Національний університет «Одеська політехніка»
Інститут комп'ютерних систем
Кафедра інформаційних систем

КУРСОВА РОБОТА

з дисципліни «Об'єктно-орієнтоване програмування»

Тема «Створення сайту-додатка ToDo лист за допомогою нативного JavaScript »

Студента (ки) __2__ курсу AI-222 групи
Спеціальності 122 – «Комп’ютерні науки»
_____Турбарова І.О._____
(прізвище та ініціали)
Керівник доц. Годовиченко М.А.
_____
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Національна шкала _____
Кількість балів: _____
Оцінка: ECTS _____

Члени комісії

_____ _____
(підпис) (прізвище та ініціали)
_____ _____
(підпис) (прізвище та ініціали)
_____ _____
(підпис) (прізвище та ініціали)

м. Одеса – 2024 рік

Національний університет «Одеська політехніка»
Інститут комп'ютерних систем
Кафедра інформаційних систем

ЗАВДАННЯ

НА КУРСОВУ РОБОТУ

студенту Турбарову Ігорю Олександровичу група AI-222

1. Тема роботи
«Створення сайту-додатка ToDo лист за допомогою нативного JavaScript»

2. Термін здачі студентом закінченої роботи 06.06.2024

3. Початкові дані до проекту (роботи)
Програма повинна виконувати: додавання елемента; видалення елемента; виведення всіх даних; перевірка заповнення всіх полів форми створення ToDO-елемента; можливості отримати всіх збережених даних на одному пристрої.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які належить розробити)
Вступ. Теоретичні відомості про JavaScript, HTML, CSS, а також підхід MVC. Програмна реалізація сайту-додатка ToDo лист. Інструкція користувача. Висновки.

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)
скріншоти сайту-додатку, які демонструють його роботу.

Завдання видано 20.03.23 _____
(підпис викладача)

Завдання прийнято до виконання 20.03.23 _____
(підпис студента)

АНОТАЦІЯ

У курсовій роботі запропонована програмна реалізація сайту-додатка ToDo лист за допомогою нативного JavaScript . Розроблено програму, яка виконує усі вимоги, зазначені в завданні до курсової роботи, та описано її складові. У теоретичній частині проаналізовано JavaScript, HTML, CSS, а також підхід MVC, зокрема загальні відомості, сферу застосування, позитивні і негативні сторони підходу MVC. Запропоновано інструкцію користувача, яка містить рекомендації щодо практичного застосування зазначеного програмного продукту.

ABSTRACT

In the course work, the software implementation of the ToDo list WEB-application site using native JavaScript is proposed. A program has been developed that fulfills all the requirements specified in the assignment for the term paper, and its components are described. The theoretical part analyzes JavaScript, HTML, CSS, as well as the MVC pattern, including general information, scope, positive and negative sides of the MVC pattern. A user manual is offered, which contains recommendations on the practical use of the specified software product.

Зміст

ВСТУП

За століття людством накопичено певний прошарок інформації, яка є по суті є даними, які можуть підлягати обробці.

Сьогодні немає жодного смартфону, планшету, котрий не мав би у своєму інтерфейсі такого розділу як «Замітки», який створений як TODO-list. Його функції досить широкі, зокрема планер, записна книжка та ін. Такий саме принцип мають і застосунки типу Padlet, Miro, Twiddla та ін., що активно використовуються у сфері освіти та бізнес-комунікації. Тому, обрана нами тема курсового проєкту «Створення сайту-додатка ToDo лист за допомогою нативного JavaScript», є **актуальною,** а також дозволить удосконалити і закріпити навички, отримані протягом вивчення дисципліни «Об'єктно-орієнтованого програмування» та факультативного курсу Android.

Мета курсової роботи – розробити, описати та перевірити працездатність сайту-додатка ToDo-list за допомогою нативного JavaScript. Програму написано на HTML, CSS, JavaScript з використанням підходу MVC. Вона повинна виконувати додавання елемента; видалення елемента; виведення всіх даних; здійснювати перевірку заповнення всіх полів форми створення ToDO-елемента; забезпечити можливості отримання всіх збережених даних на одному пристрої.

HTML, JavaScript, CSS – це три кити, на яких працюють веб-застосунки. HTML – каркас, CSS –візуальне оформлення (в CSS3 з'явилась можливість реалізовувати й анімації), а JavaScript – логіка, інтерактивність та взаємодія із користувачем.

Соціальні мережі, інтернет-магазини, портали новин: основу більшості веб-сторінок складає HTML. Тому не буде перебільшенням назвати його головною мовою інтернету. Завдяки HTML можна створювати структуровані сторінки з багатьма функціями.

JavaScript забезпечує інтерактивність сайтів. У найближчому майбутньому JavaScript залишиться основною на цій позиції, бо все, що

створюється як альтернатива JavaScript, у кінцевому підсумку конвертується в JavaScript.

Сфера JavaScript-розробки швидко розвивається. За останні п’ять років з’явилось стільки бібліотек й фреймворків, що фізично не вистачає часу все спробувати. Навіть є меми про те, що кожного дня з’являється новий JavaScript-фреймворк. Це підтверджує попит на мову з боку клієнтів та компаній, а отже спеціалісти ще мають, куди розвиватись у професії.

CSS це мова стилізації веб-сторінок і використовується для оформлення візуальної складової сайтів, дозволяє створити стиль сайту: кольори, шрифти, розташування блоків інформації, різні візуальні ефекти і т.д.

Без CSS веб-сторінки були б нудними і не привабливими для користувачів, а використання інших технологій, таких як таблиці HTML, спричинило б труднощі в управлінні стилями. Крім того, CSS використовується не тільки для оформлення веб-сторінок, але й в інших областях веб-розробки, таких як створення тем оформлення для блогів та сайтів на платформах керування контентом. CSS також є ключовою технологією створення адаптивних веб-додатків, які повинні бути доступні на багатьох різних пристроях.

MVC перетворює складну розробку застосунків на більш керований процес. Він дозволяє декільком розробникам працювати над кожним аспектом програми окремо. Цей підхід особливо корисний для веб-застосунків та мобільних застосунків, оскільки дозволяє легко керувати тим, як дані відображаються перед користувачем.

Робота над курсовим проектом – розробкою програми з властивостями, описаними вище, мала такі етапи: аналіз завдання та формування задач його реалізації; реалізація цих задач у програмному коді; оформлення програми; перевірка її працездатності; оформлення пояснювальної записки та потрібної документації.

1. Теоретичні відомості про мови програмування JavaScript, HTML, CSS, а також підхід MVC

1.1 Основи JavaScript

JavaScript — це динамічна, об'єктно-орієнтована мова програмування, створена Бренданом Ейхом, завдання якої полягає в тому, що код має виконуватися на стороні користувача та асинхронно обмінюватися даними із сервером [6]. Основний її функціональний обов'язок — "склеювання" складових елементів веб-сторінки: зображення, анімації, плагінів за умови легкості освоєння web-дизайнерами та web-програмістами початківцями [5].

Варто зазначити, що незважаючи на схожість назв, мови Java та JavaScript є двома різними мовами, що мають відмінну семантику, хоча й мають схожі риси в стандартних бібліотеках та правилах іменування. Синтаксис обох мов успадкований від мови C, але семантика та дизайн JavaScript є результатом впливу мов Self та Scheme.

Історія створення мови Java пов’язана із компанією Netscape Communications, яка у 1995 році бурхливо розвивається у сфері веб-технологій й відвойовує позиції у першого браузера NCSA Mosaic. Веб потребував мови, якою просто програмувати. Так з’явилася ідея скриптової мови Mocha, з якою можна працювати в браузері. Водночас Sun Microsystems завершувала роботу над своєю мовою програмування Java. І Netscape були готові інтегрувати їхню мову у свій браузер. Але Java призначалася для більш обізнаної в програмуванні аудиторії. Це суперечило призначенню Mocha, що мала стати провідником між Java і користувачами, які потребували її для розробки веб-сайтів. Після доопрацювання прототип Mocha почали використовувати в Netscape Communicator і він отримав назву LiveScript. А у грудні 1995 року угода між Netscape Communications і Sun була закрита. Так народилася мова JavaScript, а Java служила для створення складних компонентів у браузері.

У 2015-му нова версія мови, ES6, дала JavaScript друге життя – з’явилися нові стандарти, можливість працювати з константами

та виконувати багато функцій при скороченому коді. Цей реліз – єдиний, що підтримується всіма браузерами, хоча є й покращені версії.

У 2020 році JavaScript вперше випередила Java і стала найпопулярнішою мовою програмування. Згідно із рейтингом DOU, у 2021 році цією мовою пишуть 18% розробників, і вона залишається на першому місці [5; 7].

