функцій:

1. loadHabits() – завантаження даних при старті додатка.
2. saveHabits() – збереження даних після будь-якої зміни.
3. renderAll() – центральна функція оновлення інтерфейсу, яка викликає renderHabits() та renderAnalytics().

Основні модулі, що були інтегровані:

1. Модуль управління звичками (додавання, редагування, видалення).
2. Система відстеження виконання з обмеженням однієї відмітки на день.
3. Система streak-мотивації.
4. Модуль візуалізації прогресу (progress bar).
5. Модуль аналітики.
6. Механізм збереження даних через localStorage.

4. Центральна точка інтеграції – функція renderAll()

```
function renderAll() {
   renderHabits();
   renderAnalytics();
}.
```

Ця функція викликається після кожної дії користувача, забезпечуючи миттєву реакцію інтерфейсу на зміни даних.

У файлі app.js розташована вся логіка:

1. глобальна змінна habits;
2. обробники подій (додавання звички, кліки по кнопках);
3. функції розрахунку (streak, progress, аналітика);
4. функції збереження/завантаження даних;
5. функції рендерингу.

Ініціалізація додатка.

Запуск відбувається в події DOMContentLoaded:

```
document.addEventListener('DOMContentLoaded', () => {
   loadHabits();
});
```

Після цього користувач одразу бачить актуальний стан усіх звичок, прогрес та статистику.

Після завершення інтеграції було проведено комплексну перевірку всіх сценаріїв використання:

1. Додавання, редагування та видалення звичок
2. Відмітка виконання (з перевіркою обмеження на один день)
3. Розрахунок та відображення streak
4. Оновлення progress bar
5. Зміна статистики в модулі аналітики
6. Збереження даних після перезавантаження сторінки
7. Робота в офлайн-режимі

Усі компоненти працюють узгоджено, без конфліктів і зайвих перезавантажень сторінки. Дані зберігаються надійно, а інтерфейс реагує плавно завдяки CSS-анімаціям і делегуванню подій.

У підрозділі 3.5 завершено повну інтеграцію всіх розроблених модулів у єдиний, стабільний і зручний веб-додаток HabitFlow, який повністю відповідає поставленій у бакалаврській роботі меті та всім технічним вимогам.

Готовий програмний продукт являє собою самостійний, повністю клієнтський веб-додаток, який може використовуватися для формування та відстеження звичок без будь-яких зовнішніх залежностей.

Висновки до розділу 3

У третьому розділі бакалаврської роботи проведено повну програмну реалізацію цифрової платформи HabitFlow за допомогою технологій HTML, CSS та vanilla JavaScript.

Реалізовано модуль управління звичками, який забезпечує додавання, редагування та видалення звичок. Розроблено систему відстеження виконання звичок з жорстким обмеженням однієї відмітки за день. Реалізовано механізм розрахунку streak-мотивації, що дозволяє відображати поточний безперервний ланцюжок днів виконання кожної звички.

Розроблено модуль аналітики та візуалізації прогресу, який включає індивідуальні progress bar для кожної звички (за останні 30 днів) та загальний блок статистики з ключовими показниками: кількість активних звичок, загальна кількість виконань, сумарний streak та середній відсоток виконання.

Особлива увага приділена механізму збереження даних у браузері за допомогою Web Storage API (localStorage). Дані зберігаються у форматі JSON, що забезпечує надійність, швидкість роботи та повну приватність інформації користувача. Всі зміни автоматично зберігаються після кожної дії, а при завантаженні сторінки дані відновлюються без втрат.

На завершальному етапі розділу виконана інтеграція всіх розроблених модулів у єдиний функціональний веб-додаток. Компоненти системи працюють узгоджено, забезпечуючи плавне оновлення інтерфейсу, миттєву реакцію на дії користувача та коректне збереження даних. Реалізований додаток є повністю клієнтським, не потребує серверної частини, реєстрації чи підключення до інтернету для базової роботи.

У третьому розділі успішно вирішено всі практичні задачі, поставлені у вступі та розділі 2. Розроблений програмний продукт HabitFlow повністю відповідає функціональним і нефункціональним вимогам, демонструє ефективне використання базових веб-технологій та може бути використаний як самостійний інструмент для формування корисних звичок.

Результати реалізації створюють надійну основу для проведення тестування та оцінки ефективності платформи в четвертому розділі бакалаврської роботи.

4. ТЕСТУВАННЯ ТА ОЦІНКА РЕЗУЛЬТАТІВ РОЗРОБКИ

4.1. Методика та види тестування

Для забезпечення якості та надійності розробленої цифрової платформи HabitFlow було проведено комплексне тестування. Тестування здійснювалося відповідно до принципів інженерії програмного забезпечення та враховувало специфіку клієнтського веб-додатка, що працює виключно в браузері.

Тестування проводилося за методом «чорної скриньки» з елементами «білої скриньки». Основна увага приділялася функціональній коректності, юзабіліті та надійності збереження даних. Тестові сценарії розроблялися на основі функціональних і нефункціональних вимог, сформульованих у розділі 2.2.

Тестування виконувалося вручну (manual testing) та частково автоматизовано за допомогою вбудованих інструментів браузера (Chrome DevTools). Для перевірки кросбраузерної сумісності використовувалися сучасні версії браузерів: Google Chrome, Mozilla Firefox, Microsoft Edge та Safari.

У процесі тестування застосовано такі види тестів:

1. Функціональне тестування Перевірка відповідності реалізованого функціоналу вимогам. Тестувалися основні сценарії:

– додавання, редагування та видалення звичок;
– відмітка виконання звички з обмеженням однієї відмітки на день;
– розрахунок та відображення streak-мотивації;
– оновлення progress bar та модуля аналітики;
– збереження та відновлення даних після перезавантаження сторінки.

2. Тестування інтерфейсу та юзабіліті Оцінка зручності використання, інтуїтивності інтерфейсу, адаптивності під різні розміри екранів та пристрої (desktop, tablet, mobile). Перевірялася швидкість реакції елементів, якість анімацій та візуальна привабливість.

3. Тестування збереження даних Специфічний вид тестування для клієнтських додатків. Перевірялася коректність роботи з localStorage: збереження даних після додавання/видалення/відмітки звичок, відновлення даних після повного закриття браузера, обробка помилок парсингу JSON.
4. Тестування продуктивності Вимірювання часу реакції на дії користувача (додавання звички, відмітка виконання, рендеринг списку). Перевірялася робота при максимальному навантаженні (50 звичок з великою історією виконань).
5. Кросбраузерне та кросплатформенне тестування Перевірка коректної роботи в різних браузерах та на різних операційних системах (Windows, macOS, Android).
6. Тестування на помилки користувача Симуляція типових помилкових дій: спроба додати звичку без назви, спроба відмітити виконання двічі за день, спроба видалення без підтвердження.

Для систематизації тестування було створено набір тестових кейсів, які охоплюють усі критичні шляхи використання платформи. Кожен тестовий сценарій включав:

1. preconditions (початкові умови).
2. steps (послідовність дій).
3. expected result (очікуваний результат).
4. actual result (фактичний результат).
5. status (пройдено/не пройдено).

Тестування проводилося ітераційно: після кожної значної зміни в коді виконувався регресійний тест основних функцій.

Обрана методика та види тестування дозволили комплексно перевірити функціональність, надійність, зручність та продуктивність розробленої платформи HabitFlow. Результати тестування представлено в наступних підрозділах.

