Міністерство освіти і науки України
Криворізький національний університет
Факультет інформаційних технологій
Кафедра автоматизації, комп'ютерних наук і технологій

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти – магістр
за освітньо-професійною програмою
«Комп'ютерні науки»

зі спеціальності
122 – Комп'ютерні науки

тема роботи:

«Розробка веб-сервісу для оформлення бібліографічних посилань у відповідності до ДСТУ 8302:2015»

Виконала студентка гр. КН-23м _____ Ситник А.Р.

Керівник _____ Харламенко В. Ю.

Нормоконтроль _____ Маринич І. А.

Завідувач кафедри _____ Рубан С. А.

Кривий Ріг – 2024

КРИВОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет: *інформаційних технологій*

Кафедра: *автоматизації, комп'ютерних наук і технологій*

Ступінь вищої освіти: *Магістр*

Спеціальність: *122 – Комп'ютерні науки*

ЗАТВЕРДЖУЮ

Зав. кафедри: *к.т.н. Рубан С.А.*

_____

« *5* » *липня* 2024 р.

ЗАВДАННЯ

на кваліфікаційну роботу магістра

студентці групи КН-23м *Ситник Анастасії Русланівні*

1. Тема кваліфікаційної роботи: *«Розробка веб-сервісу для оформлення бібліографічних посилань у відповідності до ДСТУ 8302:2015»*

затверджено наказом по університету № *594с* від *04.07.2024 р.*

2. Термін здачі кваліфікаційної роботи: *01.12.2024 р.*

3. Склад кваліфікаційної роботи: *Пояснювальна записка обсягом 94 с., додатки, презентація у Microsoft PowerPoint (12 слайдів) в електронному та друкованому вигляді*

4. Консультанти кваліфікаційної роботи:

Розділ 1-3 ст. викл. Харламенко В. Ю.

_____

Нормоконтроль доц. Маринич І. А.

5. Календарний план:

№	Етапи роботи	Термін виконання
1	*Вступ*	*10.07.24*
2	*Розділ 1*	*15.07.24*
3	*Розділ 2*	*15.08.24*
4	*Розділ 3*	*15.09.24*
5	*Висновки*	*15.10.24*
6	*Оформлення кваліфікаційної роботи*	*20.11.24*
7	*Підготовка презентації та графічного матеріалу*	*28.11.24*
8	*Підготовка доповіді до захисту*	*01.12.24*

6. Дата видачі завдання: *28.06.2024 р.*

Керівник _____ /Харламенко В. Ю./

7. Запевнення: *Я, Ситник Анастасія Русланівна, запевняю, що ця кваліфікаційна робота виконана самостійно, не містить академічного плагіату, фабрикації, фальсифікації. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.*

Із чинним Положенням про академічну доброчесність Криворізького національного університету ознайомлена.

Чітко усвідомлюю, що в разі виявлення у кваліфікаційній роботі умисних порушень робота не допускається до захисту або оцінюється незадовільно.

Студентка _____ / Ситник А. Р./

АНОТАЦІЯ

Ситник А.Р. Розробка веб-сервісу для оформлення бібліографічних посилань у відповідності до ДСТУ 8302:2015.

Кваліфікаційна робота на здобуття ступеню вищої освіти магістр за освітньо-професійною програмою «Комп'ютерні науки» зі спеціальності 122 – Комп'ютерні науки. – Криворізький національний університет, Кривий Ріг, 2024.

Робота складається зі вступу, трьох розділів, висновків, переліку використаної літератури з 30 позицій. Загальний обсяг роботи становить 96 сторінок, з яких основний зміст роботи викладено на 85 сторінках, включає 4 таблиці і 49 рисунки.

Було розглянуто основні аспекти розробки веб-сервісу для оформлення бібліографічних посилань у відповідності до ДСТУ 8302:2015. Проаналізовано існуючі сервіси, їх переваги та недоліки, що дозволило сформулювати функціональні та нефункціональні вимоги до системи. Було обґрунтовано вибір сучасних технологій для створення веб-додатку, таких як React, Redux Toolkit, та розроблено адаптивний інтерфейс із використанням Bootstrap. Реалізовано ключові компоненти системи, включаючи обробку форм, управління станом додатку та створення бібліографічних посилань із точним дотриманням стандарту. Проведено практичну апробацію, яка підтвердила відповідність сервісу поставленим вимогам і його ефективність для автоматизації роботи з бібліографічними джерелами.

Результати роботи можуть бути використані для подальшого розвитку платформи, включаючи інтеграцію з іншими бібліографічними системами, розширення функціоналу та підтримку нових форматів даних.

Ключові слова:

БІБЛІОГРАФІЯ, БІБЛІОГРАФІЧНЕ ПОСИЛАННЯ, ІНФОРМАЦІЙНА СИСТЕМА, ВЕБ-СЕРВІС, REACT, REDUX, МЕНЕДЖЕР СТАНУ

ANNOTATION

Sytnyk A.R. Development of a web service for the registration of bibliographic references in accordance with DSTU 8302:2015.

Graduation master`s work for obtaining an educational degree «Master» for the educational and professional program «Computer science» in specialty 122 – «Computer science». – Kryvyi Rih National University, Kryvyi Rih, 2024.

The work consists of an introduction, three sections, conclusions, a list of used literature from 30 items. The total volume of the work is 96 pages, of which the main content of the work is set out on 85 pages, includes 4 tables and 49 figures.

The main aspects of the development of a web service for the registration of bibliographic references in accordance with DSTU 8302:2015 were considered. Existing services, their advantages and disadvantages were analyzed, which allowed us to formulate functional and non-functional requirements for the system. The choice of modern technologies for creating a web application, such as React, Redux Toolkit, was justified, and an adaptive interface using Bootstrap was developed. Key components of the system were implemented, including form processing, application state management, and creation of bibliographic references with strict adherence to the standard. Practical testing was conducted, which confirmed the service's compliance with the requirements and its effectiveness for automating work with bibliographic sources.

The results of the work can be used for further development of the platform, including integration with other bibliographic systems, expanding functionality, and supporting new data formats.

Keywords:

BIBLIOGRAPHY, BIBLIOGRAPHIC REFERENCE, INFORMATION SYSTEM, WEB SERVICE, REACT, REDUX, STATE MANAGER

ЗМІСТ

ВСТУП

У сучасних умовах розвитку науки та освіти оформлення бібліографічних посилань є важливим елементом будь-якої наукової роботи, публікації або документа. Коректне цитування джерел гарантує достовірність наукових досліджень, забезпечує етичність використання інтелектуальної власності та допомагає уникати плагіату. Проте, з огляду на складність правил та їх кількість, оформлення бібліографічних посилань нерідко викликає труднощі у студентів, науковців та авторів. В Україні ця проблема регулюється національним стандартом ДСТУ 8302:2015 «Бібліографічне посилання. Загальні положення та правила складання», який встановлює детальні вимоги до оформлення джерел.

Актуальність теми розробки веб-сервісу для автоматичного оформлення бібліографічних посилань у відповідності до ДСТУ 8302:2015 полягає в тому, що ручне форматування таких посилань займає значний час і підвищує ризик помилок. Незважаючи на те, що існують різні системи для автоматичного створення бібліографій (наприклад, міжнародні стандарти APA, MLA, Chicago тощо), спеціалізовані рішення для українського стандарту відсутні або недостатньо розвинені. У зв'язку з цим, наукові роботи, дипломи, дисертації та інші документи часто мають неточності в оформленні бібліографічного апарату, що знижує їх якість та може вплинути на результат захисту чи публікації.

Також важливо зазначити, що правильне оформлення бібліографії — це не лише технічний аспект роботи, а й елемент академічної доброчесності. Чітке дотримання вимог щодо цитування джерел сприяє розвитку наукової етики, зменшує ймовірність виникнення плагіату та підтверджує оригінальність дослідження. Автоматизація цього процесу, яку можна забезпечити за допомогою веб-сервісу, полегшить дотримання етичних стандартів як студентами, так і науковцями.

Особливу актуальність ця тема набуває в умовах цифровізації всіх сфер життя, зокрема науки й освіти. З розвитком технологій виникає необхідність

створення цифрових інструментів, які б сприяли спрощенню роботи з інформацією. Створення веб-сервісу для автоматизації оформлення бібліографічних посилань дозволить не лише зекономити час на виконання рутинної роботи, а й забезпечить точність дотримання вимог стандарту. Важливо, що такий сервіс буде доступний у будь-який час і на будь-якому пристрої, що значно полегшить роботу студентам і науковцям незалежно від їхнього місця перебування.

Крім того, використання веб-сервісу забезпечить широкі можливості для інтеграції з іншими системами управління бібліографічними даними, зокрема з науковими базами даних, бібліотеками, академічними платформами тощо. Це дозволить автоматично генерувати бібліографічні посилання, які відповідають національному стандарту, на основі метаданих джерел.

Окрему увагу слід звернути на можливості адаптації такого веб-сервісу для різних потреб користувачів. Враховуючи, що ДСТУ 8302:2015 застосовується не лише у навчальних закладах, але й у наукових установах, видавництвах, бібліотеках та наукових журналах, веб-сервіс може стати універсальним інструментом для забезпечення якості бібліографічного оформлення в різних сферах. Він зможе допомогти не лише студентам та викладачам, а й редакторам, які займаються підготовкою наукових видань.

У підсумку, розробка веб-сервісу для оформлення бібліографічних посилань згідно з ДСТУ 8302:2015 є актуальним завданням, яке відповідає сучасним потребам науки та освіти. Такий інструмент полегшить процес написання наукових робіт, підвищить їхню якість і забезпечить дотримання вимог академічної етики та стандартів. Автоматизація цього процесу дозволить уникнути помилок і заощадити час, що зробить наукову діяльність ефективнішою та більш професійною.

РОЗДІЛ 1
ДОСЛІДЖЕННЯ МОЖЛИВИХ ПІДХОДІВ ДО ВИРІШЕННЯ ЗАВДАННЯ РОЗРОБКИ ІНФОРМАЦІЙНОЇ СИСТЕМИ ДЛЯ ОФОРМЛЕННЯ БІБЛІОГРАФІЧНИХ ПОСИЛАНЬ

1.1 Огляд існуючих сервісів для оформлення бібліографічних посилань у відповідності до ДСТУ 8302:2015

Правильне оформлення бібліографічних посилань є важливим компонентом будь-якої наукової або академічної роботи. В Україні цей процес регулюється національним стандартом ДСТУ 8302:2015 «Бібліографічне посилання. Загальні положення та правила складання» [1]. Стандарт визначає вимоги до бібліографічних посилань, зокрема до структури посилань на різні види джерел: книги, статті, інтернет-ресурси, нормативні документи тощо. Оскільки правила ДСТУ 8302:2015 є досить деталізованими, науковцям, студентам та викладачам може бути складно вручну оформлювати посилання відповідно до стандарту. Для автоматизації цього процесу існують різні сервіси, які спрощують створення бібліографічних посилань і допомагають уникнути помилок. У цьому розділі буде розглянуто найпопулярніші сервіси, які підтримують або можуть бути адаптовані для роботи з ДСТУ 8302:2015.

Grafiati – це онлайн-сервіс для автоматичного генерування та управління бібліографічними посиланнями [2]. Його головною відмінністю є підтримка широкого спектра національних і міжнародних стандартів оформлення посилань, зокрема українського стандарту ДСТУ 8302:2015. Сервіс призначений для науковців, студентів, викладачів, журналістів і всіх, хто працює з академічними текстами, в яких необхідно правильно оформлювати цитування джерел (рис. 1.1).

Основні функції Grafiati:

– Підтримка різноманітних стандартів бібліографічного оформлення, включаючи APA, MLA, Chicago, Harvard та ДСТУ 8302:2015.

– Автоматичне формування бібліографічних посилань для різних типів джерел, включаючи книги, статті, веб-сайти, дисертації, доповіді, аудіовізуальні матеріали та інші.

– Можливість формування списку використаних джерел у відповідному форматі для текстового процесора.

– Збереження та управління бібліографічними списками для подальшого використання або редагування.

– Інтуїтивно зрозумілий інтерфейс, що дозволяє легко користуватися сервісом навіть без попереднього досвіду.

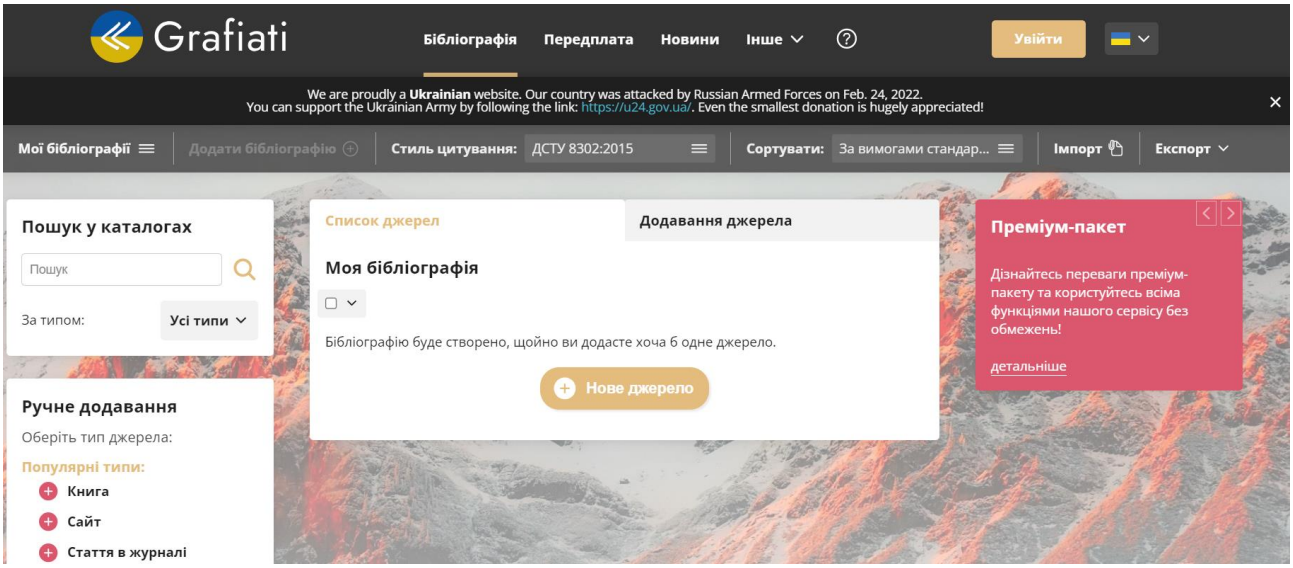

Рисунок 1.1 – Інтерфейс онлайн-сервісу Grafiati для автоматичного створення бібліографічних посилань і списків літератури за українськими та міжнародними стандартами

Переваги онлайн-сервісу Grafiati:

1. Підтримка ДСТУ 8302:2015 – на відміну від багатьох міжнародних аналогів, Grafiati пропонує повну підтримку українського стандарту оформлення бібліографічних посилань, що є ключовою перевагою для українських користувачів. Це робить його особливо корисним для студентів, викладачів і науковців, які зобов’язані використовувати цей стандарт у своїх роботах.

2. Легкість використання – інтуїтивно зрозумілий інтерфейс і зручна навігація дозволяють швидко створювати посилання навіть новачкам. Для створення посилання користувачеві достатньо ввести базові дані про джерело (назву, автора, дату публікації тощо), а система автоматично генерує правильне оформлення згідно з обраним стандартом.

3. Підтримка різних типів джерел – сервіс дозволяє формувати посилання для широкого спектра джерел, включаючи наукові статті, книги, веб-ресурси, законодавчі акти, дисертації, аудіовізуальні матеріали тощо. Це дає можливість працювати з різними типами інформаційних ресурсів, що актуально для наукових і дипломних робіт.

4. Актуальність стандартів – сервіс постійно оновлюється відповідно до змін у міжнародних і національних стандартах цитування. Це дозволяє користувачам бути впевненими, що всі посилання відповідають останнім вимогам і нормативам.

5. Інтеграція з текстовими процесорами – Grafiati підтримує можливість копіювання та вставлення бібліографічних списків у форматах, сумісних із популярними текстовими редакторами, такими як Microsoft Word, Google Docs та інші. Це спрощує процес додавання готових посилань до наукових документів.

Недоліки онлайн-сервісу Grafiati:

1. Обмежений безкоштовний функціонал – сервіс надає основні функції для створення бібліографічних записів безкоштовно, проте для доступу до розширених можливостей потрібна платна підписка. Це може стати недоліком для студентів або користувачів, які мають обмежений бюджет.

2. Відсутність підтримки роботи офлайн – Grafiati є виключно веб-сервісом, що вимагає постійного підключення до Інтернету. Для користувачів, які працюють у середовищах без стабільного Інтернет-з’єднання, це може бути незручністю.

Cite This For Me – це популярний веб-сервіс для автоматичного створення бібліографічних посилань у форматах APA, MLA, Chicago, Harvard тощо [3].

Хоча він не підтримує український стандарт ДСТУ 8302:2015, користувачі можуть використовувати його для базової підготовки посилань, а потім вручну редагувати їх відповідно до вимог ДСТУ (рис. 1.2).

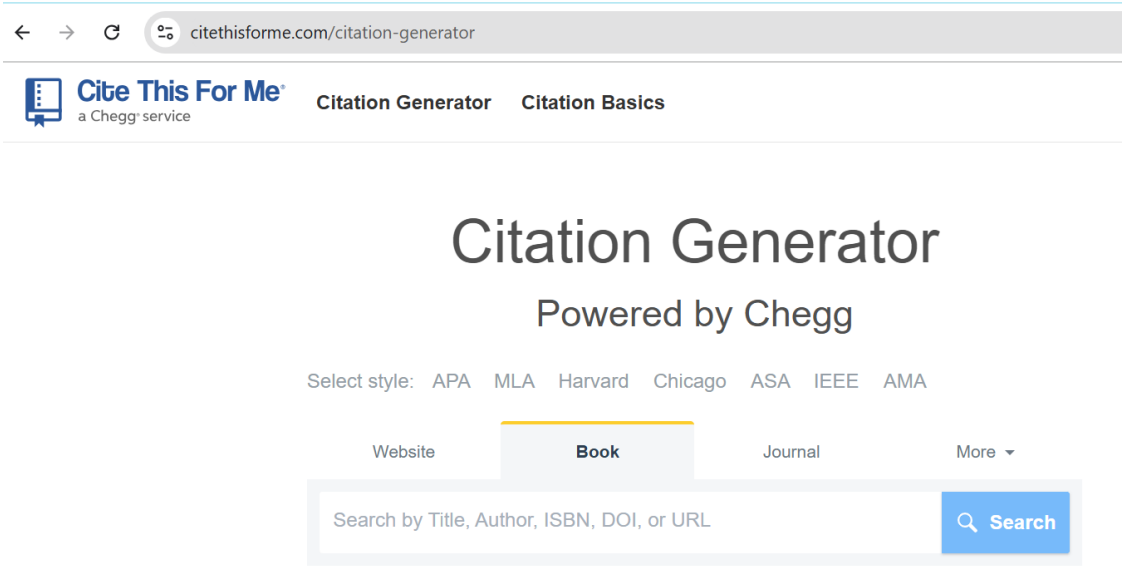

Рисунок 1.2 – Інтерфейс онлайн-сервісу Cite This For Me

Основні можливості:

– генерація посилань для різних типів джерел, включаючи книги, веб-сайти, статті, відео тощо;

– зручний інтерфейс з можливістю автоматичного формування списків літератури;

– інтеграція з текстовими редакторами для швидкого копіювання посилань у роботу.

Переваги:

– простота у використанні – сервіс підходить для швидкого створення посилань у різних міжнародних форматах;

– безкоштовний доступ – основні функції доступні безкоштовно.

Недоліки:

– відсутність підтримки ДСТУ 8302:2015 – для відповідності українському стандарту необхідно редагувати посилання вручну після їх створення.

– обмежена кількість безкоштовних функцій – для збереження більше трьох бібліографічних списків потрібно підписатися на платну версію.

Zotero – це відкрита платформа для управління бібліографією, яка дозволяє збирати, організовувати та цитувати джерела [4]. Хоча Zotero не підтримує ДСТУ 8302:2015 за замовчуванням, користувачі можуть додати цей стиль через додаткові плагіни або редагування стилю (рис. 1.3).

