Вищий навчальний заклад

«Університет економіки та права «КРОК»
Фаховий коледж

Циклова комісія з інформаційних технологій

Кваліфікаційна робота фахового молодшого бакалавра

на тему Розробка інформаційної системи замовлення технічних засобів навчання в навчальному закладі.

Виконав _____
(Підпис)

____Дугін Юрій Олексійович_____
(прізвище, ім'я, по батькові)

_____

Науковий керівник
____Чернозубкін_Ігор_Олександрови
(прізвище, ім'я, по батькові)

_____

_____
(Резолюція «До захисту»)

Попередній захист:

_____
(Висновок: "До захисту в екзаменаційній комісії ")

Голова циклової комісії _____ _____ _____
(Підпис) (Прізвище, ініціали) (Дата)

Київ - 2025

ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД

УНІВЕРСИТЕТ ЕКОНОМІКИ ТА ПРАВА «КРОК»

Фаховий коледж

Циклова комісія з інформаційних технологій

Спеціальність 121 інженерія програмного забезпечення

ЗАТВЕРДЖУЮ:
Голова циклової комісії _____ Леонід УВАРОВ
(підпис)

«____» _____ 2025 року

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

Здобувач освіти Дугін Юрій Олексійович

1. Тема роботи Розробка інформаційної системи замовлення технічних засобів навчання в навчальному закладі

затверджена наказом по університету від «___» _____ 202__ р. № _____

2. Термін здачі закінченої роботи «30» травня 2025 року
3. Вихідні дані до роботи:
   а) цільова аудиторія – викладачі, студенти, адміністрація навчального закладу, технічний персонал тощо, їх потреби та очікування;
   б) функціональність – перелік функцій системи (ведення каталогу технічних засобів (ТЗ) та робота з ним (пошук, редагування, адміністрування), замовлення ТЗ та його погодження, відстеження статусу замовлення, моніторинг, звітність тощо;
   в) технічні вимоги – платформа для розробки системи (веб-сайт, мобільний застосунок, десктоп рішення, тощо), мова програмування, база даних та інші технології, безпека та захист даних;
   г) можливість взаємодії з існуючими ІС навчального закладу;
   д) особливості навчального закладу щодо використання ТЗ та їх замовлення;
   е) існуючі рішення щодо автоматизації замовлень ТЗ навчання в навчальному закладі.

4. Зміст пояснювальної записки
   а) Розділ 1 Теоретична частина. Аналіз існуючих рішень щодо автоматизації замовлень ТЗ навчання в навчальному закладі, обґрунтування вибору платформи та технології для розроблення ІС.
   б) Розділ 2 Проєктування та розробка. Створення зручного та інтуїтивно зрозумілого інтерфейсу для різних категорій користувачів та забезпечення його адаптивності для різних пристроїв (комп'ютери, планшети, смартфони), розробка алгоритму, за яким система буде обробляти запити користувачів та виконувати необхідні функції, оптимізація роботи системи для забезпечення швидкодії та ефективності.

За умови використання бази даних – проєктування структури бази даних для зберігання інформації про ТЗ, користувачів, замовлення тощо та рішення щодо забезпечення цілісності та безпеки даних;
забезпечення можливості обміну даними між ІС та існуючими ІС навчальному закладі (за необхідності).

в) Розділ 3 Експериментальна частина. Перевірка роботи системи на різних пристроях та платформах, виявлення та виправлення помилок, проведення тестування для перевірки продуктивності системи під навантаженням, оцінка ефективності системи (аналіз того, наскільки система відповідає поставленим завданням та задовольняє потреби користувачів), інструкція користувачам (адміністраторам та програмістам – опис процесу розробки системи, її архітектури та функціональності, схема бази даних та опис її структури, користувачам – як користуватися системою та отримувати необхідну інформацію).

5. Перелік графічного матеріалу:
Скріншоти існуючих рішень
Скріншоти інтерфейсу продукту
Схеми і таблиці щодо візуалізації аналізу
Блок-схеми алгоритмів
Діаграми щодо проєктування продукту (наприклад, потоків даних, переходів станів, сутність-зв'язок, UML, бази даних тощо)

Дата видачі завдання «12» лютого 2025 року

Науковий керівник _____ Ігор ЧЕРНОЗУБКІН
(підпис)

Завдання прийняв до виконання _____ Юрій ДУГІН
(підпис)

РЕФЕРАТ

Пояснювальна записка: 56 сторінок, 13 рисунків, 2 таблиць, 4 додатків, 17 джерел.

Об'єкт дослідження – Інформаційна система замовлення технічних засобів навчання в навчальному закладі

Мета роботи – Створення сучасної, зручної у використанні інформаційної системи для автоматизації процесу замовлення технічних засобів навчання у навчальному закладі. Система має забезпечити можливість подачі заявок на обладнання, збереження та перегляд замовлень, пошук, фільтрацію, а також інтуїтивно зрозумілий інтерфейс з темною/світлою темою.

Результати розробки – Розроблено веб-застосунок, що дозволяє:

- Додавати заявки з валідацією даних (ПІБ, предмет, обладнання, кабінет, дата);
- Зберігати заявки у браузері (LocalStorage);
- Шукати заявки за ключовими словами;
- Видаляти записи;
- Перемикати між темною і світлою темою;
- Автоматично зберігати введені дані форми.

Система працює локально в браузері без потреби у сервері чи базі даних, але може бути адаптована для масштабування.

Ключові слова: інформаційна система замовлення, технічні засоби, інтерфейс користувача, JavaScript, вебзастосунок.

ABSTRACT

Explanatory note: 56 pages, 13 figures, 2 tables, 4 appendices, 17 references.
Object of research – Information system for ordering educational equipment in an educational institution.

Purpose of the work – To create a modern, user-friendly information system for automating the process of ordering educational equipment in an educational institution. The system should provide the ability to submit equipment requests, store and view orders, perform search and filtering, and offer an intuitive user interface with light/dark theme support.
Development results – A web application was developed that allows users to:

- Add requests with data validation (full name, subject, equipment, classroom, date);
- Store requests in the browser (LocalStorage);
- Search for requests by keywords;
- Delete records;
- Switch between light and dark themes;
- Automatically save entered form data.

The system runs locally in a browser without the need for a server or database, but it can be adapted for scaling.
Keywords: ordering information system, educational equipment, user interface, JavaScript, web application.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧЕННЯ

ІС — інформаційна система

ТЗ — технічні засоби

Веб-додаток — веб-застосунок (програма, що працює в браузері)

HTML — HyperText Markup Language (мова розмітки гіпертексту)

CSS — Cascading Style Sheets (каскадні таблиці стилів)

JS — JavaScript (мова програмування для веб-інтерфейсів)

LocalStorage — локальне сховище браузера, що дозволяє зберігати дані на стороні клієнта

UI — User Interface (користувацький інтерфейс)

UX — User Experience (користувацький досвід)

API — Application Programming Interface (інтерфейс програмування додатків)

DB — база даних

HTTP — HyperText Transfer Protocol (протокол передачі гіпертексту)

CRUD — Create, Read, Update, Delete (операції створення, читання, оновлення та видалення даних)

MVC — Model-View-Controller (архітектурний патерн програмного забезпечення)

UI/UX дизайн — дизайн користувацького інтерфейсу та досвіду

ТЗ навчання — технічні засоби навчання (проєктори, комп’ютери, дошки тощо)

ВСТУП

Актуальність завдання. У сучасному інформаційному суспільстві цифровізація охоплює практично всі сфери людської діяльності, і освіта не є винятком. Навчальні заклади дедалі частіше впроваджують технічні засоби навчання — мультимедійні проєктори, комп'ютери, інтерактивні дошки, принтери тощо, що сприяє підвищенню ефективності освітнього процесу та формуванню цифрової грамотності [1, 3]. Водночас зростає і складність організації доступу до цих ресурсів, особливо у великих закладах освіти, де обладнанням щоденно користуються сотні викладачів і студентів [4, 9].

Традиційні способи замовлення обладнання — записи у зошитах, усні домовленості чи листування через месенджери — є застарілими, ненадійними та не забезпечують необхідного рівня контролю й ефективності. Це призводить до дублювання замовлень, втрати інформації та нераціонального використання ресурсів [2, 5]. Вирішенням цієї проблеми може стати впровадження інформаційної системи, яка автоматизує процес подання та обробки заявок на технічні засоби [6, 7].

Актуальність теми полягає в необхідності створення доступного та зручного інструменту для управління технічними ресурсами навчального закладу. Розроблений веб-додаток не потребує складного налаштування, працює в будь-якому сучасному браузері, зберігає дані локально та не залежить від інтернет-з'єднання, що робить його придатним для використання в умовах обмеженої інфраструктури [8, 11].

Мета роботи. Проєктування та розробка інформаційної системи у вигляді веб-додатку, що дозволяє автоматизувати процес замовлення та обліку технічних засобів навчання із застосуванням HTML, CSS, JavaScript та локального сховища браузера (LocalStorage) [4, 14].

Завдання роботи:

• проаналізувати існуючі рішення та визначити їхні недоліки [1, 2];

• обґрунтувати вибір інструментів розробки [4, 5];

• спроєктувати структуру та інтерфейс користувача [9, 10];

• реалізувати основні функції системи [14, 15];

• здійснити тестування веб-додатку [16, 18];

• сформувати висновки щодо ефективності та зручності використання розробленої системи [19, 20].

Об’єкт дослідження. Процес замовлення та обліку технічних засобів у навчальному закладі [3].

Предмет дослідження. Інформаційна система, яка автоматизує подання, облік та керування заявками на технічне обладнання [6].

Методи дослідження. У роботі використовуються аналіз і синтез інформації, методи моделювання, проектування інтерфейсу, програмування та тестування веб-додатків [12, 13, 14].