Основні особливості JavaScript – динамічність, гнучкість роботи із функціями та універсальність. Вона підтримується всіма сучасними браузерами, легко інтегрується з версткою (HTML) та дає змогу налаштувати комунікацію із сервером. Серед інших переваг:

- тип даних визначається, коли змінній або константі присвоюється значення;
- у JavaScript функції можна як виконувати, так і повертати, передавати їх як параметри іншим функціями і привласнювати як значення змінних;
- методологія об’єктно-орієнтованого програмування дає змогу представити програму у вигляді сукупності об’єктів;
- дозволяє частково перенести бізнес-логіку із сервера на бік користувача, тобто виконувати код у браузері, що своєю чергою зменшує навантаження на сервери.
- має розвинену інфраструктуру та активну спільноту. Так, веб-розробники можуть працювати з великою кількістю бібліотек і фреймворків, таких, як: React, Angular і Vue, декількома пакувальниками (Webpack, Gulp) та допоміжними бібліотеками (Lodash, axios, та ін.). [7]
- швидкість написання значно вища, ніж на Java або C #.

Варто вказати й на декілька обмежень. Основне – це робота з файловою системою, тобто недоступне зчитування файлів. Крім того, JavaScript не підтримує віддалений доступ до системи, а тому мову незручно використовувати для мережевих застосунків [7].

Навколо JavaScript сформована ціла інфраструктура. Наприклад:

- бібліотеки та фреймворки для створення додатків (React, Vue);
- пакувальники (Webpack, Gulp);
- допоміжні бібліотеки;
- генератори статичних сайтів (Gatsby.js, Next.js);
- тестові Фреймворки (Jest, Mocha, Chai, Jasmine, Cypress, Protractor) [8].

Усі переваги JavaScript настільки вагомі, що разом з HTML і CSS ця мова програмування входить в базовий набір того, що вивчають початківці-програмісти: можна відкрити консоль розробника в браузері і одразу почати писати код і практикувати.

1.2 Характеристика HTML

HTML (англ. HyperText Markup Language – мова розмітки гіпертекстових документів) – текстова мова розмітки, призначена для розмітки документів, які містять текст, зображення, гіперпосилання, тощо [9]. Процес розміщення в текст кодів HTML називають розміткою. В основі мови розмітки лежать теги, що складаються sз кількох слів або набору символів. Вони задають певні властивості даним або тексту, які знаходяться всередині сформованої конструкції [9].

Мова HTML є додатком SGML (стандартної узагальненої мови розмітки) і відповідає міжнародному стандарту ISO 8879. У всесвітній павутині HTML-сторінки, як правило, передаються браузерам від сервера за протоколами HTTP або HTTPS, у вигляді простого тексту або із використанням шифрування. Структурними і семантичними елементами є дескриптори. Дескриптори також часто називають «тегами». Також, в HTML внесена підтримка гіпертексту [9].

Тридцять чотири роки тому, 20 грудня 1990 року, британський учений з Європейської організації (CERN), Тім Бернерс-Лі, запустив перший у світі веб-сайт. Там описувалася нова технологія глобального обміну інформацією – «Всесвітня павутина», яка й стала основою для розвитку сучасного

інтернету. Він розробив мову HTML і створив веб-оглядач та серверне програмне забезпечення для запропонованої ним системи гіпертекстових документів. За його задумом автора, проект мав допомагати вченим шукати інформацію і ділитися нею. Ідею глобальної системи комунікацій World Wide Web Тім Бернерс-Лі представив ще в 1989 році. «Розпливчасто, але цікаво», – прокоментував концепцію його начальник Майк Сендалл, але дав добро на її втілення [10].

Наприкінці 1991 року Тім Бернерс-Лі опублікував е мережі Інтернет перший загальнодоступний опис мови HTML, відомий як документ «HTML теги» (HTML Tags). У ньому були описані 20 елементів найпершої схеми розмітки HTML-документів. 13 з них і сьогодні використовують у мові HTML.

Бернерс-Лі розглядав HTML як похідну мову від SGML, і в середині 1993 року її офіційно визнали такою, опублікувавши першу специфікацію HTML: "Hypertext Markup Language (HTML)", авторами якої були Тім Бернерс-Лі та Ден Конолі.

У 1993 році Дейв Раджетт запропонував стандартизувати розмітку для визначення таблиць та інтерактивних форм.

На початку 1994 року створено робочу групу HTML (HTML Working Group), яка займалась проблемами стандартизації мови розмітки HTML.

У 1995 році робоча група HTML завершила роботу над першою специфікацією «HTML 2.0», що мала бути використана як базовий стандарт для подальших удосконалень HTML. Версія 2.0 окреслювала чіткі відмінності між новим виданням специфікації та попередніми проектами.

З 1996 року специфікації HTML затверджувались консорціумом W3C, враховуючи доповнення до розмітки, що впроваджувалися компаніями-розробниками веб-оглядачів.

У 2000 році HTML стала міжнародним стандартом (ISO/IEC 15445:2000).

Остання специфікація HTML, опублікована W3C наприкінці 1999 року, має назву «HTML 4.01 Recommendation» [10].

Мова HTML має кілька основних елементів, які використовуються для визначення структури та вигляду веб-сторінок. Деякі з них включають:

1. Заголовки (Headings) – використовуються для створення різних рівнів заголовків на сторінці. Від <h1> (найвищий рівень) до <h6> (найнижчий рівень), вони допомагають організувати та виділити важливі розділи тексту.

2. Параграфи (Paragraphs) – тег <p> використовується для розміщення абзаців тексту. Він дозволяє структурувати вміст сторінки на логічні блоки і полегшує читання для користувачів.

3. Посилання (Links) – тег <a> використовується для створення гіперпосилань на інші сторінки, документи або ресурси в Інтернеті. Він є навігатором по сайту та дозволяє користувачам переходити до інших джерел інформації.

4. Зображення (Images) – тег <img> використовується для вставки зображень на веб-сторінку. Він дозволяє додавати ілюстрації, фотографії та інші графічні елементи до контенту сторінки [11].

Це лише кілька прикладів основних елементів мови гіпертекстової розмітки. Існує багато інших тегів, які можна використовувати для розмітки різних типів вмісту.

Основне використання мови гіпертекстової розмітки включає:

1. Створення веб-сторінок: HTML дозволяє розмітити текст, додати зображення, вбудовані відео, аудіо та інші медіа елементи на сторінку.
2. Розробка веб-додатків: за допомогою HTML, разом з CSS (Cascading Style Sheets) та JavaScript, можна створювати складні веб-додатки зі змінним вмістом та інтерактивними елементами.
3. Оптимізація для пошукових систем: Використання правильної розмітки HTML допомагає зрозуміти пошуковим системам

структуру сторінки та зміст, що сприяє покращенню SEO (Search Engine Optimization) [12].

Мільярди людей щодня бачать на своїх комп'ютерах і мобільних пристроях результати інтерпретації HTML-документів своїм браузером. Незважаючи на те, що основними читачами HTML є браузери, будь-яка людина може легко зрозуміти його структуру, оскільки ця мова проста й логічна, її добре знання часто дуже допомагає в сучасному світі.

HTML набув великої популярності завдяки своїм незаперечним перевагам:

- мову легко вивчати та використовувати;
- вона підтримується всіма поширеними браузерами;
- її можна інтегрувати з мовами програмування.

Тім Бернерс-Лі, творець HTML, якось сказав: «Мережа перетворила дані на золото 21 століття» [12]. Розширюючи цю метафору, можемо сказати, що HTML є основним інструментом для видобутку та зберігання дорогоцінного металу – інформації.

1.3 Теоретичні відомості про CSS

CSS – це абревіатура від **Cascading Style Sheets** і перекладається з англійської як каскадні аркуші або таблиці стилів. Термін «каскадні таблиці стилів» було запропоновано Хоконом Лі у 1994 році. Разом із Бертом Босом він почав розвивати CSS [13]. Етимологія назви пов'язана із тим, що CSS це не мова програмування, у ній не описується логіка, а її код виглядає як список певних конструкцій і нагадує статичні табличні дані. А каскадні тому, що стиль того самого елемента може формуватися з різних джерел, за певним пріоритетом.

Принцип закладений у CSS зародився вже як півстоліття тому, але у своєму сучасному вигляді ця технологія сформувалася в 1994-95 роках, на зорі появи організації W3C, яка займається стандартизацією багатьох технологій, що використовуються при створенні сайтів.

Перед появою CSS стилі для веб-сторінок визначалися безпосередньо у розмітці HTML. Це призводило до багатьох проблем, таких як: складність підтримки та зміни дизайну, а також відсутність можливості повторно використовувати стилі на різних сторінках.

Створення CSS стало відповіддю на ці проблеми. З її появою розробники змогли відокремити зовнішній вигляд від вмісту веб-сторінки, що зробило код більш зрозумілим та легким для обслуговування. Завдяки CSS, стилі можна було визначати в окремому файлі та застосовувати їх до багатьох сторінок. Це відкрило нові можливості для створення ефектів, макетів та дизайнів, що зробило веб-сторінки більш естетичними та гнучкими [13].

CSS1 був першим стандартом CSS, опублікованим у 1996 році. Це був значний крок у розвитку веб-розробки, оскільки CSS1 стандартизував багато властивостей та атрибутів, які до цього часу були визначені тільки в різних браузерах. За допомогою CSS1 розробники могли створювати більш якісний та сучасний дизайн для своїх веб-сторінок.

CSS2 був випущений у 1998 році і представив безліч нових можливостей для розробників веб-сторінок, які дозволяли створювати більш складний та гнучкий дизайн. Крім того, CSS2 включав нові функції, такі як: позиціонування елементів, фони та межі, які робили розробку веб-сторінок зручнішою та ефективнішою.