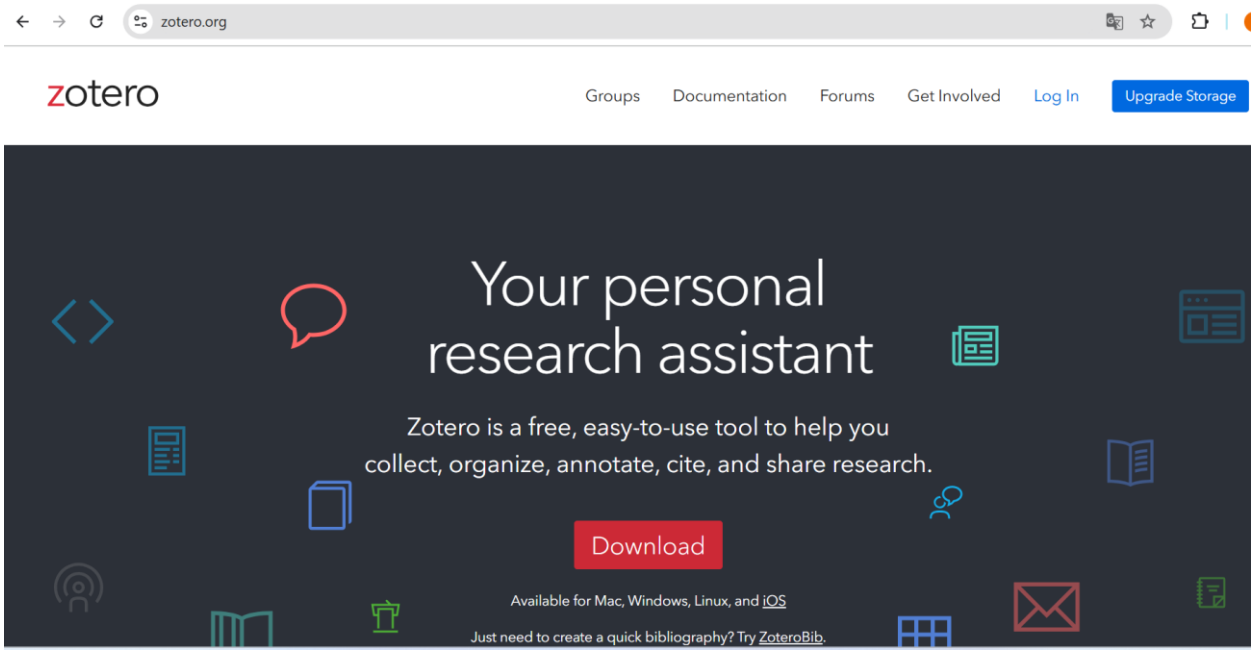

Рисунок 1.3 – Інтерфейс платформи Zotero

Основні можливості:

– можливість створення бібліографічних записів для різних типів джерел і організація їх у бібліотеки;

– інтеграція з браузерами для автоматичного збирання даних з веб-сторінок;

– підтримка численних стилів оформлення посилань, включаючи міжнародні стандарти.

Переваги:

– безкоштовність – всі основні функції сервісу доступні безкоштовно;

– можливість адаптації під ДСТУ 8302:2015 – завдяки відкритій архітектурі користувачі можуть додати підтримку українського стандарту;

– інтеграція з текстовими процесорами – Zotero дозволяє швидко

вставляти бібліографічні посилання у документи Word і Google Docs.

Недоліки:

– відсутність прямої підтримки ДСТУ 8302:2015 – для створення посилань у цьому форматі потрібні додаткові налаштування або редагування стилю;

– складність налаштувань – для користувачів, які не знайомі з технічними деталями налаштування бібліографічних інструментів, процес адаптації може бути складним.

Mendeley – це платформа для управління науковими дослідженнями та бібліографією, яка підтримує численні міжнародні стилі цитування [5]. Як і Zotero, Mendeley не має вбудованої підтримки ДСТУ 8302:2015, але його можна налаштувати для роботи з українським стандартом за допомогою плагінів або редагування стилів (рис. 1.4).

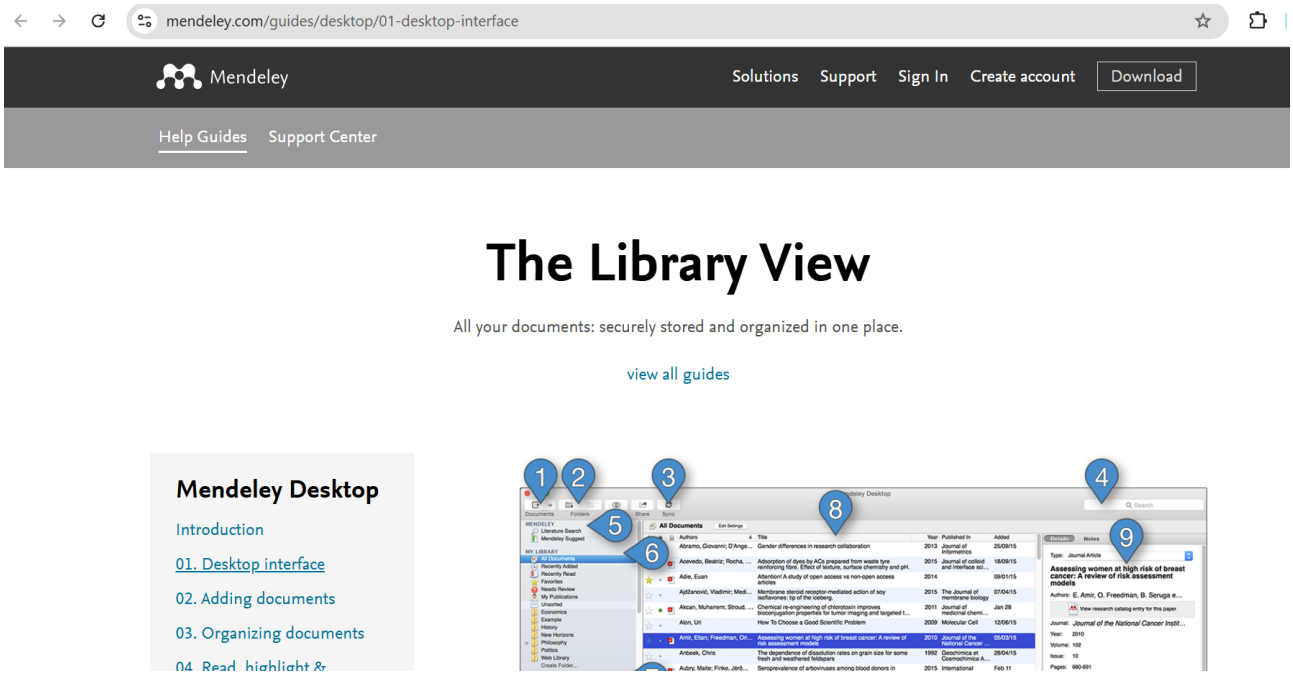

Рисунок 1.4 – Інтерфейс платформа Mendeley для управління науковими дослідженнями та бібліографією

Основні можливості:

– можливість створення та організації бібліографічних записів;

– інтеграція з науковими базами даних і можливість зберігати статті та

документи;

– підтримка численних стилів оформлення цитувань.

Переваги:

– широка функціональність – Mendeley пропонує не лише можливості для оформлення посилань, але й інструменти для управління науковими дослідженнями;

– інтеграція з текстовими процесорами – як і Zotero, Mendeley інтегрується з Microsoft Word і Google Docs для автоматизації цитування.

Недоліки:

– відсутність підтримки ДСТУ 8302:2015 – як і в випадку із Zotero, для роботи з українським стандартом потрібні додаткові налаштування;

– складність для новачків – для повного освоєння можливостей Mendeley потрібен час і технічні навички.

BibGuru – це онлайн-інструмент для автоматичного створення бібліографічних посилань [6]. Він орієнтований на студентів і дослідників, які потребують швидких і простих рішень для створення списків літератури. Як і багато інших сервісів, BibGuru не підтримує ДСТУ 8302:2015, але може використовуватися для швидкого генерування базових посилань, які потім можна відредагувати вручну (рис. 1.5).

Основні можливості:

– автоматичне створення посилань на основі метаданих джерел;

– підтримка популярних форматів цитування, таких як APA, MLA, Chicago;

– інтеграція з текстовими редакторами для вставки бібліографічних списків.

Переваги:

– швидкість і простота – сервіс дозволяє швидко генерувати базові посилання;

– безкоштовний доступ – BibGuru є безкоштовним для основних завдань.

Недоліки:

– відсутність підтримки ДСТУ 8302:2015 – для роботи з українським стандартом потрібно вручну редагувати посилання після їх створення;

– обмежений функціонал – сервіс не пропонує можливостей для глибокого управління бібліографією.

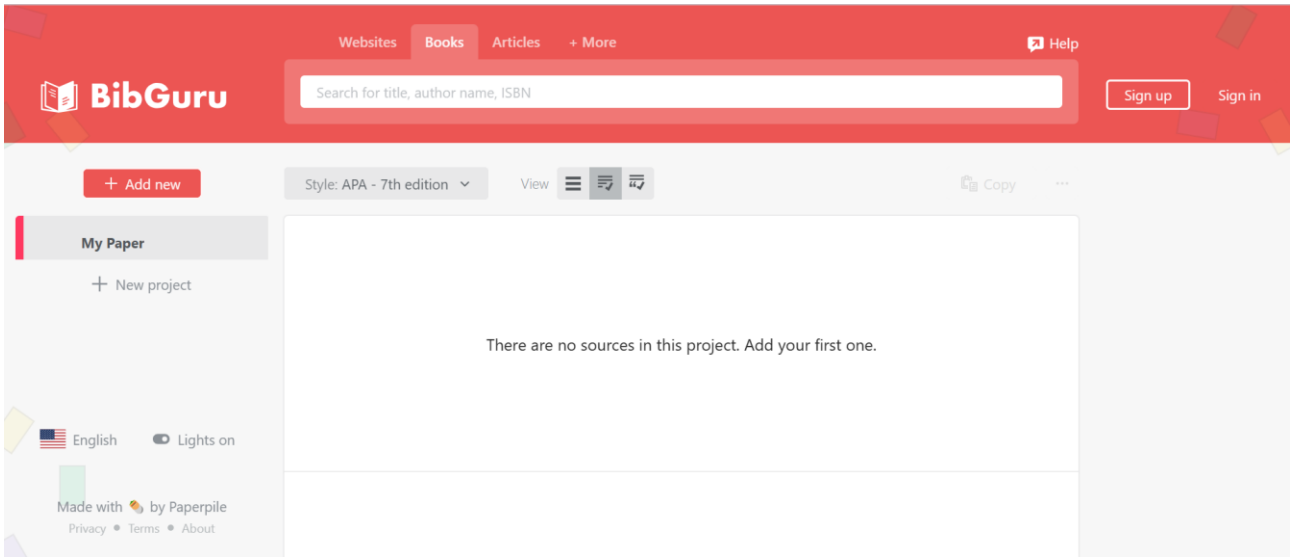

Рисунок 1.5 – Інтерфейс онлайн-інструменту BibGuru

1.2 Аналіз технологій для реалізації веб-сервісу

Для реалізації веб-сервісу для оформлення бібліографічних посилань розглянемо два підходи: архітектуру MVC (Model-View-Controller) та архітектуру WebAPI з клієнтським застосунком на основі JavaScript-фреймворків. Кожен із цих підходів має свої переваги та недоліки, тому їхній аналіз дозволить визначити найбільш ефективне рішення для цього типу проєкту [7, 8].

Архітектура MVC

MVC (Model-View-Controller) - це архітектурний шаблон, який дозволяє розділити логіку веб-додатка на три окремі частини: модель (Model), представлення (View) та контролер (Controller).

Переваги MVC для реалізації веб-сервісу [7]:

– Чітка структура коду: завдяки поділу на моделі, контролери та представлення легше підтримувати та розширювати код. Кожен елемент

відповідає за свою частину функціональності, що полегшує роботу над проєктом, особливо в командній розробці.

– Швидкий старт для серверного рендерингу: MVC дозволяє реалізувати інтерфейс і серверну логіку на стороні сервера. Це може бути зручним для формування та подальшого рендерингу сторінок з бібліографічними посиланнями.

– Стабільність і перевіреність підходу: MVC широко застосовується, має багатий набір інструментів і бібліотек, що робить його надійним і зручним для реалізації.

Недоліки MVC:

– Обмежена інтерактивність: оскільки більшість інтерфейсів рендеряться на сервері, клієнтські інтерфейси MVC підходять для веб-застосунків з мінімальною інтерактивністю. Для веб-сервісу, де користувач може створювати або налаштовувати бібліографічні посилання в реальному часі, MVC може бути недостатньо гнучким.

– Складність масштабування клієнтської частини: додатки MVC можуть стати громіздкими при спробі додати інтерактивні можливості, що зменшує гнучкість користувацького інтерфейсу.

MVC буде ефективним вибором, якщо веб-сервіс передбачає прості операції та стандартний інтерфейс без великої кількості динамічних елементів. Для проєкту з оформлення бібліографічних посилань MVC підходить, якщо вимоги до клієнтського інтерфейсу не є високими.

Архітектура WebAPI з клієнтським застосунком на базі JavaScript-фреймворків

Підхід WebAPI + клієнтський застосунок передбачає створення RESTful API, який обробляє запити від клієнтського застосунка, реалізованого на основі JavaScript-фреймворків, таких як React, Angular або Vue.js [8].

Переваги WebAPI з клієнтським застосунком

– Висока інтерактивність і гнучкість: використання JavaScript-фреймворків на стороні клієнта дозволяє створити сучасний, інтерактивний

інтерфейс з можливістю динамічних змін без перезавантаження сторінки. Це зручно для сервісів, де користувач взаємодіє з формами, наприклад, формуючи посилання в реальному часі.

– Масштабованість і реюзабельність: WebAPI дозволяє використовувати API з різними клієнтами. Це може бути зручно для мобільних застосунків, а також для сторонніх сервісів, що можуть підключитися до API для обробки бібліографічних посилань.

– Розділення відповідальностей: серверна частина (API) займається обробкою даних, тоді як клієнтська частина відповідає за відображення та взаємодію з користувачем. Це полегшує підтримку та масштабування застосунка.

Недоліки WebAPI з клієнтським застосунком [7, 8]:

– Більш складна реалізація та налаштування: у порівнянні з MVC, цей підхід потребує більшої кількості налаштувань. Потрібно окремо налаштувати WebAPI та клієнтську частину, а також забезпечити їхню взаємодію.

– Необхідність підтримки API: створення WebAPI вимагає додаткової уваги до безпеки та тестування, оскільки API може бути доступним для зовнішніх запитів, що накладає вимоги до автентифікації та авторизації.

Цей підхід ідеально підходить для проєктів, які передбачають високу інтерактивність або можуть розвиватися у багатоканальні рішення. Якщо сервіс для оформлення бібліографічних посилань має бути інтерактивним, зручним для редагування, зі швидким рендерингом і можливістю подальшого розширення — архітектура WebAPI + JavaScript-фреймворк буде кращим варіантом. Порівняння підходів за критеріями наведено у табл. 1.1.

MVC підходить для створення простих веб-сервісів із мінімальною інтерактивністю, де зручність підтримки серверного коду важливіша за гнучкість клієнтського інтерфейсу. У свою чергу, архітектура WebAPI з клієнтським застосунком, побудованим на JavaScript-фреймворках, є оптимальним вибором для розробки інтерактивних сервісів із розширеними функціями, можливістю масштабування та інтеграції з іншими платформами.

Для веб-сервісу з оформлення бібліографічних посилань, орієнтованого на користувача, найкращим варіантом буде використання архітектури WebAPI разом із клієнтським застосунком на основі JavaScript-фреймворків. Такий підхід забезпечить високий рівень зручності для користувача та відкриє перспективи для подальшого розвитку проєкту.

Таблиця 1.1 – Порівняння підходів архітектури MVC та WebAPI з клієнтським застосунком на базі JavaScript-фреймворків

Критерій	**MVC**	**WebAPI + JS-фреймворки**
Інтерактивність	Низька	Висока
Розділення компонентів	Часткове	Повне
Масштабованість	Менша, залежить від сервера	Вища, легко інтегрується з іншими сервісами
Швидкість розробки	Швидка для простих застосунків	Потребує більше часу
Підтримка	Зручна	Складніша через розподіл клієнта та сервера
Інтеграція з іншими сервісами	Обмежена	Легка

1.3 Розробка функціональних та нефункціональних вимог до сервісу

Розробка веб-сервісу для оформлення бібліографічних посилань потребує чіткого визначення функціональних і нефункціональних вимог, які забезпечать правильне функціонування системи, зручність її використання, відповідність стандарту ДСТУ 8302:2015, а також її ефективність і надійність.

1. Функціональні вимоги

Функціональні вимоги визначають основний набір функцій, які має виконувати веб-сервіс. Вони охоплюють:

1.1. Реєстрація та аутентифікація користувачів

– Сервіс повинен забезпечувати можливість реєстрації нових користувачів із вказанням електронної пошти, пароля та імені.

– Має бути функціонал для входу в систему (аутентифікація) з перевіркою облікових даних.

– Передбачено можливість скидання пароля у випадку його втрати.

1.2. Створення бібліографічних посилань

– Користувач має мати змогу створювати нові бібліографічні посилання відповідно до ДСТУ 8302:2015.

– Має бути передбачена можливість вибору типу джерела (книга, стаття, електронний ресурс тощо).

– Форма створення повинна включати поля для введення необхідних даних (автор, рік публікації, назва тощо).

1.3. Редагування та видалення посилань

– Користувач може редагувати раніше створені посилання для внесення змін.

– Має бути можливість видалення посилань, які більше не потрібні.

1.4. Збереження та організація посилань

– Система повинна дозволяти зберігати створені посилання у профілі користувача.

– Має бути можливість організовувати посилання в категорії чи колекції для зручного доступу.

1.5. Перевірка коректності оформлення

– Сервіс має автоматично перевіряти коректність введених даних на відповідність вимогам ДСТУ 8302:2015.

1.6. Експорт посилань

– Користувач повинен мати змогу експортувати посилання у популярних форматах, таких як PDF, Word або BibTeX.

1.7. Пошук і фільтрація

– Має бути реалізований функціонал пошуку посилань за ключовими словами (автор, рік, назва).

– Система повинна дозволяти фільтрувати посилання за різними критеріями (тип джерела, рік публікації, категорія).

1.8. Управління профілем користувача

– Користувач може змінювати свої особисті дані (ім’я, електронну пошту, пароль).

– Має бути можливість перегляду історії створених посилань.

2. Нефункціональні вимоги

Для забезпечення ефективного функціонування веб-сервісу з оформлення бібліографічних посилань відповідно до ДСТУ 8302:2015 необхідно визначити нефункціональні вимоги до технологій розробки, системи управління базами даних (СУБД) та способу розгортання системи на хостингу. Ці аспекти забезпечують стабільність, масштабованість і зручність обслуговування системи.

Технологія розробки

Для розробки веб-сервісу обрано стек технологій, що забезпечує високу продуктивність, гнучкість і зручність у розробці. До складу технологій входять:

1. Серверна частина (бекенд):

– Мова програмування: JavaScript (Node.js) або Python (Django/Flask). Node.js дозволяє створювати високопродуктивні асинхронні сервери. Python із Django/Flask забезпечує зручність роботи з базами даних і масштабованість.

– Фреймворк: Express.js для Node.js або Django для Python. Express.js забезпечує мінімалістичний підхід для створення RESTful WebAPI. Django містить вбудовані інструменти для прискорення розробки.

2. Клієнтська частина (фронтенд):

– JavaScript-фреймворки: React.js або Vue.js. React.js забезпечує створення інтерактивних і динамічних інтерфейсів. Vue.js підходить для проєктів із середньою складністю, завдяки простоті у використанні.

3. Протокол взаємодії:

– REST API для обміну даними між сервером і клієнтом.

Цей підхід забезпечує універсальність і сумісність з іншими системами.

– JSON як формат передачі даних для забезпечення зручності та компактності.

Система управління базами даних (СУБД)

База даних є критичним компонентом веб-сервісу, оскільки забезпечує зберігання інформації про бібліографічні посилання, користувачів та їхні дії.

1. Вимоги до СУБД:

– Надійність: База даних повинна забезпечувати збереження даних навіть у разі збою системи.

– Продуктивність: СУБД має забезпечувати високу швидкість обробки запитів, особливо при виконанні пошуку та фільтрації.

– Масштабованість: Система повинна підтримувати збільшення обсягів даних і кількості користувачів.

2. Обраний інструмент:

– PostgreSQL: забезпечує високу продуктивність і підтримує складні запити. Містить потужні засоби для роботи з даними, такими як індекси, транзакції та перевірки цілісності.

– Альтернативи: MySQL: легкий у налаштуванні, але менш ефективний для роботи зі складними структурами. MongoDB: використовується для неструктурованих даних, але для даного проєкту менш підходить.

Розгортання на хостингу

Розгортання веб-сервісу є ключовим етапом для забезпечення доступу користувачів до системи.

1. Вимоги до хостингу:

– Надійність і доступність: хостинг повинен забезпечувати безперервну роботу сервісу з доступністю не менше 99.9%.

– Масштабованість: інфраструктура повинна дозволяти розширення ресурсів (CPU, RAM) без зупинки роботи сервісу.

– Безпека: хостинг має підтримувати шифрацію даних (SSL/TLS), регулярне резервне копіювання та захист від атак (DDoS, SQL-ін’єкції).

– Інтеграція CI/CD: платформа повинна підтримувати автоматизацію процесу розгортання через системи CI/CD (наприклад, GitHub Actions, GitLab CI).

2. Обрані платформи:

– Heroku: легкість у налаштуванні та інтеграції з системами CI/CD. Підходить для невеликих проєктів і швидкого запуску.

– AWS (Amazon Web Services): надійна інфраструктура, що підтримує масштабування. Підходить для великих проєктів і забезпечує високу продуктивність.

– DigitalOcean: простий інтерфейс і гнучкі налаштування. Підходить для середніх проєктів із помірним навантаженням.

3. Розгортання:

– Серверна частина (бекенд) розгортається як окремий сервіс (контейнер Docker або віртуальна машина).

– Клієнтська частина (фронтенд) може бути розміщена як статичний веб-застосунок у CDN, наприклад, через Cloudflare.

– Використання інструментів для моніторингу (Prometheus, Grafana) для відстеження стану системи.

Розробка веб-сервісу для оформлення бібліографічних посилань відповідно до ДСТУ 8302:2015 базується на сучасному стеку технологій, таких як Node.js або Python для серверної частини, React.js або Vue.js для клієнтського застосунку та PostgreSQL як СУБД. Розгортання сервісу передбачає використання надійних платформ, таких як Heroku або AWS, що забезпечують масштабованість, безпеку та зручність обслуговування. Такий підхід забезпечує надійну інфраструктуру та створює основу для подальшого розвитку проєкту.