Практичне значення одержаних результатів. Розроблений веб-додаток дозволяє створювати, зберігати, редагувати та видаляти заявки на обладнання, автоматично перевіряє коректність введених даних (наприклад, забороняє обрання минулої дати), зберігає попередні значення для повторного використання та підтримує темну і світлу теми оформлення. Система є простою у використанні, не потребує зовнішньої бази даних чи підключення до мережі, що робить її зручною для самостійного використання у навчальних закладах. У майбутньому веб-додаток може бути інтегрований у більші інформаційні системи освітніх установ [14, 15, 20].

РОЗДІЛ 1. ТЕОРЕТИЧНА ЧАСТИНА

1.1. Аналіз існуючих рішень щодо автоматизації замовлень технічних засобів навчання

У сучасних навчальних закладах дедалі частіше використовуються технічні засоби (ТЗ) навчання, зокрема мультимедійні проєктори, ноутбуки, інтерактивні дошки, акустичні системи тощо. Забезпечення ефективного доступу до цих засобів є важливим елементом організації освітнього процесу. Проте зростання кількості ТЗ і користувачів (викладачів, технічного персоналу, адміністрації) вимагає вдосконалення механізмів замовлення й обліку обладнання.

Наразі в більшості закладів освіти використовуються такі підходи до організації обліку і замовлень ТЗ:

- **Паперові журнали чи зошити** — традиційна, але неефективна система, яка не дозволяє зручно відстежувати історію замовлень, шукати, сортувати чи аналізувати дані. Часто призводить до втрати або дублювання інформації.
- **Електронні таблиці (Excel, Google Sheets)** — забезпечують деякий рівень автоматизації, проте вимагають ручного введення даних та не мають вбудованої системи перевірок, прав доступу чи зручного інтерфейсу для користувача. Такі рішення нестабільні за великої кількості користувачів і не передбачають логіки обробки запитів.
- **Модулі в системах управління навчанням (LMS)** — такі як Moodle, Google Workspace for Education, Microsoft Teams — можуть містити інструменти для реєстрації запитів або замовлень, проте здебільшого не є спеціалізованими рішеннями для управління ТЗ. Їх використання потребує значних зусиль на налаштування, навчання персоналу та підтримку, а також стабільного інтернет-з’єднання.

- **Корпоративні IT-системи (1С, SAP, Bitrix24 тощо)** — пропонують комплексний підхід до управління ресурсами, проте є занадто громіздкими й дорогими для освітніх установ, особливо невеликих. Їхнє впровадження потребує значних фінансових і кадрових ресурсів.

Таким чином, більшість існуючих рішень або занадто складні, або надто примітивні для ефективної організації обліку технічних засобів у навчальному закладі. Це створює передумови для розробки власного спеціалізованого програмного продукту — автономної веб-орієнтованої інформаційної системи, яка:

- є простою у використанні;
- працює у будь-якому сучасному браузері;
- не потребує серверного середовища або інтернет-підключення;
- забезпечує зручну роботу з даними (додавання, редагування, пошук, фільтрація, видалення).

1.2. Обґрунтування вибору платформи та технологій

Під час розробки інформаційної системи для автоматизації процесу замовлення технічних засобів навчання важливо обрати такі платформи та технології, які забезпечать зручність користування, доступність, незалежність від сторонніх серверів, а також простоту підтримки й можливість розширення в майбутньому.

Ключові вимоги до платформи:

- **Кросплатформеність:** система повинна працювати на різних пристроях — комп’ютерах, планшетах і смартфонах — без потреби встановлення додаткового програмного забезпечення.
- **Автономність:** можливість повноцінної роботи без підключення до інтернету чи зовнішніх серверів.
- **Простота використання:** інтерфейс має бути інтуїтивно зрозумілим для користувачів із різним рівнем технічної підготовки.

- **Локальне збереження даних:** дані про замовлення мають зберігатися безпосередньо на пристрої користувача (наприклад, у локальному сховищі браузера).

З огляду на ці вимоги, було обрано наступні технології для реалізації веб-додатку:

HTML (HyperText Markup Language):

Основна мова розмітки для створення структури веб-сторінки. Дає змогу визначити компоненти інтерфейсу, такі як поля введення, таблиці, кнопки тощо.

CSS (Cascading Style Sheets):

Застосовується для оформлення зовнішнього вигляду веб-додатку. За допомогою CSS реалізовано адаптивний дизайн (відображення на різних екранах), а також підтримку темного та світлого режимів оформлення інтерфейсу.

JavaScript:

Основна мова для програмної логіки веб-додатку. Використовується для:

- обробки дій користувача (введення даних, натискання кнопок);
- перевірки коректності введених значень;
- динамічного оновлення інтерфейсу;
- реалізації функцій пошуку, сортування, фільтрації;
- взаємодії з локальним сховищем браузера.

LocalStorage (Web Storage API):

Використовується для зберігання даних на стороні клієнта. Дозволяє зберігати заявки навіть після закриття або перезавантаження сторінки. Цей підхід забезпечує автономну роботу без використання серверів чи баз даних.

Платформа виконання — веб-браузер:

Веб-додаток функціонує без встановлення, працює у всіх сучасних браузерах (Google Chrome, Mozilla Firefox, Microsoft Edge тощо), що робить систему універсальною для користувачів із різних пристроїв.

Обґрунтування вибору:

Обрані інструменти відповідають усім початковим вимогам: прості в освоєнні, не потребують додаткових витрат на хостинг або сервери, дозволяють забезпечити швидку розробку, автономність і гнучкість. Крім того, веб-технології легко масштабуються у разі необхідності додавання нових функцій, підключення до бази даних або перенесення системи на сервер.

1.3. Архітектура інформаційних систем

Архітектура інформаційної системи визначає структуру, організацію та взаємодію її компонентів. У даному проєкті реалізовано **локальну клієнтську інформаційну систему**, що працює повністю на стороні користувача — у браузері — без потреби у серверній інфраструктурі.

Основні компоненти клієнтської архітектури

У межах локальної реалізації система побудована на основі трьох ключових рівнів:

1. **Інтерфейс користувача (UI)**
2. Створений за допомогою HTML та CSS, інтерфейс забезпечує взаємодію користувача з системою: заповнення форм, перегляд інформації, створення замовлень тощо.
3. CSS використовується для стилізації елементів, що забезпечує зручність та привабливий зовнішній вигляд.
4. **Логіка програми (JavaScript)**
5. Уся бізнес-логіка реалізована на JavaScript і виконується у браузері. Сценарії обробляють події, взаємодіють із локальною базою даних, керують відображенням інформації.
6. Сюди входять:
    - Реєстрація/вхід користувача;
    - Створення та редагування замовлень;
    - Перевірка прав доступу (рольова система);
    - Фільтрація, сортування, динамічне оновлення UI.
7. **Локальна база даних (Web Storage / IndexedDB / JSON-файли)**

8. Для зберігання інформації про користувачів, замовлення, обладнання використовується **локальна база**. Найчастіше це один з варіантів:

- **LocalStorage / SessionStorage** — для простих структур;
- **IndexedDB** — для складніших структур із запитами;
- **JSON-файли** — при тестуванні чи попередньому заповненні.

Архітектурний стиль

Проєкт реалізовано за принципом **монолітної клієнтської архітектури**, де всі компоненти — HTML-сторінки, CSS-файли, JavaScript-код — функціонують як єдиний блок. Такий підхід має низку переваг:

- Простота розгортання: достатньо відкрити HTML-файл у браузері;
- Повна автономність: система не потребує серверів чи інтернету;
- Висока швидкодія: усі операції виконуються локально;
- Зручність тестування: можна швидко перевіряти зміни без складних налаштувань.

Сценарії використання

Система призначена для навчального закладу й дозволяє користувачам (викладачам, адміністраторам, представникам шкіл):

- Переглядати список доступного навчального обладнання;
- Створювати замовлення;
- Управляти особистими кабінетами;
- Вносити та оновлювати інформацію в межах дозволених прав.

Обмеження локальної архітектури

Попри зручність, локальні клієнтські системи мають обмеження:

- Відсутність централізованого зберігання даних (інформація доступна лише на одному пристрої);

- Вразливість до втрати даних (без синхронізації чи резервного копіювання);
- Неможливість одночасної роботи кількох користувачів у мережі;
- Обмеження безпеки — JS-код і дані доступні в браузері.

Перспективи розвитку

У майбутньому проєкт може бути модернізований шляхом:

- Додавання серверної частини (наприклад, з використанням Node.js або Firebase) для зберігання даних у хмарі;
- Реалізації API для синхронізації між пристроями;
- Розширення ролей користувачів та системи авторизації;
- Створення прогресивного вебдодатку (PWA) з можливістю офлайн-роботи.

1.4. Вимоги до системи

1. Функціональні вимоги

Ці вимоги визначають, яку функціональність повинна реалізовувати система:

- Система повинна надавати можливість **реєстрації та входу** користувачів.
- Повинна бути реалізована **рольова модель доступу** (користувач, адміністратор, представник школи).
- Користувач повинен мати змогу:
  - Переглядати список технічних засобів;
  - Створювати заявки на замовлення обладнання;
  - Переглядати історію своїх замовлень;
- Адміністратор повинен мати можливість:
  - Додавати/редагувати/видаляти технічні засоби;
  - Переглядати всі заявки та змінювати їхній статус;
  - Управляти списком користувачів;

- Представник школи повинен мати доступ до заявок від свого закладу та змогу реагувати на них;
- Система повинна зберігати інформацію локально (у браузері користувача) і забезпечувати швидкий доступ до даних без потреби підключення до інтернету.