CSS3 був випущений в 1999 році. Він розширив можливості CSS2 та представив безліч нових властивостей та атрибутів. Серед нових можливостей CSS3 можна виділити:

- нові селектори: CSS3 представив багато нових селекторів, які дозволяють вибирати елементи на основі їх положення в документі, а також їх атрибутів та вмісту;
- анімації та переходи: CSS3 дозволяє створювати анімації та переходи між станами елементів без використання JavaScript;

- градієнти: CSS3 представив нові типи градієнтів, які дозволяють створювати більш складні та барвисті фони;
- тіні та обведення: CSS3 дозволяє створювати більш складні тіні та обведення для елементів сторінки [13].

Однією із значних змін у CSS3 було введення модульної архітектури. Модулі CSS є набором властивостей та атрибутів, пов'язаних із певною функціональністю, такою як кольори, фони або шрифти. Це спрощує розробку та підтримку CSS, оскільки розробники можуть використовувати тільки ті модулі, які їм потрібні для свого проекту, і не завантажувати код, що не використовується [13]. CSS3 також представив безліч нових модулів. Наприклад, модуль Grid Layout дозволяє створювати складні сітки на сторінці, а модуль Flexbox дозволяє створювати гнучкі макети, які легко адаптуються до різних пристроїв і екранів.

Таким чином, CSS – це загальновизнаний міжнародний стандарт, найпотужніший інструмент, один з найосновніших складових практично будь-якої веб-сторінки, без якої неможливо уявити сучасну веб-розробку. Разом sз тим технологія CSS досить проста в освоєнні.

1.4 Характеристика підходу MVC

Усе більше веб-розробників розуміють важливість використання структурованих та модульних підходів до розробки Front-end.

Один з таких підходів — це підхід Модель-Вигляд-Контролер (MVC). Ця архітектурна концепція дозволяє розбити ваш застосунок на логічні компоненти та ефективно керувати даними, їх відображенням та логікою користувача.

Таким чином, **MVC (*Model-View-Controller*)** — це архітектурний шаблон для розробки програмного забезпечення, який допомагає розділити програму на три основні частини. Розглянемо нижче, з чого складається схема MVC.

Модель (Model): Модель відповідає за обробку даних та бізнес-логіку програми. Вона зберігає інформацію і визначає, як ця інформація обробляється і змінюється.

Вигляд (View): Вигляд відповідає за відображення даних користувачу. Він представляє інформацію з Моделі у зручному для користувача вигляді.

Контролер (Controller) Контролер взаємодіє із користувачем і відправляє команди до Моделі та Вигляду. Він контролює потік інформації між Моделлю і Виглядом та реагує на дії користувача [14] (рис. 1.1).

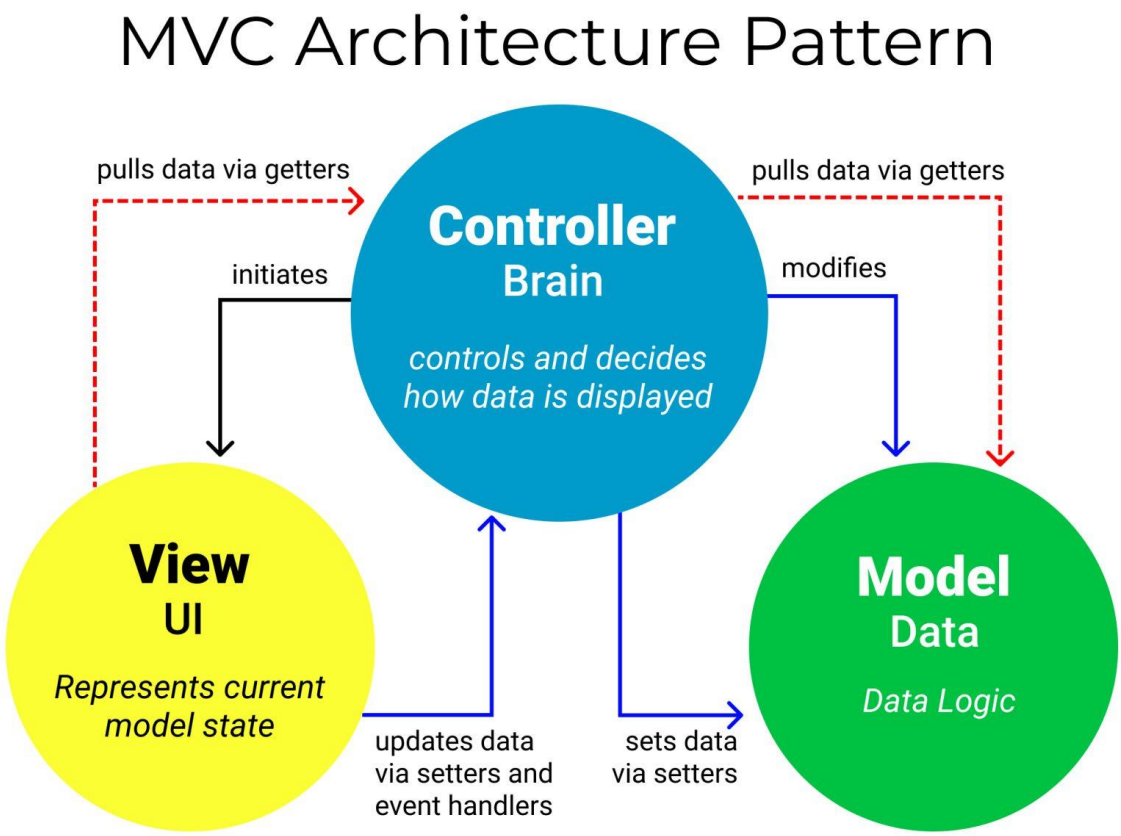

Рис. 1.1 Архітектура підходу MVC

Сьогодні підхід MVC використовується для сучасних веб-застосунків, оскільки він дозволяє застосунку бути масштабованим, підтримуваним і легко розширюваним.

В архітектурі MVC кожен із компонентів має свою чітко визначену область відповідальності та допомагає розділити код Front-end та Back-end на окремі компоненти:

- модель зосереджена на логіці та обробці даних. Вона зберігає інформацію і визначає, як ця інформація має взаємодіяти з програмою;
- вигляд відповідає за відображення даних користувачеві. Це включає в себе графічний інтерфейс, веб-сторінки або інші способи представлення інформації користувачу;
- контролер взаємодіє з користувачем та визначає, які дії виконувати на основі взаємодії користувача з Виглядом. Він також відповідає за маршрутизацію і обробку запитів [14].

Використання MVC сприяє створенню більш модульного коду, що робить його гнучкішим та легшим для розширення. Виокремлення логіки в Модель і відображення у Вигляд допомагає підтримувати код більш зрозумілим і читабельним. Розділення логіки і відображення дозволяє легко використовувати одну Модель із різними Виглядами або навпаки, що сприяє повторному використанню коду. Крім того, різні розробники можуть працювати над різними компонентами одночасно, не втручаючись у роботу один одного.

Для коректного використання MVC потрібно мати на увазі наступне:

- контролер має обробляти вхідні події та дії користувача і спрямовувати їх на відповідні методи Моделі для обробки даних та на Вигляд для оновлення інтерфейсу користувача;
- модель має забезпечувати методи для збереження, оновлення, видалення та отримання даних, а також виконання логіки;
- вигляд має відслідковувати зміни в Моделі та відображати їх в інтерфейсі користувача. Він також повинен відправляти повідомлення Контролеру про дії користувача [14].

Існує багато фреймворків, які спеціалізуються на реалізації архітектурного шаблону MVC та допомагають спростити розробку веб-застосунків. Ось деякі із найпопулярніших фреймворків з підтримкою MVC:

- Ruby on Rails — це фреймворк для розробки веб-застосунків мовою програмування Ruby. Rails має вбудовану підтримку MVC та допомагає розробникам створювати застосунки швидко і ефективно.
- Django — це фреймворк для розробки веб-застосунків мовою програмування Python. Django також має вбудований шаблон MVC (або, точніше, шаблон Model-View-Template) і надає розробникам потужні засоби для створення веб-застосунків.
- Laravel — це фреймворк для розробки веб-застосунків мовою програмування PHP. Він має вбудовану підтримку MVC та надає інструменти для реалізації бізнес-логіки та вигляду.
- Spring MVC — це фреймворк для розробки веб-застосунків мовою програмування Java. Spring MVC допомагає створювати веб-застосунків із використанням підходу Model-View-Controller та надає широкий набір інструментів для роботи з веб-технологіями.
- Express.js — це фреймворк для розробки веб-застосунків на JavaScript із використанням платформи Node.js. Він допомагає створювати серверну частину застосунків із використанням шаблону MVC та використовувати JavaScript на обох сторонах — клієнтській і серверній [14].

Вибір конкретного фреймворку залежить від мови програмування, потреб проєкта та особистих вподобань розробника.

Використання підходу розробки MVC — це не тільки ефективний спосіб розробляти програмні продукти, але й цінний ресурс для навчання інших людей у сфері програмування.