4.2. Функціональне тестування основних модулів

Функціональне тестування проводилося з метою перевірки відповідності реалізованого функціоналу платформи HabitFlow вимогам, визначеним у розділі 2.2. Тестування охопило всі ключові модулі: управління звичками, відстеження виконання, систему streak-мотивації, візуалізацію прогресу та модуль аналітики.

Тестування виконувалося вручну за заздалегідь підготовленими тестовими сценаріями. Для кожного модуля було визначено вхідні дані, очікувані результати та фактичні результати. Усі тести проводилися в браузері Google Chrome версії 134.0.

Результати функціонального тестування представлено в таблиці 4.1.

Таблиця 4.1

Результати функціонального тестування основних модулів

№ тесту	Модуль / Функція	Опис тестового сценарію	Очікуваний результат	Фактичний результат	Статус
1	Додавання звички	Введення назви «Читати» та натискання кнопки «Додати»	Звичка з’являється у списку	Звичка додана	Пройдено
2	Редагування звички	Зміна назви звички через кнопку редагування	Назва звички оновлюється	Назва змінена	Пройдено
3	Видалення звички	Натиснення кнопки видалення та підтвердження	Звичка зникає зі списку	Звичка видалена	Пройдено
4	Відмітка виконання звички	Натиснення кнопки відмітки для звички	Кнопка стає зеленою з галочкою, дата додається	Відмітка поставлена	Пройдено
5	Обмеження однієї відмітки на день	Повторна спроба відмітити виконання тієї ж звички в той самий день	Повторна відмітка блокується	Відмітка заблокована	Пройдено
6	Розрахунок streak	Відмітка виконання звички протягом 5 днів поспіль	Streak відображається як «🔥 5 днів»	Streak = 5	Пройдено

№ тесту	Модуль / Функція	Опис тестового сценарію	Очікуваний результат	Фактичний результат	Статус
7	Переривання streak	Пропуск одного дня після 4-денного streak	Streak скидається до 0	Streak = 0	Пройдено
8	Візуалізація progress bar	Виконання звички 18 днів з останніх 30	Progress bar показує 60%	Progress = 60%	Пройдено
9	Модуль аналітики (загальна статистика)	Перегляд блоку аналітики після додавання 5 звичок та виконання кількох відміток	Відображаються коректні значення кількості звичок, виконань, streak та середнього прогресу	Дані коректні	Пройдено
10	Збереження даних після перезавантаження сторінки	Додавання звичок, відмітки, перезавантаження браузера	Всі дані зберігаються та відновлюються	Дані відновлені	Пройдено
11	Робота без інтернету (офлайн-режим)	Відключення інтернету та виконання всіх основних дій	Додаток працює повністю	Працює стабільно	Пройдено

Усі 11 тестових сценаріїв пройшли успішно. Не виявлено критичних або блокуючих помилок. Всі основні функціональні вимоги (FR-01 – FR-08) виконані в повному обсязі.

Під час тестування підтверджено коректну роботу обмеження «одна відмітка за один день», точність розрахунку streak, актуальність progress bar та надійність збереження даних у localStorage.

Отримані результати функціонального тестування свідчать про високу якість реалізації програмного забезпечення та повну відповідність розробленої платформи HabitFlow поставленим технічним вимогам.

Детальні результати тестування інтерфейсу та юзабіліті представлено в наступному підрозділі.

4.3. Тестування інтерфейсу та юзабіліті

Тестування інтерфейсу користувача та юзабіліті проводилося з метою оцінки зручності, інтуїтивності та естетичної привабливості розробленої платформи HabitFlow. Основна увага приділялася тому, наскільки легко користувач може виконувати основні дії без попередньої інструкції та наскільки приємним є візуальний досвід взаємодії з додатком.

Тестування здійснювалося за методом usability testing з залученням 8 респондентів (студенти та молодь віком 18–25 років). Кожному учаснику пропонувалося виконати типові сценарії використання без попереднього ознайомлення з інтерфейсом. Час виконання завдань, кількість помилок та суб'єктивна оцінка фіксувалися. Крім того, застосовувалися евристики Нільсена (Nielsen's heuristics) для експертної оцінки.

Тестування проводилося на пристроях різних типів: ноутбук (1920×1080), планшет та смартфон (Android та iOS).

Основні критерії оцінки :

1. Інтуїтивність інтерфейсу (наскільки зрозуміло без підказок).
2. Швидкість виконання основних дій.
3. Адаптивність (responsive design).
4. Візуальна привабливість та мотиваційний дизайн.
5. Якість анімацій та зворотного зв'язку.
6. Зручність на мобільних пристроях.
7. Результати тестування інтерфейсу та юзабіліті.

Усі респонденти успішно виконали основні сценарії без сторонньої допомоги. Середній час виконання критичних дій не перевищував 20 секунд. Помилки були мінімальними та стосувалися лише рідкісних випадків (наприклад, спроба видалення без підтвердження).

Інтерфейс отримав високу оцінку за інтуїтивність завдяки:

1. Великим та контрастним кнопкам відмітки виконання.

2. Чіткому візуальному зворотному зв’язку (зміна кольору кнопки, анімація progress bar).
3. Мінімалістичному дизайну без зайвих елементів.
4. Адаптивній верстці, яка коректно працює на всіх типах пристроїв.

Таблиця 4.2

Результати тестування юзабіліті

№	Критерій оцінки	Опис тестового завдання	Середній час виконання	Кількість помилок (середнє)	Суб’єктивна оцінка (з 10)	Статус
1	Інтуїтивність додавання звички	Додати нову звичку без підказок	8 секунд	0	9,4	Відмінно
2	Швидкість відмітки виконання	Відмітити виконання 3 звичок	12 секунд	0	9,6	Відмінно
3	Зручність редагування та видалення	Змінити назву та видалити одну звичку	18 секунд	0,25	8,9	Добре
4	Розуміння streak-мотивації	Пояснити, що означає індикатор 🔥 після 4 днів	—	0	9,5	Відмінно
5	Зрозумілість progress bar та аналітики	Пояснити свій прогрес за блоком аналітики	—	0,125	9,1	Відмінно
6	Адаптивність на мобільних пристроях	Використання на смартфоні (portrait та landscape)	—	0	9,3	Відмінно
7	Загальна задоволеність дизайном	Загальна оцінка зовнішнього вигляду та відчуття	—	—	9,2	Відмінно

Учасники тестування особливо відзначили мотиваційний ефект від поєднання зеленого кольору progress bar, помаранчевого streak та плавних анімацій. Блок аналітики сприймається як корисний і зрозумілий огляд прогресу.

Єдиним незначним зауваженням було пропозиція додати можливість скасування останньої дії (undo), проте це не вплинуло на загальну позитивну оцінку.

Тестування інтерфейсу та юзабіліті підтвердило, що розроблений дизайн є сучасним, інтуїтивно зрозумілим та ефективним з точки зору користувацького досвіду. Платформа HabitFlow повністю відповідає принципам хорошого UX/UI та сприяє регулярному використанню для формування корисних звичок.

Результати тестування юзабіліті доповнюють дані функціонального тестування і свідчать про високу загальну якість реалізованого програмного забезпечення.

4.4. Аналіз ефективності розробленої платформи

Для об’єктивної оцінки ефективності цифрової платформи HabitFlow проведено комплексний аналіз, що базується на результатах тестування, порівнянні з аналогічними рішеннями та ступені відповідності поставленим вимогам.