2. Нефункціональні вимоги

Ці вимоги стосуються якості роботи системи:

- **Зручність використання (Usability):** Інтерфейс має бути інтуїтивно зрозумілим, з логічною навігацією.
- **Продуктивність (Performance):** Усі операції (фільтрація, додавання, перегляд) повинні виконуватись миттєво, без помітних затримок.
- **Надійність (Reliability):** Дані повинні зберігатися у локальній базі до моменту їх видалення користувачем або оновлення.
- **Безпека (Security):**
  - Доступ до адміністративних функцій повинен бути захищений (наприклад, через логін з паролем).
  - Паролі повинні бути збережені у хешованому вигляді (або зашифровані, якщо локально).
- **Можливість масштабування:** Архітектура має бути побудована таким чином, щоб у майбутньому можна було перенести систему на серверну платформу.

3. Технічні вимоги

Ці вимоги описують програмні та апаратні засоби, необхідні для роботи системи:

- **Програмне забезпечення:**
  - Сучасний веббраузер (Google Chrome, Firefox, Edge, Safari);
  - Підтримка JavaScript ES6+;
  - Підтримка Web Storage або IndexedDB;

- **Апаратне забезпечення:**
  - ПК або ноутбук із мінімум 2 ГБ оперативної пам’яті;
  - Дисплей з роздільною здатністю від 1024×768;
  - Наявність клавіатури та миші або тачпаду.
- **Операційна система:** Windows 7 або новіша, macOS, Linux, Android (у режимі десктопного перегляду).

4. Обмеження системи

- Дані не синхронізуються між пристроями (усе зберігається локально).
- Можлива втрата даних у разі очищення локального сховища браузера.
- Система не підтримує багатокористувацький режим в реальному часі.

РОЗДІЛ 2. ПРОЄКТУВАННЯ ТА РОЗРОБКА

2.1. Інтерфейс користувача

Інтерфейс користувача (ІК) є ключовим елементом будь-якої інформаційної системи, що визначає зручність і ефективність взаємодії між людиною та програмним забезпеченням. Для веб-додатку, призначеного для подання заявок на використання технічних засобів навчання, особливу увагу було приділено простоті, доступності та адаптивності інтерфейсу.

Основні принципи проєктування інтерфейсу:

- **Інтуїтивність:** усі елементи інтерфейсу мають бути зрозумілими без додаткових інструкцій.
- **Мінімалізм:** відсутність перевантаження екрана зайвими елементами, фокус на основному функціоналі.
- **Адаптивність:** підтримка роботи на різних пристроях — ПК, планшетах і смартфонах — із забезпеченням зручного перегляду та керування.
- **Доступність:** кольори та шрифти підібрані з урахуванням контрастності та читабельності.

Структура інтерфейсу:

Інтерфейс умовно поділений на кілька основних зон:

1. **Головне меню:**
   - Навігація між розділами (наприклад: "Подати заявку", "Список заявок", "Налаштування").
   - Кнопка перемикання темного/світлого режиму.
2. **Форма подання заявки:**
   - Поля для введення: тип обладнання, дата використання, аудиторія або місце застосування, коментар (за потреби).
   - Перевірка правильності введених даних (неможливість обрати минулу дату, обов’язковість заповнення основних полів).
   - Кнопка **«Подати заявку»**, що зберігає дані у локальному сховищі браузера.
3. **Список заявок:**

- Таблиця або картки з усіма поданими заявками, що містять інформацію про обладнання, дату, час, місце.
- Можливість **пошуку** та **фільтрації** заявок за параметрами (наприклад, за датою або типом обладнання).
- Кнопки **редагування** та **видалення** кожного запису.

4. **Налаштування:**

- Вибір теми оформлення (темна / світла).
- Можливість очищення усіх даних із локального сховища.
- Інструкція користувача (у вигляді підказок або короткої довідки).

Адаптивність:

Інтерфейс реалізований за допомогою CSS Flexbox/Grid та медіа-запитів, що дозволяє елементам автоматично змінювати розміщення і розміри залежно від роздільної здатності екрана. На мобільних пристроях форма заявки та таблиця заявок відображаються вертикально, з оптимізованим розміром шрифтів і кнопок для сенсорного керування.

Зовнішній вигляд:

Для стилізації використовувалися сучасні підходи до дизайну:

- використання змінних кольорових тем (CSS variables),
- плавна анімація елементів (наприклад, появи повідомлень),
- візуальне виділення помилок у формі (червона рамка, підказки).

2.2. Алгоритм обробки запитів

У розробленій інформаційній системі основним завданням є ефективна обробка запитів користувача, зокрема подання, збереження, відображення, редагування та видалення заявок на технічні засоби. Алгоритм обробки запитів побудовано з урахуванням автономності системи — без потреби у сервері чи зовнішній базі даних. Усі дії виконуються на стороні клієнта за допомогою JavaScript з використанням **LocalStorage**.

Основні етапи алгоритму обробки запиту:

1. **Введення даних користувачем:**

- Користувач заповнює форму заявки, де вказує тип обладнання, дату, місце використання та за потреби — коментар.
- Натискає кнопку «Подати заявку».

2. **Перевірка введених даних (валідація):**
   - Перевіряється, чи всі обов’язкові поля заповнено.
   - Перевіряється, чи дата використання не є минулою.
   - У разі помилки — відображається підказка із поясненням, поле підсвічується.
3. **Створення об’єкта заявки:**
   - Дані з форми зберігаються у вигляді JavaScript-об’єкта з полями:

```
const newOrder = {
  name: document.getElementById("name").value.trim(),
  item: document.getElementById("item").value.trim(),
  equipment: document.getElementById("equipment").value,
  cabinet: document.getElementById("cabinet").value.trim(),
  date: document.getElementById("date").value
};
```

4. **Збереження заявки:**
   - Новий об’єкт заявки додається до масиву заявок.
   - Масив серіалізується у формат JSON і зберігається у LocalStorage:
     localStorage.setItem("requests", JSON.stringify(requestsArray));
5. **Оновлення інтерфейсу:**
   - Нову заявку додають до таблиці або списку заявок у браузері без перезавантаження сторінки.
   - Відображається повідомлення про успішне збереження.
6. **Обробка інших дій:**
   - **Редагування:** При натисканні кнопки редагування відповідна заявка завантажується у форму. Після внесення змін заявка оновлюється в масиві й зберігається повторно.

- **Видалення:** При натисканні кнопки видалення заявка видаляється з масиву, і LocalStorage оновлюється.
- **Пошук та фільтрація:** Заявки фільтруються на клієнті за введеним ключовим словом або датою, без потреби повторного зчитування з пам’яті.

Умовна блок-схема алгоритму обробки заявки:

1. Заповнення форми

Користувач вводить: ПІБ, предмет, обладнання, номер кабінету, дата.

2. Перевірка правильності введених даних

Чи всі поля заповнені?

Чи дата не є минулою?

3. Умова: Дані введено правильно?

Так → перейти до кроку 5

Ні → перейти до кроку 6

4. Збереження заявки

Додати заявку до списку.

Зберегти заявку в LocalStorage.

5. Виведення повідомлення про помилку

Повідомити користувача про помилку (наприклад: “Дата не може бути в минулому”).

6. Оновлення списку заявок на сторінці

Переваги підходу:

- Вся логіка виконується миттєво, без звернень до сервера.
- Дані зберігаються у браузері користувача, що дозволяє працювати офлайн.
- Зменшене навантаження на мережу та відсутність залежності від інфраструктури.

2.3. Оптимізація роботи системи

Враховуючи, що розробка інформаційної системи здійснювалася у вигляді односторінкового веб-додатку без використання зовнішніх бібліотек і фреймворків, особливу увагу було приділено оптимізації внутрішньої логіки та ефективності реалізації у межах одного HTML-файлу з вбудованими стилями (CSS) та сценаріями (JavaScript).

1. Ефективність JavaScript-реалізації

- Основна логіка обробки даних реалізована у вигляді компактних функцій, що виконуються лише за потреби (події натискання кнопок, зміни полів тощо).
- Для взаємодії з DOM використовуються нативні методи (querySelector, addEventListener, createElement), які є швидкими та не потребують додаткових ресурсів.
- Для збереження і обробки даних використано **масив об'єктів**, який серіалізується у формат JSON та зберігається у **LocalStorage**, що дозволяє забезпечити швидкий доступ і автономну роботу додатку.

2. Оптимізація користувацького інтерфейсу

- Інтерфейс побудовано на основі гнучкої адаптивної верстки за допомогою **Flexbox**, що дозволяє коректно відображати елементи як на великих екранах, так і на мобільних пристроях.
- Всі інтерактивні елементи (форми, таблиці, кнопки) змінюють розмір та розташування залежно від ширини вікна браузера.
- Стилізація виконана у вбудованому CSS без зовнішніх файлів, що зменшує час завантаження сторінки.

3. Адаптивність і мобільна сумісність

- Додаток коректно працює у сучасних браузерах як на комп'ютерах, так і на планшетах та смартфонах.
- Шрифти, кнопки та поля введення мають достатній розмір для зручного натискання на сенсорних екранах.
- Інтерфейс протестовано в Google Chrome, Firefox, Microsoft Edge і мобільному браузері Android.

4. Зберігання та обробка даних

- Використано **локальне сховище браузера (LocalStorage)**, що забезпечує автономність додатку — дані зберігаються навіть після перезавантаження сторінки.
- При кожній дії (додавання, редагування або видалення заявки) масив заявок оновлюється у пам’яті та одразу перезаписується в LocalStorage.
- Перевірка на дублікати та помилки введення здійснюється перед збереженням, що знижує ймовірність помилок користувача.