Цей підхід допомагає структурувати код, робить його більш зрозумілим та підходить для спільної роботи, а також сприяє модульності, тестуванню, підтримці та рефакторингу.

Отже, використання MVC покращує розробку програм і полегшує навчання програмуванню, допомагаючи новачкам та професіоналам розуміти та покращувати свої навички розробки.

2. ПРОГРАМНА РЕАЛІЗАЦІЯ ЗАВДАННЯ

2.1 Опис програми

Ми розробили програму для будь-якого користувача, який потребує планер на своєму гаджеті з метою упорядкування поточних справ і ведення заміток безкоштовно та без реєстрації. У програмі ми продемонстрували додавання, видалення елемента; виведення всіх елементів на екран; перевірка заповнення всіх полів форми створення ToDO-елемента; можливості отримати всіх збережених даних на одному пристрої.

Опис структури TODO-list

JS файли

TodoListV2.js завантажується на сторінку останнім, бо є ініціалізатором контролера. У собі має self-invoke function, яка спрацьовує лише один раз і в якій викликається функція ініціалізації контролера.

View.js є частиною підходу MVC, відповідає за представлення даних у цьому проєкті, створює все те, що бачить користувач на екрані. Тут є дві константи, які посилаються на DOM-елементи, а саме: на форму для створення заміток та на всі замітки, що містяться на сторінці.

- resetForm – цей метод очищує поля форми після того, як ці поля пройшли валідацію та дані з форми є валідованими.
- createTodoItemTemplate – створює макет нової замітки за допомогою створення нового DOM-вузла, який є нашою новою заміткою. Як аргумент приймає об’єкт з полями id, title, description, що відповідають id – унікальний ідентифікатор замітки, title – заголовок замітки, description – опис замітки. До обгортки замітки додається кастомний атрибут з властивістю, що дорівнює id.
- appendTodoItem – додає нашу замітку на сторінку.

- addTodoItemToList – додає нову замітку до існуючих. Як аргумент приймає об’єкт з полями id, title, description, що відповідають id – унікальний ідентифікатор замітки, title – заголовок замітки, description – опис замітки.
- removeTodoItem – відповідає за видалення замітки з інтерфейсу, пошук замітки, яку треба видалити, здійснюється за допомогою кастомного атрибуту з властивістю id.

Model.js є частиною підходу MVC, відповідає за маніпулювання даними (отримання, установка, зберігання) та їх оновлення. Тут є змінна todos та константа key. Опишемо їх відповідно. Змінна todos зберігає масив об’єктів заміток (назва, опис, id замітки). Константа key зберігає ключ для доступу до наших даних у localStorage.

- get todos – відповідає за отримання даних з localStorage за ключем, що зберігається в key та є гетером цієї моделі.
- set todos – відповідає за додавання даних до localStorage за ключем, що зберігається в key та є сетером цієї моделі.
- saveTodoItem – відповідає за додавання нової замітки та присвоює їй id. Як аргумент приймає об’єкт нової замітки.
- removeTodoItem – видаляє замітку з певним id із масиву заміток, а також приймає id замітки, яку треба видалити.

Controller.js є частиною підходу MVC (головний модуль нашого додатку) відповідає за керування взаємодією користувача з додатком. Тут є дві константи, які посилаються на DOM-елементи, а саме: на форму для створення заміток та на всі замітки, що містяться на сторінці.

- Init – викликається у файлі TodoListV2.js і являє собою функцію ініціалізації нашого проєкту, у якій створюється три слухача події, а саме: на завантаження DOM, на контейнер із замітками, на відправлення форми, при цьому процесі біндиться контекст виконання.

- formHandler – відповідає за обробку події відправлення форми. Як аргумент отримує подію. Спочатку відміняється дефолтна подія форми, далі збираються з неї дані, перевіряються на наявність будь-якого вмісту. Після чого валідовані дані потрапляють до Моделі, де і опрацьовуються. У той самий час до Представлення додається нова замітка, яку опрацьовує Представлення.
- loadedHandler – відповідає за обробку події завантаження DOM, а саме: дістає дані з Моделі і передає у Представлення для подальшого відображення.
- handleDeleteItem – відповідає за обробку події, натискання на контейнер із замітками. Як аргумент приймає подію, при цьому зупиняє спливання події; знаходить найближчий елемент в DOM-і, що містить кастомний атрибут data-todo-id та намагається видалити його за допомогою передачі його ідентифікатора до Моделі, щоб вилучити його із масиву заміток та з localStorage, а також до Представлення, щоб ця замітка вийшла з нашого інтерфейсу.

package.json – файл конфігурації проєкту, який містить інформацію про залежності.

Node_modules – каталог, де зберігаються всі залежності проєкту та додаткові бібліотеки.

.prettierrc.json, .eslintrc.json – файли конфігурації притера (форматування коду) та ESlint (аналіз коду, пошук помилок, автоматичне виправлення та написання коду в єдиному стилі).

CSS-файли

Style.css – файл стилів для нашого HTML, описані стилі для обгортки заміток, самих заміток, форми та розміру нашого додатку.

Bootstrap – підключається в index.html за допомогою cdn і використовується для контейнерів, стилізації форми, кнопок, тексту, певної адаптивності сайту.

HTML-файл

index.html – є основним файлом нашого проєкту. У ньому описується кодування сторінки, контейнери елементів (форма, список заміток); підключаються стилі (Style.css, Bootstrap), скрипти у певному порядку, а саме: Контролер, Модель, Представлення та ініціалізуючий файл.

3. ІНСТРУКЦІЯ КОРИСТУВАЧА

3.1 При запуску веб-додатку за посиланням одразу відкривається сам веб-додаток із формою (рис. 3.1)

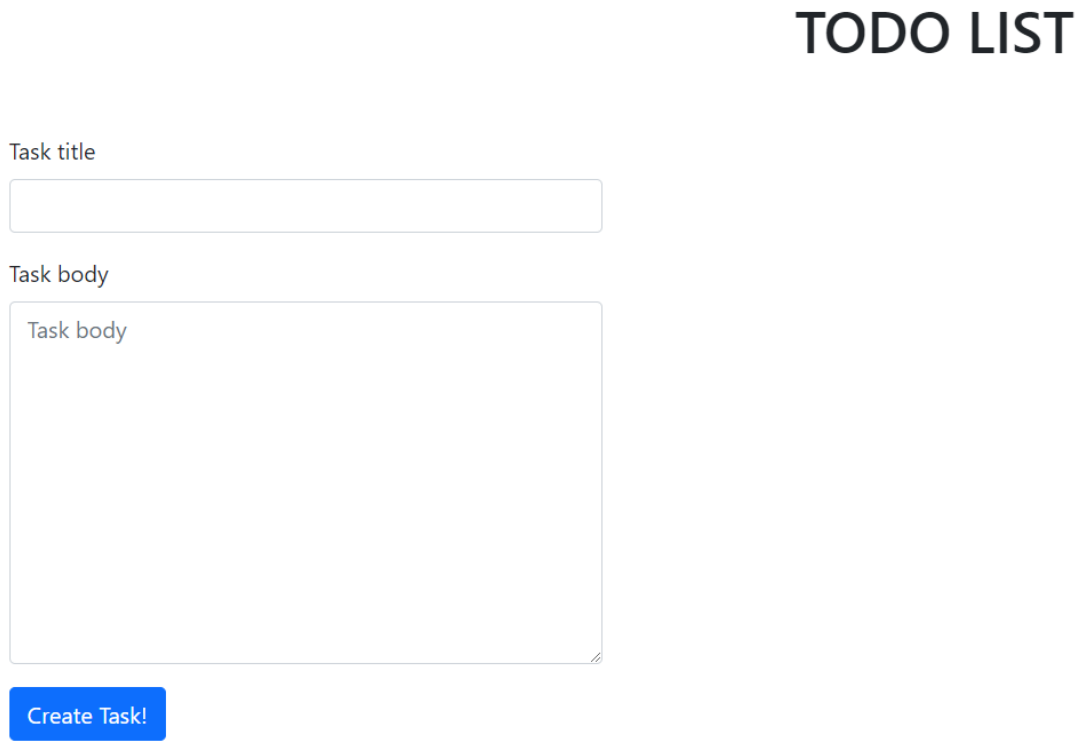

Рис. 3.1 Головне вікно

3.2 Якщо Ви вже користувались цим веб-додатком та вже створили декілька заміток на тому ж пристрої, то Ви їх й побачите (рис. 3.2).