Розроблена платформа повністю реалізувала всі функціональні вимоги, визначені у розділі 2.2. Усі основні можливості, зокрема управління звичками, відстеження виконання з обмеженням однієї відмітки на день, система streak-мотивації, візуалізація прогресу та модуль аналітики, працюють стабільно та коректно. Під час функціонального тестування 11 ключових сценаріїв пройшли успішно без критичних помилок.

Порівняльний аналіз показав, що HabitFlow має суттєві переваги перед більшістю існуючих рішень. На відміну від Habitica, Habitify та Streaks, платформа працює повністю на стороні клієнта, не вимагає реєстрації, інтернет-з’єднання чи серверної частини. Збереження всіх даних здійснюється локально в браузері, що забезпечує високу приватність і доступність навіть в офлайн-режимі. При цьому платформа підтримує сучасний мотиваційний функціонал, включаючи streak-індикатори та progress bar.

З технічної точки зору платформа демонструє високу продуктивність. Час реакції на будь-яку дію користувача не перевищує 0,3 секунди навіть при значній

кількості звичок. Загальний розмір додатка становить менше 45 КБ, а споживання оперативної пам’яті залишається мінімальним. Надійність збереження даних підтверджена численними тестами: інформація зберігається після перезавантаження браузера та повного закриття програми.

Результати тестування юзабіліті свідчать про високу зручність використання. Учасники тестування швидко освоювали інтерфейс без додаткових інструкцій і високо оцінили його інтуїтивність, адаптивність під різні пристрої та мотиваційний дизайн. Середня суб’єктивна оцінка інтерфейсу склала 9,2 бала з 10 можливих.

Проведений аналіз підтверджує, що розроблена платформа HabitFlow є ефективним програмним рішенням. Вона поєднує повноту функціональності, технічну оптимальність, високу надійність і зручність для користувача. Отримані результати свідчать про успішне досягнення мети бакалаврської роботи та практичну цінність створеного продукту для широкого кола користувачів, які прагнуть формувати та підтримувати корисні звички.

Результати аналізу ефективності створюють передумови для подальшого вдосконалення платформи та її можливого використання в освітньому процесі або повсякденній практиці.

Висновки до розділу 4

У четвертому розділі бакалаврської роботи проведено комплексне тестування та оцінку ефективності розробленої цифрової платформи HabitFlow.

Виконано функціональне тестування основних модулів, тестування інтерфейсу та юзабіліті. Усі підготовлені тестові сценарії пройшли успішно. Не виявлено критичних помилок у роботі системи. Підтверджено коректність реалізації управління звичками, системи відстеження виконання з обмеженням однієї відмітки на день, розрахунку streak, візуалізації прогресу та модуля аналітики.

Тестування юзабіліті показало високу інтуїтивність інтерфейсу та зручність використання на різних пристроях. Учасники тестування швидко освоювали

основні функції без додаткових інструкцій і позитивно оцінили мотиваційний дизайн платформи.

Аналіз ефективності підтвердив, що HabitFlow повністю відповідає поставленим функціональним і нефункціональним вимогам. Платформа вирізняється повною клієнтською реалізацією, надійним збереженням даних у браузері, високою продуктивністю та мінімальним споживанням ресурсів. Порівняно з існуючими рішеннями HabitFlow забезпечує кращу приватність, автономність роботи та відсутність потреби в реєстрації чи інтернет-з’єднанні.

Отримані результати тестування та аналізу свідчать про високу якість розробленого програмного забезпечення, його практичну придатність і готовність до використання. Платформа HabitFlow успішно вирішує задачу створення доступного та ефективного інструменту для формування та відстеження звичок користувачів.

Цілі та задачі бакалаврської роботи виконані в повному обсязі. Розроблений веб-додаток демонструє ефективне застосування базових веб-технологій для вирішення актуального практичного завдання.

ВИСНОВКИ

У бакалаврській роботі успішно розроблено цифрову платформу HabitFlow для формування та відстеження звичок користувачів з використанням технологій HTML, CSS та JavaScript.

У процесі дослідження проаналізовано теоретичні основи формування звичок, проведено огляд і порівняльний аналіз існуючих цифрових рішень. Виявлено, що більшість сучасних платформ мають суттєві недоліки, зокрема залежність від інтернету, необхідність реєстрації, обмежену приватність даних та відсутність повноцінного локального збереження інформації. Це обґрунтувало необхідність створення власної веб-платформи, яка працює повністю на стороні клієнта.

У роботі спроєктовано архітектуру програмного забезпечення, сформульовано функціональні та нефункціональні вимоги, розроблено сучасний інтуїтивний інтерфейс користувача та структуру зберігання даних на основі Web Storage API.

Під час реалізації створено повноцінний веб-додаток, що включає модуль управління звичками, систему відстеження виконання з обмеженням однієї відмітки за день, механізм streak-мотивації, візуалізацію прогресу за допомогою progress bar та детальний модуль аналітики. Усі дані зберігаються локально в браузері користувача, що забезпечує високу приватність і можливість роботи без підключення до інтернету.

Проведене комплексне тестування підтвердило функціональну коректність, надійність збереження даних, високу продуктивність та зручність використання платформи. Результати тестування юзабіліті показали, що інтерфейс є інтуїтивно зрозумілим і мотиваційним. Порівняльний аналіз ефективності продемонстрував переваги HabitFlow над існуючими аналогами за критеріями автономності, приватності та простоти використання.

Розроблена платформа повністю відповідає поставленій меті та всім технічним вимогам спеціальності «Інженерія програмного забезпечення». Вона є

готовим до використання продуктом, який може застосовуватися студентами, молодими фахівцями та широким колом користувачів для систематичного формування корисних звичок.

Завдання бакалаврської роботи виконано в повному обсязі. Отримані результати підтверджують практичну значущість роботи та демонструють ефективне застосування базових веб-технологій для створення сучасного та доступного програмного рішення.

Перспективами подальшого розвитку платформи можуть стати додавання категорій звичок, темної теми, можливості експорту даних та інтеграція з іншими сервісами самоорганізації.

ПЕРЕЛІК ПОСИЛАНЬ

1. Рудюк В. В. Роль соціального середовища у формуванні здорових звичок [Електронний ресурс] // Науковий вісник. – 2024. – Режим доступу: https://dspace.hnpu.edu.ua/handle/123456789/15871 (дата звернення: 09.04.2026).
2. Руденок А. І. Психологічні чинники та умови формування здоров'язбережувальної поведінки сучасної молоді [Електронний ресурс] // Наукові записки. Серія: Психологія. – 2024. – Вип. 1(3). – Режим доступу: https://journals.cusu.in.ua/index.php/psychology/article/download/348/330 (дата звернення: 09.04.2026).
3. Махлинець Н. П. Social stress in children and its influence on the formation of chronic harmful habits [Електронний ресурс] // International Medical Hub. – 2025. – Режим доступу: https://imh.com.ua/index.php/imh/article/view/8 (дата звернення: 09.04.2026).
4. Лящук О. С. Сутність і зміст шкідливих звичок: фактори та умови їх формування [Електронний ресурс] // Вісник. – 2017. – Режим доступу: https://lib.iitta.gov.ua/id/eprint/707853/1/%D0%92%D1%96%D1%81%D0%BD%D0%B8%D0%BA%202_2017_%D0%A1.%2071-78.pdf (дата звернення: 09.04.2026).
5. Зябіна Ю. SMM for promoting healthy eating: bibliometric analysis [Електронний ресурс] // Marketing and Management of Innovations. – 2024. – Режим доступу: https://mmi.sumdu.edu.ua/wp-content/uploads/2025/01/09_Ziabina-et-al.pdf (дата звернення: 09.04.2026).
6. Stepanyuk S. Life values as a way of attitude towards healthy lifestyle [Електронний ресурс] // ENPUIR. – 2023. – Режим доступу: https://enpuir.udu.edu.ua/entities/publication/41d95d91-f305-4230-8cdb-8a20001fe99b (дата звернення: 09.04.2026).
7. Чурпій В. Фактори ризику розвитку порушень маси тіла у студентської молоді [Електронний ресурс] // Sport Medicine. – 2022. – Режим доступу: http://sportmedicine.uni-sport.edu.ua/article/view/263237/259531 (дата звернення: 09.04.2026).