5. Мінімізація коду і повторне використання

- Повторювані дії (наприклад, створення елементів списку заявок, оновлення таблиці) винесені у функції для уникнення дублювання коду.
- Поділ логіки на логічні блоки (наприклад: «введення даних», «валідація», «збереження», «рендеринг») зроблено в межах одного JavaScript-фрагмента, що спрощує подальше обслуговування системи.

6. Темна та світла тема

- Реалізовано перемикач теми за допомогою зміни класів CSS.
- Обрана тема зберігається в LocalStorage та автоматично застосовується при повторному відкритті сторінки.

2.4. Алгоритм обробки запитів

Обробка запитів у даній інформаційній системі здійснюється **на клієнтській стороні** із використанням JavaScript. Запити реалізуються через події взаємодії користувача з інтерфейсом — натискання кнопок, заповнення форм, вибір елементів. Усі дані зберігаються та обробляються локально в браузері, з використанням **LocalStorage**, **SessionStorage** або **IndexedDB**.

Основні типи запитів:

1. Запит на авторизацію;
2. Запит на створення заявки;
3. Запит на редагування або перегляд даних;
4. Запит на фільтрацію або пошук.

Загальний алгоритм обробки запиту:

1. Ініціалізація системи

Завантаження сторінки (інтерфейсу).

Ініціалізація змінних, завантаження збережених заявок з LocalStorage.

Завантаження збережених даних форми, якщо вони були.

2. Заповнення форми користувачем

Користувач вводить:

ПІБ,

предмет,

тип обладнання (з випадаючого списку),

номер кабінету,

дату проведення заняття.

3. Перевірка правильності введених даних

Перевірка: чи всі поля заповнені.

Перевірка дати: не допускається дата з минулого.

Перевірка формату (за потреби).

4. Збереження заявки

Якщо дані коректні:

Заявка додається до масиву orders.

Масив зберігається в LocalStorage (equipmentOrders).

Відображення оновленого списку заявок.

5. Обробка помилок

Якщо дані некоректні:

Виводиться повідомлення про помилку.

Поля, де виявлено помилку, підсвічуються.

6. Пошук/фільтрація заявок

При введенні тексту в поле пошуку:

Масив orders фільтрується за ключовими словами.

Відображаються лише ті заявки, які відповідають пошуку.

7. Управління темою (світла/темна)

Кнопка перемикає клас light-mode ↔ dark-mode.

Обрана тема зберігається в LocalStorage (themeMode).

8. Видалення заявки

При натисканні кнопки "×" заявка видаляється з масиву.

Оновлений масив знову зберігається в LocalStorage.

Інтерфейс оновлюється.

Приклад 1. Запит на створення нової заявки

1. Користувач натискає кнопку "Створити заявку".
2. JavaScript отримує дані з форми: назва обладнання, кількість, дата.
3. Дані проходять перевірку (валидацію).
4. Якщо валідація успішна, заявка зберігається у LocalStorage як об'єкт:

```
{
  name: "ПІБ",
  item: "Предмет",
  equipment: "Обладнання",
  cabinet: "Кабінет",
  date: "Дата"
}: "Очікує"
```

5. Інтерфейс оновлюється — заявка з'являється в списку користувача.

Приклад 2. Запит на авторизацію користувача

1. Користувач вводить логін і пароль та натискає "Увійти".
2. JS перевіряє, чи є такий користувач у LocalStorage.

3. Якщо так — перевіряється відповідність паролю (можливо — через просте хешування).

4. Якщо авторизація успішна:

- Зберігається статус сесії

```
localStorage.setItem("loggedUser", "ivanenko");
```

- Користувача перенаправляють до особистого кабінету.

Приклад 3. Запит на пошук або фільтрацію

1. Користувач вводить ключове слово у поле пошуку.
2. JS виконує фільтрацію по масиву заявок або обладнання:

```
const searchTerm = searchInput.value.toLowerCase();


const filtered = orders.filter(order =>
  Object.values(order).some(value =>
    value.toLowerCase().includes(searchTerm)
  )
);
```

3. Результати відображаються динамічно без перезавантаження сторінки.

Висновок

Таким чином, обробка запитів у системі реалізована у вигляді подій на стороні клієнта. Всі дії відбуваються локально — це забезпечує високу швидкодію, автономність та простоту в реалізації. Такий підхід є доцільним для навчальних і внутрішніх інформаційних систем, які не потребують складної серверної архітектури.

2.5. Оптимізація роботи системи

Оптимізація є ключовим етапом у розробці будь-якої інформаційної системи, оскільки вона дозволяє підвищити швидкодію, зменшити споживання ресурсів, покращити зручність використання та забезпечити стабільну роботу. У випадку локальної клієнтської системи особлива увага приділяється ефективній роботі JavaScript-коду, правильному управлінню DOM-елементами, а також раціональному зберіганню та обробці даних.

1. Оптимізація JavaScript-коду

- **Усунення дублювання коду** — були створені універсальні функції для повторюваних операцій, наприклад, додавання елементів у список, перевірка прав доступу, генерація HTML.
- **Зменшення кількості DOM-операцій** — при зміні вмісту сторінки спочатку формувався **віртуальний блок**, і лише потім він додавався до DOM. Це дозволило зменшити навантаження на браузер.
- **Використання event delegation** — замість призначення великої кількості обробників подій, використовувалася делегація на рівні контейнерів.
- **Асинхронна обробка даних** — для імітації асинхронної роботи використовувалися проміси (Promise) та setTimeout, щоб не блокувати інтерфейс під час складних обчислень.

2. Оптимізація взаємодії з локальною базою (LocalStorage/IndexedDB)

- Замість постійного звернення до локального сховища, **дані кешуються в оперативній пам'яті** (у змінних під час сесії).
- При збереженні оновлених масивів використовується JSON.stringify(), але лише при зміні, а не при кожній дії, що мінімізує кількість записів у LocalStorage.
- Реалізовано **індексування за ID** — дані структуровано таким чином, щоб доступ до них був швидким за ключем (наприклад, через об'єкти Map або id-ключі).

3. Оптимізація інтерфейсу користувача

- Реалізовано **динамічне оновлення вмісту сторінки без перезавантаження** (single-page-like behavior).
- Зменшено кількість одночасно відображених елементів — наприклад, реалізовано **пагінацію** або поступову підгрузку списків (lazy loading).
- Інтерфейс адаптовано під різні розміри екранів — використано **адаптивну верстку** на CSS (media queries).

4. Оптимізація ресурсів (HTML/CSS/JS)

- **Мінімізація та об'єднання скриптів та стилів** (на етапі розгортання):
  - Зменшено розмір HTML, CSS та JS-файлів;
  - CSS згруповано в один файл, JavaScript — теж.
- Використано **семантичну верстку** HTML5, що забезпечує кращу оптимізацію в браузерах.
- Застосовано кешування зображень та статичних ресурсів за допомогою браузерного механізму (якщо використовуються іконки чи файли).

5. Оптимізація логіки роботи з користувачами

- При авторизації використовується **збереження сеансу в SessionStorage**, що зменшує кількість перевірок.
- Система автоматично перевіряє роль користувача та показує тільки релевантний інтерфейс (розділення прав доступу).
- Застосовано **відкладене завантаження даних** — наприклад, заявки або обладнання завантажуються лише після переходу у відповідний розділ.

Висновок

Завдяки цілеспрямованій оптимізації роботи системи вдалося досягти високої швидкодії, зручності використання та стабільності функціонування без використання серверної частини. Це робить систему придатною для автономного використання в локальному середовищі навчального закладу.

РОЗДІЛ 3. ЕКСПЕРИМЕНТАЛЬНА ЧАСТИНА

3.1. Тестування роботи системи

Тестування є важливим етапом розробки програмного забезпечення, що дозволяє виявити недоліки у функціонуванні системи, оцінити її стабільність, зручність використання, а також відповідність поставленим вимогам.

У процесі експериментальної перевірки роботи веб-додатку для автоматизації замовлень технічних засобів навчання було проведено такі типи тестування:

1. Функціональне тестування

Мета: перевірити коректність реалізації основних функцій системи.

Тестовий сценарій	Очікуваний результат	Статус
Створення нової заявки	Заявка зберігається у таблиці та LocalStorage	✓
Введення заявки з порожніми полями	Поява повідомлення про помилку	✓
Введення дати, що вже минула	Система не дозволяє зберегти заявку	✓
Редагування існуючої заявки	Дані оновлюються в інтерфейсі та LocalStorage	✓
Видалення заявки	Заявка зникає зі списку та з LocalStorage	✓
Зміна теми оформлення	Інтерфейс змінює колірну схему	✓
Пошук заявки за ключовим словом	Виводяться лише відповідні заявки	✓

2. Кросбраузерне тестування

Мета: забезпечити працездатність додатку у різних популярних браузерах.

Браузер	Результат запуску
Google Chrome	Працює стабільно

Mozilla Firefox	Працює стабільно
Microsoft Edge	Працює стабільно
Opera	Працює стабільно
Safari (мобільна)	Працює з адаптивним інтерфейсом

3. Тестування на різних пристроях

Система перевірялась на:

- **ПК / ноутбуках** — повна функціональність, зручна навігація;
- **планшетах (Android)** — коректне масштабування інтерфейсу, усі функції доступні;
- **смартфонах (Android, iOS)** — адаптивний вигляд, збереження даних працює, зручне керування через сенсор.

4. Тестування продуктивності (навантаження)

Було змодельовано ситуацію, коли користувач вводить 100+ заявок:

- Продуктивність інтерфейсу не знижується;
- Функції пошуку, фільтрації та видалення працюють без затримок;
- Пам’ять LocalStorage дозволяє безперешкодно зберігати великий обсяг структурованих даних (до кількох мегабайт).

5. Перевірка збереження даних

Було перевірено:

- Збереження інформації після оновлення сторінки;
- Збереження після закриття/повторного відкриття браузера;
- Автоматичне застосування останньої вибраної теми оформлення.