Розроблені функціональні та нефункціональні вимоги визначають ключові аспекти роботи веб-сервісу для оформлення бібліографічних посилань відповідно до ДСТУ 8302:2015. Вони забезпечують чітке розуміння функціональності системи, її технічних характеристик, а також вимог до безпеки, зручності та продуктивності. Ці вимоги стануть основою для

подальшого проєктування та реалізації системи.

1.4 Постановка задачі досліджень

Основними варіанти використання для веб-сервісу, який оформлює бібліографічні посилання відповідно до стандарту ДСТУ 8302:2015 можуть бути:

1. Реєстрація користувача:

– користувач може створити обліковий запис для доступу у системі;

– система перевіряє введені дані, забезпечує унікальність облікових записів та зберігає інформацію про користувачів.

2. Аутентифікація та авторизація:

– користувач входить у систему, вводячи свої облікові дані, для доступу до функціоналу.

– система перевіряє введені дані та надає користувачу доступ до відповідних функцій системи.

3. Створення посилання:

– користувач може створити нове бібліографічне посилання, вводячи необхідні дані (автор, назва, рік публікації тощо).

– включає автоматичну перевірку коректності оформлення відповідно до стандарту ДСТУ 8302:2015.

4. Редагування посилання:

– користувач може змінювати інформацію в уже створених посиланнях.

– після редагування дані автоматично перевіряються на відповідність стандарту.

5. Збереження посилання

– збереження створених або відредагованих посилань у профілі користувача.

– доступ до збережених посилань для подальшого використання, редагування або експорту.

6. Експорт посилання

– користувач може експортувати посилання у формати PDF, Word або BibTeX.

– експорт забезпечує інтеграцію створених посилань у документи чи бібліографічні менеджери.

7. Перевірка коректності

– автоматична перевірка оформлення посилання на відповідність вимогам ДСТУ 8302:2015.

– виконується під час створення, редагування або за запитом користувача.

8. Пошук посилань

– користувач може шукати посилання за ключовими словами (наприклад, автор, рік публікації, тип джерела).

– пошук працює в межах збережених посилань у профілі користувача.

9. Фільтрація посилань

– функція для відбору посилань за заданими критеріями (рік, тип джерела, автор тощо).

– полегшує навігацію серед великої кількості збережених посилань.

10. Управління профілем

– користувач може змінювати особисті дані, такі як ім’я, електронна пошта, пароль.

– також у профілі доступний перегляд історії створених посилань з можливістю редагування або видалення.

11. Адміністрування системи (для адміністратора)

– адміністратор здійснює загальне управління системою, моніторинг роботи, налаштування параметрів та забезпечує стабільність її роботи.

Для створення UML-діаграми використання було використано синтаксис PlantUML [9]:

```
@startuml
left to right direction

actor Користувач as User
actor Адміністратор as Admin
User <|-- Admin

rectangle "Веб-сервіс для оформлення бібліографічних посилань\nвідповідно до ДСТУ 8302:2015" as System {

    usecase "Створення посилання" as CreateCitation
    usecase "Редагування посилання" as EditCitation
    usecase "Збереження посилання" as SaveCitation
    usecase "Експорт посилання" as ExportCitation
    usecase "Перевірка коректності" as ValidateCitation
    usecase "Управління профілем" as ManageProfile
    usecase "Реєстрація користувача" as RegisterUser
    usecase "Авторизація користувача" as AuthenticateUser

    usecase "Адміністрування системи" as SystemAdministration
    usecase "Налаштування параметрів стандарту" as ConfigureStandard
    usecase "Управління користувачами" as ManageUsers

    User --> RegisterUser
    User --> AuthenticateUser
    AuthenticateUser --> CreateCitation
    AuthenticateUser --> EditCitation
    AuthenticateUser --> SaveCitation
    AuthenticateUser --> ExportCitation
    AuthenticateUser --> ValidateCitation
    AuthenticateUser --> ManageProfile

    Admin --> SystemAdministration
    SystemAdministration --> ConfigureStandard : include
    SystemAdministration --> ManageUsers : include

    CreateCitation .> ValidateCitation : include
    EditCitation .> ValidateCitation : include

    ExportCitation -> (Формат PDF)
    ExportCitation -> (Формат Word)
    ExportCitation -> (Формат BibTeX)
}
@enduml
```

У середовищі розробки Visual Studio Code побудовано діаграму та експортовано файл із зображенням діаграми (рис. 1.6).

На рисунку 1.6 діаграма показує, як користувачі можуть створювати, редагувати, зберігати та експортувати посилання відповідно до українського стандарту оформлення, а адміністратори можуть керувати системою та налаштовувати параметри відповідно до вимог стандарту ДСТУ 8302:2015.

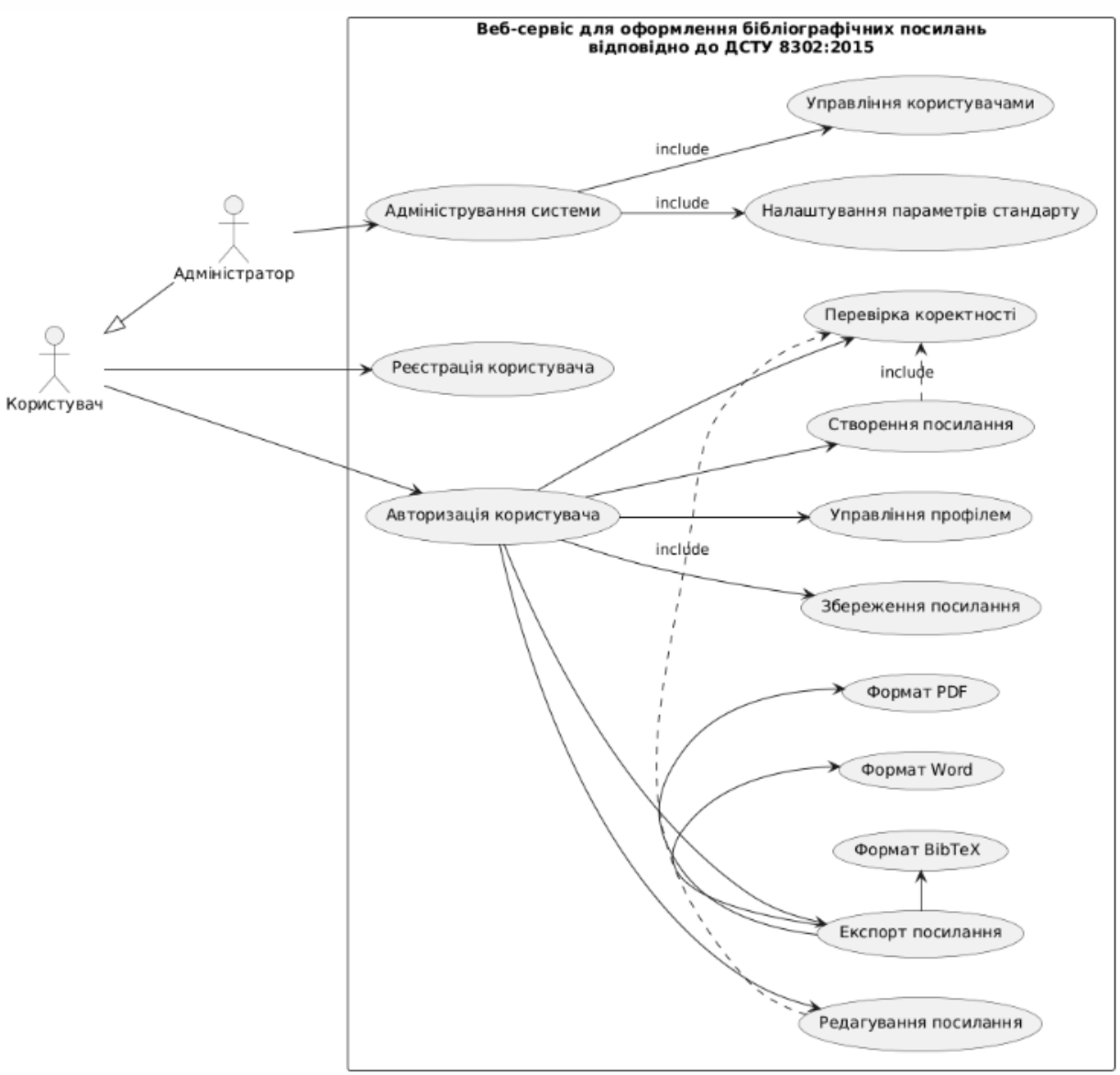

Рисунок 1.6 – Діаграма використання для веб-сервісу, який оформлює бібліографічні посилання відповідно до стандарту ДСТУ 8302:2015

Опис діаграми.

1. Актори:

– Користувач: типовий користувач сервісу, який може створювати, редагувати та експортувати бібліографічні посилання, перевіряти їх на відповідність стандарту, а також керувати своїм профілем.

– Адміністратор: користувач із розширеними правами, який може адмініструвати систему, налаштовувати параметри стандарту ДСТУ 8302:2015 та керувати користувачами.

2. Варіанти використання для Користувача:

– Створення посилання: користувач створює нове бібліографічне посилання. При цьому включено Перевірку коректності для відповідності ДСТУ 8302:2015.

– Редагування посилання: користувач може редагувати існуюче посилання з перевіркою його коректності.

– Збереження посилання: збереження посилання для подальшого використання.

– Експорт посилання: функція для виведення посилання у форматах PDF, Word або BibTeX.

– Перевірка коректності: перевірка правильності оформлення посилання відповідно до стандарту.

– Управління профілем: користувач може керувати своїм профілем, змінювати налаштування облікового запису.

3. Варіанти використання для Адміністратора:

– Адміністрування системи: основна функція для управління системою, що включає Налаштування параметрів стандарту (налаштування правил відповідно до ДСТУ 8302:2015) та Управління користувачами (додавання, видалення та редагування користувачів).

4. Зв’язки:

– Зв’язок include використовується для обов'язкових підпроцесів. Наприклад, Перевірка коректності є обов'язковою частиною як Створення

посилання, так і Редагування посилання.

– Експорт посилання має кілька форматів для збереження: PDF, Word і BibTeX.

Висновки до розділу:

Огляд існуючих сервісів для оформлення бібліографічних посилань свідчить про широкий вибір інструментів, що можуть значно полегшити роботу науковців та студентів. Однак лише Grafiati пропонує повну підтримку ДСТУ 8302:2015, що робить його найкращим вибором для українських користувачів. Інші сервіси, такі як Zotero, Mendeley і Cite This For Me, не мають вбудованої підтримки українського стандарту, але можуть використовуватися для автоматизації процесу з подальшим ручним редагуванням. Розробка нових сервісів або покращення існуючих для підтримки ДСТУ 8302:2015 є актуальним завданням, оскільки це дозволить українським науковцям зекономити час і підвищити точність оформлення бібліографічних посилань.

Для розробки веб-сервісу, призначеного для оформлення бібліографічних посилань і орієнтованого на створення зручного та гнучкого інтерфейсу для користувача, доцільним є вибір архітектури WebAPI у поєднанні з клієнтським застосунком, побудованим на JavaScript-фреймворках. Такий підхід дозволить забезпечити оптимальний користувацький досвід, високу інтерактивність сервісу та створить умови для подальшого розширення й модернізації проєкту.

Для створення веб-сервісу для оформлення бібліографічних посилань відповідно до ДСТУ 8302:2015 були визначені ключові вимоги до функціональності інформаційної системи. Ці вимоги було візуалізовано за допомогою діаграми варіантів використання, створеної із застосуванням синтаксису PlantUML. Такий підхід дозволяє чітко структурувати функціональні можливості системи та забезпечує базу для її подальшого проєктування й розробки.

РОЗДІЛ 2
ПРОЄКТУВАННЯ ВЕБ-СЕРВІСУ ДЛЯ ОФОРМЛЕННЯ БІБЛІОГРАФІЧНИХ ПОСИЛАНЬ

2.1 Вибір та обґрунтування технологій реалізації веб-сервісу для оформлення бібліографічних посилань

Для реалізації веб-сервісу для оформлення бібліографічних посилань відповідно до ДСТУ 8302:2015 у клієнтській частині, побудованій на React, було використано низку популярних бібліотек. Їх застосування забезпечило розширення функціональності, спрощення розробки та підвищення ефективності роботи застосунку.

Бібліотека react-router-dom є основним інструментом для організації маршрутизації в односторінкових веб-застосунках, створених за допомогою React [10]. Вона дозволяє розробникам забезпечити динамічну навігацію між різними сторінками або розділами застосунку, не виконуючи повного перезавантаження сторінки. Це забезпечує більш швидкий відгук системи та покращений користувацький досвід.

Однією з ключових особливостей react-router-dom є підтримка динамічних маршрутів, які дозволяють передавати параметри через URL [13]. Це важливо для застосунків, де сторінки генеруються на основі певних даних, наприклад, перегляд конкретного бібліографічного посилання за його ідентифікатором. Крім того, бібліотека надає можливість створення захищених маршрутів, доступ до яких отримують лише авторизовані користувачі, що є важливим для збереження конфіденційності та безпеки даних.

Ще однією важливою функцією react-router-dom є підтримка вкладених маршрутів. Це означає, що одна сторінка може містити кілька рівнів навігації, наприклад, основний профіль користувача та вкладки для

редагування особистих даних чи перегляду історії посилань. Такий підхід спрощує структуру застосунку, робить її більш модульною та легкою для розуміння й підтримки.

Завдяки react-router-dom, розробники можуть створювати інтуїтивно зрозумілу навігацію, що відповідає сучасним стандартам. Бібліотека дозволяє використовувати різні методи навігації, такі як переходи за допомогою посилань або програмні переходи через виклики функцій. Крім того, вона підтримує інтеграцію з історією браузера, що дозволяє користувачам використовувати кнопки "Назад" та "Вперед" для навігації по застосунку.

Таким чином, react-router-dom є незамінним інструментом для побудови масштабованих і зручних для користувачів React-застосунків, які потребують багатої навігації та інтерактивності. Завдяки своїй гнучкості та простоті у використанні, ця бібліотека дозволяє ефективно розвивати та підтримувати веб-застосунки будь-якої складності.

Бібліотека redux є ключовим інструментом для управління станом застосунку, що дозволяє централізовано зберігати та керувати даними, необхідними для роботи всіх компонентів. Вона ідеально підходить для додатків, де необхідно зберігати загальний стан, наприклад, інформацію про користувача, список створених бібліографічних посилань, результати пошуку або фільтрації.

Основною перевагою redux є його архітектура store -> action -> reducer, яка чітко визначає, як змінюється стан додатку [20]. Store виступає центральним сховищем даних, action описує, що потрібно змінити, а reducer визначає, як саме стан буде змінено. Такий підхід гарантує передбачуваність і контрольованість змін у стані навіть у складних додатках.

Крім цього, redux полегшує обмін даними між компонентами застосунку, зокрема в тих випадках, коли ці компоненти розташовані на

різних рівнях ієрархії. Наприклад, інформація про авторизованого користувача або стан пошуку може використовуватися кількома компонентами одночасно без необхідності передавати ці дані через властивості. Завдяки цьому код стає більш читабельним і легшим у підтримці.

Також redux пропонує інструменти для зручного налагодження, зокрема Redux DevTools, що дозволяють переглядати історію змін стану, відслідковувати виконані дії та навіть "відмотувати" стан до попереднього стану для тестування [20]. Ця функціональність особливо корисна для великих команд і проєктів, де важливо забезпечити стабільність системи під час розробки.

Таким чином, використання redux забезпечує ефективне управління даними, високу передбачуваність змін і значно спрощує розробку великих та інтерактивних веб-застосунків.

Бібліотека react-redux забезпечує інтеграцію Redux із React, спрощуючи процес управління станом застосунку. Вона дозволяє легко підключати компоненти React до центрального сховища Redux, використовуючи контекст для передачі даних. Це значно полегшує роботу з глобальним станом, оскільки компоненти отримують доступ до потрібних даних без необхідності глибокої передачі через властивості.

Однією з ключових переваг react-redux є інструменти для роботи з Redux, такі як connect, useSelector та useDispatch. Ці засоби забезпечують простий і зрозумілий спосіб підключення компонентів до сховища та виконання дій. Наприклад, useSelector дозволяє отримувати дані безпосередньо зі сховища, тоді як useDispatch надає можливість викликати дії для зміни стану.

Завдяки використанню react-redux, кількість коду, необхідного для роботи з Redux, значно зменшується, а його структура стає більш читабельною. Це особливо важливо для великих застосунків, де управління

станом може стати складним завданням. Бібліотека допомагає зробити взаємодію між React і Redux максимально ефективною, підтримуючи модульність та спрощуючи процес розробки.

Таким чином, react-redux є незамінним інструментом для реалізації зручного та структурованого управління станом у React-застосунках, що використовують Redux, забезпечуючи як зручність розробки, так і масштабованість системи [20].

Бібліотека axios використовується для виконання HTTP-запитів до сервера, забезпечуючи ефективну взаємодію між клієнтською та серверною частинами застосунку. Вона надає зручний API для роботи з REST API, що дозволяє легко отримувати, створювати, редагувати або видаляти дані. Це особливо корисно для веб-сервісів, де потрібно виконувати запити для обробки інформації, наприклад, створення або оновлення бібліографічних посилань [11].

Однією з важливих особливостей axios є підтримка асинхронних запитів, що забезпечує швидке реагування застосунку без блокування основного потоку виконання. Бібліотека автоматично обробляє відповіді сервера та помилки, що значно спрощує розробку, особливо у складних сценаріях із великою кількістю запитів.

Додатковою перевагою є можливість налаштування глобальних параметрів. За допомогою axios можна встановити базовий URL для всіх запитів, що спрощує роботу з API, а також додавати токени авторизації до заголовків, що необхідно для захищених застосунків.

Таким чином, axios є потужним інструментом для реалізації HTTP-запитів, надаючи зручний і гнучкий механізм для взаємодії з сервером, який підходить для створення сучасних веб-застосунків [11].

Бібліотека Bootstrap використовується для створення адаптивного та сучасного дизайну інтерфейсу, значно прискорюючи процес розробки завдяки наданню готових стилів і компонентів [12]. Вона забезпечує

розробників інструментами для швидкого створення формулярів, кнопок, таблиць та інших елементів інтерфейсу, що відповідають сучасним стандартам дизайну.

Однією з головних переваг Bootstrap є підтримка адаптивного дизайну, завдяки чому інтерфейс автоматично підлаштовується під різні типи пристроїв — від смартфонів до великих моніторів. Це дозволяє створювати універсальні застосунки, що коректно відображаються на будь-яких екранах.

Для розробників, які використовують React, Bootstrap легко інтегрується через спеціалізовані бібліотеки, такі як react-bootstrap, що забезпечує безшовну взаємодію між React-компонентами та стилями Bootstrap. Це поєднання дозволяє не лише швидко створювати красивий інтерфейс, але й легко підтримувати та розширювати його.

Таким чином, Bootstrap є важливим інструментом для створення естетично привабливого, функціонального та адаптивного дизайну в сучасних веб-застосунках.

Використання бібліотек react-router-dom, redux, react-redux, axios та bootstrap стало ключовим етапом у розробці клієнтської частини веб-сервісу. Ці інструменти дозволили реалізувати функціональність, яка відповідає сучасним стандартам розробки та забезпечує високий рівень зручності як для користувачів, так і для розробників.

Завдяки react-router-dom вдалося організувати ефективну маршрутизацію, яка підтримує динамічні маршрути, захищені сторінки та гнучку структуру навігації. Це зробило можливим створення інтуїтивно зрозумілого застосунку з логічною системою переходів між різними його частинами.

Інтеграція redux і react-redux забезпечила централізоване управління станом застосунку, дозволивши зберігати й обробляти дані, такі як інформація про користувача або створені бібліографічні посилання, у

передбачуваній та контрольованій архітектурі. Це спростило роботу з глобальним станом і зробило застосунок більш стабільним та легким у масштабуванні.

Використання axios дало змогу реалізувати асинхронну взаємодію з сервером через REST API. Завдяки підтримці HTTP-запитів різного типу та автоматичній обробці відповідей сервера бібліотека забезпечила швидку та надійну роботу з даними. Це особливо важливо для застосунків, які інтенсивно взаємодіють із сервером, таких як веб-сервіс для створення й управління бібліографічними посиланнями.

Бібліотека bootstrap, у свою чергу, дозволила створити адаптивний і сучасний дизайн інтерфейсу з використанням готових компонентів, таких як кнопки, таблиці та форми. Це забезпечило високу швидкість розробки й уніфікований зовнішній вигляд застосунку на різних пристроях, що сприяє покращенню користувацького досвіду.

Поєднання цих бібліотек створило міцну основу для побудови гнучкого, масштабованого та зручного для користувача інтерфейсу. Такий підхід дозволив спростити процес розробки, забезпечити високу інтерактивність застосунку та створити можливості для його подальшого розвитку й інтеграції з іншими платформами.

2.2 Проєктування інтерфейсу та верстка основних сторінок веб-сервісу для оформлення бібліографічних посилань

На першому етапі розробки додатку були вирішено реалізувати форми, за допомогою яких користувач системи міг би додавати (або редагувати) інформацію про літературні джерела, для яких потрібно створити бібліографічне посилання у відповідності до ДСТУ 8302:201. Для визначення набору необхідних полів було проаналізовано обсяг інформації, яку повинен вводити користувач для створення

бібліографічного опису кожного типу джерела. У якості зразку дизайну було використано інтерфейс сервісу Grafiati. Для верстки сторінок та візуального оформлення було використано бібліотеку Bootstrap 5.3.