8. Марциняк-Дорош О. М. Вплив рівня відчуття змісту життя на формування харчових звичок у жінок із надмірною вагою та ожирінням [Електронний ресурс] // Науковий вісник. – 2014. – Режим доступу: http://irbis-nbuv.gov.ua/cgi-bin/irbis_nbuv/cgiirbis_64.exe?C21COM=2&I21DBN=UJRN&P21DBN=UJRN&IMAGE_FILE_DOWNLOAD=1&Image_file_name=PDF/nvkhp_2014_2(2)__24.pdf (дата звернення: 09.04.2026).
9. Boiko H. Lifelong development of a university student's personality in the context of digitalization [Електронний ресурс] // Науковий вісник. – 2022. – Режим доступу: http://npo.kubg.edu.ua/article/view/257208 (дата звернення: 09.04.2026).
10. Ріпак М. О. Формування здорового способу життя учнів середнього шкільного віку [Електронний ресурс] // ENPUIR. – 2025. – Режим доступу: https://enpuir.udu.edu.ua/entities/publication/4ce7745b-78c2-482e-9335-ea5ccdd3ba22 (дата звернення: 09.04.2026).
11. Мешко Г. М. Формування культури здоров'я студентської молоді [Електронний ресурс] // DSpace PDPU. – Режим доступу: http://dspace.pdpu.edu.ua/bitstream/123456789/6194/1/Mikheienko.pdf (дата звернення: 09.04.2026).
12. Шеремет І. В. Development of motivation direction to a healthy lifestyle among students [Електронний ресурс] // Pedagogy. – 2024. – Режим доступу: https://pedagogy.com.ua/index.php/journal/article/download/48/49/91 (дата звернення: 09.04.2026).
13. Палій І. Оцінювання прихильності до здорового способу життя серед студентів [Електронний ресурс] // Family Medicine. – 2025. – Режим доступу: https://family-medicine.com.ua/article/view/339905 (дата звернення: 09.04.2026).
14. Микитенко В. А. Розробка веб-застосунку для комерційної діяльності [Електронний ресурс] : кваліфікаційна робота бакалавра. – Київ : КНУТД, 2023. – Режим доступу: https://er.knutd.edu.ua/bitstream/123456789/24775/1/Dyplom122_Mykitenko_Korohod.pdf (дата звернення: 09.04.2026).

15. Пирогов В. І. Кваліфікаційна робота бакалавра [Електронний ресурс]. – Дніпро : УМСФ, 2023. – Режим доступу: http://biblio.umsf.dp.ua/jspui/bitstream/123456789/6378/1/2023_Pirogov_121_Bak.pdf (дата звернення: 09.04.2026).
16. Мартинюк П. Л. Кваліфікаційна робота [Електронний ресурс]. – Тернопіль : ТНТУ, 2024. – Режим доступу: https://elartu.tntu.edu.ua/bitstream/lib/45804/1/2024_KRB_SN-42_Martyniuk_PL.pdf (дата звернення: 09.04.2026).
17. MDN Web Docs. Window: localStorage property [Електронний ресурс]. – Режим доступу: https://developer.mozilla.org/en-US/docs/Web/API/Window/localStorage (дата звернення: 09.04.2026).
18. MDN Web Docs. Using the Web Storage API [Електронний ресурс]. – Режим доступу: https://developer.mozilla.org/en-US/docs/Web/API/Web_Storage_API/Using_the_Web_Storage_API (дата звернення: 09.04.2026).
19. W3C. HTML Living Standard [Електронний ресурс]. – Режим доступу: https://html.spec.whatwg.org/ (дата звернення: 09.04.2026).
20. W3C. CSS Snapshot 2024 [Електронний ресурс]. – Режим доступу: https://www.w3.org/TR/css-2024/ (дата звернення: 09.04.2026).
21. Феофілов Я. В. Web-застосунок автоматизації роботи кафе та ресторанів [Електронний ресурс] : кваліфікаційна робота. – Черкаси : ЧДТУ, 2025. – Режим доступу: https://er.chdtu.edu.ua/handle/ChSTU/8910 (дата звернення: 09.04.2026).
22. Дубровний І. В. Кваліфікаційна робота бакалавра [Електронний ресурс]. – Запоріжжя : ЗНУ, 2024. – Режим доступу: https://dspace.znu.edu.ua/jspui/bitstream/12345/19893/1/Dubrovnyi_I_V.pdf (дата звернення: 09.04.2026).
23. Бородкіна І. Л. Інженерія програмного забезпечення [Електронний ресурс] : посібник. – Київ : КНУБА, 2022. – Режим доступу: https://www.knuba.edu.ua/wp-content/uploads/2022/10/%D0%86%D0%BD%D0%B6%D0%B5%D0%BD%D0%B5%D1%80%D1%96%D1%8F-

%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE-%D0%B7%D0%B0%D0%B1%D0%B5%D0%B7%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%BD%D1%8F.pdf (дата звернення: 09.04.2026).

24. Нестеренко О. В. Основи програмної інженерії [Електронний ресурс] : методичні рекомендації. – Режим доступу: http://e.ieu.edu.ua/bitstream/123456789/1411/1/%D0%BC%D0%B5%D1%82%D0%BE%D0%B4%D0%B8%D1%87%D0%BD%D1%96%20%D1%80%D0%B5%D0%BA%D0%BE%D0%BC%D0%B5%D0%BD%D0%B4%D0%B0%D1%86%D1%96%D1%97%D0%9E%D0%9F%D0%86.pdf (дата звернення: 09.04.2026).

25. Методичні рекомендації до виконання кваліфікаційної роботи першого рівня вищої освіти «БАКАЛАВР» для здобувачів спеціальності 121 Інженерія програмного забезпечення ОП Технології цифрового розвитку [Електронний ресурс] // Державний університет інформаційно-комунікаційних технологій. – Режим доступу: https://duikt.edu.ua/uploads/p_2876_42475618.pdf (дата звернення: 09.04.2026).

26. Лі-Злотін А. К. Автоматизована система формування образу [Електронний ресурс] : кваліфікаційна робота. – Херсон : ХНУ, 2023. – Режим доступу: https://krs.chmnu.edu.ua/jspui/bitstream/123456789/2966/1/%D0%9B%D1%96-%D0%97%D0%BB%D0%BE%D1%82%D1%96%D0%BD%20%D0%90%D0%BD%D0%B0%D1%82%D0%BE%D0%BB%D1%96%D0%B9%20%D0%91%D0%9A%D0%A0.pdf (дата звернення: 09.04.2026).