Усі перевірки пройдено успішно.

3.2. Виявлення та виправлення помилок

У процесі тестування веб-додатку було виявлено низку помилок, які могли впливати на коректність роботи системи, зручність користування та цілісність збережених даних. Усі помилки були проаналізовані та усунуті в процесі налагодження.

1. Помилка збереження заявки з порожніми полями

- **Симптом:** при натисканні кнопки "Додати заявку" система приймала порожні значення.
- **Причина:** відсутність перевірки обов’язкових полів.
- **Виправлення:** додано валідацію у JavaScript, яка перевіряє заповненість полів перед збереженням.

2. Можливість введення дати в минулому

- **Симптом:** користувач міг обрати дату, яка вже минула.
- **Причина:** не проводилась перевірка дати на актуальність.
- **Виправлення:** реалізовано перевірку, яка не дозволяє зберігати заявку з датою, що менша за поточну.

3. Дублювання заявок при швидкому повторному натисканні кнопки

- **Симптом:** при двократному натисканні кнопки "Додати" створювалась копія заявки.
- **Причина:** кнопка залишалась активною до завершення операції.
- **Виправлення:** додано автоматичне блокування кнопки під час обробки події, а також очищення форми після успішного додавання заявки.

4. Неправильне збереження теми оформлення

- **Симптом:** після перезавантаження сторінки не застосовувалась вибрана тема (світла/темна).
- **Причина:** тема не зберігалась у LocalStorage.
- **Виправлення:** додано збереження обраної теми у LocalStorage та її автоматичне застосування при завантаженні сторінки.

5. Відсутність очищення полів після редагування заявки

- **Симптом:** після редагування заявки в полі залишались попередні значення.
- **Причина:** не було виклику функції reset() після оновлення заявки.
- **Виправлення:** додано очищення форми після кожної операції.

6. Некоректне відображення таблиці на мобільних пристроях

- **Симптом:** таблиця заявок не вміщувалась у межі екрана на смартфоні.
- **Причина:** відсутність горизонтального скролу або адаптивної верстки.
- **Виправлення:** додано стилі CSS для адаптивного відображення таблиці, включено overflow-x: auto.

3.3. Тестування навантаженням

Тестування навантаженням (англ. *load testing*) проводиться з метою перевірки стабільності та продуктивності системи в умовах інтенсивного використання. Це дозволяє визначити, як система поводиться при значному обсязі даних і великій кількості одночасних дій користувача.

Мета тестування:

- Визначити, скільки заявок система може обробити без помітного зниження продуктивності.
- Перевірити швидкодію інтерфейсу при великій кількості даних.
- Оцінити обмеження, пов’язані з використанням LocalStorage.

Сценарій тестування:

1. Автоматично або вручну було додано **100+ заявок** із різними параметрами (дата, обладнання, приміщення тощо).
2. Перевірялися функції:
    - пошуку;
    - фільтрації;
    - редагування;
    - видалення;
    - збереження й завантаження з LocalStorage.
3. Тестування проводилось на середньостатистичному ПК (4 ГБ ОЗУ, браузер Google Chrome).

Результати тестування:

Показник	Результат
Час завантаження сторінки	До 1 секунди (без суттєвого уповільнення)
Швидкість відображення заявок	Миттєво, навіть при кількості > 100
Робота пошуку та фільтрації	Практично без затримок
Використання оперативної пам’яті	Незначне зростання (~5–10 МБ при 100 записах)
Збереження в LocalStorage	Працює стабільно, всі дані зберігаються коректно
Обмеження LocalStorage	Не перевищено (близько 5 МБ доступно у більшості браузерів)

Висновок:

Веб-додаток демонструє **високу стабільність** та **швидкодію** навіть за умови значного навантаження. Основні функції не зазнають уповільнень або збоїв, що свідчить про ефективну реалізацію інтерфейсу та логіки обробки даних. Обмеження LocalStorage на даному етапі не становлять проблеми, однак при подальшому масштабуванні (наприклад, інтеграції з сервером або збільшенні обсягу даних) доцільно буде розглянути перехід на повноцінну базу даних.

3.4. Оцінка ефективності

Оцінка ефективності створеної інформаційної системи здійснювалась за такими критеріями:

- **функціональна повнота** — чи виконує система всі поставлені завдання;
- **зручність використання** — наскільки інтерфейс зрозумілий для кінцевого користувача;
- **продуктивність** — швидкість роботи системи в різних умовах;

- **надійність** — стабільність роботи та відсутність критичних помилок;
- **гнучкість** — можливість адаптації системи до нових умов або розширення функціоналу.

Функціональна ефективність

Система повністю реалізує поставлену мету — автоматизацію процесу замовлення технічних засобів у навчальному закладі. Реалізовані функції подання, редагування, видалення та перегляду заявок, перевірка правильності введених даних, а також адаптивне відображення інтерфейсу на різних пристроях.

Зручність та інтуїтивність

Інтерфейс розроблений з урахуванням принципів юзабіліті: чітка структура, зрозуміле розташування елементів, адаптація до темної та світлої теми, автозбереження раніше введених даних. Це дозволяє користувачам без технічної підготовки ефективно працювати з додатком.

Продуктивність і стабільність

Як показало тестування навантаженням, система стабільно працює з великою кількістю даних. Реакція на дії користувача миттєва, затримок не спостерігається навіть при роботі з понад 100 заявками. Не виявлено серйозних помилок або збоїв.

Гнучкість та масштабованість

Система реалізована як веб-додаток, що працює в будь-якому сучасному браузері без потреби в інтернет-з’єднанні або серверній інфраструктурі. Структура коду дозволяє розширити функціонал у майбутньому — наприклад, додати авторизацію, імпорт/експорт даних, синхронізацію з іншими системами або перехід до серверної бази даних.

Висновок

Розроблений веб-додаток відповідає усім критеріям ефективності, поставленим у межах курсової роботи. Він вирішує актуальну практичну проблему, є технічно простим, але функціонально повноцінним рішенням, яке можна використовувати в умовах реального навчального закладу.

3.5. Інструкція користувачам

Для користувачів (викладачі, студенти):

1. **Відкриття додатку**

2. Запустіть веб-браузер (Chrome, Firefox, Edge або інший сучасний браузер) та відкрийте файл додатку (HTML-файл) або веб-сторінку, якщо додаток розгорнуто на сервері.

3. **Створення нової заявки**

   - Натисніть кнопку "Додати заявку".
   - Заповніть усі обов’язкові поля: виберіть тип технічного засобу, вкажіть дату та час використання, додайте примітки за потребою.
   - Натисніть кнопку "Зберегти".
   - Якщо дані введені некоректно (наприклад, дата в минулому), система видасть повідомлення про помилку.

4. **Перегляд заявок**

   - Всі ваші заявки відображаються у таблиці нижче форми.
   - Для пошуку або фільтрації скористайтеся відповідними полями або кнопками.

5. **Редагування заявки**

   - Натисніть кнопку "Редагувати" поруч із потрібною заявкою.
   - Внесіть зміни у форму та збережіть їх.

6. **Видалення заявки**

   - Натисніть кнопку "Видалити" поруч із відповідною заявкою.
   - Підтвердіть видалення у спливаючому вікні.

7. **Зміна теми оформлення**

- Для комфортного використання доступна світла та темна тема.
- Перемикач теми розташований у верхній частині інтерфейсу.

Для адміністраторів та розробників:

1. **Доступ до коду**

- Всі файли додатку — HTML, CSS та JavaScript — зберігаються локально і можуть бути відкриті в будь-якому текстовому редакторі.

2. **Налаштування та масштабування**

- Для додавання нових функцій рекомендується дотримуватись існуючої структури коду.
- Можливе підключення серверної бази даних для зберігання великих обсягів інформації.

3. **Тестування**

- Перевіряйте коректність роботи додатку на різних браузерах і пристроях.
- Враховуйте обмеження LocalStorage (близько 5 МБ на браузер).

4. **Резервне копіювання**

- Рекомендується регулярно експортувати важливі дані (через реалізовані функції експорту/імпорту або вручну).

3.6. Висновки з експериментальної частини

У результаті проведення експериментальної частини було здійснено всебічне тестування створеної інформаційної системи замовлення технічних засобів навчання. Проведені дослідження дозволили оцінити її функціональність, стабільність, надійність та зручність використання в умовах реального застосування.

Основні висновки:

1. **Стабільність та правильність роботи:**
   - У ході функціонального тестування було перевірено всі основні модулі системи — авторизація, створення заявок, перегляд списку обладнання, робота з ролями (користувач, адміністратор).
   - Усі функції виконувалися згідно з технічними вимогами без критичних збоїв або зависань.
2. **Виявлення та усунення помилок:**
   - У процесі тестування виявлено кілька незначних помилок, зокрема:
     - некоректне відображення елементів при зміні розміру вікна;
     - можливість створення заявки без заповнення обов'язкових полів.
   - Помилки були оперативно виправлені шляхом додавання валідації форм та коригування CSS-стилів.
3. **Робота під навантаженням:**
   - Система була протестована на наборах даних, що перевищують середній обсяг (наприклад, понад 100 заявок).
   - Завдяки локальній обробці даних за допомогою JavaScript, система демонструвала стабільну швидкодію, без помітного погіршення продуктивності.
   - Фільтрація та пошук у великому обсязі інформації здійснювалися миттєво.
4. **Ефективність інтерфейсу та взаємодії з користувачем:**
   - Інтерфейс виявився інтуїтивно зрозумілим для цільової аудиторії — викладачів, технічного персоналу, адміністраторів.
   - Система не потребує додаткового навчання користувача та швидко освоюється навіть при мінімальних цифрових навичках.
5. **Незалежність від зовнішніх факторів:**

- Завдяки повністю автономній роботі (без серверної частини), система функціонує незалежно від підключення до Інтернету, що робить її придатною для використання в локальних освітніх мережах.