На рис. 2.1 наведено зовнішній вигляд розробленої форми для введення інформації, необхідної для оформлення посилання на дисертацію.

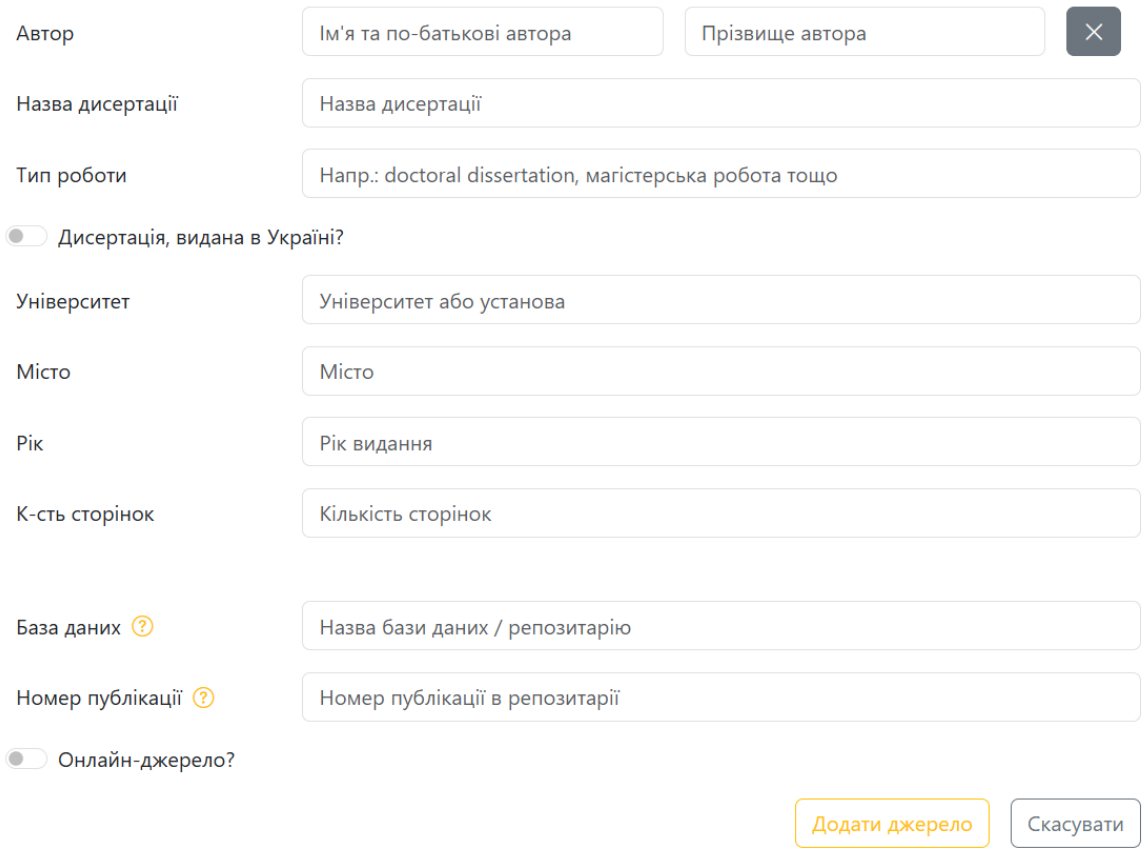

Рисунок 2.1 – Розробка макету сторінки додавання посилання на дисертацію

На рис. 2.2 наведено частину HTML- файлу React-компоненту ThesisForm.js з реалізацією форми для додавання посилання на дисертацію.

```
import { useState } from "react";
import specialities from "../data/specialities";

const ThesisForm = () => {
    const [isUkrainian, setIsUkrainian] = useState(false);
    const [isOnline, setIsOnline] = useState(false);
    return (
        <>
            <div className="row my-3 gx-3">
                <label className="col-3 col-form-label">Автор</label>
                <div className="col-4">
                    <input type="text" placeholder="Ім'я та по-батькові автора"
                        className="form-control"></input>
                </div>
                <div className="col-4">
                    <input type="text" placeholder="Прізвище автора"
                        className="form-control col-3"></input>
                </div>
                <div className="col-1">
                    <button className="btn btn-secondary">
                        <i class="bi bi-x-lg"></i>
                    </button>
                </div>
            </div>
            <div className="row my-3 gx-3">
                <label className="col-3 col-form-label">Назва дисертації</label>
                <div className="col-9">
                    <input type="text" placeholder="Назва дисертації"
                        className="form-control "></input>
                </div>
            </div>
            <div className="row my-3 gx-3">
                <label className="col-3 col-form-label">Тип роботи</label>
                <div className="col-9">
                    <input type="text" placeholder="Напр.: doctoral dissertation, магістерська робота тощо"
                        className="form-control "></input>
                </div>
            </div>
```

Рисунок 2.2 – HTML-код реалізації сторінки додавання посилання на дисертацію

Форма містить два додаткові перемикачі (чекбокси), що дозволяють ввести додаткову інформацію, якщо дисертація захищена в Україні, та чи потрібно оформити посилання на онлайн-джерело. Для реалізації відображення прихованих полів використано хук useState. За допомогою даного хука створені дві реактивні змінні isUkrainian та isOnline для керування відповідними перемикачами, а також методи setIsUkrainian та setIsOnline для оновлення їх значень. Відповідні змінні використані для реалізації умовного рендерингу, при якому елементи форм відображаються (тобто мають значення властивості display, що дорівнює “block”), коли змінна має значення true. Якщо змінна має значення false,

елементи форми не відображаються (значення властивості display дорівнює “none”). На рис. 2.3 зображено реалізацію умовного рендерингу для полів, необхідних для додавання інформації, необхідної для оформлення посилання на дисертацію, що була захищена в Україні.

```
<div style={{display: (isUkrainian ? "block" : "none")}}>
<div className="row my-3 gx-3">
    <label className="col-3 col-form-label">Науковий ступінь</label>
    <div className="col-9">
        <select type="number" defaultValue={"немає"} className="form-select"
        {...register("academicDegree")}>
            <option value={"немає"}>-------------</option>
            <option value={"канд."}>кандидат наук</option>
            <option value={"д-ра"}>доктор наук</option>
            <option value={"д-ра філософії в галузі"}>доктор філософії</
            option>
        </select>
    </div>
</div>
```

Рисунок 2.3 – Приклад реалізації умовного рендерингу за допомогою реактивної змінної та синтаксису JSX

На рис. 2.4 наведено зовнішній вигляд розробленої форми при переведенні перемикача у встановлене значення, внаслідок чого відкриваються додаткові елементи форми для введення інформації стосовно дисертації, захищеної в Україні. Як видно, на формі з’являються додаткові поля – «Науковий ступінь», «Спеціальність», «Код спеціальності». Для заповнення переліку спеціальностей у структурі проєкту створено файл specialities.js, який містить масив об’єктів з описом, що включає унікальний ідентифікатор id, назву спеціальності, та відповідне скорочення назви у відповідності до ДСТУ 8302:2015. Структуру файлу specialities.js наведено на рис. 2.5.

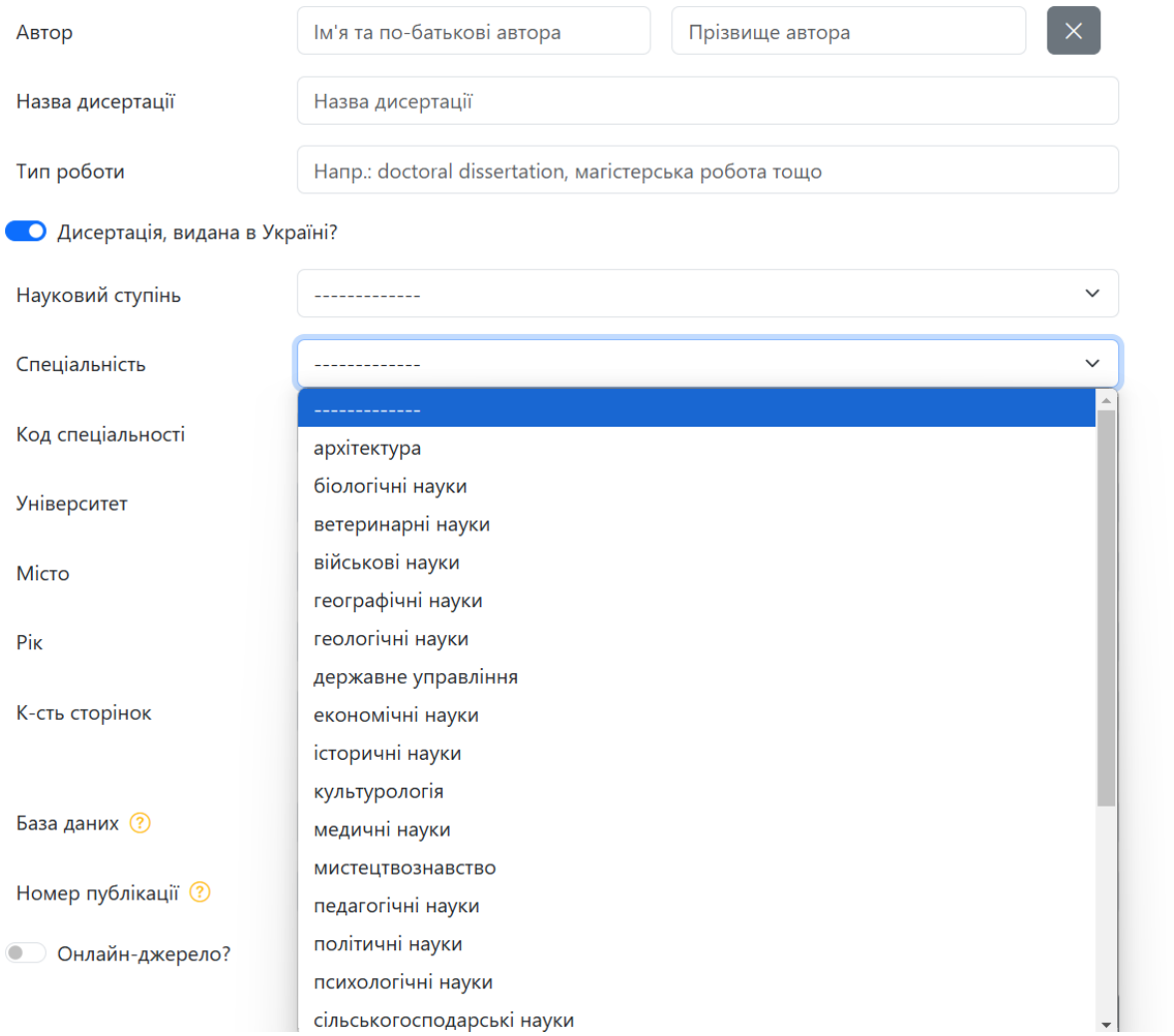

Рисунок 2.4 – Зовнішній вигляд форми для додавання опису дисертації, захищеної в Україні

Масив об’єктів використовується для формування HTML-елементів <option/> шляхом синтаксису рендерингу списків JSX з використанням функції Array.map(). У якості значення використовується поле value, а для відображення тексту для користувача – поле name. Для оптимізації роботи зі списками у React об’єкти з файлу specialities.js, що описують спеціальності, також мають унікальний ідентифікатор id, що використовуються в якості ключа елементів <option/> (атрибут «key»).

Код перетворення масиву об’єктів, що описують спеціальності, у елементи <option/>, наведено на рис. 2.6.

```
const specialities = [
    {id: 1, name: "архітектура", value: "архітектури"},
    {id: 2, name: "біологічні науки", value: "біол. наук"},
    {id: 3, name: "ветеринарні науки", value: "вет. наук"},
    {id: 4, name: "військові науки", value: "військ. наук"},
    {id: 5, name: "географічні науки", value: "геогр. наук"},
    {id: 6, name: "геологічні науки", value: "геол. наук"},
    {id: 7, name: "державне управління", value: "наук з держ. упр."},
    {id: 8, name: "економічні науки", value: "екон. наук"},
    {id: 9, name: "історичні науки", value: "іст. наук"},
    {id: 10, name: "культурологія", value: "культурології"},
    {id: 11, name: "медичні науки", value: "мед. наук"},
    {id: 12, name: "мистецтвознавство", value: "мистецтвознавства"},
    {id: 13, name: "педагогічні науки", value: "пед. наук"},
    {id: 14, name: "політичні науки", value: "політ. наук"},
```

Рисунок 2.5 – Файл з масивом назв спеціальностей та їх позначенням у відповідності до ДСТУ 8302:2015

```
<div className="row my-3 gx-3">
    <label className="col-3 col-form-label">Спеціальність</label>
    <div className="col-9">
        <select type="number" className="form-select" {...register("specialityName")}
        defaultValue={"немає"}>
            <option value={"немає"}>-------------</option>
            {Array.from(specialities).map(speciality=>{
                return (
                <option key={speciality.id} value={speciality.value}>
                    {speciality.name}
                </option>)
            })}
        </select>
    </div>
</div>
```

Рисунок 2.6 – Приклад реалізації рендерингу списку спеціальностей за допомогою синтаксису JSX

При виборі користувачем конкретної спеціальності використовується значення, що зберігається у атрибуті value елемента option.

На рис. 2.7 наведено вигляд розробленої форми для введення інформації, необхідної для оформлення посилання на книгу. Серед особливостей варто відзначити наявність кнопки, яка дозволяє додавати авторів, а також два додаткові перемикачі (чекбокси), що дозволяють ввести додаткову інформацію, якщо книга є частиною багатотомного видання, та чи потрібно оформити посилання на онлайн-джерело. Для реалізації відображення прихованих полів також використано хук useState [14]. За допомогою даного хука створені дві реактивні змінні isPart та isOnline для керування відповідними перемикачами, а також методи setIsPart та setIsOnline для оновлення їх значень. Відповідні змінні використані для реалізації умовного рендерингу.

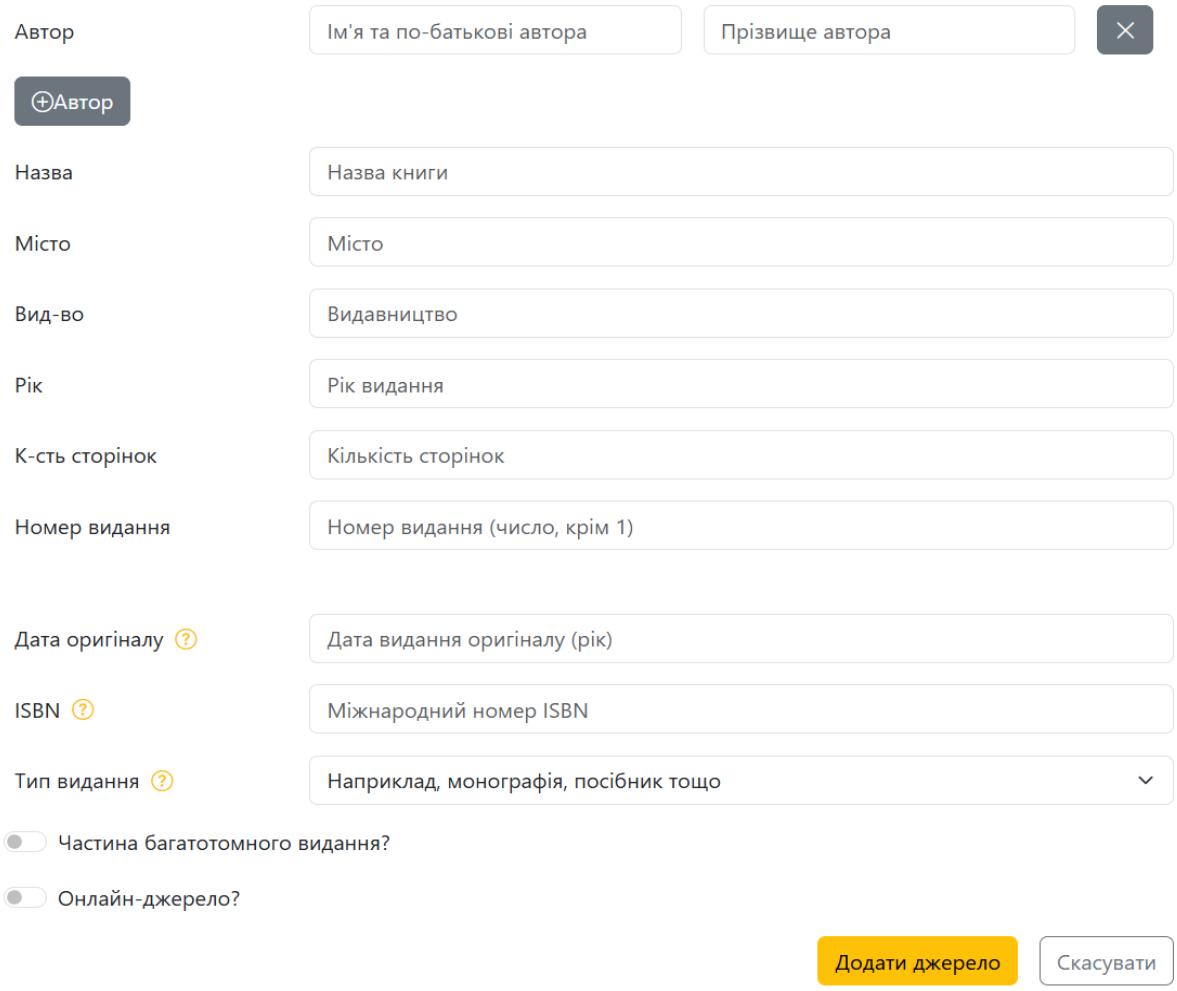

Рисунок 2.7 – Розробка макету сторінки додавання посилання на дисертацію

На рис. 2.8 наведено частину HTML- файлу React-компоненту BookForm.js з реалізацією форми для додавання посилання на дисертацію.

```
import { useState } from "react";

const BookForm = () => {
    const [isPart, setIsPart] = useState(false);
    const [isOnline, setIsOnline] = useState(false);
    return (
        <>
            <div className="row my-3 gx-3">
                <label className="col-3 col-form-label">Автор</label>
                <div className="col-4">
                    <input type="text" placeholder="Ім'я та по-батькові автора"
                        className="form-control"></input>
                </div>
                <div className="col-4">
                    <input type="text" placeholder="Прізвище автора"
                        className="form-control col-3"></input>
                </div>
                <div className="col-1">
                    <button className="btn btn-secondary">
                        <i class="bi bi-x-lg"></i>
                    </button>
                </div>
            </div>
            <div className="row my-3 gx-3">
                <div className="col">
                    <button className="btn btn-secondary">
                        <i class="bi bi-plus-circle"></i>Автор
                    </button>
                </div>
            </div>
            <div className="row my-3 gx-3">
                <label className="col-3 col-form-label">Назва</label>
                <div className="col-9">
                    <input type="text" placeholder="Назва книги"
                        className="form-control "></input>
                </div>
            </div>
            <div className="row my-3 gx-3">
```

Рисунок 2.8 – HTML-код реалізації сторінки додавання посилання на книгу

На рис. 2.9 зображено додаткові поля, які відображаються на формі, якщо книга є частиною багатотомного видання та є онлайн-джерелом.

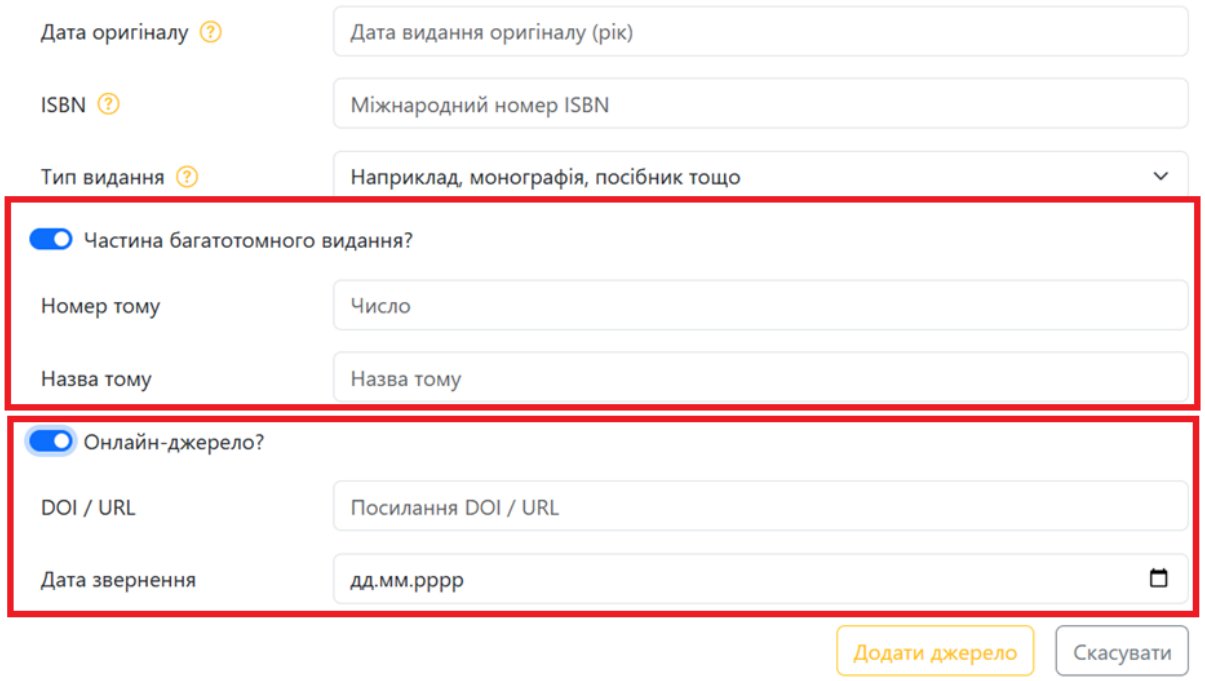

Рисунок 2.9 – Відображення додаткових полів у формі для додавання опису книги, що є частиною багатотомного видання та є онлайн-джерелом

На рис. 2.10 наведено вигляд розробленої форми для введення інформації, необхідної для оформлення посилання на статтю. Серед особливостей варто відзначити наявність кнопки, яка дозволяє додавати авторів, а також додаткового перемикача (чекбокса), що дозволяє відобразити поля для введення додаткової інформації, якщо для бібліографічного опису статті потрібно додати посилання на онлайн-джерело та дату звернення. Для реалізації відображення прихованих полів також використано хук useState. За допомогою даного хука створено реактивну змінну isOnline для керування відповідним перемикачем, а також метод setIsOnline для оновлення її значення. Відповідні змінні використані для реалізації умовного рендерингу.