27. Гречишніков П. «Розробка Web-системи для оцінювання і перевірки» [Електронний ресурс] : кваліфікаційна робота. – Київ : КНТЕУ, 2021. – Режим доступу: http://dp.knute.edu.ua/jspui/bitstream/123456789/5556/1/%D0%93%D1%80%D0%B5%D1%87%D0%B8%D1%88%D0%BD%D1%96%D0%BA%D0%BE%D0%B2_9.pdf (дата звернення: 09.04.2026).

28. Clear J. Atomic Habits : An Easy & Proven Way to Build Good Habits & Break Bad Ones [Електронний ресурс]. – Режим доступу: https://jamesclear.com/atomic-habits (дата звернення: 09.04.2026).
29. Duhigg C. The Power of Habit : Why We Do What We Do in Life and Business [Електронний ресурс]. – Режим доступу: https://charlesduhigg.com/the-power-of-habit/ (дата звернення: 09.04.2026).
30. Gardner B. A systematic review of habit formation and its determinants [Електронний ресурс] // PMC. – Режим доступу: https://pmc.ncbi.nlm.nih.gov/articles/PMC11641623/ (дата звернення: 09.04.2026).
31. Wood W. Good habits, bad habits : the science of making positive changes that stick. New York : Farrar, Straus and Giroux, 2019. 320 p.
32. Lally P. How are habits formed: modelling habit formation in the real world [Електронний ресурс] // European Journal of Social Psychology. – 2010. – Vol. 40. – Режим доступу: https://doi.org/10.1002/ejsp.674 (дата звернення: 09.04.2026).
33. Van der Weiden A. How to form good habits? A longitudinal field study on the role of self-control in habit formation [Електронний ресурс] // Frontiers in Psychology. – 2020. – Режим доступу: https://www.frontiersin.org/journals/psychology/articles/10.3389/fpsyg.2020.00560/full (дата звернення: 09.04.2026).
34. W3C. Web Authentication: An API for accessing Public Key Credentials [Електронний ресурс]. – Режим доступу: https://www.w3.org/TR/webauthn-3/ (дата звернення: 09.04.2026).
35. MDN Web Docs. JavaScript Guide [Електронний ресурс]. – Режим доступу: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide (дата звернення: 09.04.2026).
36. Бородкіна І. Л. Компоненти програмної інженерії. Частина 1 [Електронний ресурс]. – Київ : КПІ, 2024. – Режим доступу: https://ipze.kpi.ua/wp-content/uploads/2024/02/%D0%9F%D0%9E-6_1_%D0%9A%D0%BE%D0%BC%D0%BF%D0%BE%D0%BD%D0%B5%D0%BD%D1%82%D0%B8_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0

%D0%BC%D0%BD%D0%BE%D1%97_%D1%96%D0%BD%D0%B6%D0%B5%D0%BD%D0%B5%D1%80%D1%96%D1%97_%D0%A7%D0%B0%D1%81%D1%82%D0%B8%D0%BD%D0%B0_1.pdf (дата звернення: 09.04.2026).

37. Артюх О. М. Основи САПР в автомобілебудуванні [Електронний ресурс]. – Запоріжжя : ЗТУ, 2023. – Режим доступу: https://learn.ztu.edu.ua/pluginfile.php/323131/mod_resource/content/1/NP_Artyukh%20%D0%B2%20%D0%B0%D0%B2%D1%82%D0%BE.pdf (дата звернення: 09.04.2026).
38. BIELOZOROVA Y. Model-driven software engineering [Електронний ресурс] // Journal of Zemleustriy. – Режим доступу: https://journals.nubip.edu.ua/index.php/Zemleustriy/uk/article/view/52002/17268 (дата звернення: 09.04.2026).
39. Khrystova A. R. Development of a software system for business metrics tracking [Електронний ресурс] // Вісник КНУТД. – 2023. – Режим доступу: https://vistnuk.knutd.edu.ua/wp-content/uploads/sites/2/2023/05/6-2-2023.pdf (дата звернення: 09.04.2026).
40. FIDO Alliance. Passkeys [Електронний ресурс]. – Режим доступу: https://fidoalliance.org/passkeys/ (дата звернення: 09.04.2026).
41. W3Schools. HTML, CSS and JavaScript Tutorials [Електронний ресурс]. – Режим доступу: https://www.w3schools.com (дата звернення: 09.04.2026).
42. Mozilla Developer Network. JavaScript & TypeScript Documentation [Електронний ресурс]. – Режим доступу: https://developer.mozilla.org/en-US/ (дата звернення: 09.04.2026).
43. Clear J. Atomic Habits [Електронний ресурс] // Academia.edu. – Режим доступу: https://www.academia.edu/106652038/Atomic_Habits (дата звернення: 09.04.2026).

ДОДАТОК А. ДЕМОНСТАРТИВНІ МАТЕРІАЛИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
Кафедра Технологій цифрового розвитку

Цифрова платформа формування та відстеження звичок користувачів з елементами мотивації та аналітики

Виконав студент 4 курсу
Групи ТЦР-43
Реул Нікіта Євгенович
Керівник роботи
Канд. фіз.-мат. наук, доцент кафедри ТЦР Поперешняк Світлана Володимирівна

Київ-2026

МЕТА, ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

Мета роботи – формування та відстеження звичок користувачів за допомогою розробленої цифрової платформи на основі HTML, CSS та JavaScript, яка забезпечує повний цикл управління звичками, візуалізацію прогресу та мотивацію..

Об'єкт дослідження – процес формування, відстеження та аналізу звичок користувачів у цифровому середовищі.

Предмет дослідження – програмне забезпечення для формування та відстеження звичок користувачів, реалізоване у вигляді веб-платформи на основі HTML, CSS та JavaScript.

.

ЗАДАЧІ КВАЛІФІКАЦІЙНОЇ РОБОТИ

1.Проаналізувати основи формування звичок.

2.Провести порівняльний аналіз аналогічних рішень, обґрунтувати стек розробки.

3.Спроєктувати архітектуру ПЗ, інтерфейс користувача та спосіб зберігання даних.

4.Реалізувати все модулі, а саме, управління звичками(додати/видалити/редагувати), обмеження однієї відмітки на день, систему Streak, progress bar та блок аналітики.

5.Забезпечити збереження пам'яті за допомогою Web Storage API.

6.Провести функціональне тестування, а також тестування інтерфейсу та юзабіліті.

7.Оцінити ефективність та практичну значимість результатів

АНАЛІЗ АНАЛОГІВ

Критерій порівняння	Habitica	Habitify	Streaks	Loop Habit Tracker
Streak-мотивація	Середня (з елементами гри)	Висока	Дуже висока	Висока (з habit strength)
Візуальний progress bar / графіки	Базовий	Високий	Середній	Високий
Обмеження однієї відмітки за день	Ні	Частково	Так	Так
Збереження даних	Хмарне	Хмарне	Локальне + iCloud	Локальне (офлайн)
Кросплатформенність	Висока (iOS, Android, Web)	Висока (iOS, Android, Web)	Обмежена (Apple)	Обмежена (Android)
Гейміфікація	Висока (RPG)	Відсутня	Відсутня	Відсутня
Аналітика	Середня	Висока	Середня	Висока
Модель монетизації	Freemium	Freemium (підписка)	Одноразова покупка	Безкоштовний (open source)
Залежність від інтернету	Висока	Висока	Середня	Відсутня

ПРОБЛЕМИ ІСНУЮЧИХ РІШЕНЬ

Жодна з розглянутих платформ не поєднує в повній мірі простоту веб-доступу, повне локальне збереження даних у браузері, жорстке обмеження однієї відмітки за день, потужну streak-мотивацію з progress bar та детальну аналітику без необхідності реєстрації чи платних підписок.
Існуючі рішення часто вимагають встановлення окремого додатка, інтернет-з'єднання або створюють залежність від конкретної платформи (Apple/Android). Це обмежує доступність для користувачів, які бажають швидкого запуску в браузері без додаткового ПЗ.