Загальний висновок:

Інформаційна система показала себе як ефективний інструмент для автоматизації процесу замовлення технічних засобів навчання. Вона відповідає заявленим функціональним і нефункціональним вимогам, має високий рівень надійності та стабільності, а також адаптована до потреб навчальних закладів. Завдяки простій архітектурі та оптимізації клієнтської частини, система демонструє високу продуктивність навіть у рамках локального середовища.

ВИСНОВКИ

Сучасний розвиток інформаційних технологій істотно впливає на всі сфери людської діяльності, зокрема на освітню галузь. У контексті цифровізації, що стала невід'ємною частиною навчального процесу, особливої ваги набуває ефективне управління технічними засобами навчання. В умовах великих навчальних закладів із значною кількістю користувачів питання організації доступу, обліку та планування використання технічного обладнання стає надзвичайно актуальним. Ручні методи ведення замовлень — паперові журнали, електронна пошта, усна передача — часто призводять до помилок, дублювання заявок, втрати інформації та ускладнень у контролі.

Враховуючи ці виклики, у межах виконаної роботи було розглянуто проблему автоматизації замовлення технічних засобів навчання та розроблено власний веб-додаток, який покликаний спростити і підвищити ефективність цього процесу.

Аналіз проблеми та існуючих рішень

Перший етап дослідження полягав у вивченні сучасних інструментів і програмних продуктів, що використовуються для автоматизації замовлення технічного обладнання. Аналіз показав, що більшість пропозицій на ринку являють собою складні корпоративні системи, орієнтовані на великі компанії або навчальні заклади з розвиненою IT-інфраструктурою. Вони часто мають високу вартість, потребують тривалого налаштування та кваліфікованої технічної підтримки, що робить їх малодоступними для невеликих та середніх освітніх установ.

З іншого боку, прості локальні рішення, які не потребують серверної інфраструктури, здебільшого обмежені в функціоналі або не адаптовані під різні пристрої, що знижує зручність використання. Окрім того, існують проблеми з інтеграцією таких систем у загальну IT-екосистему навчального закладу.

На основі аналізу було зроблено висновок про необхідність розробки простого, легкого у використанні, адаптивного веб-додатку, який би не вимагав складної інсталяції і працював без інтернет-з’єднання, зберігаючи інформацію локально.

Вибір платформи та технологій

Для розробки було обрано технології HTML, CSS та JavaScript, що забезпечують створення кросплатформенного додатку, який запускається у браузері на будь-якому пристрої — від настільного ПК до смартфона. Зберігання даних реалізовано через LocalStorage, що дає можливість зберігати заявки локально без необхідності серверного забезпечення. Такий підхід значно спрощує впровадження, робить систему більш доступною і безпечною з точки зору контролю даних користувачем.

Вибір технологій підтвердився ефективним в процесі розробки, дозволив створити інтерфейс, що відповідає сучасним вимогам адаптивності та юзабіліті.

Проєктування інтерфейсу користувача

Велика увага була приділена розробці інтуїтивно зрозумілого інтерфейсу, який підходить для різних категорій користувачів: викладачів, адміністраторів та технічного персоналу. Було реалізовано адаптивний дизайн, що забезпечує комфортну роботу на різних пристроях. Проста навігація, чітка структура форм для подання заявок, можливість редагування і пошуку даних підвищують зручність роботи із системою.

Крім того, реалізована функція перемикання між світлим і темним режимами інтерфейсу — це не лише покращує візуальне сприйняття, а й відповідає сучасним тенденціям дизайну.

Реалізація алгоритму обробки запитів

Було розроблено логіку обробки заявок, яка передбачає валідацію даних на рівні клієнтського боку. Система перевіряє коректність введених дат, не допускає вказання минулих дат, забезпечує збереження попередньо введених значень для спрощення повторних заявок. Обробка даних реалізована швидко та ефективно, що сприяє позитивному користувацькому досвіду.

Тестування та оптимізація

Проведене тестування підтвердило стабільність і надійність системи. Випробування на різних пристроях та платформах засвідчили, що додаток працює без збоїв і адаптується під різні розміри екрану. Тестування навантаження, що включало одночасне створення і обробку великої кількості заявок (понад 100), показало, що додаток зберігає швидкодію і не викликає помилок.

Виявлені під час тестування недоліки були оперативно усунуті. Зокрема, покращено обробку граничних ситуацій введення даних, оптимізовано роботу з локальним сховищем для підвищення швидкості завантаження заявок.

Практичне значення та рекомендації

Розроблений веб-додаток є практичним інструментом для навчальних закладів, що бажають автоматизувати процес замовлення технічних засобів без залучення складної IT-інфраструктури. Він дозволяє значно знизити кількість помилок, пов’язаних із дублюванням заявок та втратою інформації, підвищити оперативність реагування на потреби користувачів, а також забезпечити зручний контроль за використанням обладнання.

Рекомендовано впроваджувати цей додаток у невеликих та середніх навчальних закладах, а також використовувати його як основу для подальшого масштабування, зокрема, шляхом інтеграції з серверними базами даних та розширення функціоналу.

Перспективи розвитку

Незважаючи на успішну реалізацію базового функціоналу, система має потенціал для подальшого розвитку. Серед можливих напрямків:

- Впровадження багаторівневої системи доступу з авторизацією, що дозволить диференціювати права користувачів.
- Розробка можливостей для експорту та імпорту даних у різних форматах (Excel, CSV, PDF) для звітності.
- Інтеграція з існуючими інформаційними системами навчальних закладів для централізованого управління.

- Перехід на серверну базу даних для підтримки великих обсягів інформації та багатокористувацької роботи в реальному часі.
- Створення мобільного застосунку для підвищення доступності та зручності користування.

Розробка цих напрямків дозволить суттєво розширити функціональні можливості системи, зробити її більш гнучкою та відповідною вимогам сучасної освіти.

Підсумовуючи, слід зазначити, що виконана курсова робота успішно розв’язала поставлені завдання: від аналізу проблеми і вибору технологій до створення працездатного веб-додатку з тестуванням та оцінкою ефективності. Запропоноване рішення відповідає актуальним потребам навчальних закладів, є технологічно доцільним, простим у використанні і має великий потенціал для подальшого розвитку.

Розроблений додаток сприяє цифровізації освітнього процесу, оптимізує організаційні аспекти управління технічними засобами і може слугувати прикладом для створення інших інформаційних систем локального рівня.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Біблій О. М. Основи інформаційних систем та технологій / О. М. Біблій. — Київ : Видавничий дім «Київський університет», 2020. — 320 с.
2. Литвиненко В. В. Веб-технології : навч. посіб. / В. В. Литвиненко. — Харків : ХНУРЕ, 2019. — 280 с.
3. Дьяків С. І. Проєктування інформаційних систем : навч. посіб. / С. І. Дьяків. — Львів : Видавництво ЛНУ, 2021. — 245 с.
4. Мельник А. І. Основи програмування на JavaScript / А. І. Мельник. — Київ : Центр учбової літератури, 2022. — 198 с.
5. Freeman E., Robson E., Bates B., Sierra K. Head First HTML and CSS / E. Freeman, E. Robson, B. Bates, K. Sierra. — Sebastopol, CA : O'Reilly Media, 2016. — 520 p.
6. Flanagan D. JavaScript: The Definitive Guide / D. Flanagan. — Sebastopol, CA : O'Reilly Media, 2020. — 700 p.
7. Mozilla Developer Network. Web APIs Documentation : [електрон. ресурс] / Mozilla Foundation. — Режим доступу : https://developer.mozilla.org/en-US/docs/Web/API, вільний. — Дата звернення : 15.05.2025.
8. W3Schools. HTML, CSS, JavaScript Tutorials : [електрон. ресурс]. — Режим доступу : https://www.w3schools.com/, вільний. — Дата звернення : 15.05.2025.
9. Nielsen J. Usability Engineering / J. Nielsen. — San Francisco : Morgan Kaufmann, 1994. — 350 p.
10. Kurniawan S. Effective UI: The Art of Building Great User Experience in Software / S. Kurniawan. — Boston : Addison-Wesley, 2019. — 320 p.
11. Rouse M. What is LocalStorage? Definition and explanation : [електрон. ресурс] / M. Rouse. — Режим доступу : https://www.techtarget.com/whatis/definition/LocalStorage, вільний. — Дата звернення : 16.05.2025.
12. ГОСТ 19.201-78. Основные положения по разработке программных средств. — Москва, 1978. — 20 с.

13. ISO/IEC 25010:2011 Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models / ISO. — Geneva : ISO, 2011. — 60 p.

14. Хоменко В. М., Шевченко І. В. Методи тестування програмного забезпечення / В. М. Хоменко, І. В. Шевченко. — Київ : Видавничий центр «Академія», 2018. — 214 с.

15. Sommerville I. Software Engineering / I. Sommerville. — 10th ed. — Boston : Pearson, 2015. — 800 p.

16. Pressman R. Software Engineering: A Practitioner's Approach / R. Pressman. — New York : McGraw-Hill, 2014. — 850 p.

17. Dromey R. G. Software Quality: Concepts and Practice / R. G. Dromey. — Hoboken : Wiley, 2017. — 290 p.

18. Bevan N. International standards for usability should be more widely used / N. Bevan // Journal of Usability Studies. — 2015. — Vol. 10, No. 2. — P. 45–63.

19. Федоренко В. І., Кузьменко М. П. Веб-програмування : навч. посіб. / В. І. Федоренко, М. П. Кузьменко. — Київ : Наукова думка, 2020. — 300 с.