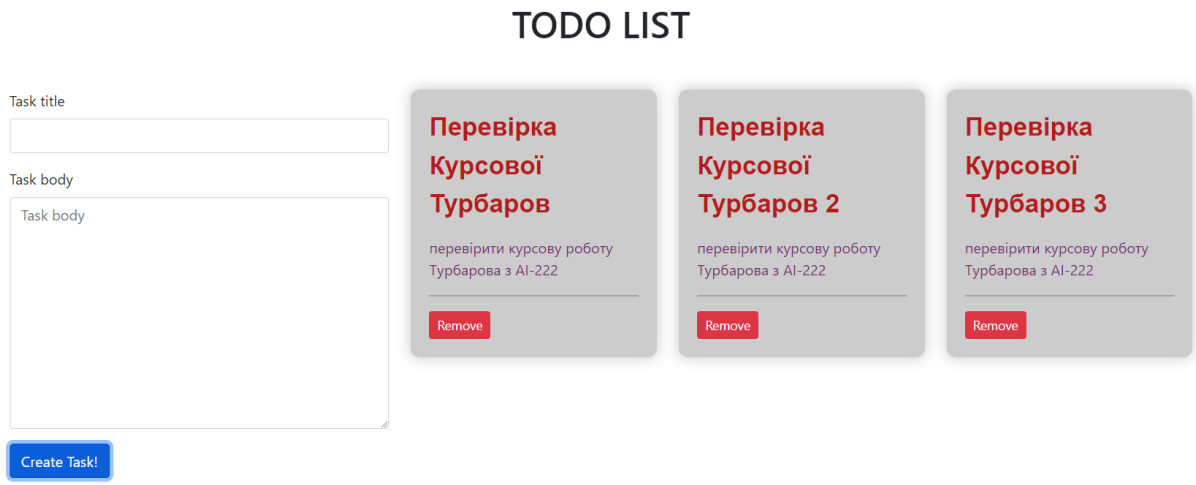

Рис. 3.2 Вікно програми із трьома замітками

3.3 Додавання нової замітки

Для додавання нової замітки Вам потрібно заповнити форму створення нової замітки (рис. 3.3), Task title – назва замітки, Task Body – опис замітки, після чого натиснути кнопку Create Task. Форма перевірить, чи всі поля заповнені. Якщо ні, то сайт повідоме про це: певне поле буде підсвічуватися червоним (рис. 3.4, 3.5). Якщо ж всі поля форми заповнені, користувач побачить, що всі поля зелені на 0,75 с, а нова замітка буде додана до інших (рис. 3.6).