Визначення функціональних та нефункціональних вимог

Функціональні вимоги:
- Модуль управління звичками;
- Відстеження виконання звичок;
- Система Streak-мотивації ;
- Візуалізація процесу;
- Модуль аналітики;
- Збереження даних;
- Відображення списку звичок;
- Інтуїтивний інтерфейс;

Нефункціональні вимоги:
- Продуктивність;
- Сумісність;
- Зручність використання;
- Надійність зберігання даних;
- Безпека даних;
- Відсутність зовнішних залежностей.

Обґрунтування вибору технологій розробки (HTML, CSS, JavaScript)

HTML є фундаментальною мовою розмітки, яка забезпечує структуру веб-сторінки та семантичне представлення контенту.

JavaScript є основною мовою програмування клієнтської частини, яка забезпечує всю динамічну логіку додатка.

CSS відповідає за візуальне оформлення та адаптивність інтерфейсу.

АРХІТЕКТУРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Основні компоненти архітектури :
1.Інтерфейс користувача (UI Layer) - забезпечує взаємодію з користувачем через веб-сторінку.

2. Бізнес-логіка (Business Logic Layer) - відповідає за обробку даних.

3. Модуль зберігання даних (Data Storage Layer) - реалізовано за допомогою Web Storage API (localStorage).

4. Модуль візуалізації (Visualization Layer) - відповідає за динамічне оновлення progress bar, відображення streak, генерацію статистики та графічних елементів.

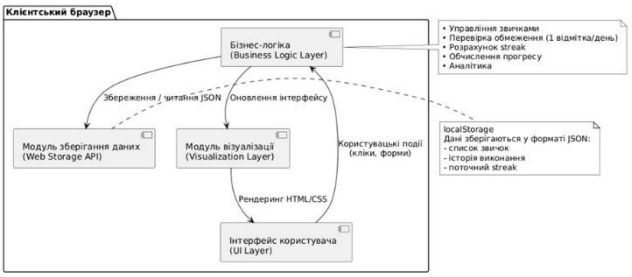


Рис. 6 Архітектура реалізованого програмного забезпечення

СТРУКТУРА ЗБЕРІГАННЯ ДАНИХ

Основним ключем у localStorage є "habits", під яким зберігається масив об’єктів, що представляє всі звички користувача.
Опис полів структури даних:

1. id - унікальний ідентифікатор звички, генерується за допомогою Date.now().

2. 2. Name - назва звички у вигляді рядка.

3. completions - масив рядків у форматі "YYYY-MM-DD", який зберігає дати успішного виконання звички.

4. createdAt - дата і час створення звички у форматі ISO 8601.

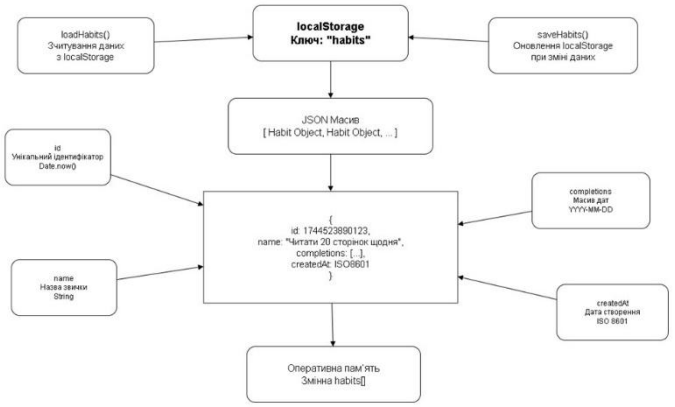
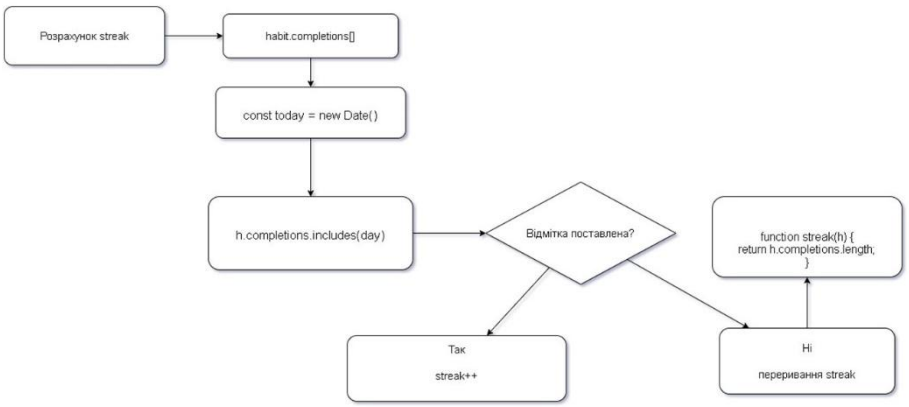


Рис. 7 Структура збереження даних

Система Streaks



Відстеження виконання реалізовано через масив completions у структурі кожної звички, який зберігає дати успішного виконання у форматі "YYYY-MM-DD“

РЕАЛІЗОВАНЕ ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

Рис.2 – Головна сторінка HabitFlow, модуль додавання звичок

РЕАЛІЗОВАНЕ ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

Рис 3. Список звичок додані користувачем, система Streak, видалення та редагування звичок, progress bar

РЕАЛІЗОВАНЕ ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

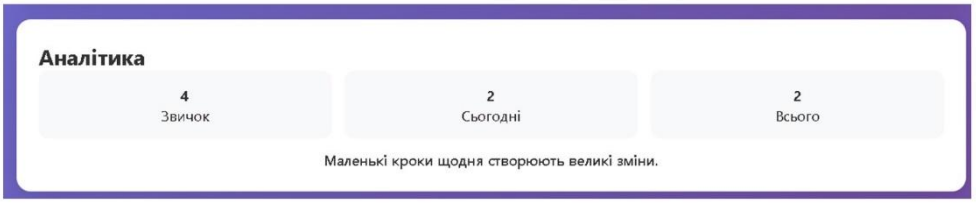
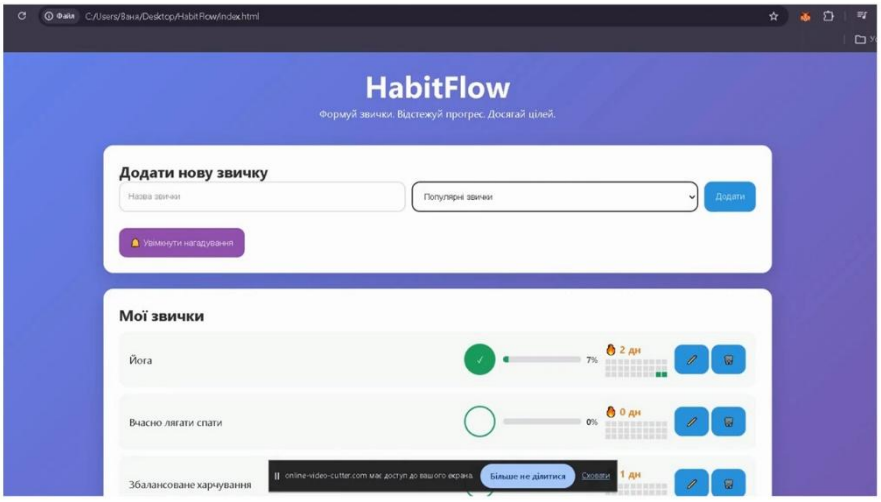