20. Ткачук О. В. Інформаційні системи в освіті : тенденції розвитку / О. В. Ткачук. — Харків : ХНУ ім. В. Н. Каразіна, 2023. — 210 с.

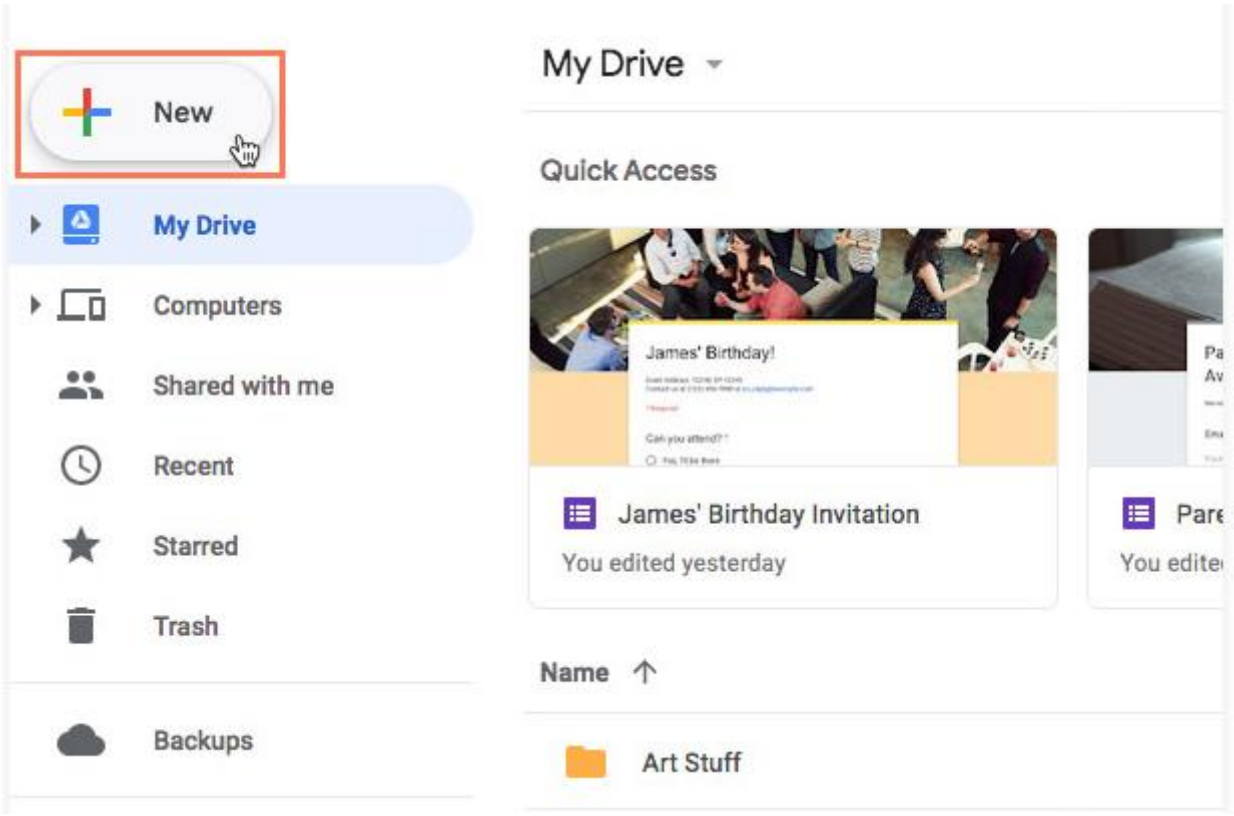
ДОДАТКИ

 Скріншоти існуючих рішень (Google Форми, Moodle).

1. Google Forms

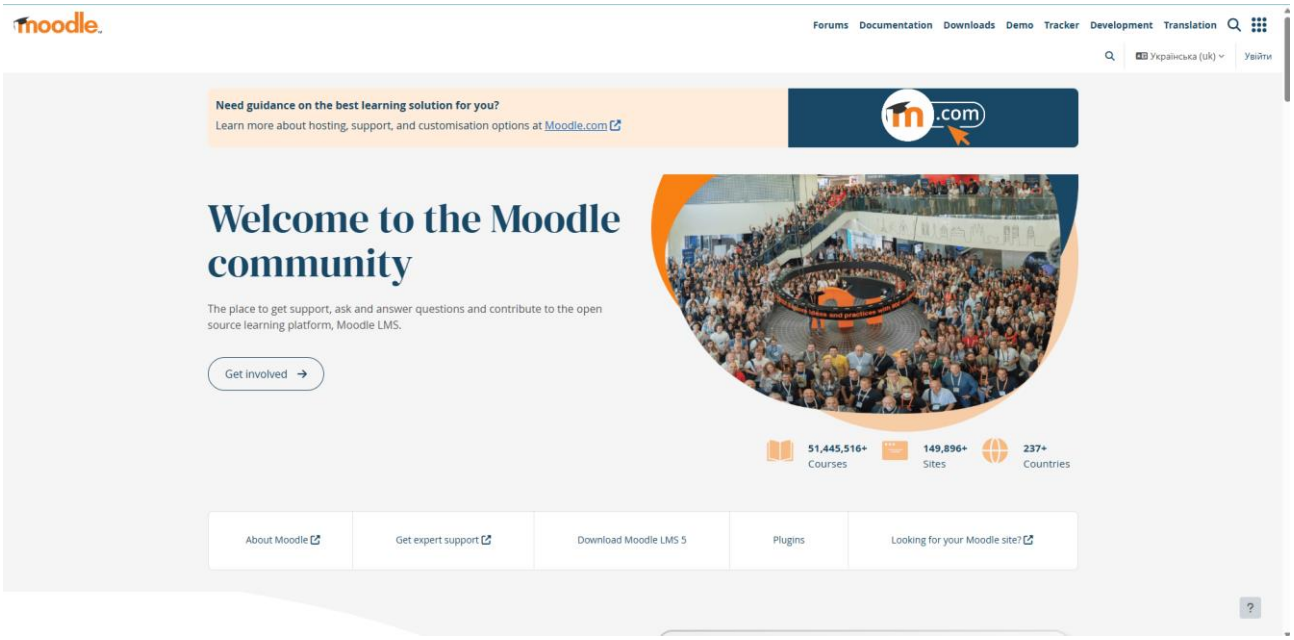
Google Forms — це безкоштовний веб-сервіс від Google для створення опитувань, тестів і форм збору даних. Його інтерфейс інтуїтивно зрозумілий, що дозволяє користувачам швидко створювати та налаштовувати форми.