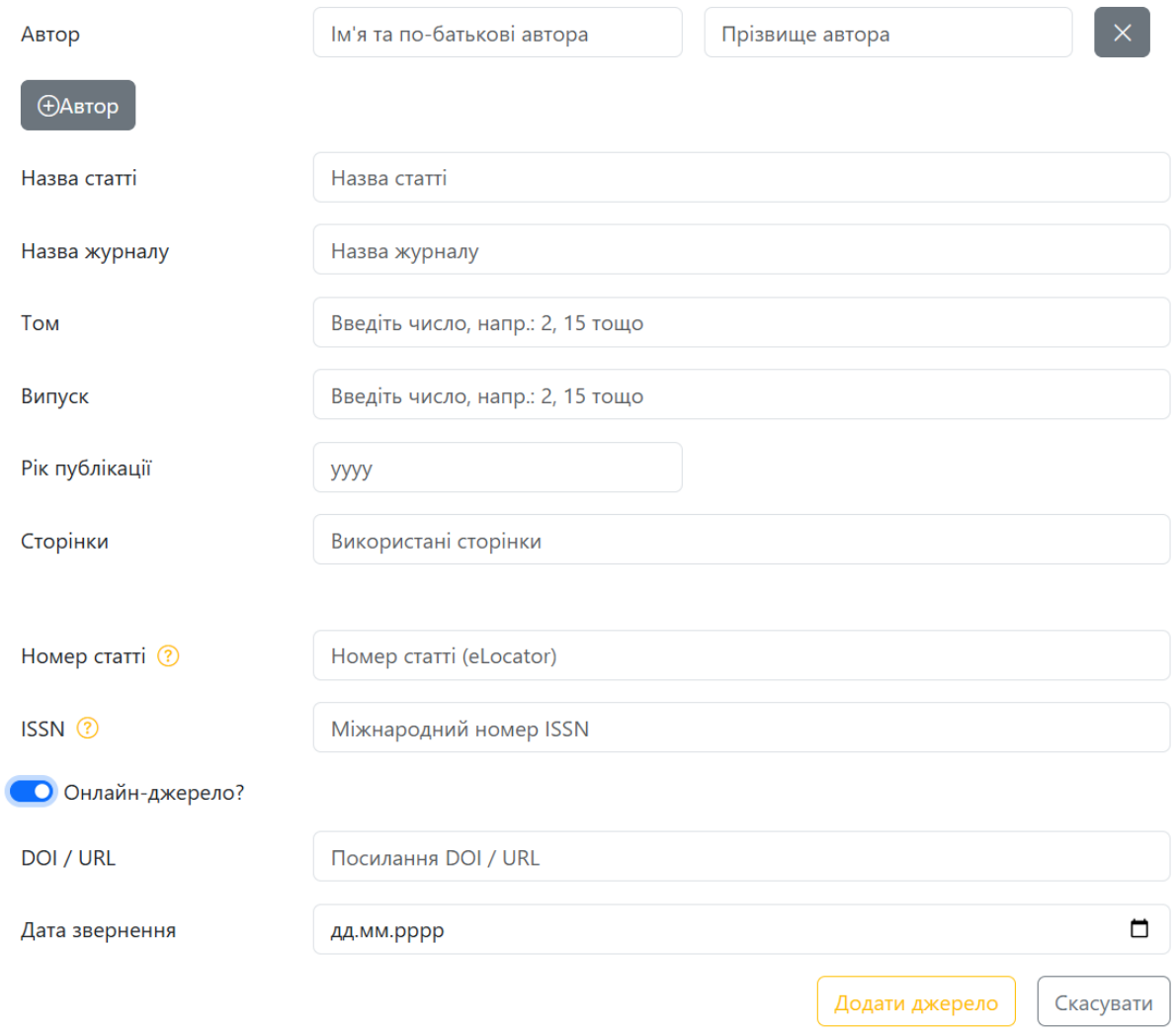

Рисунок 2.10 – Розробка макету сторінки додавання посилання на статтю

На рис. 2.11 наведено частину HTML- файлу React-компоненту ArticleForm.js з реалізацією форми для додавання посилання на статтю. На формі також розміщені два поля, які не використовуються для формування бібліографічного опису згідно стандарту ДСТУ 8302:2015 – «Номер статті» та «ISSN». Відповідні дані використовуються у інших стилях цитування (зокрема, номер статті використовується у стилях APA, Chicago, Harvard, IEEE, ISSN – у стилі ISO 690:2010. Ці поля було вирішено розмістити по аналогії з сервісом Grafiati для можливості подальшого удосконалення додатку та реалізації можливості переключення відображення посилання у відповідності до різних стилів.

```
import { useState } from "react";

const ArticleForm = () => {
    const [isOnline, setIsOnline] = useState(false);
    return (
        <>
            <div className="row my-3 gx-3">
                <label className="col-3 col-form-label">Автор</label>
                <div className="col-4">
                    <input type="text" placeholder="Ім'я та по-батькові автора"
                        className="form-control"></input>
                </div>
                <div className="col-4">
                    <input type="text" placeholder="Прізвище автора"
                        className="form-control col-3"></input>
                </div>
                <div className="col-1">
                    <button className="btn btn-secondary">
                        <i class="bi bi-x-lg"></i>
                    </button>
                </div>
            </div>
            <div className="row my-3 gx-3">
                <div className="col">
                    <button className="btn btn-secondary">
                        <i class="bi bi-plus-circle"></i>Автор
                    </button>
                </div>
            </div>
            <div className="row my-3 gx-3">
                <label className="col-3 col-form-label">Назва статті</label>
                <div className="col-9">
                    <input type="text" placeholder="Назва статті"
                        className="form-control "></input>
                </div>
            </div>
            <div className="row my-3 gx-3">
                <label className="col-3 col-form-label">Назва журналу</label>
```

Рисунок 2.11 – HTML-код реалізації сторінки додавання посилання на статтю

На рис. 2.12 наведено вигляд розробленої форми для введення інформації, необхідної для оформлення посилання на веб-сторінку (сайт). Серед особливостей варто відзначити наявність кнопки, яка дозволяє додавати авторів. Поле «Сайт» використовується не в усіх стилях

оформлення бібліографічних посилань і його також вирішено залишити для забезпечення можливості подальшого розширення функціоналу.

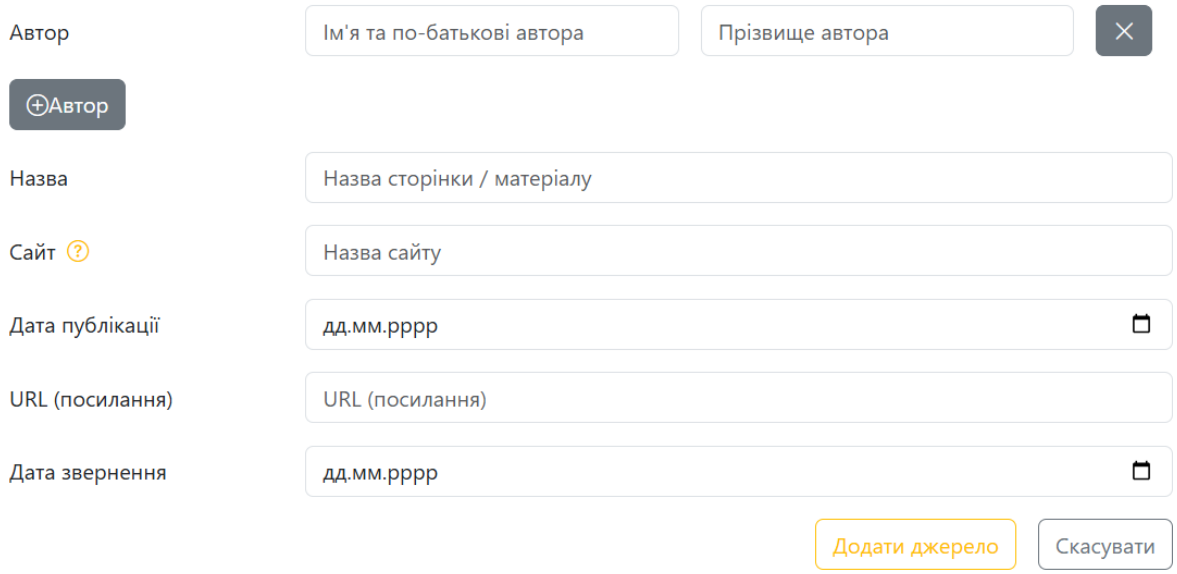

Рисунок 2.12 – Розробка макету сторінки додавання посилання на веб-сторінку (сайт)

На рис. 2.13 наведено частину HTML-файлу React-компоненту SiteForm.js з реалізацією форми для додавання посилання на веб-сторінку (сайт).

Як видно з HTML-коду сторінок основних форм (рис. 2.2, 2.8, 2.11, 2.13), для верстки активно використовувалися класи бібліотеки Bootstrap 5.3 [12]:

– row – відповідає за формування рядків, у яких колонки (елементи з класом col) розміщуються як flex-елементи у flex-контейнері);

– my-3 – для реалізації зовнішніх відступів (margin) у вертикальній площині розміром 1 rem;

– gx-3 – для реалізації відступів між елементами з класом col всередині row у горизонтальній площині розміром 1 rem;

– col-form-label – для стилізації елементів <label/> та розміщення їх в одному рядку з елементами вводу;

```
import { useState } from "react";

const SiteForm = () => {
    return (
        <>
            <div className="row my-3 gx-3">
                <label className="col-3 col-form-label">Автор</label>
                <div className="col-4">
                    <input type="text" placeholder="Ім'я та по-батькові автора"
                        className="form-control"></input>
                </div>
                <div className="col-4">
                    <input type="text" placeholder="Прізвище автора"
                        className="form-control col-3"></input>
                </div>
                <div className="col-1">
                    <button className="btn btn-secondary">
                        <i class="bi bi-x-lg"></i>
                    </button>
                </div>
            </div>
            <div className="row my-3 gx-3">
                <div className="col">
                    <button className="btn btn-secondary">
                        <i class="bi bi-plus-circle"></i>Автор
                    </button>
                </div>
            </div>
            <div className="row my-3 gx-3">
                <label className="col-3 col-form-label">Назва</label>
                <div className="col-9">
                    <input type="text" placeholder="Назва сторінки / матеріалу"
                        className="form-control "></input>
                </div>
            </div>
            <div className="row my-3 gx-3">
                <label className="col-3 col-form-label">
                    Сайт  
```

Рисунок 2.13 – HTML-код реалізації сторінки додавання посилання на веб-сторінку (сайт)

– form-input – для стилізації основних елементів введення форм;

– form-check form-switch – для оформлення блоку, що містить перемикач (чекбокс) та відповідна мітка з описом;

– form-check-input – для оформлення чекбоксу;

– form-check-label – для оформлення мітки з описом біля перемикача;

– d-flex – для перетворення елементу у flex-контейнер (встановлення властивості display у значення flex);

– justify-content-end – для розміщення дочірніх елементів (контенту) в кінці flex-контейнера;

– btn – для базової стилізації кнопок та посилань;

– btn-outline-warning, btn-outline-secondary – для стилізації кнопок збереження результатів та скасування введення.

Таким чином, на першому етапі реалізації додатку було виконано аналіз обсягу необхідної для оформлення кожного типу літературного джерела інформації. З використанням бібліотеки Bootstrap розроблено та стилізовано форми для додавання основних типів посилань на літературній джерела, зокрема дисертацію, книгу, статтю та сайт. Для реалізації динамічності, зокрема можливості введення додаткової інформації, використані можливості вбудованого хука React useState та умовний рендеринг мови JSX.

2.3 Реалізація обробки форм

2.3.1 Вибір бібліотеки для реалізації обробки форм

На наступному етапі розробки додатку необхідно було забезпечити отримання інформації з полів форм для її подальшої обробки.

При використанні бібліотеки React існує декілька способів отримання інформації з полів форм – використання посилань (або рефів, refs), використання змінних стану (що повертаються хуком useState, та зміна значення змінних у обробнику події onChange() відповідного елементу вводу), а також використання спеціальних бібліотек [15, 16]. На сьогодні існує декілька достатньо зручних бібліотек для роботи з формами у React.

Під час навчання ми найчастіше використовували 2 бібліотеки – react-hook-form та Formik. Проте є альтернативи, тож було вирішено дослідити можливі варіанти.

Redux Form призначений для спільного використання з бібліотекою Redux та легко інтегрується з нею для зберігання стану форм у глобальному сховищі. До основних переваг можна віднести власне легку інтеграцію з Redux, але є і суттєві недоліки – важкий стан через повторні оновлення; існує ймовірність сповільнення великого проєкт через часті дії Redux; відносно великий розмір – біля 27 KB (залежно від налаштувань).

Полегшеною альтернативою Redux Form без жорсткої прив'язки до Redux є бібліотека Final Form / React Final Form. Перевагами бібліотеки є висока продуктивність та гнучкі можливості інтеграції без залежності від Redux [17]. Основними недоліками можна вважати те, що API є трохи складнішим, ніж у інших бібліотек, та менш активна спільнота розробників порівняно з Formik або React Hook Form. Проте бібліотека є дуже легкою – лише біля 6 KB.

Останнім часом при розробці React-застосунків також використовується версія бібліотеки Final Form, яка має назву React Final Form Hooks [18, 19]. Як видно з назви, бібліотека підтримує хуки, що спрощує інтеграцію в сучасні React-застосунки, є легкою та продуктивною. Головним недоліком можна вважати меншу популярність і наявність спільноти розробників, ніж у React Hook Form.

Для неконтрольованих форм часто застосовують бібліотеку Unform, яка створена для забезпечення високої продуктивності в масштабних застосунках. Основною перевагою є те, що вона підходить для великих форм із малою кількістю повторних рендерів, а також легку інтеграцію з будь-яким UI-фреймворком. До недоліків можна віднести меншу популярність, тому може бути складніше знайти підтримку. Як і React Final Form, є дуже легкою – лише біля 6 KB.

Для створення форм на основі JSON-схем розроблено бібліотеку React JSON Schema Form. Дана бібліотека ідеально підходить для динамічних форм, що генеруються автоматично на основі опису схеми у форматі JSON [19]. Але бібліотека має ряд недоліків, зокрема залежність від JSON Schema і через це великий розмір (біля 80 KB). Фактично функціональність бібліотеки є надлишковою для простих форм, тож цей варіант був одразу відкинутий.

Також варто згадати комбінацію бібліотек Formik та Zod для валідації. У такому поєднанні перевагами є те, що Zod має гнучке та типізоване API для валідації, і формат валідації ближчий до TypeScript. Недоліком є відносно слабка підтримка (особливо у порівнянні з Yup).

Проведений аналіз показав, що з урахуванням обсягу та особливостей проєкту створення ресурсу для оформлення бібліографічних посилань доцільно сконцентруватися на порівнянні можливостей двох найпопулярніших бібліотек – Formik та React Hook Form. Обидві дані бібліотеки призначені для управління формами в React-застосунках та дозволяють суттєво спростити обробку введених користувачем даних, валідацію та інтеграцію з іншими бібліотеками.

У таблиці 2.1 наведено результати порівняльного аналізу вказаних бібліотек із зазначенням їх основних переваг та недоліків.

Як видно з наведених у таблиці результатів, до основних переваг React Hook Form можна віднести високу продуктивність, легке налаштування та інтеграції з UI-бібліотеками, відносно менший розмір.

До основних недоліків бібліотеки React Hook Form можна віднести те, що її застосування може бути складнішим для початківців через необхідність роботи з неконтрольованими компонентами, а також менш інтуїтивний підхід до обробки стану форми.

Таблиця 2.1 – Порівняльний аналіз бібліотек React Hook Form та Formik

Критерій	React Hook Form	Formik
Підхід до роботи	Використовує React-хуки та не контролює компоненти. Маніпулює формами через рефлекси DOM	Працює з контрольованими компонентами, оновлюючи стан на кожне введення
Продуктивність	Вища продуктивність завдяки мінімальній кількості повторних рендерів	Повільніше через часті оновлення стану та повторні рендери
Розмір бібліотеки	Легша: ~14 KB (включаючи залежності)	Важча: ~35 KB (включаючи залежності)
API та навчання	Просте API, легке для швидкого навчання. Використовує мінімум коду	Більш розгорнуте API, яке може бути складнішим для початківців
Валідація	Підтримує валідацію через Yup, Zod, або кастомні функції	Широко використовує Yup, але можна використовувати й інші засоби
Інтеграція зі сторонніми бібліотеками	Легко інтегрується з бібліотеками UI (Material UI, Ant Design тощо) без необхідності створення спеціальних обгорток	Може потребувати додаткових налаштувань для інтеграції з UI-бібліотеками
Можливість роботи з великими формами	Оптимізований для великих форм завдяки зменшенню кількості рендерів	Може призводити до затримок при роботі з великими формами через часті оновлення стану

Основними перевагами Formik є добре задокументоване API, зручність для початківців через контрольований компонентний підхід, широка підтримка бібліотеки валідації Yup.

Але бібліотека має також і недоліки – зокрема, низьку продуктивність на великих формах та більший розмір пакета з бібліотекою.

Як зазначено у [19], обираючи між цими бібліотеками, слід враховувати розмір та складність форм, а також рівень досвіду розробників. Якщо головним пріоритетом є продуктивність і легкість коду, React Hook Form буде кращим вибором. Якщо ж перевага надається детальному контролю та простішому навчанню для початківців, варто використовувати бібліотеку Formik.

Для остаточно вибору було враховано 2 аспекти. При заповненні інформації, необхідної для оформлення посилання, опрацьовується достатньо велика кількість полів. Також необхідно забезпечити високу продуктивність, оскільки фактично робота з додаванням інформації для створення бібліографічного опису відбувається на одній сторінці SPA-додатку. Тож, з урахуванням описаних вище висновків, було вирішено зупинитися на бібліотеці React Hook Form.

2.3.2 Реалізація обробки форм додавання бібліографічних посилань з використанням бібліотеки React Hook Form

На рис. 2.14 наведено фрагмент коду компонента BookForm, що містить код деструктуризації об’єктів, що повертаються хуком useForm, імпортованим з бібліотеки React Hook Form. Варто відзначити, що у якості параметрів до useForm() передано об’єкт, що містить поле зі значеннями за замовчанням (defaultValues). Зокрема, для забезпечення відображення початкових полів для введення імені та прізвища першого автора, передано об’єкт, що містить властивість authors.

src > components > JS BookForm.js > BookForm

```
1   import { useState } from "react";
2   import { useFieldArray, useForm } from "react-hook-form";
3
4   const BookForm = () => {
5       const [isPart, setIsPart] = useState(false);
6       const [isOnline, setIsOnline] = useState(false);
7       const {
8           register,
9           handleSubmit,
10          control
11      } = useForm({
12          defaultValues: {
13              authors: [{
14                  firstname: "",
15                  surname: ""
16              }]
17          }
18      });
19      const {
20          fields,
21          append,
22          remove
23      } = useFieldArray(
24          {
25              control,
26              name: "authors"
27          }
28      );
29      const onSubmit = (data) => {
30          console.log(data);
31      }
```

Рисунок 2.14 – Імпорт та подальша деструктуризація основних функцій бібліотеки React Hook Form

У розробленому програмному забезпеченні використано наступні об’єкти:

– функція register – призначена для реєстрації поля у формі, зв’язуючи його з внутрішнім механізмом управління; приймає 2 параметри: name – назву поля та options – об’єкт із правилами валідації;

– функція handleSubmit – використовується для обробки події надсилання форми; приймає функцію зворотного виклику (callback), який викликається після успішної валідації;

– об’єкт formState – містить інформацію про поточний стан форми;

– функція watch – дозволяє повернути поточне значення поля або слідкувати за змінами в полі;

– функція reset – дозволяє скинути значення форми до початкових або вказаних значень;

– функція trigger – дозволяє викликати примусову валідацію одного або декількох полів форми;

– функція setValue – дозволяє вручну встановити значення конкретного поля;

– функція getValues – дозволяє повернути поточні значення всіх полів форми або конкретного поля.

На рис. 2.15 наведено приклад використання функції register() для реєстрації полів у формі. Зокрема, приклад містить реєстрацію поля для назви книги («title»), міста видання («city»), та видавництва («publisher»). Варто відзначити, що функція register() повертає об’єкт із атрибутами, які необхідні для реєстрації цього поля у формі, а саме:

– name: назва поля, яку використовує React Hook Form;

– onChange: функція для відстеження змін у полі;

– onBlur: функція для обробки події втрати фокусу;

– ref: посилання на DOM-елемент, що дозволяє React Hook Form відслідковувати поле.