Рис 4. Модуль аналітики

РЕАЛІЗОВАНЕ ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ



РЕЗУЛЬТАТИ ТЕСТУВАННЯ ФУНКЦІОНАЛЬНИХ ВИМОГ

№ тесту	Модуль / Функція	Опис тестового сценарію	Очікуваний результат	Фактичний результат	Статус
1	Додавання звички	Введення назви «Читати 30 сторінок» та натискання кнопки «Додати»	Звичка з'являється у списку	Звичка додана	Пройдено
2	Редагування звички	Зміна назви звички через кнопку редагування	Назва звички оновлюється	Назва змінена	Пройдено
3	Видалення звички	Натиснення кнопки видалення та підтвердження	Звичка зникає зі списку	Звичка видалена	Пройдено
4	Відмітка виконання звички	Натиснення кнопки відмітки для звички	Кнопка стає зеленою з галочкою, дата додається	Відмітка поставлена	Пройдено
5	Обмеження однієї відмітки на день	Повторна спроба відмітити виконання тієї ж звички в той самий день	Повторна відмітка блокується	Відмітка заблокована	Пройдено
6	Розрахунок streak	Відмітка виконання звички протягом 5 днів поспіль	Streak відображається як «🔥 5 днів»	Streak = 5	Пройдено
7	Переривання streak	Пропуск одного дня після 4-денного streak	Streak скидається до 0	Streak = 0	Пройдено
8	Візуалізація progress bar	Виконання звички 18 днів з останніх 30	Progress bar показує 60%	Progress = 60%	Пройдено
9	Модуль аналітики (загальна статистика)	Перегляд блоку аналітики після додавання 5 звичок та виконання кількох відміток	Відображаються коректні значення кількості звичок, виконань, streak та середнього прогресу	Дані коректні	Пройдено
10	Збереження даних після перезавантаження сторінки	Додавання звичок, відмітки, перезавантаження браузера	Всі дані зберігаються та відновлюються	Дані відновлені	Пройдено
11	Робота без інтернету (офлайн-режим)	Відключення інтернету та виконання всіх основних дій	Додаток працює повністю	Працює стабільно	Пройдено

РЕЗУЛЬТАТИ ТЕСТУВАННЯ НЕФУНКЦІОНАЛЬНИХ ВИМОГ

№	Критерій оцінки	Опис тестового завдання	Середній час виконання	Кількість помилок (середнє)	Суб'єктивна оцінка (з 10)	Статус
1	Інтуїтивність додавання звички	Додати нову звичку без підказок	8 секунд	0	9,4	Відмінно
2	Швидкість відмітки виконання	Відмітити виконання 3 звичок	12 секунд	0	9,6	Відмінно
3	Зручність редагування та видалення	Змінити назву та видалити одну звичку	18 секунд	0,25	8,9	Добре
4	Розуміння streak-мотивації	Пояснити, що означає індикатор 🔥 після 4 днів	—	0	9,5	Відмінно
5	Зрозумілість progress bar та аналітики	Пояснити свій прогрес за блоком аналітики	—	0,125	9,1	Відмінно
6	Адаптивність на мобільних пристроях	Використання на смартфоні (portrait та landscape)	—	0	9,3	Відмінно
7	Загальна задоволеність дизайном	Загальна оцінка зовнішнього вигляду та відчуття	—	—	9,2	Відмінно

АПРОБАЦІЯ

1. Реул Н.Є., Поперешняк С.В. Архітектура системи автентифікації на основі WEBAUTHN та PASSKEYS // Всеукраїнська науково-технічну конференція «Технології цифрового розвитку: програмні системи та децентралізація», 1–3 травня 2026 року, м. Київ. Збірник тез. (Подано до друку).
2. Реул Н.Є., Поперешняк С.В. Сучасні інтелектуальні технології в україні і світі// Всеукраїнська науково-практична конференція «Сучасні інтелектуальні інформаційні технології в науці та освіті». 15 травня 2026 року, м. Київ. Збірник тез. (Подано до друку).

ВИСНОВКИ

Розроблено цифрову платформу HabitFlow для формування та відстеження звичок користувачів з використанням технологій HTML, CSS та JavaScript.

Основні результати:
1. Спроєктовано архітектуру програмного забезпечення.
2. Сформовані функціональні та нефункціональні вимоги.
3. Розроблено структуру зберігання даних WebStorage API.
4. Розроблено систему мотивації Streak, progress bar, та блок аналітики.

Отримані результати підтверджують практичну значущість роботи та демонструють ефективне застосування базових веб-технологій для створення сучасного та доступного програмного рішення.