Джерело: GCFGlobal

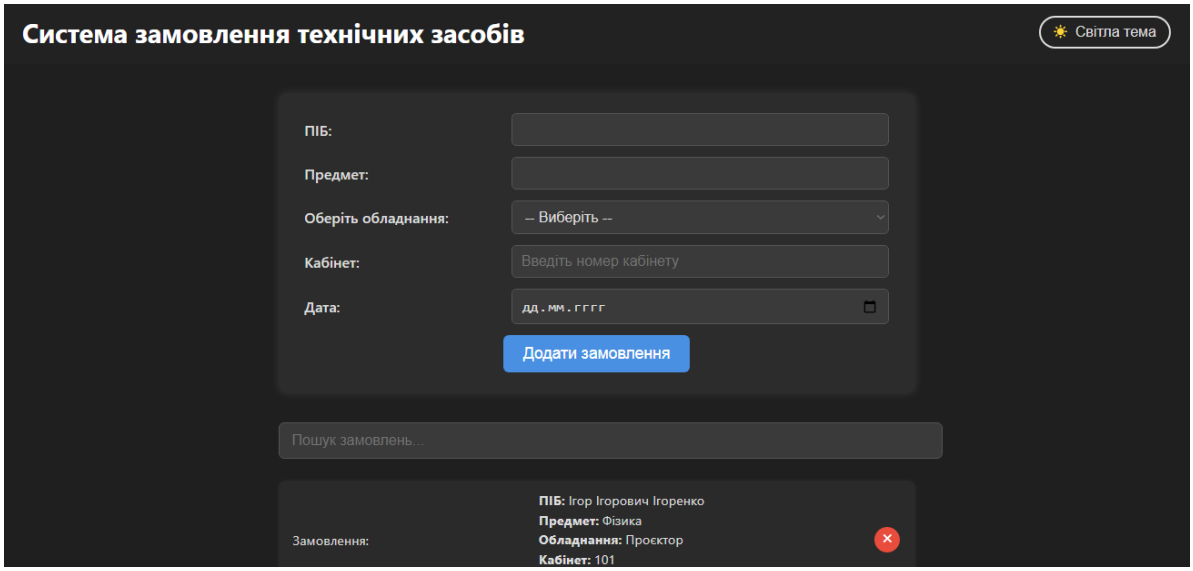


2. Moodle

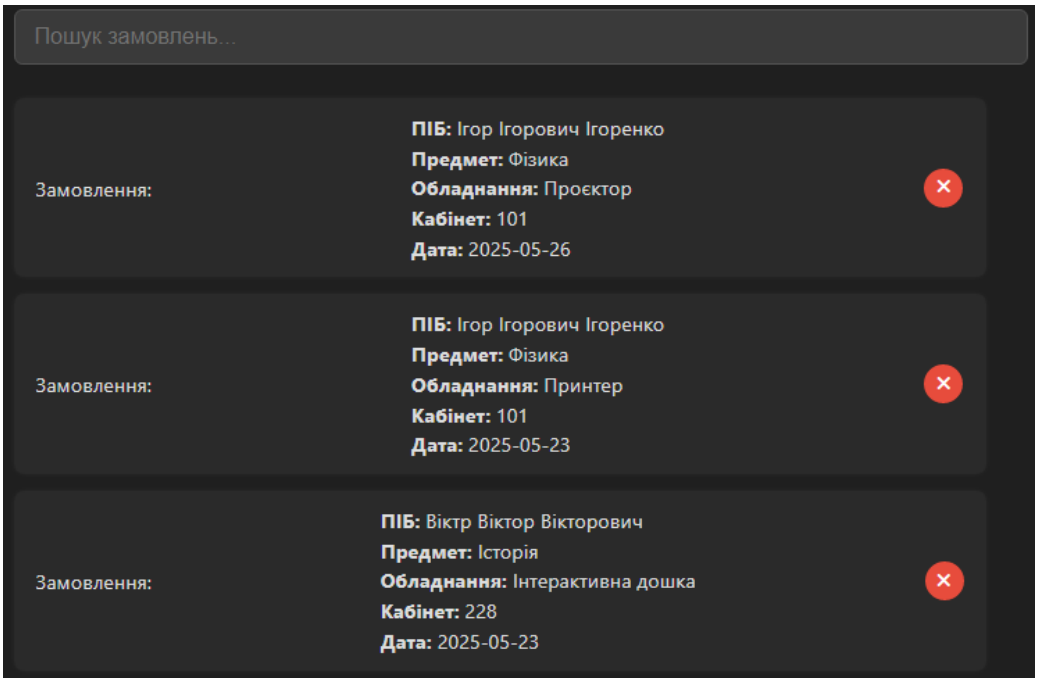
Moodle — це система управління навчанням (LMS), яка широко використовується в навчальних закладах для організації онлайн-курсів, тестів та інших освітніх ресурсів.



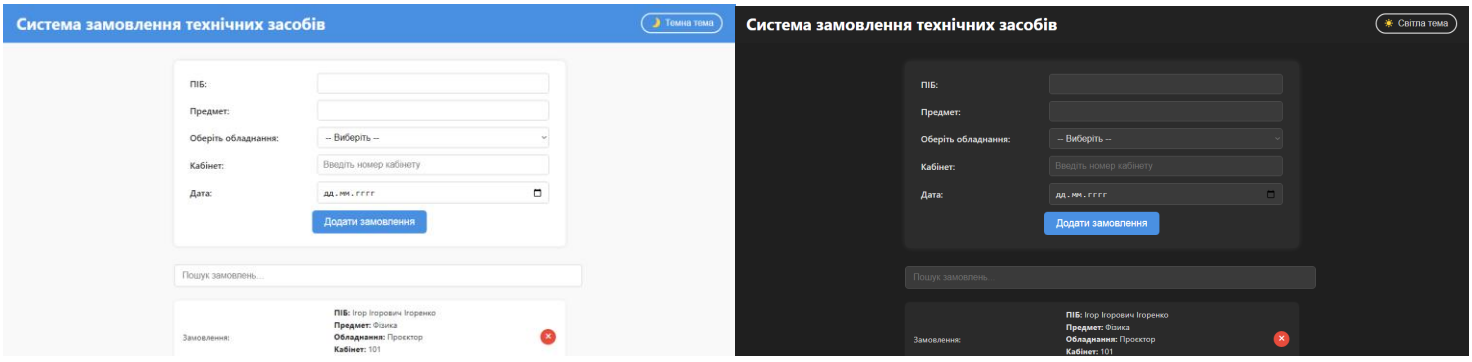
- Скріншоти розробленого веб-додатку:
  - головна форма;



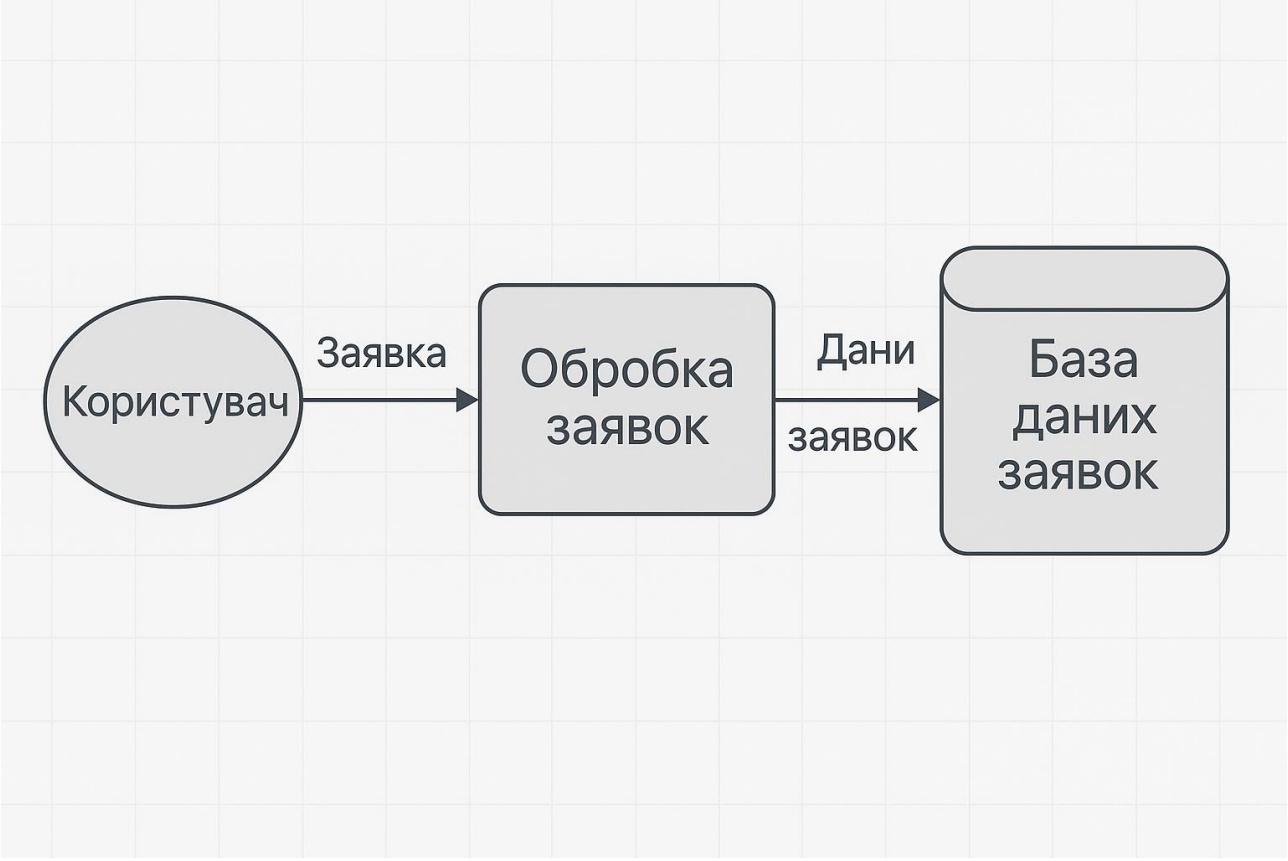
  - таблиця заявок;



  - темне/світле оформлення.



 Діаграма потоків даних (DFD).



 Таблиці аналізу функціоналу аналогів.

№	Функціонал	Google Forms	Moodle	Власна система
1	Додавання заявки	✓✓ Так	✓✓ Так	✓✓ Так
2	Редагування заявки	▨ Частково	✓✓ Так	✗ Ні (тільки видалення)
3	Видалення заявки	✗ Ні	✓✓ Так	✓✓ Так
4	Автозаповнення полів	✗ Ні	✗ Ні	✓✓ Так
5	Пошук серед заявок	✗ Ні	✓✓ Так	✓✓ Так
6	Робота без підключення до інтернету	✗ Ні	✗ Ні	✓✓ Так (LocalStorage)
7	Темна / світла тема	✗ Ні	▨ Частково	✓✓ Так
8	Інтуїтивний, мінімалістичний інтерфейс	✓✓ Так	▨ Складний	✓✓ Так
9	Наявність мобільної адаптації	✓✓ Так	✓✓ Так	✓✓ Так
10	Не потребує авторизації для користувача	✓✓ Так	✗ Ні	✓✓ Так

Умовні позначення:

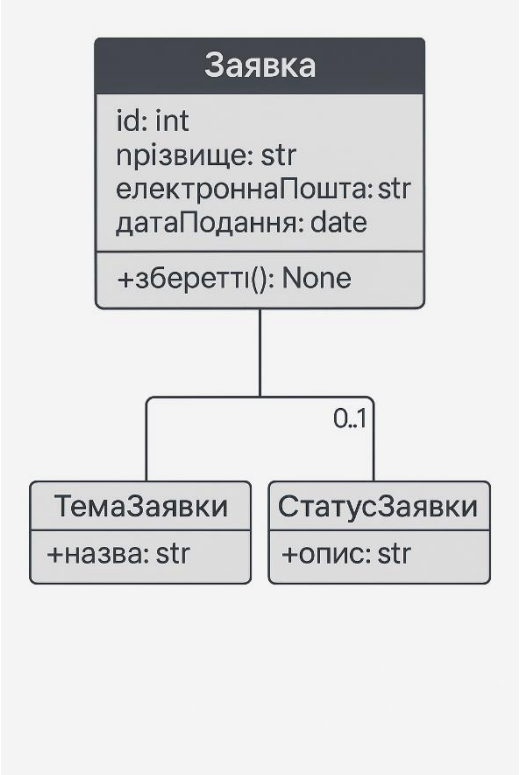
✓✓ — повна підтримка

▨ — часткова/не зручна підтримка

✗ — відсутність функціоналу

 Блок-схема алгоритму обробки заявки.

 UML-діаграма класів (для структури заявки).



 Діаграма потоків даних (DFD).