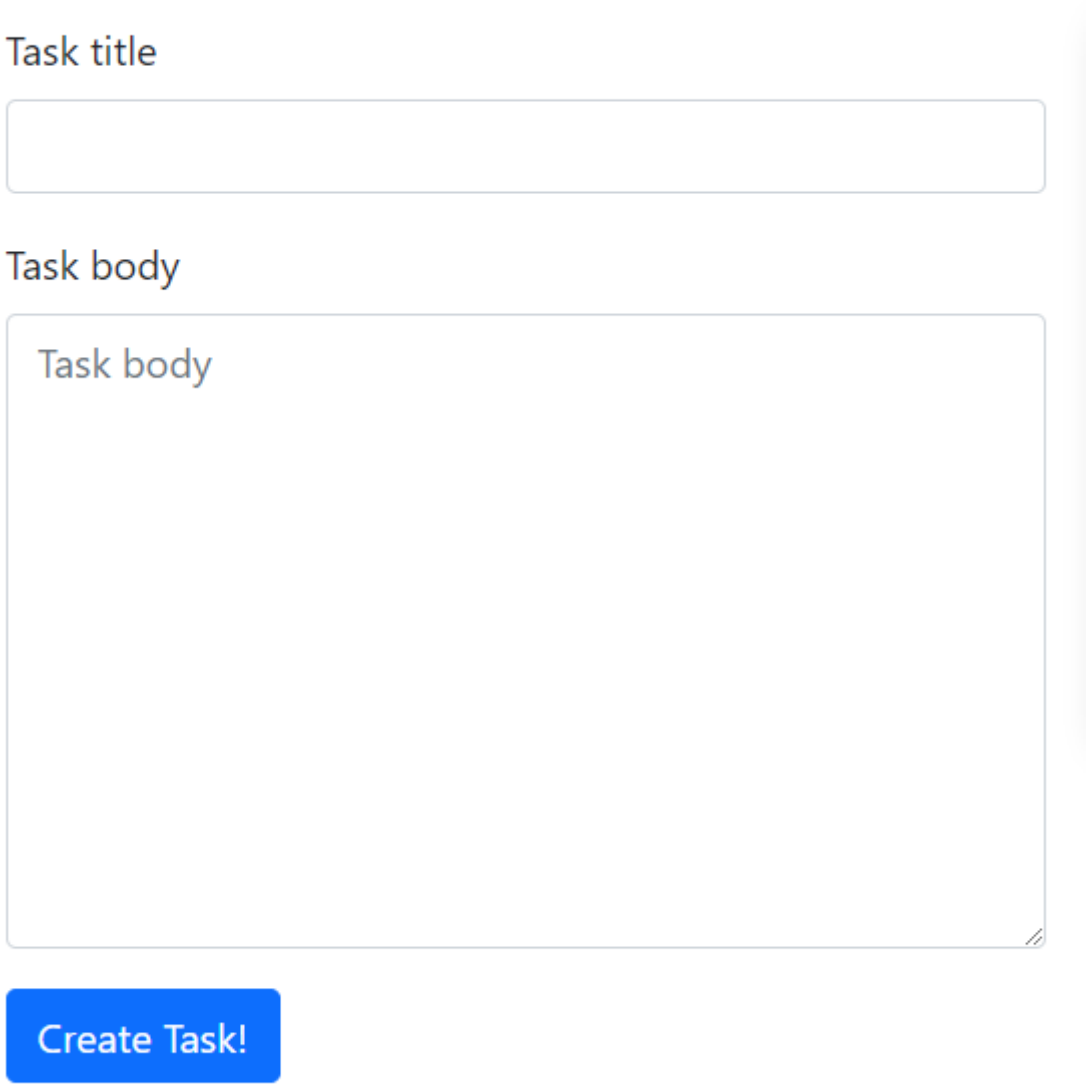

Рис. 3.3 Пуста форма

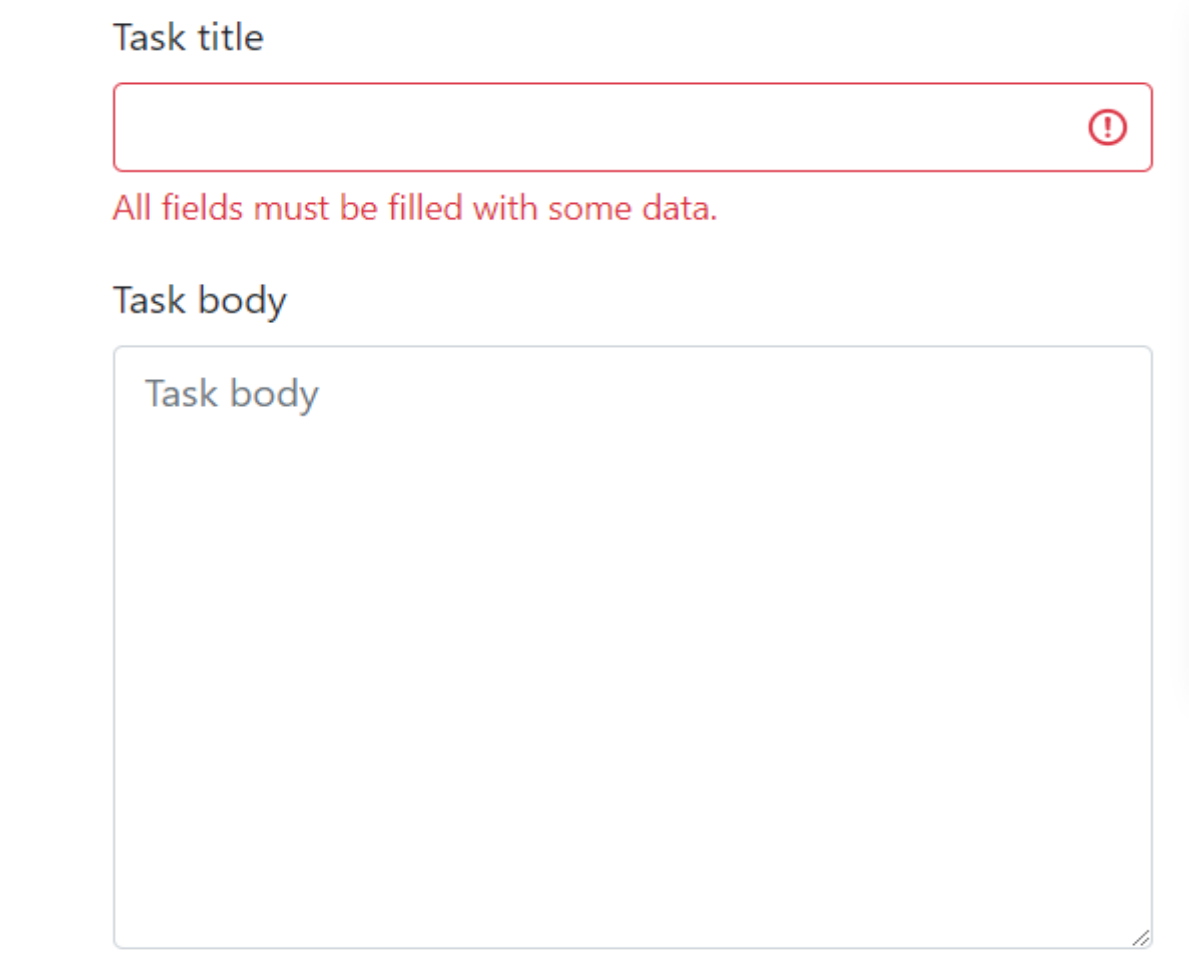

Рис. 3.4 Форма з незаповненим першим полем

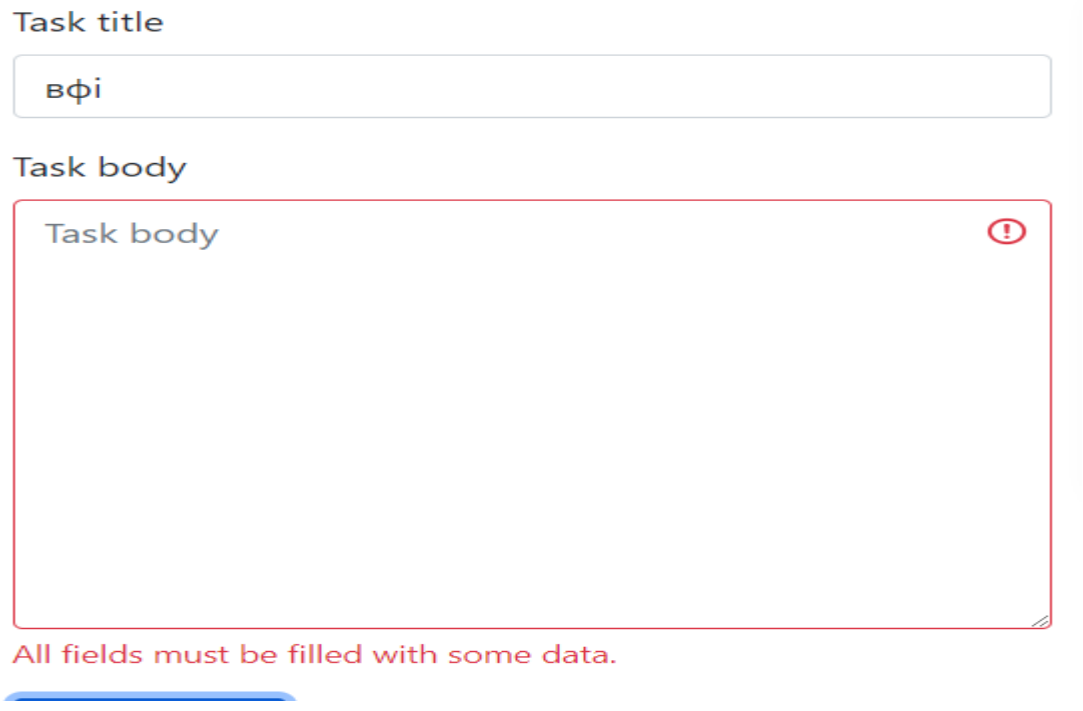

Рис. 3.5 Форма з незаповненим другим полем

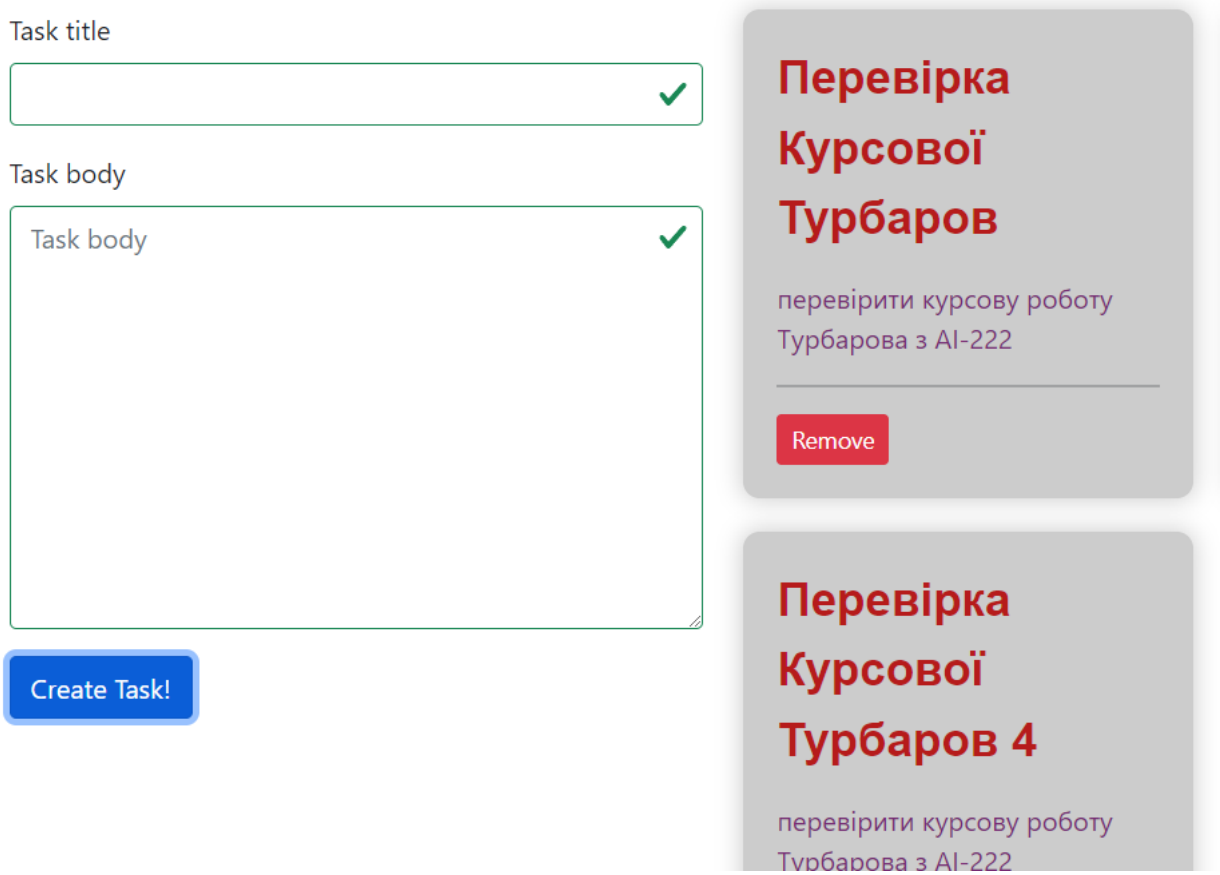

Рис. 3.6 Форма після створення нової замітки та сама замітка

3.4 Видалення замітки.

Для цієї дії вам потрібно лише на замітці, яку треба видалити, натиснути кнопку Remove й замітка буде видалена (рис.3.7)

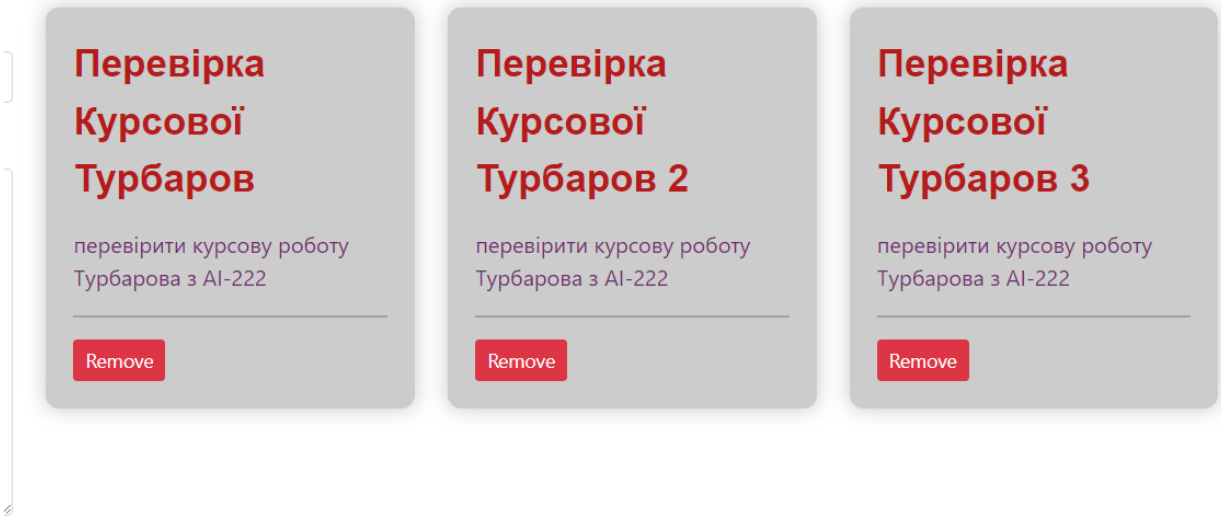

Рис. 3.7 Демонстрація видаленої замітки

3.5 Якщо користувач закриє сайт (рис. 3.8) і зайде, наприклад, через будь-який час на тому ж приладі, то він побачить, що всі його замітки залишилися (рис. 3.9).