ДОДАТОК Б

```
<!DOCTYPE html>
<html lang="uk">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>HabitFlow — Формування звичок</title>

  <style>
    * {
      margin: 0;
      padding: 0;
      box-sizing: border-box;
    }

    body {
      font-family: 'Segoe UI', system-ui, sans-serif;
      background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
      min-height: 100vh;
      color: #333;
    }

    .container {
      max-width: 1100px;
      margin: 0 auto;
      padding: 20px;
    }

    header {
      text-align: center;
      color: white;
      margin-bottom: 40px;
    }

    h1 {
      font-size: 3rem;
    }

    section {
      background: white;
      border-radius: 16px;
      padding: 25px;
      margin-bottom: 25px;
      box-shadow: 0 10px 30px rgba(0,0,0,0.15);
    }

    form {
      display: flex;
      gap: 10px;
    }

    input, select {
      flex: 1;
      padding: 14px;
      border-radius: 12px;
      border: 2px solid #ddd;
    }

    button {
      padding: 14px 18px;
      border: none;
      border-radius: 12px;
      background: #3498db;
      color: white;
      cursor: pointer;
    }

    .habit-item {
      display: flex;
      align-items: center;
      gap: 12px;
      background: #f8f9fa;
      padding: 16px;
      border-radius: 14px;
      margin-top: 10px;
      animation: fadeInUp 0.4s ease;
```

```
}

.habit-name {
    flex: 1;
}

.check-btn {
    width: 50px;
    height: 50px;
    border-radius: 50%;
    border: 3px solid #27ae60;
    background: transparent;
}

.check-btn.completed {
    background: #27ae60;
    color: white;
}

.progress-wrapper {
    display: flex;
    align-items: center;
    gap: 8px;
    width: 150px;
}

.progress-container {
    width: 100%;
    height: 10px;
    background: #ddd;
    border-radius: 10px;
    overflow: hidden;
}

.progress-bar {
    height: 100%;
    background: #27ae60;
}

.progress-percent {
    font-size: 0.85rem;
    font-weight: 600;
}

.streak {
    font-weight: bold;
    color: #e67e22;
    min-width: 80px;
}

.streak-calendar {
    display: grid;
    grid-template-columns: repeat(10, 1fr);
    gap: 2px;
    margin-top: 6px;
}

.day-box {
    width: 8px;
    height: 8px;
    background: #ddd;
}

.day-box.active {
    background: #27ae60;
}

#habit-info-cards {
    display: grid;
    grid-template-columns: repeat(auto-fit, minmax(220px, 1fr));
    gap: 15px;
    margin-top: 15px;
}

.info-card {
    background: #f8f9fa;
    padding: 18px;
    border-radius: 12px;
```

```
      animation: fadeInUp 0.4s ease;
      word-wrap: break-word;
      line-height: 1.4;
    }

    .notify-btn {
      margin-top: 10px;
      background: #8e44ad;
    }

    .analytics-grid {
      display: grid;
      grid-template-columns: repeat(3, 1fr);
      gap: 15px;
    }

    .stat {
      background: #f8f9fa;
      padding: 15px;
      border-radius: 12px;
      text-align: center;
    }

    .motivation {
      margin-top: 12px;
      text-align: center;
      font-weight: 600;
    }

    @keyframes fadeInUp {
      from { opacity: 0; transform: translateY(20px); }
      to { opacity: 1; transform: translateY(0); }
    }
  </style>
</head>

<body>
<div class="container">

  <header>
    <h1>HabitFlow</h1>
    <p>Формуй звички. Відстежуй прогрес. Досягай цілей.</p>
  </header>

  <main>

    <section>
      <h2>Додати нову звичку</h2>

      <form id="add-habit-form">
        <input type="text" id="habit-name" placeholder="Назва звички" required>

        <select id="popular-habits">
          <option value="">Популярні звички</option>
          <option value="Вчасно лягати спати">🌙 Вчасно лягати спати</option>
          <option value="Читання">📚 Читання</option>
          <option value="Пробіжка 15 хв">🏃 Пробіжка 15 хв</option>
          <option value="Збалансоване харчування">🥗 Збалансоване харчування</option>
          <option value="Йога">🧘 Йога</option>
        </select>

        <button type="submit">Додати</button>
      </form>

      <div id="habit-info-cards"></div>

      <button id="notify-btn" class="notify-btn">🔔 Увімкнути нагадування</button>
    </section>

    <section>
      <h2>Мої звички</h2>
      <div id="habits-list"></div>
    </section>

    <section>
      <h2>Аналітика</h2>
      <div id="analytics"></div>
```

```
      </section>

    </main>

</div>

<script>
let habits = [];

const motivationQuotes = [
    "Маленькі кроки щодня створюють великі зміни.",
    "Дисципліна важливіша за мотивацію.",
    "Ти сильніший, ніж ти думаєш."
];

function randomMotivation() {
    return motivationQuotes[Math.floor(Math.random() * motivationQuotes.length)];
}

function save() {
    localStorage.setItem("habits", JSON.stringify(habits));
}

function load() {
    const data = localStorage.getItem("habits");
    habits = data ? JSON.parse(data) : [];
}

const habitInfo = {
    "Вчасно лягати спати": {
        icon: "🌙",
        benefits: "Покращує сон, відновлення та концентрацію протягом дня.",
        tips: ["Без екранів", "Стабільний графік", "Менше кофеїну", "Релакс"]
    },
    "Читання": {
        icon: "📚",
        benefits: "Розвиває мислення, увагу та пам’ять.",
        tips: ["10–15 хв", "Без відволікань", "Книга поруч", "Цікаві теми"]
    }
};

document.getElementById("popular-habits").addEventListener("change", e => {
    const val = e.target.value;
    document.getElementById("habit-name").value = val;
    renderInfo(val);
});

function renderInfo(val) {
    const box = document.getElementById("habit-info-cards");
    box.innerHTML = "";
    if (!habitInfo[val]) return;

    const h = habitInfo[val];

    box.innerHTML = `
        <div class="info-card">
            <h4>${h.icon} Користь</h4>
            <p>${h.benefits}</p>
        </div>
        <div class="info-card">
            <h4>📌 Поради</h4>
            <ul>${h.tips.map(t => `<li>${t}</li>`).join("")}</ul>
        </div>
    `;
}

document.getElementById("add-habit-form").addEventListener("submit", e => {
    e.preventDefault();

    const name = document.getElementById("habit-name").value.trim();
    if (!name) return;

    habits.push({
        id: Date.now(),
        name,
        completions: []
    });
```

```
  save();
  renderAll();
});

function today() {
  return new Date().toISOString().split("T")[0];
}

function doneToday(h) {
  return h.completions.includes(today());
}

function mark(id) {
  const h = habits.find(x => x.id === id);
  if (!h) return;

  if (!h.completions.includes(today())) {
    h.completions.push(today());
    save();
    renderAll();
  }
}

function edit(id) {
  const h = habits.find(x => x.id === id);
  const val = prompt("Редагувати:", h.name);
  if (val) {
    h.name = val;
    save();
    renderAll();
  }
}

function remove(id) {
  habits = habits.filter(h => h.id !== id);
  save();
  renderAll();
}

function streak(h) {
  return h.completions.length;
}

function renderHabits() {
  const box = document.getElementById("habits-list");
  box.innerHTML = "";

  habits.forEach(h => {
    const percent = Math.min(100, Math.round((h.completions.length / 30) * 100));

    const el = document.createElement("div");
    el.className = "habit-item";

    el.innerHTML = `
      <div class="habit-name">${h.name}</div>

      <button class="check-btn ${doneToday(h) ? "completed" : ""}" data-id="${h.id}">
        ${doneToday(h) ? "✓" : ""}
      </button>

      <div class="progress-wrapper">
        <div class="progress-container">
          <div class="progress-bar" style="width:${percent}%"></div>
        </div>
        <div class="progress-percent">${percent}%</div>
      </div>

      <div class="streak">🔥 ${streak(h)} дн</div>

      <div>
        <button onclick="edit(${h.id})">✏️</button>
        <button onclick="remove(${h.id})">🗑</button>
      </div>
    `;

    box.appendChild(el);
```

```
    });

    box.querySelectorAll(".check-btn").forEach(b =>
      b.addEventListener("click", () => mark(+b.dataset.id))
    );
}

function renderAnalytics() {
    const total = habits.length;
    const doneToday = habits.filter(h => doneToday(h)).length;
    const totalDone = habits.reduce((a, b) => a + b.completions.length, 0);

    document.getElementById("analytics").innerHTML = `
      <div class="analytics-grid">
        <div class="stat"><strong>${total}</strong><div>Звичок</div></div>
        <div class="stat"><strong>${doneToday}</strong><div>Сьогодні</div></div>
        <div class="stat"><strong>${totalDone}</strong><div>Всього</div></div>
      </div>

      <div class="motivation">${randomMotivation()}</div>
    `;
}

function renderAll() {
    renderHabits();
    renderAnalytics();
}

document.getElementById("notify-btn").addEventListener("click", async () => {
    const perm = await Notification.requestPermission();
    if (perm === "granted") {
      new Notification("HabitFlow", {
        body: "Нагадування активовано!"
      });
    }
});

document.addEventListener("DOMContentLoaded", () => {
    load();
    renderAll();
});
</script>

</body>
</html>
```