При додаванні функції у якості атрибуту використано синтаксис спред-оператора ({...}). Даний оператор розкладає об’єкт на окремі атрибути та додає їх до елемента <input>. Це дозволяє автоматично підключити поле до логіки React Hook Form без необхідності вручну додавати кожен обробник.

```
78                  <div className="row my-3 gx-3">
79                      <label className="col-3 col-form-label">Назва</label>
80                      <div className="col-9">
81                          <input type="text"
82                              placeholder="Назва книги"
83                              className="form-control " {...register("title")}>
84                          </input>
85                      </div>
86                  </div>
87                  <div className="row my-3 gx-3">
88                      <label className="col-3 col-form-label">Місто</label>
89                      <div className="col-9">
90                          <input type="text"
91                              placeholder="Місто"
92                              className="form-control "
93                              {...register("city")}>
94                          </input>
95                      </div>
96                  </div>
97                  <div className="row my-3 gx-3">
98                      <label className="col-3 col-form-label">Вид-во</label>
99                      <div className="col-9">
100                         <input type="text"
101                             placeholder="Видавництво"
102                             className="form-control "
103                             {...register("publisher")}>
104                         </input>
105                     </div>
```

Рисунок 2.15 – Приклад реєстрації полів форми з використанням функції register() бібліотеки React Hook Form

Варто відзначити, що форми для створення бібліографічного опису книги, статті та сайту повинні мати можливість додавати декількох авторів. Це вимагає динамічного керування кількістю полів, які використовуються для введення імені та прізвища авторів (а також

видаляти зайві, випадково створені поля). Ця задача в цілому є достатньо складною для реалізації. Але до переваг обраної бібліотеки React Hook Form належить, зокрема, і підтримка динамічного створення форм з можливістю додавання та видалення полів у формах, які містять списки або масиви даних. Для цього в бібліотеці реалізовано дуже зручний хук useFieldArray().

Хук useFieldArray() у React Hook Form приймає об'єкт параметрів із кількома ключовими властивостями, які визначають його поведінку, зокрема:

– об'єкт control (обов'язковий), що повертається з useForm() і який використовується хуком для управління станом форми;

– параметр name типу string (обов'язковий) – задає назву масиву полів у структурі даних форми; використовується для доступу до цього масиву за допомогою методу register, що повертається хуком useForm;

– параметр keyName типу string (необов'язковий) – ім'я ключа, який використовується для унікальної ідентифікації кожного елемента в масиві (за замовчуванням має значення «id»).

Основними об'єктами, що повертає хук useFieldArray(), є:

– масив fields – масив об'єктів, що представляють поля; кожен об'єкт містить ідентифікатор (id) та інші значення;

– функція append – дозволяє додати нове поле або поля до кінця масиву;

– функція prepend – дозволяє додати нове поле або поля до початку масиву;

– функція remove– дозволяє видалити поле за його індексом;

– функція insert– дозволяє вставити поле за вказаним індексом;

– функція swap– дозволяє обміняти місцями два поля за їх індексами;

– функція move– дозволяє перемістити поле з одного індексу на інший;

– функція replace– дозволяє замінити весь масив полів новим масивом.

На рис. 2.16 наведено використання даних функцій для динамічного керування полями для отримання інформації про ім’я та прізвище авторів у компоненті BookForm.

```
<form onSubmit={handleSubmit(onSubmit)}>
    {fields.map((item, index) => (<>
        <div className="row my-3 gx-3" key={index}>
            <label className="col-3 col-form-label">Автор</label>
            <div className="col-4">
                <input type="text"
                placeholder="Ім'я та по-батькові автора"
                    className="form-control"
                    {...register(`authors.${index}.firstname`,
                    { required: true })}>
                    </input>
            </div>
            <div className="col-4">
                <input type="text"
                placeholder="Прізвище автора"
                    className="form-control col-3"
                    {...register(`authors.${index}.surname`)}>
                    </input>
            </div>
            <div className="col-1">
                <button className="btn btn-secondary"
                    onClick={() => remove(index)}>
                    <i className="bi bi-x-lg"></i>
                </button>
            </div>
        </div>
    </>))}
    <div className="row my-3 gx-3">
        <div className="col">
            <button className="btn btn-secondary"
                onClick={() => { append({ firstname: "", surname: "" }) }}>
                <i className="bi bi-plus-circle"></i>Автор
            </button>
        </div>
    </div>
```

Рисунок 2.16 – Приклад використання хука useFieldArray() для динамічного керування полями введення інформації про авторів

Даний фрагмент коду призначений для реалізації можливості додавання полів для введення інформації про автора (рис. 2.17).

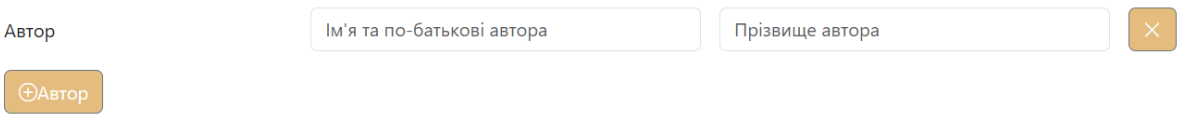

Рисунок 2.17 – Зовнішній вигляд компоненту для динамічного керування полями введення інформації про авторів

Як видно з коду на рис. 2.16, конструкція fields.map() дозволяє створити рядки з полями для введення імені та прізвища кожного автора зі списку поля authors, зареєстрованого на формі; метод append({ firstname: "", surname: "" }) – додати новий рядок; метод remove(index) – видалити існуючий рядок за його індексом.

2.4 Реалізація головної сторінки додатку

На головній сторінці додатку було вирішено розмістити панель з двома вкладками – перша призначена для відображення бібліографічного опису доданих джерел, а друга – для додавання нового посилання на джерело (вибір та відображення форм введення інформації).

Для реалізації вкладок було вирішено використати компоненти Tabs (контейнер вкладок) та Tab (окрема вкладка) бібліотеки react-bootstrap. Було вирішено використати саме бібліотеку з React-компонентами, а не чистий Bootstrap, оскільки при такому варіанті спрощується інтеграція з менеджерами стану (такими як Redux або Redux Toolkit) та реалізація керування вкладками.

На рис. 2.18 наведено код компонента MainTabs, що містить вкладку зі списком джерел (ключ вкладки eventKey=HOMETAB) та вкладку додавання джерела (ключ вкладки eventKey=FORMTAB). Вкладка «Список джерел» містить компонент Main, призначений для відображення

списку доданих користувачем джерел, а вкладка додавання джерела містить компонент FormSwitcher, що відображає форму, яка відповідає типу джерела, який обрав користувач у модальному вікні вибору типу джерела.

src > components > JS MainTabs.js > ...

```
import Main from "./Main";
import "./MainTabs.css";
import { useState } from "react";
import SourceSelectorModal from "./SourceSelectorModal";
import FormSwitcher from "./FormSwitcher";
import { useDispatch, useSelector } from "react-redux";
import { setActiveTab, setSelectedForm } from "../redux/reducers/currentSlice";
import { NONE } from "../data/sourceTypes";
import { FORMTAB, HOMETAB } from "../data/tabNames";
const MainTabs = () => {
    const [showModal, setShowModal] = useState(false);
    const dispatch = useDispatch();
    const selectedSourceType = useSelector(state=>state.current.selectedSourceType);
    const activeTab = useSelector(state=>state.current.activeTab);
    const onConfirm = ()=>{
        dispatch(setSelectedForm(selectedSourceType));
        setShowModal(false);
    }
    const onCancel = ()=>{
        dispatch(setSelectedForm(NONE));
        setShowModal(false);
    }
    const onShowModalClick = () => {
        console.log("Modal starts!");
        setShowModal(true);
    }
    return (
        <>
            <Tabs fill activeKey={activeTab} onSelect={(k)=>dispatch(setActiveTab(k))}>
                <Tab eventKey={HOMETAB} title="Список джерел">
                    <div className="bg-white rounded-bottom p-3">
                        <Main />
                    </div>
                </Tab>
                <Tab eventKey={FORMTAB} title="Додавання джерела">
                    <div className="bg-white rounded-bottom p-3">
                        <FormSwitcher onShowModalClick={onShowModalClick}></FormSwitcher>
                    </div>
                </Tab>
            </Tabs>
            <SourceSelectorModal show={showModal}
                onConfirm={onConfirm} onCancel={onCancel}
                ></SourceSelectorModal>
        </>
    )
}
```

Рисунок 2.18 – JSX-код реалізації компоненту для відображення вкладок на головній сторінці

Компонент Tabs містить властивість activeKey, з використанням якої можна керувати вкладками та встановлювати активну вкладку. Оскільки при закритті компонентів-форм, призначених для введенні інформації про різні типи бібліографічних джерел, необхідно робити активною вкладку зі списком доданих джерел, для зменшення зв’язності коду при передачі стану між компонентами було вирішено застосувати одну з бібліотек, що дозволяють реалізувати керування станом у React-додатках. З метою збереження поточних налаштувань та стану візуальних елементів інтерфейсу додатку було вирішено винести у сховище об’єкт current, властивість activeTab якого відповідатиме за поточну обрану вкладку в компоненті MainTabs.

Також компонент MainTabs повертає компонент <SourceSelectorModal/>, який реалізує модальне вікно вибору типу джерела, яке відкривається після того, як користувач натискає кнопку <AddSourceSwitcher/>, рендер якої відбувається у компоненті <FormSwitcher/> на другій вкладці.

Код компонента SourceSelectorModal наведено на рис. 2.19. Модальне вікно створено на базі компонентів бібліотеки react-bootstrap. Варто відзначити, що до модального вікна через механізм пропсів (props) передаються методи onConfirm та onCancel. Перший метод (onConfirm) призначений для встановлення властивості selectedForm об’єкту current, що зберігається у сховище менеджеру стану. В залежності від встановленого значення, компонент <FormSwitcher/> відображає один з компонентів-форм для введення необхідної інформації для створення посилання на дисертацію (компонент ThesisForm), книгу (компонент BookForm), статтю (ArticleForm) або сайт (SiteForm).

src > components > JS SourceSelectorModal.js > SourceSelectorModal

```jsx
import SourceSelectorList from "./SourceSelectorList";
import Button from 'react-bootstrap/Button';
import Modal from 'react-bootstrap/Modal';

const SourceSelectorModal = (props) => {
    return (
        <>
            <Modal
                {...props}
                size="lg"
                aria-labelledby="contained-modal-title-vcenter"
                centered
            >
                <Modal.Header closeButton>
                    <Modal.Title id="contained-modal-title-vcenter">
                    <span>Тип джерела</span>
                    </Modal.Title>
                </Modal.Header>
                <Modal.Body>
                    <h5>Оберіть тип джерела, яке Ви збираєтеся додати:</h5>
                    <SourceSelectorList />
                </Modal.Body>
                <Modal.Footer>
                    <Button variant="primary"
                    onClick={props.onConfirm}
                    >
                        Обрати
                    </Button>
                    <Button variant="secondary"
                    onClick={props.onCancel}
                    >
                        Скасувати
                    </Button>
                </Modal.Footer>
            </Modal>
        </>
    )
}
```

Рисунок 2.19 – JSX-код реалізації компоненту SourceSelectorModal для відображення модального вікна вибору типу джерела

У тілі модального вікна (дочірній елемент для Modal.Body) відбувається рендер компоненту SourceSelectorList, який завантажує масив sourceTypes з описом основних типів бібліографічних джерел, що використовується у проєкті. Далі за допомогою умовного рендерингу та конструкції sourceTypesList.map() генеруються кнопки, при натисканні на

які відбувається диспетчеризація події (виклик методу dispatch) Redux з встановлення обраного типу літературного джерела (selectedSourceType).

src > components > JS SourceSelectorList.js > SourceSelectorList

```
1  import sourceTypes from "../data/sourceTypes";
2  import { useDispatch, useSelector } from "react-redux";
3  import { setSelectedSourceType } from "../redux/reducers/currentSlice";
4
5  const SourceSelectorList = () => {
6      const sourceTypesList = sourceTypes;
7      const selectedSourceType = useSelector(state => state.current.selectedSourceType);
8      const dispatch = useDispatch();
9      return (
10         <>
11             <div className="list-group">
12                 {
13                     sourceTypesList && sourceTypesList.map(item => (
14                         <button type="button" key={item.id}
15                             className={item.value === selectedSourceType
16                                 ? "list-group-item list-group-item-action active"
17                                 :
18                                 "list-group-item list-group-item-action"
19                             }
20                         // aria-current="true"
21                         onClick={()=>dispatch(setSelectedSourceType(item.value))}
22                         >
23                             {item.title}
24                         </button>
25                     ))
26                 }
27             </div>
28         </>
29     )
30 }
31 export default SourceSelectorList;
```

Рисунок 2.20 – JSX-код реалізації компоненту SourceSelectorList для відображення списку кнопок для вибору типу джерела

На рис. 2.21 наведено код файлу /data/sourceTypes.js з оголошенням констант, що описують типи літературних джерел, а також масив sourceTypes, що використовуються для спрощення генерації кнопок у компоненті SourceSelectorList.

Константа NONE використовується для обробки потенційної помилки, якщо в результаті непередбачуваного збою у роботі програми змінна стану state.current.selectedForm не отримає одне з значень THESIS,

ARTICLE, BOOK або SITE. У такому випадку буде відображатися компонент AddSouceButton, за допомогою якого можна повторно викликати модальне вікно вибору типу джерела.

src > data > JS sourceTypes.js > ...

```
export const THESIS = "Thesis";
export const ARTICLE = "Article";
export const BOOK = "Book";
export const SITE = "Site";
export const NONE = "None";


const sourceTypes = [
    {
        id: 1,
        title: "Стаття",
        value: ARTICLE,
        selected: true
    },
    {
        id: 2,
        title: "Книга",
        value: BOOK,
        selected: false
    },
    {
        id: 3,
        title: "Дисертація",
        value: THESIS,
        selected: false
    },
    {
        id: 4,
        title: "Сайт",
        value: SITE,
        selected: false
    },

]

export default sourceTypes;
```

Рисунок 2.21 – Файл з визначенням констант, що описують типи літературних джерел

Як видно на рис. 2.22, компонент FormSwitcher за допомогою хука бібліотеки Redux useSelector() (виділено червоним) отримує зі сховища значення обраного типу форми (state.current.selectedForm), та у блоці switch перевіряє їх з можливими типами форм, визначеними як константи у файлі /data/sourceTypes.js (рис. 2.21).

src > components > JS FormSwitcher.js > FormSwitcher

```
1  import { ARTICLE, BOOK, SITE, THESIS } from "../data/sourceTypes";
2  import ThesisForm from "./ThesisForm";
3  import SiteForm from "./SiteForm";
4  import BookForm from "./BookForm";
5  import ArticleForm from "./ArticleForm";
6  import AddSourceButton from "./AddSourceButton";
7  import { useSelector } from "react-redux";
8  const FormSwitcher = ({onShowModalClick})=>{
9      const selectedForm = useSelector(state=>state.current.selectedForm);
10     switch(selectedForm){
11         case THESIS:
12             return (<ThesisForm/>)
13         case BOOK:
14             return (<BookForm/> )
15         case ARTICLE:
16             return (<ArticleForm/>)
17         case SITE:
18             return (<SiteForm/>)
19         default:
20         return (<AddSourceButton onShowModalClick={onShowModalClick}/>)
21     }
22 }
23 export default FormSwitcher;
```

Рисунок 2.22 – JSX-код реалізації компоненту FormSwitcher для відображення потрібної форми в залежності від обраного користувачем типу джерела

У результаті на вкладці «Додавання джерела» буде відображатися саме та форма, яка відповідає вибору користувача, зробленому у модальному вікні.

Для обробки введеної користувачем інформації використано метод обробки відправки форми handleSubmit, що повертається хуком useForm (рис. 2.14). У нього потрібно передати метод, який буде виконувати фактичну обробку результату відправки форми (натискання кнопки типу «submit»).

На рис. 2.23 наведено зміни, що були внесені у компонент BookForm, для обробки відправки форми. Реалізовано метод onSubmit, що передається у метод handleSubmit, який встановлено як обробник події відправки форми onSubmit.

Рисунок 2.23 – Приклад JSX-коду реалізації обробки відправки форми у компоненті BookForm

Зокрема, відбувається встановлення типу літературного джерела (у даному випадку – книги) – рядок 41 (data.type = BOOK). У залежності від встановленого типу при відображенні літературного джерела буде використовуватися відповідний компонент.

Також додається інформація, чи є книга частиною багатотомного видання (data.isPart = isPart), та чи потрібно додавати інформацію для опису джерела як доступного онлайн (data.isOnline = isOnline).

Для додавання унікального ключа з метою оптимізації рендерингу списку джерел з використанням функції map() до даних форми також вирішено було додати мітку часу (data.ts = + new Date()).

Також при відправці форми диспетчеризуються 3 події Redux [20]:

– dispatch(addReference(data)) – для додавання отриманих з форми даних до колекції створених користувачем посилань;

– dispatch(setSelectedForm(NONE)) – для приховування форми додавання літературного джерела і відображення кнопки, яка відкриває модальне вікно вибору типу створюваного посилання;

– dispatch(setActiveTab(HOMETAB)) – для переключення на вкладку зі списком створених користувачем бібліографічних посилань («Список посилань»).

Аналогічні зміни були внесені у компоненти форм ArticleForm, ThesisForm, SiteForm.

У результаті переключення на вкладку зі списком посилань буде виконано рендеринг компоненту Main, відповідального за відображення оформлених згідно стандарту ДСТУ 8302:201 бібліографічних посилань.

У коді даного компонента (рис. 2.24) з використанням хука useSelector зі сховища повертається масив доданих через форми з використанням методу addReference() описів посилань (у вигляді об’єктів, що містять основні властивості опису літературного джерела).

Далі за допомогою методу map() формується нумерований список, кожний елемент якого (
) буде відображати коректний бібліографічний опис літературного джерела залежно від його типу (властивості item.type, що приймає одне зі значень констант, визначених у файлі /data/sourceTypes.js).

src > components > JS Main.js > Main > map() callback

```
1   import { useSelector } from "react-redux";
2   import DisplaySwitcher from "./DisplaySwitcher";
3
4   const Main = () => {
5       const references = useSelector(state => state.reference.list);
6       console.log(references);
7       return (
8           <>
9           <h2>Моя бібліографія</h2>
10              <div className="row my-3 gx-3">
11                  {
12                      references && (
13                          <ol className="source-list">{
14                              Array.from(references).map((item) => (
15                                      <li key={item.ts}>
16                                      <DisplaySwitcher {...item}>
17                                      </DisplaySwitcher>
18                                      </li>
19                                  )
20                              )
21                          }</ol>
22                      )
23                  }
24              </div>
25          </>
26      )
27  }
28
29  export default Main;
```

Рисунок 2.24 – JSX-код реалізації компоненту Main для відображення списку доданих користувачем літературних джерел

При рендерингу списку всередині методу map() кожний об’єкт item, що містить основні характеристики доданого літературного джерела, через механізм пропсів (props) передається у компонент DisplaySwitcher з використанням spread-оператора ({…item}).

Як видно на рис. 2.25, компонент DisplaySwitcher у блоці switch перевіряє тип доданого посилання і порівнює з можливими типами форм, визначеними як константи у файлі /data/sourceTypes.js (рис. 2.21). Залежно від типу посилання, обирається один з компонентів для відображення бібліографічного опису.

src > components > JS DisplaySwitcher.js > default

```
1   import { ARTICLE, BOOK, SITE, THESIS } from "../data/sourceTypes";
2   import ArticleDisplay from "./ArticleDisplay";
3   import BookDisplay from "./BookDisplay";
4   import SiteDisplay from "./SiteDisplay";
5   import ThesisDisplay from "./ThesisDisplay";
6
7   const DisplaySwitcher = (item)=>{
8       console.log("Item from DisplaySwitcher 2", item);
9       switch(item.type){
10          case THESIS:
11              return (<ThesisDisplay {...item}/>)
12          case BOOK:
13              return (<BookDisplay {...item}/>)
14          case ARTICLE:
15              return (<ArticleDisplay {...item}/>)
16          case SITE:
17              return (<SiteDisplay {...item}/>)
18          default:
19          return (<h2>Not Found</h2>)
20      }
21  }
22  export default DisplaySwitcher;
```

Рисунок 2.25 – JSX-код реалізації компоненту DisplaySwitcher для відображення потрібного компоненту в залежності від обраного користувачем типу джерела

Компоненти для відображення бібліографічного опису конкретного типу літературного джерела використовують введену користувачем інформацію та умовний рендеринг для формування потрібного формату опису. На рис. 2.26 наведено приклад реалізації відображення бібліографічного опису сайту.

src > components > JS SiteDisplay.js > SiteDisplay > item.authors.map() callback

```
const SiteDisplay = (item) => {
    const accessDate = new Date(item.accessDate);
    return (
        <>
            {
                item.authors && item.authors.length < 4 &&
                item.authors.map((author, index, arr) => (
                    <span key={index}>
                        {author.surname}{" "}
                        {author.firstname.length > 0 ? author.firstname[0] : ""}
                        {"."}
                        {index < (arr.length - 1) ? ", " : " "}
                    </span>
                ))
            }
            {item.title}
            {
                item.authors && item.authors.length > 3 &&
                (<>
                 {" / "}
                {item.authors[0].firstname.length > 0 ?
                item.authors[0].firstname[0] : ""}
                {". "}
                {item.authors[0].surname}
                {" та ін"}
                </>)
            }
            {". "}
            <i>{item.siteTitle}</i>
            {". URL: "}
            <a href={item.URL}>{item.URL}</a>
            {" (дата звернення: "}
            {`${accessDate.getDate()}.${accessDate.getMonth()}.
            ${accessDate.getYear() + 1900}`}
            {")."}
        </>
    )
}
```

Рисунок 2.26 – JSX-код реалізації компоненту SiteDisplay для відображення бібліографічного опису сайту

Аналогічним чином реалізовано компонент ArticleDisplay для відображення бібліографічного опису статті, BookDisplay – для відображення бібліографічного опису книги та ThesisDisplay – для відображення бібліографічного опису дисертації.