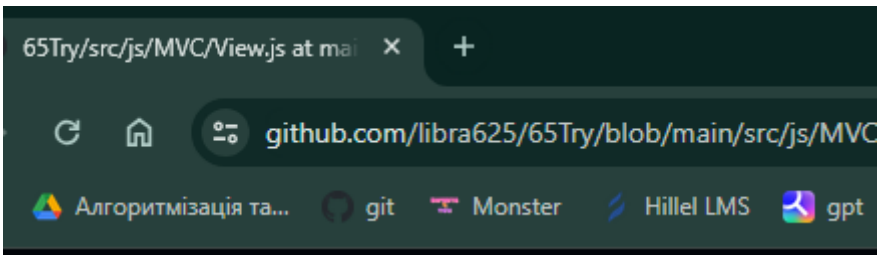

Рис. 3.8 Демонстрація того, що сайт був закритий

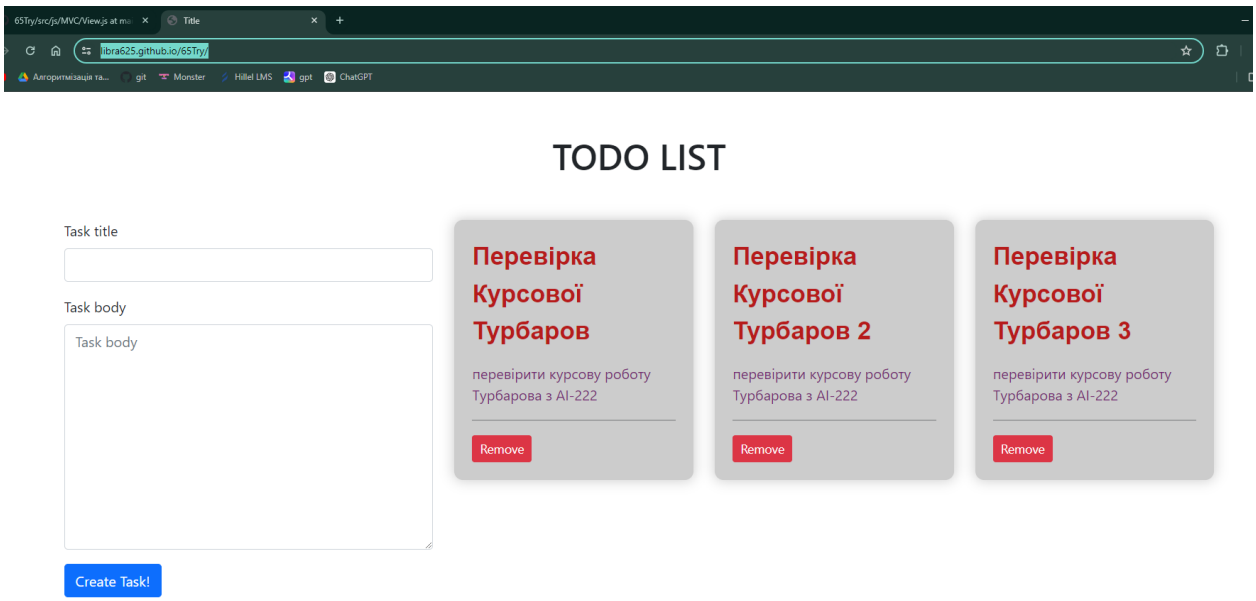

Рис. 3.9 Демонстрація того, що після того, як сайт був закритий і знову відкритий, створені замітки завантажились і відобразилися

3.6 Взаємодія в режимі «інкогніто»

При відкритті нових вікон (2 та більше) у режимі «інкогніто» вони будуть пов’язані одне із одним, але лише на час, поки хоча б одне вікно залишається відкритим. Подальші відкриті вікна або вкладки будуть також пов’язані із попередніми (попереднім). Ці вікна та вкладки не зв’язані з основним сайтом і замітками на ньому (рис. 3.10).

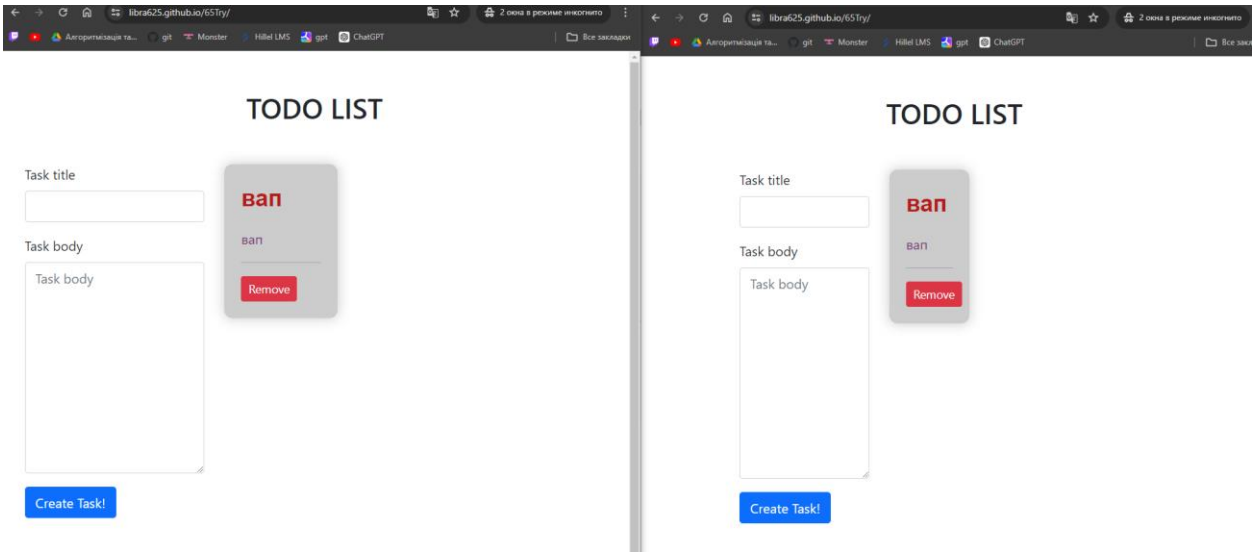

Рис. 3.10 Демонстрація роботи веб-додатку в режимі «інкогніто»

3.7 Взаємодія в звичайному (нормальному) режимі

Зв’язок між вікнами, вкладками та сховищем зберігається, але лише на одному певному пристрої. Для оновлення заміток, якщо вони були додані в іншій вкладці (рис. 3.11), треба оновити сторінку тієї вкладки, на яку треба отримати оновлення (рис. 3.12).

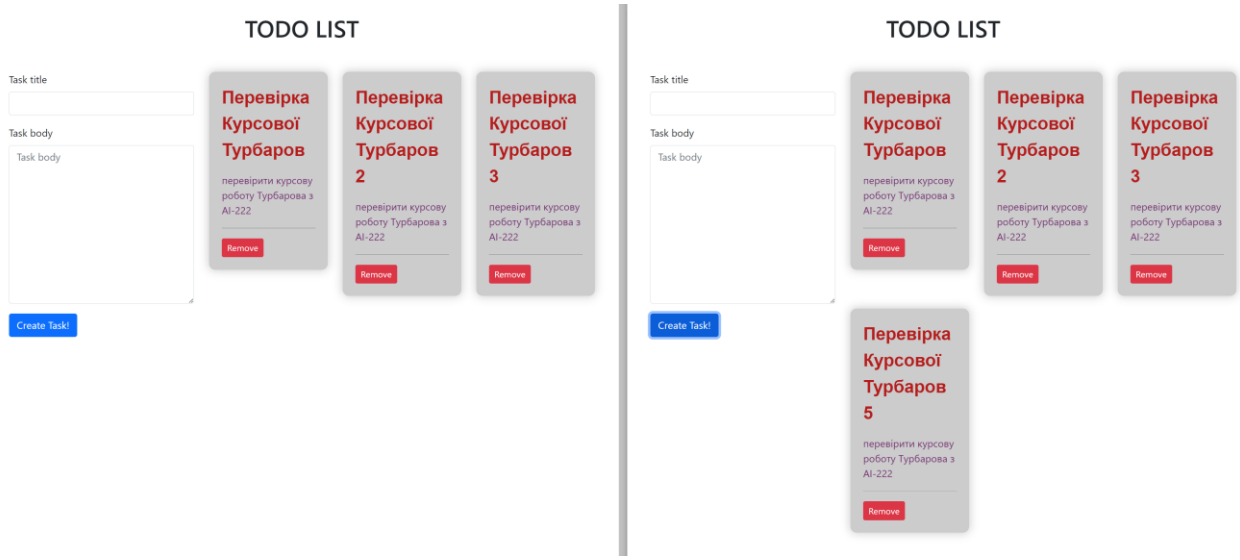

Рис. 3.11 Головне вікно веб-додатку у двох вкладках, у другій вкладці створено нову замітку