2.5 Управління станом додатку

2.5.1 Обгрунтування та вибір бібліотеки для управління станом додатку

Оскільки у розробленому додатку використовується достатньо велика кількість компонентів, які мають активно обмінюються інформацією для реалізації ефективної взаємодії, то для зменшення взаємної зв’язності компонентів та усунення ефекту props-drilling доцільно використати бібліотеку керування станом додатку, а стан винести у окреме сховище.

Тому було вирішено провести аналіз популярних на поточний момент бібліотек для керування станом клієнтських додатків на JavaScript. Нижче наведено огляд основних варіантів реалізації, що розглядалися.

1. Вбудований React Context разом з хуком useReducer [21]:

– переваги: вбудований у React, простий та зручний для використання у невеликих додатках;

– недоліки: не оптимізований для великих додатків через часті перерендери.

– типова сфера застосування: для простих або середніх проєктів без потреби в складному управлінні станом.

2. Одна з найпопулярніших та найбільш відомих бібліотек – Redux:

– переваги: сувора структура, багата екосистема, підтримка за допомогою інтеграції інструментів розробника у браузер (DevTools);

– недоліки: великий обсяг шаблонного коду, складність навчання.

– типова сфера застосування: для великих і складних додатків із необхідністю прогнозованого управління станом.

3. MobX [21, 22]:

– переваги: простота використання, реактивність, мінімум коду;

– недоліки: можлива складність у відстеженні змін через автоматичну реактивність;

– типова сфера застосування: для проєктів, де потрібна гнучкість і автоматична реактивність.

4. Recoil:

– переваги: гарна інтеграція з React, атомарний підхід до стану, зручний SSR;

– недоліки: Відносно новий, ще формується екосистема.

– типова сфера застосування: для середніх і великих додатків із потребою в модульному управлінні станом.

5. Zustand:

– переваги: Мінімалізм, простий API, відсутність шаблонного коду.

– недоліки: Обмежена кількість плагінів і розширень.

– типова сфера застосування: для невеликих та середніх проєктів, де важлива простота і продуктивність.

Результати проведеного аналізу наведено у таблиці 2.2.

Проаналізувавши основні переваги і недоліки розглянутих бібліотек, вирішено прийняти ключовими наступні критерії: продуктивність, гнучкість, можливість валідації стану, наявність спільноти. Фактично доцільно обирати або між використанням Redux, або бібліотеками MobX чи Zustand [23, 24]. З урахуванням прийнятих пріоритетів та для власного професійного розвитку, було вирішено зупинитися на варіанті з Redux.

Водночас, наразі рекомендованим варіантом використання Redux у проєктах на React є бібліотека Redux Toolkit [25].

Таблиця 2.2 – Порівняльний аналіз найпопулярніших менеджерів стану для React

Характеристика	React Context + useReducer	Redux	MobX	Recoil	Zustand
Основна концепція	Побудова глобального стану за допомогою вбудованих хуків (`Context`, `useReducer`)	Уніфікований глобальний стан із використанням action та reducers	Реактивні стани на основі observable	Дерево атомів (стани) і селекторів для гнучкого управління станом	Просте управління станом через store без складних структур
Легкість у навчанні	Середня (потрібно розуміння хуків і контексту)	Складна (потребує розуміння основ Flux, middleware)	Легка (менше шаблонного коду)	Легка (подібна до React Context)	Дуже легка (мінімум шаблонного коду)
Кількість коду	Середня (менше шаблонного коду, але все ще потребує ручного налаштування)	Велика (потрібні actions, reducers, store)	Мала (завдяки реактивності)	Середня (через атоми та селектори)	Мала (мінімум обгорток і конфігурацій)
Продуктивність	Низька/Середня (через часті перерендери контексту)	Висока (добре налаштована мемоізація)	Висока (мінімальні перерендери через реактивність)	Висока (гнучке управління атомами)	Висока (тільки необхідні компоненти перерендерюються)
Гнучкість	Середня (підходить для простих проєктів)	Висока (підтримка складних архітектур)	Висока (автоматична реактивність)	Висока (легко розподіляти стани)	Середня/Висока (проста інтеграція)
Валідація стану	Вручну через логіку компонентів	Легка через middleware	Вручну або через спостереження	Автоматично через атоми	Легка через простий API
Підтримка серверного рендерингу (SSR)	Складна, потрібно передавати початковий стан	Можлива, але потребує додаткової конфігурації	Проста, автоматично підтримує SSR	Вбудована підтримка	Проста, зазвичай без додаткових налаштувань
Спільнота та екосистема	Велика, завдяки вбудованості у React	Дуже велика, активне середовище	Менша, але лояльна спільнота	Середня, зростає популярність	Менша, але швидко зростає

Фактично, Redux Toolkit (або RTK) – це сучасний і спрощений спосіб використання Redux з меншим обсягом коду та більшою продуктивністю, що використовує бібліотеку Immer для забезпечення незмінності («іммутабельності») об’єктів у сховищі [25]. Спільнотою та документацію React рекомендується використовувати RTK для нових проєктів, щоб уникнути зайвої складності класичного Redux. У таблиці 2.3 наведено порівняльний аналіз бібліотек Redux та Redux Toolkit.

Таблиця 2.3 – Порівняльний аналіз бібліотек Redux та Redux Toolkit

Критерій	Redux	Redux Toolkit (RTK)
Встановлення	Потрібно встановлювати кілька пакетів вручну (redux, react-redux, middleware)	Все в одному пакеті (@reduxjs/toolkit), включає redux і react-redux
Кількість коду	Багато шаблонного коду (actions, reducers, types)	Менше коду завдяки createSlice, createAsyncThunk та вбудованим інструментам
Налаштування	Вимагає ручного налаштування store, middleware	Просте налаштування через configureStore(), автоматично додає middleware
Іммутативність	Потрібно вручну забезпечувати іммутативність (наприклад, через spread оператор)	Вбудований Immer для автоматичної роботи з іммутативними змінами
Асинхронність	Необхідно вручну налаштовувати middleware для асинхронних запитів	Вбудована підтримка асинхронних дій через createAsyncThunk
DevTools	Потрібна додаткова конфігурація для інтеграції з DevTools	DevTools автоматично налаштовані через configureStore()
Крива навчання	Складніша через велику кількість шаблонного коду	Простіша завдяки зменшенню шаблонності та вбудованим рішенням

Отже, для реалізації проєкту програмного забезпечення для створення бібліографічного опису літературних джерел було вирішено використати бібліотеку Redux Toolkit.

2.5.2 Реалізація управління станом у розроблюваному програмного забезпеченні

Для збереження доданих користувачем елементів бібліографії у сховищі було створено слайс referenceSlice (рис. 2.27), що у початковому стані містить перелік об’єктів-посилань на літературні джерела (властивість list).

src > redux > reducers > JS referenceSlice.js > default

```
1   import { createSlice, current } from "@reduxjs/toolkit";
2
3   const initialState = {
4       list: [],
5       value: 0
6   }
7   const referenceSlice = createSlice({
8       name: "reference",
9       initialState,
10      reducers: {
11          addReference: (state, action)=>{
12              console.log(current(state));
13              console.log("New item added", action.payload);
14              state.list = [...state.list, action.payload];
15          }
16      }
17  })
18
19  export const {addReference} = referenceSlice.actions;
20
21  const referenceReducer = referenceSlice.reducer;
22  export default referenceReducer;
```

Рисунок 2.27 – Код реалізації слайсу для керування сховищем елементів бібліографічного опису

До редюсерів слайсу referenceSlice додали функцію addReference, яка використовується для додавання елементу бібліографії до колекції list сховища.

Другим слайсом розробленого додатку є слайс currentSlice, який потрібен для керування станом візуальної частини додатку – для збереження інформації про:

– обрану вкладку (змінна activeTab);

– форму, яку потрібно відображати на вкладці додавання джерела (змінна selectedForm);

– обраний тип джерела у модальному вікні вибору типу джерела (компонент SourceSelectorModal) до натискання кнопки «Додати джерело», яка відкриває потрібну обрану форму (змінна selectedSourceType).

На рис. 2.28 наведено код реалізації слайсу currentSlice. Як видно, в коді визначено початковий стан вказаних змінних у сховищі, а також визначено методи-редюсери: setSelectedForm() – для вибору форми, що буде відображатися; setSelectedSourceType() – для вибору та відображення активного типу джерела у модальному вікні; setActiveTab() – для встановлення активної вкладки. Дані методи використовуються для оновлення сховища. Як видно з наведеного на рис. 2.28 коду, за рахунок того, що Redux Toolkit використовує бібліотеку Immer, суттєво спрощується реалізація оновлення стану сховища у порівнянні з базовою версією бібліотеки Redux, і код стає більш лаконічним.

Файл /redux/reducers/currentSlice.js експортує вказані вище дії для подальшого використання при диспетчеризації подій, а також редюсер currentReducer для подальшого використання при створенні сховища (з використанням експорту за замовчуванням).

src > redux > reducers > JS currentSlice.js > ...

```
1   import { createSlice } from "@reduxjs/toolkit";
2   import  { ARTICLE,  NONE } from "../../data/sourceTypes";
3   import { HOMETAB } from "../../data/tabNames";
4
5   const initialState = {
6       selectedForm: NONE,
7       selectedSourceType: ARTICLE,
8       activeTab: HOMETAB
9   }
10  const currentSlice = createSlice({
11      name: "current",
12      initialState,
13      reducers: {
14          setSelectedForm: (state, action)=>{
15              state.selectedForm = action.payload;
16          },
17          setSelectedSourceType: (state, action)=>{
18              state.selectedSourceType = action.payload;
19          },
20          setActiveTab: (state, action)=>{
21              state.activeTab = action.payload;
22          }
23      }
24  })
25
26  export const {
27      setSelectedForm,
28      setSelectedSourceType,
29      setActiveTab
30  } = currentSlice.actions;
31
32  const currentReducer = currentSlice.reducer;
33  export default currentReducer;
```

Рисунок 2.28 – Код реалізації слайсу для керування станом візуальної частини додатку

На рис. 2.29 наведено код реалізації файлу /redux/store/index.js, в якому з використанням функції configureStore() створюється об’єкт сховища шляхом поєднання редюсерів referenceReducer та currentReducer.

src > redux > store > JS index.js > ...

```
1   import { configureStore } from "@reduxjs/toolkit";
2   import referenceReducer from "../reducers/referenceSlice";
3   import currentReducer from "../reducers/currentSlice";
4
5   const store = configureStore({
6       reducer: {
7           reference: referenceReducer,
8           current: currentReducer
9       }
10  });
11
12  export default store;
```

Рисунок 2.29 – Код визначення об’єкту сховища Redux

Для візуалізації архітектури додатку, що використовує Redux Toolkit, побудуємо діаграму компонентів (Component Diagram) UML [9], що відображатиме структуру додатку та взаємозв’язки між основними компонентами, такими як сховище (Store), редюсери (Reducers/Slices), дії (Actions), та елементи інтерфейсу користувача (UI-компоненти).

На рис. 2.30 наведено діаграму компонентів для опису використаного підходу до керування станом у розробленому додатку з використанням бібліотеки Redux Toolkit.

Як видно з діаграми, сховище (store) використовує configureStore для об’єднання двох ред’юсерів:

– referenceSlice для роботи зі списком посилань;

– currentSlice для управління вкладками, відображуваними формами та типом джерел.

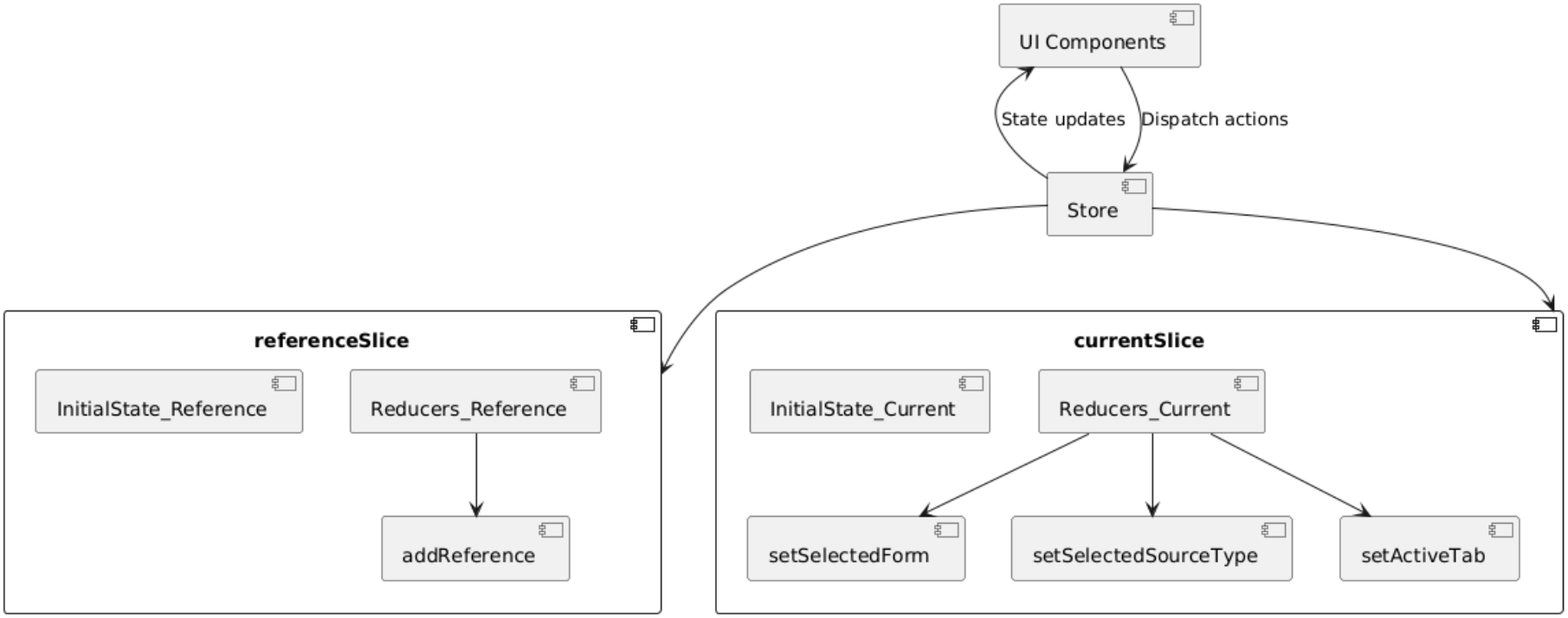

Рисунок 2.30 – Діаграма компонентів UML для опису використаного підходу до керування станом додатку

Відповідний код опису діаграми компонентів в синтаксисі PlantUML [9] наведено на рис. 2.31.

components.puml > { } components

```
@startuml

    [Store] --> [referenceSlice]
    [Store] --> [currentSlice]

    component referenceSlice {
        [InitialState_Reference]
        [Reducers_Reference]
        [addReference]
    }

    component currentSlice {
        [InitialState_Current]
        [Reducers_Current]
        [setSelectedForm]
        [setSelectedSourceType]
        [setActiveTab]
    }



    [Reducers_Reference] --> [addReference]
    [Reducers_Current] --> [setSelectedForm]
    [Reducers_Current] --> [setSelectedSourceType]
    [Reducers_Current] --> [setActiveTab]

    [UI Components] -up-> [Store] : Dispatch actions
    [Store] -up-> [UI Components] : State updates
@enduml
```

Рисунок 2.31 – Опис діаграми компонентів реалізації керування станом додатку у синтаксисі PlantUML

На рис. 2.32 наведено діаграму діяльності UML, що описує типовий порядок дій та взаємодії між компонентами Redux Toolkit при додаванні нового

бібліографічного джерела.

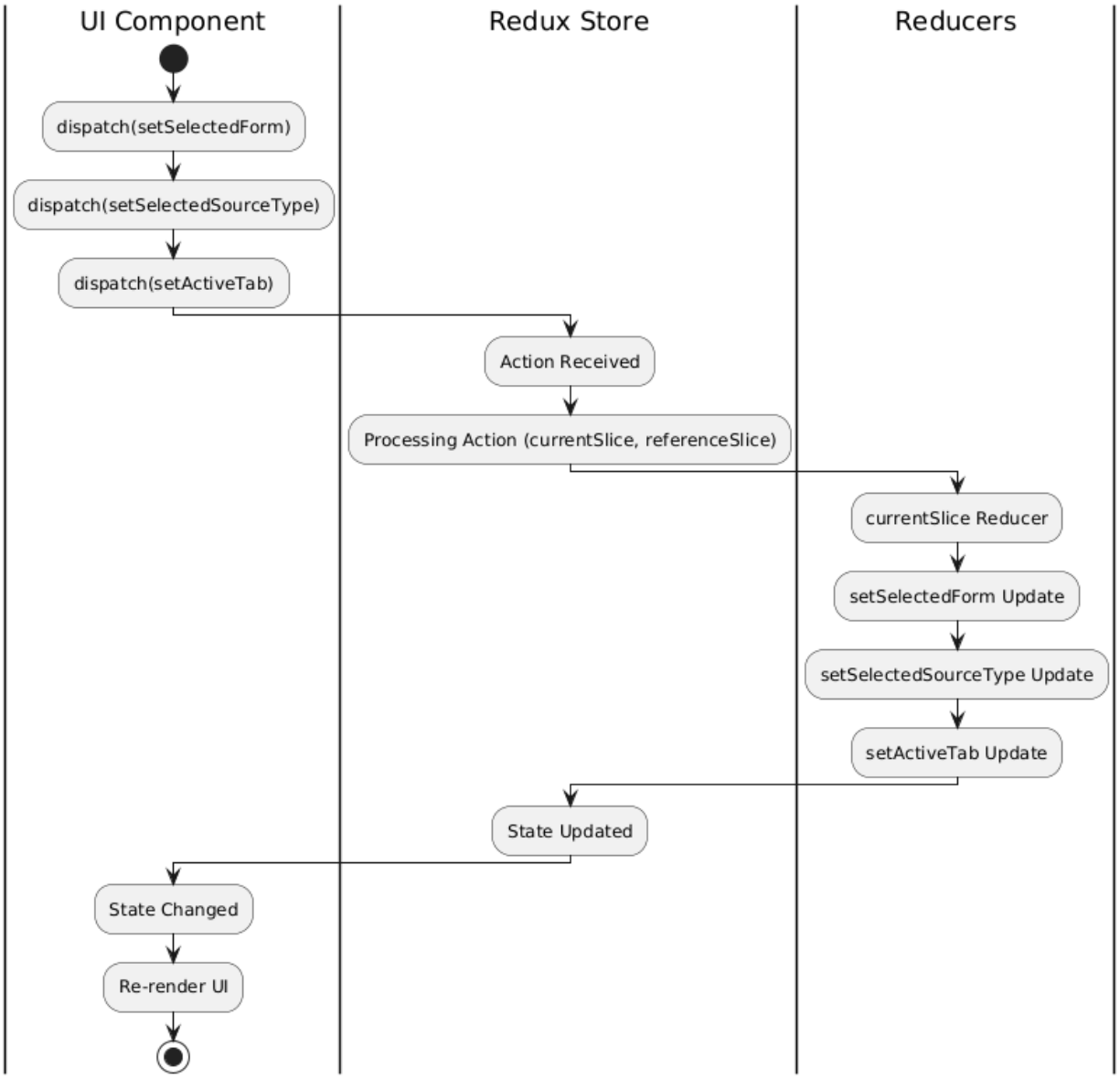

Рисунок 2.32 – Діаграма діяльності UML для опису використаного підходу до керування станом додатку

Як видно з діаграми на рис. 2.32, компоненти інтерфейсу при необхідності додати новий елемент бібліографії відправляють дію (action) до сховища Redux (store) з використанням методу dispatch.

Сховище Redux (store) отримує дію (action) та передає його до відповідного редюсера (reducer).

У редюсері здійснюється оновлення стану для кожного поля (setSelectedForm, setSelectedSourceType, setActiveTab), зокрема додається елемент бібліографії, встановлюється активна вкладка, відбувається встановлення типу форми та обраного типу джерела (скидання у початкові значення).

Після оновлення сховища (store), UI-компонент отримує новий стан і здійснює перерендер.

Компоненти інтерфейсу надсилають дії (actions) до сховища (store), яке оновлює відповідний стан.

Таким чином, з використанням бібліотеки Redux Toolkit у проєкті програмного забезпечення для створення бібліографічного опису літературних джерел було реалізовано ефективне керування станом, за рахунок чого вдалося суттєво зменшити рівень зв’язності між окремими компонентами та оптимізувати структуру проєкту на React.

Висновки до розділу:

У цьому розділі було розглянуто вибір технологій і методів для реалізації веб-сервісу з оформлення бібліографічних посилань відповідно до стандарту ДСТУ 8302:2015.

На етапі вибору та обґрунтування технологій визначено, що React є оптимальним інструментом для побудови клієнтської частини сервісу завдяки його модульності, високій продуктивності та багатій екосистемі. Інтеграція популярних бібліотек, таких як Redux для управління станом, React Hook Form для обробки форм та Bootstrap для верстки, значно розширила функціональність додатку, спростила процес розробки та покращила його візуальне оформлення.