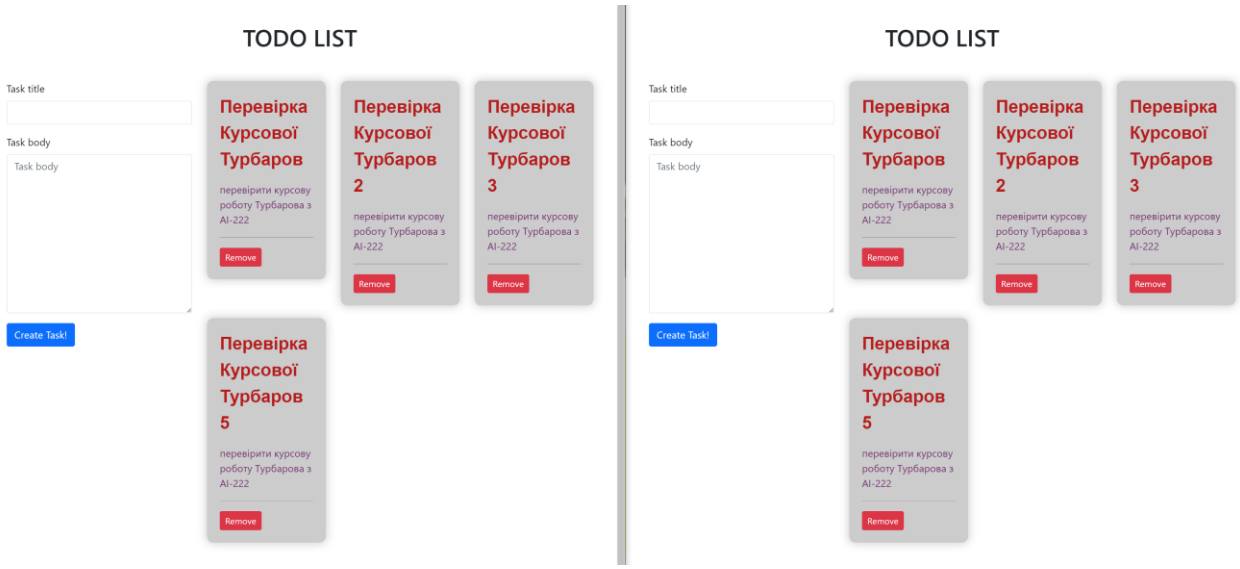

Рис. 3.12 Головне вікно веб-додатку у двох вкладках після оновлення сторінки на першій вкладці

3.8 Адаптивність

Якщо опис або назва замітки дуже великий, то вона просто розтягується за висотою (рис. 3.13).

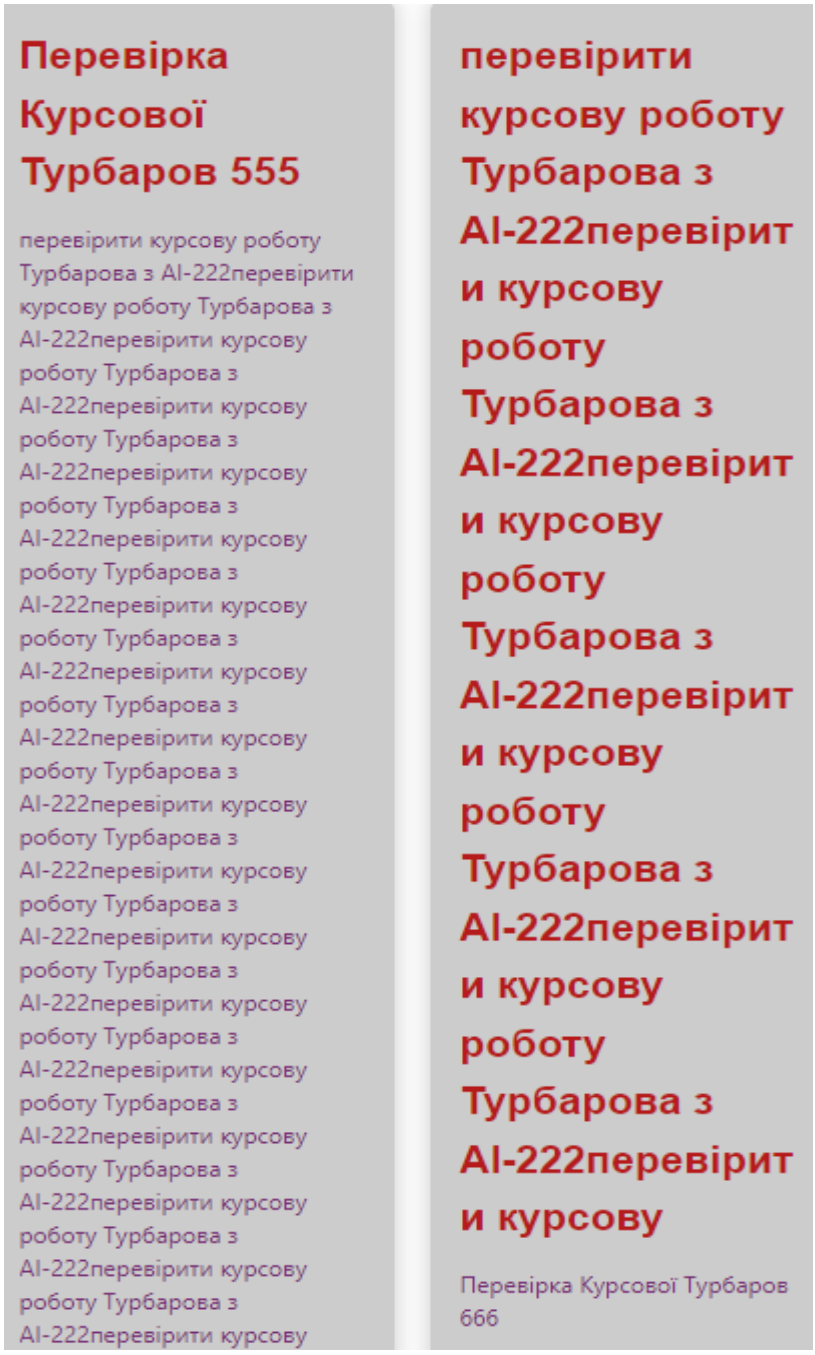

Рис. 3.13 Демонстрація адаптивності

ВИСНОВКИ

Був створений та реалізований сайт-додаток ToDo-лист за допомогою нативного JavaScript. Програма виконує: додавання замітки; видалення замітки; виведення всіх заміток; перевірку заповнення всіх полів форми створення ToDO-веб-застосунка; можливостей отримання всіх збережених заміток на одному пристрої. Курсову роботу виконано в повному обсязі згідно з вимогами до робіт такого типу.

Під час розробки програми і підготовки пояснювальної записки розширив свої знання про історію HTML, JavaScript, CSS та підходу MVC, а також удосконалив навички створення веб-застосунку з використанням підходу MVC (аналіз вимог замовника, проєктування архітектури програми, реалізація коду, використання програмного продукту).

Оцінюючи у відсотках мій вклад у дану курсову роботу, хочу зазначити, що протягом усієї роботи над нею мною було опрацьовано чимало сайтів, де я брав деякі ідеї, які після перевірки або ставали відправними точками для прийняття остаточних рішень щодо розробки описаного тут програмного продукту, або відкидалися як неробочі. Також у теоретичній частині курсової роботи мною було використано матеріали сайтів, що містять матеріал про зазначені вище мови програмування. Також мною використано матеріал лекцій з ООП та факультативного курсу Android. Посилання на них є в тексті й вміщені у списку використаної літератури. Загалом особистий вклад у роботу складає 78 %.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Алгоритми, дані і структури. [Текст], навч. посіб. / В.М. Ільман, О.П. Іванов, Л.О. Панік. Дніпропет. нац.. ун-т залізн. трансп.ім. акад. В. Лазаряна. Дніпро, 2019. 134 с.
2. Коротеева Т.О. Алгоритми та структури даних : навч. посібник. Львів : Видавництво Львівської політехніки, 2014. 280 с.
3. Ковалюк Т.В. Основи програмування. Київ : Видавнича група ВНУ, 2005. 384 с.
4. Динамічні структури даних. URL : https://ppt-online.org/889534 (дата звернення 25.05.2024).
5. Сучасний підручник з JavaScript : web-сайт. URL : https://uk.javascript.info/ (дата звернення 15.05.2024).
6. Історія сильної але абсурдної мови програмування – JAVASCRIPT. *Цікавий світ. Технології.* URL : https://senfil.net/index.php?newsid=345 (дата звернення 12.05.2024).
7. Чому JAVASCRIPT – перспективна мова програмування? Поради початківцям. URL : https://dou.ua/forums/topic/35184/ (дата звернення 26.05.2024).
8. Чому варто вивчати JAVASCRIPT? URL : https://qagroup.com.ua/publications/why-learn-javascript/ (дата звернення 26.05.2024).
9. HTML (HYPERTEXT MARKUP LANGUAGE) мова розмітки гіпертекстових документів. URL : https://www.webmaster.in.ua/p/html.html (дата звернення 26.05.2024).
10. 28 років тому запустили перший у світі веб-сайт. URL : http://surl.li/uekhv (дата звернення 23.05.2024).
11. Елементи HTML. URL : https://salo.li/f6aB3C4 (дата звернення 24.05.2024).

12. Введення в HTML. Основи. URL : https://w3schoolsua.github.io/hyperskillua/34/index.html#gsc.tab=0 (дата звернення 24.05.2024).
13. Підручник з CSS. URL : https://w3schoolsua.github.io/css/index_en.html#gsc.tab=0 (дата звернення 24.05.2024).
14. Що таке MVC-патерн для програмування динамічних веб-застосунків. URL : http://surl.li/uenxe (дата звернення 20.05.2024).

ДОДАТОК А

КОД ПРОГРАМИ

Увесь код програми можна подивитись за посиланням:

https://libra625.github.io/65Try/