На етапі проєктування інтерфейсу було враховано вимоги до зручності використання. Для реалізації форм, що дозволяють додавати або редагувати бібліографічні посилання, було визначено набір обов’язкових полів на основі стандарту ДСТУ 8302:2015. Візуальний дизайн і функціональність сторінок

сервісу були адаптовані під потреби користувачів, а для натхнення використано найкращі практики, такі як дизайн інтерфейсу Grafiati.

Також у розділі проаналізовано використання бібліотек Redux і Redux Toolkit для управління станом додатку. Було створено UML-діаграми компонентів і діяльності, що ілюструють підхід до роботи з даними та їх обробки в системі. Це забезпечило чітке розуміння процесів передачі інформації між компонентами та управління станом у реальному часі.

Таким чином, вибір технологій, проєктування інтерфейсу та інструментів для обробки форм дозволили створити ефективний, гнучкий і зручний для користувачів веб-сервіс, який відповідає сучасним вимогам до програмних продуктів. Отримані результати забезпечують основу для подальшої оптимізації і розвитку системи.

РОЗДІЛ 3
ПРАКТИЧНА АПРОБАЦІЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ДЛЯ ОФОРМЛЕННЯ БІБЛІОГРАФІЧНИХ ПОСИЛАНЬ У ВІДПОВІДНОСТІ ДО ДСТУ 8302:2015

3.1 Практична апробація та опис методики використання веб-сервісу для оформлення бібліографічних посилань у відповідності до ДСТУ 8302:2015

Для демонстрації функціональності розробленої системи, призначеної для автоматизованого оформлення бібліографічних посилань відповідно до вимог ДСТУ 8302:2015, необхідно розглянути послідовність операцій і завдань, які система виконує. Це дозволить оцінити зручність її використання, ефективність автоматизації процесів і відповідність стандартам.

Детальний аналіз функціональних можливостей розробленого програмного забезпечення продемонструє, як автоматизація рутинних завдань, таких як заповнення шаблонів, перевірка правильності даних і форматування, може полегшити роботу користувачів, підвищуючи точність і продуктивність.

Першим кроком користувач має зареєструватися в системі. Для цього він проходить процедуру реєстрації та виконує вхід через особистий кабінет користувача на веб-платформі (рис. 3.1).

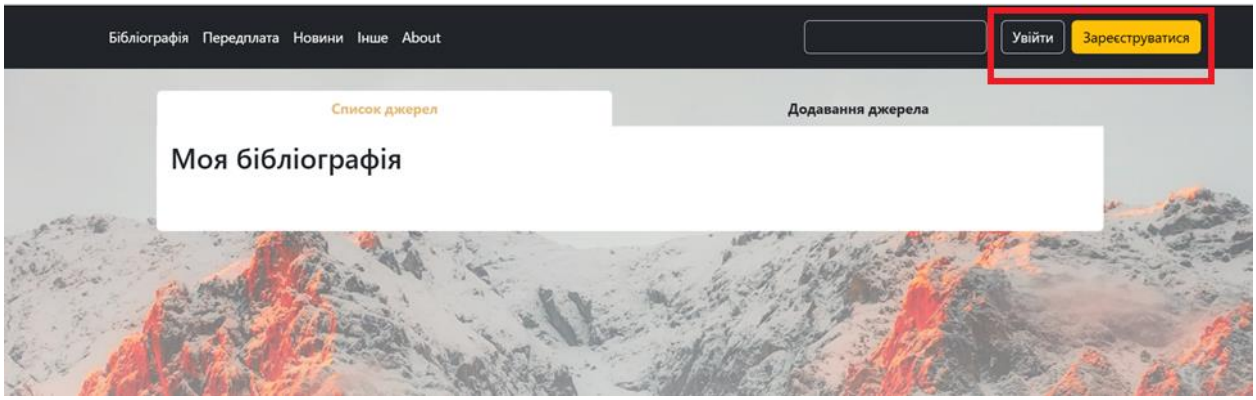

Рисунок 3.1 – Вхід у систему особистого кабінету користувача

Для створення бібліографічного посилання спочатку оберіть тип джерела з доступного списку в системі. При натисканні на кнопку «Нове джерело» можна вибрати книгу, статтю, веб-ресурс або інший вид документа, залежно від вашої потреби. Після вибору система автоматично адаптує необхідні поля для введення даних відповідно до вимог ДСТУ 8302:2015. Це забезпечує точність та відповідність стандарту при оформленні посилання (рис. 3.2).

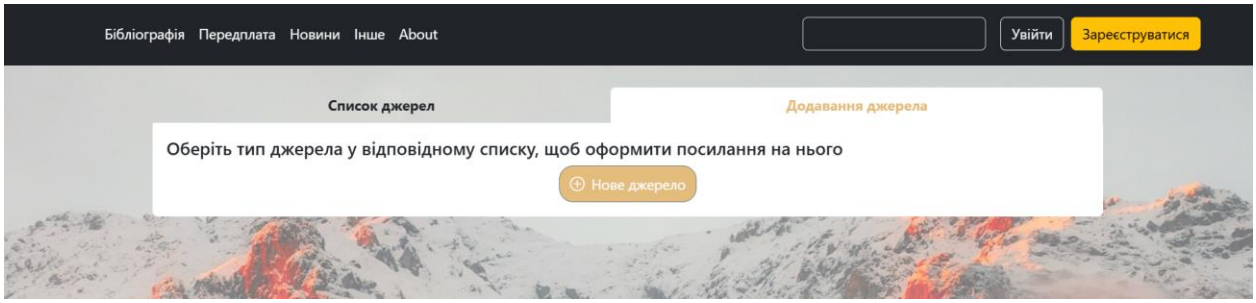

Рисунок 3.2 – Створення бібліографічного джерела

Тип джерела визначає вид матеріалу, на який створюється бібліографічне посилання, і є основним параметром для його правильного оформлення. У системі доступні такі типи джерел, як книги, статті, веб-ресурси, дисертації, нормативні документи тощо. Вибір типу джерела дозволяє системі адаптувати форму введення даних та правила їхнього оформлення відповідно до вимог ДСТУ 8302:2015. Це допомагає уникнути помилок та автоматизує створення посилань, які відповідають стандарту (рис. 3.3).

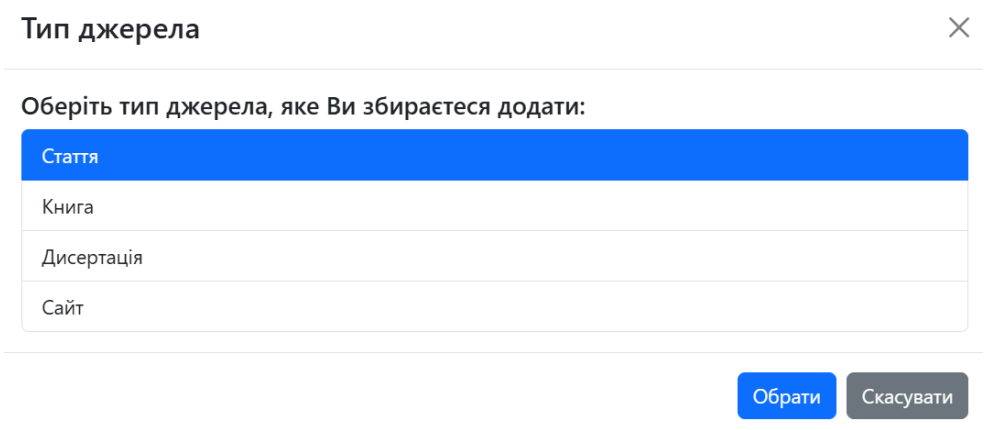

Рисунок 3.3 – Тип бібліографічного джерела

Система підтримує можливість додавання кількох авторів для бібліографічного посилання на статтю. У відповідному полі введення даних користувач може додати імена всіх авторів у послідовності, визначеній вимогами ДСТУ 8302:2015. При цьому система автоматично адаптує формат посилання відповідно до кількості авторів, враховуючи правильне розташування їхніх ініціалів та прізвищ. Це забезпечує точне та коректне оформлення статті з декількома співавторами (рис. 3.4).

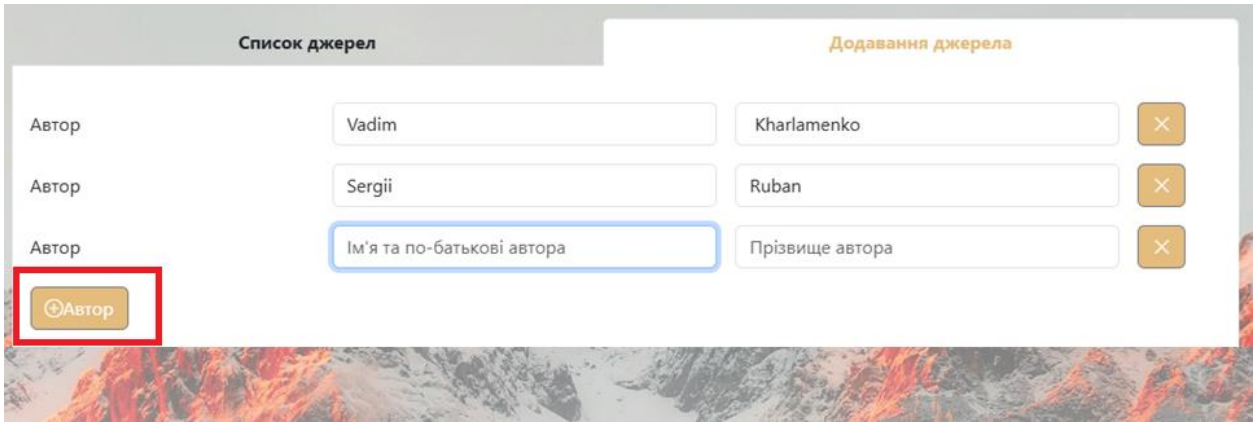

Рисунок 3.4 – Додавання кількох авторів для бібліографічного посилання на статтю

Система підтримує функцію автоматичного додавання онлайн-джерел за допомогою посилань DOI або URL. Для цього користувач може вказати у спеціальному полі унікальний ідентифікатор (DOI) або веб-адресу (URL). Це значно спрощує процес створення посилання, забезпечує його коректність та відповідність ДСТУ 8302:2015, особливо для наукових статей та електронних ресурсів (рис. 3.5).

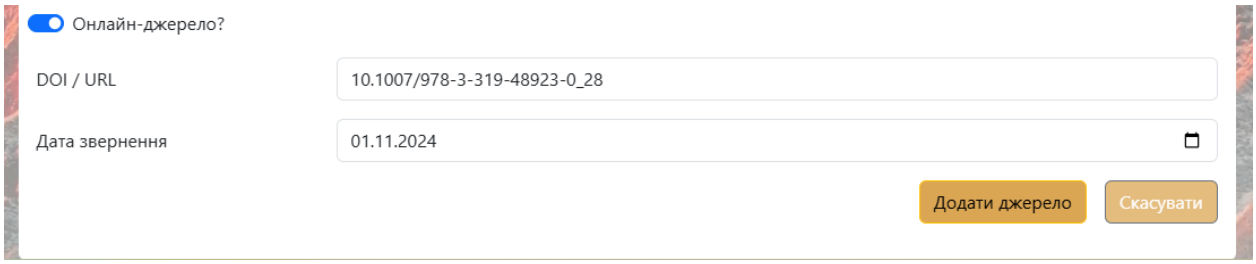

Рисунок 3.5 – Функція додавання онлайн-джерел

Після заповнення всіх необхідних даних для бібліографічного посилання система автоматично створює джерело, оформлене відповідно до вимог ДСТУ 8302:2015. Введена інформація перевіряється на коректність, і посилання формується у стандартизованому вигляді, що відповідає обраному типу джерела. Це гарантує відповідність посилання затвердженим нормам та виключає можливість помилок форматування (рис. 3.6).

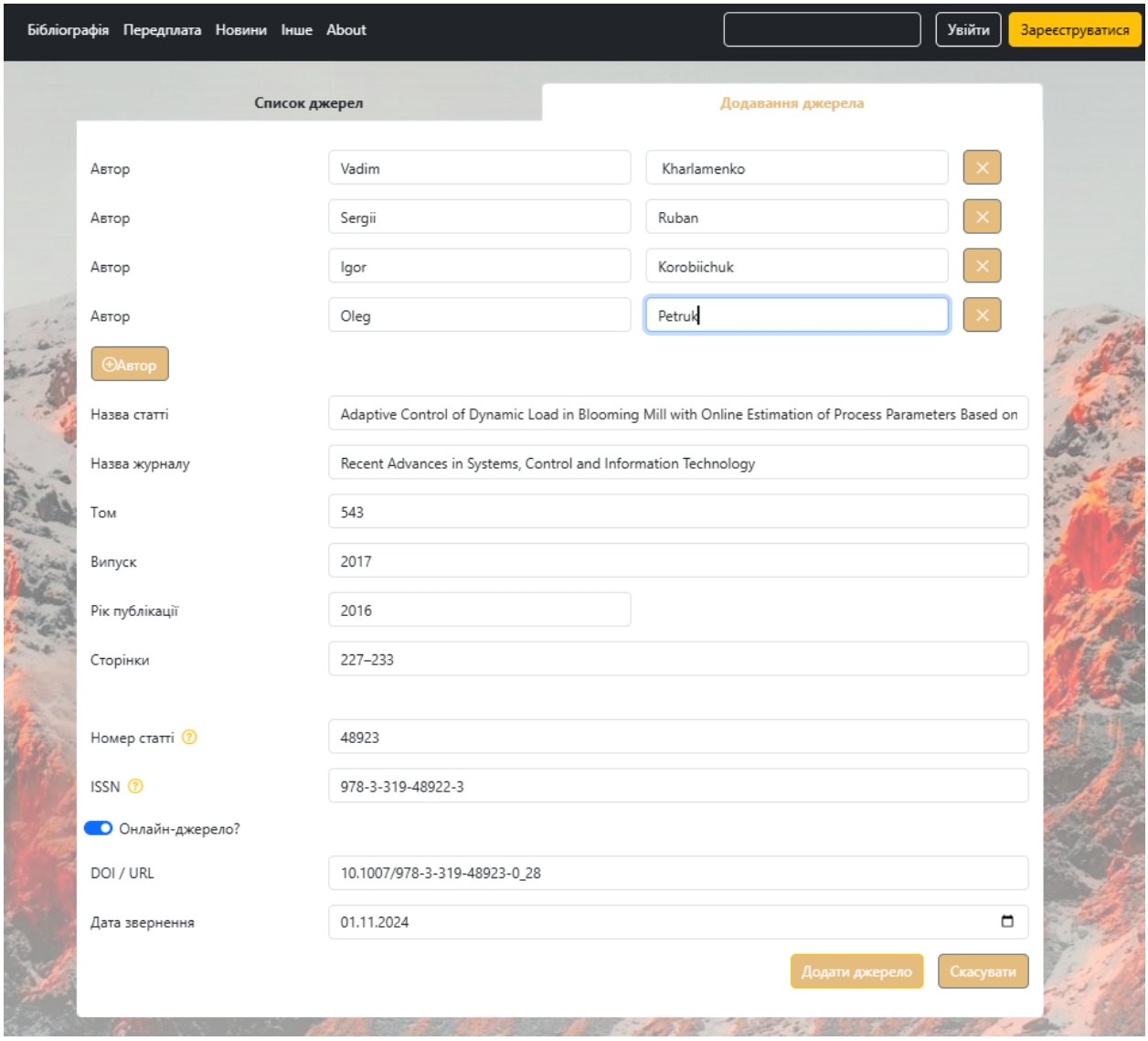

Рисунок 3.6 – Додавання бібліографічного джерела стаття

Після створення бібліографічного посилання, стаття додається до списку джерел на головній сторінці веб-сервісу. Це забезпечує зручність у роботі з бібліографією та точність оформлення бібліографічного посилання (рис. 3.7).

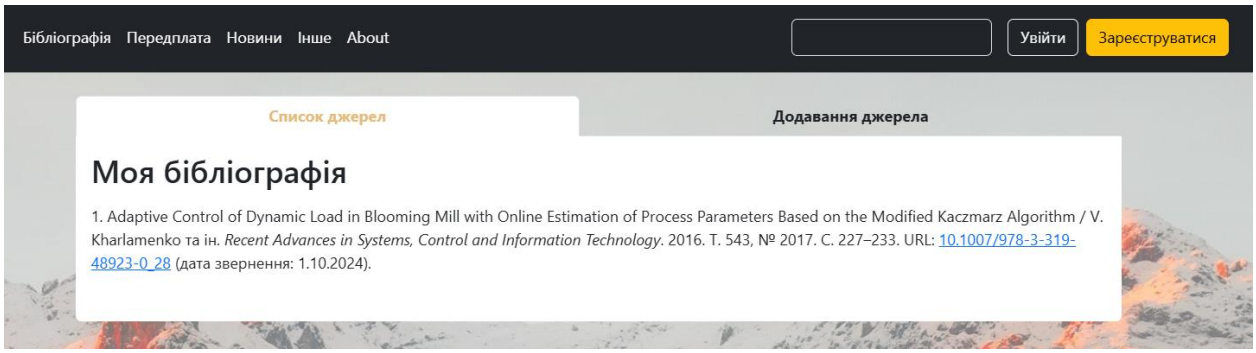

Рисунок 3.7 – Список джерела на головній сторінці веб-сервісу

Веб-сервіс надає можливість оформлювати бібліографічні посилання для різних типів джерел, так, на рисунку 3.8 представлено процес додавання підручника. Для кожного типу джерела передбачено спеціальні шаблони, що відповідають вимогам ДСТУ 8302:2015, із необхідними полями для заповнення. Це дозволяє легко створювати коректні посилання незалежно від виду джерела.

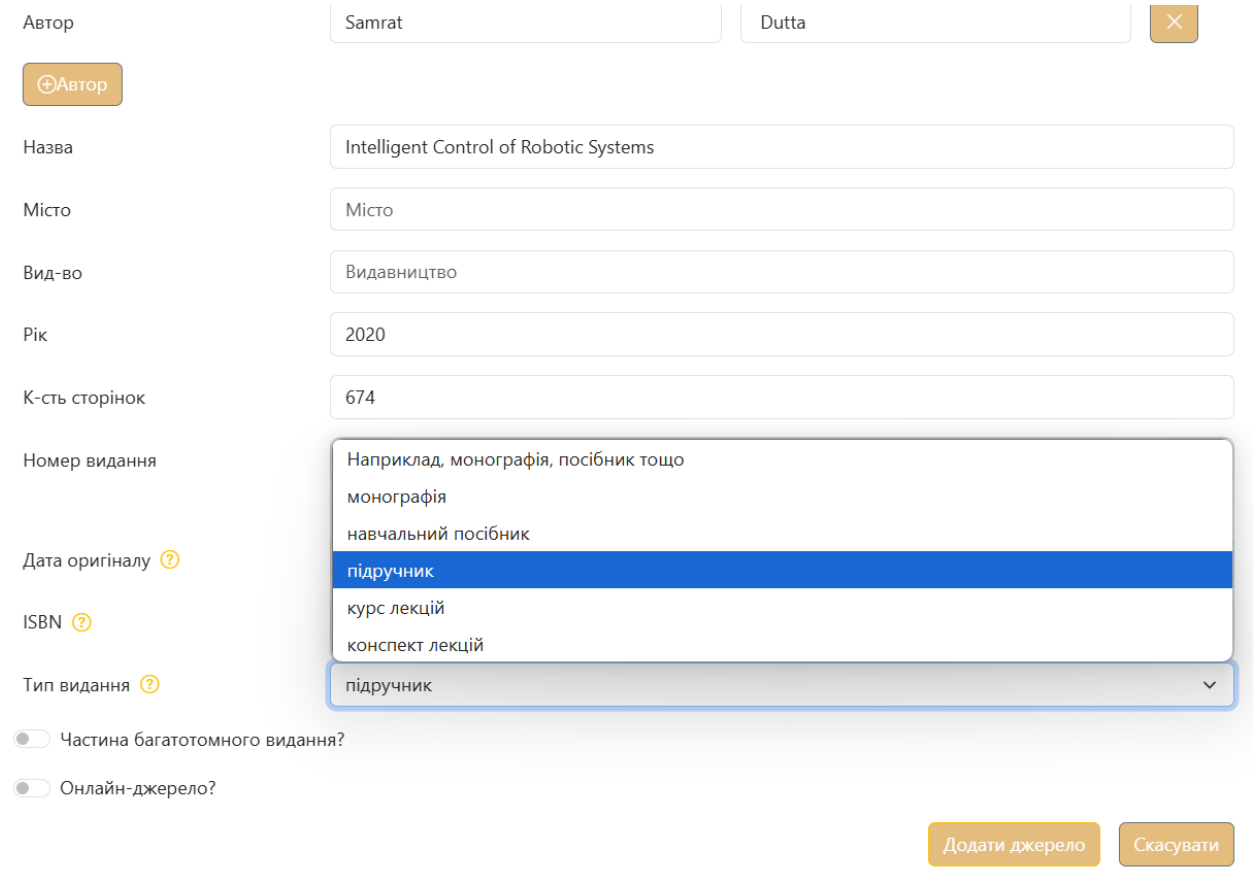

Рисунок 3.8 – Оформлення бібліографічного посилання на прикладі підручника

Веб-сервіс дозволяє створювати бібліографічні посилання для підручників, дотримуючись стандарту ДСТУ 8302:2015. Для цього потрібно обрати тип джерела «Підручник», заповнити поля, такі як автори, назва, рік видання, місце видання, видавництво та кількість сторінок та інші. Після збереження посилання автоматично формується у правильному форматі та може бути додане до списку джерел (рис. 3.9).

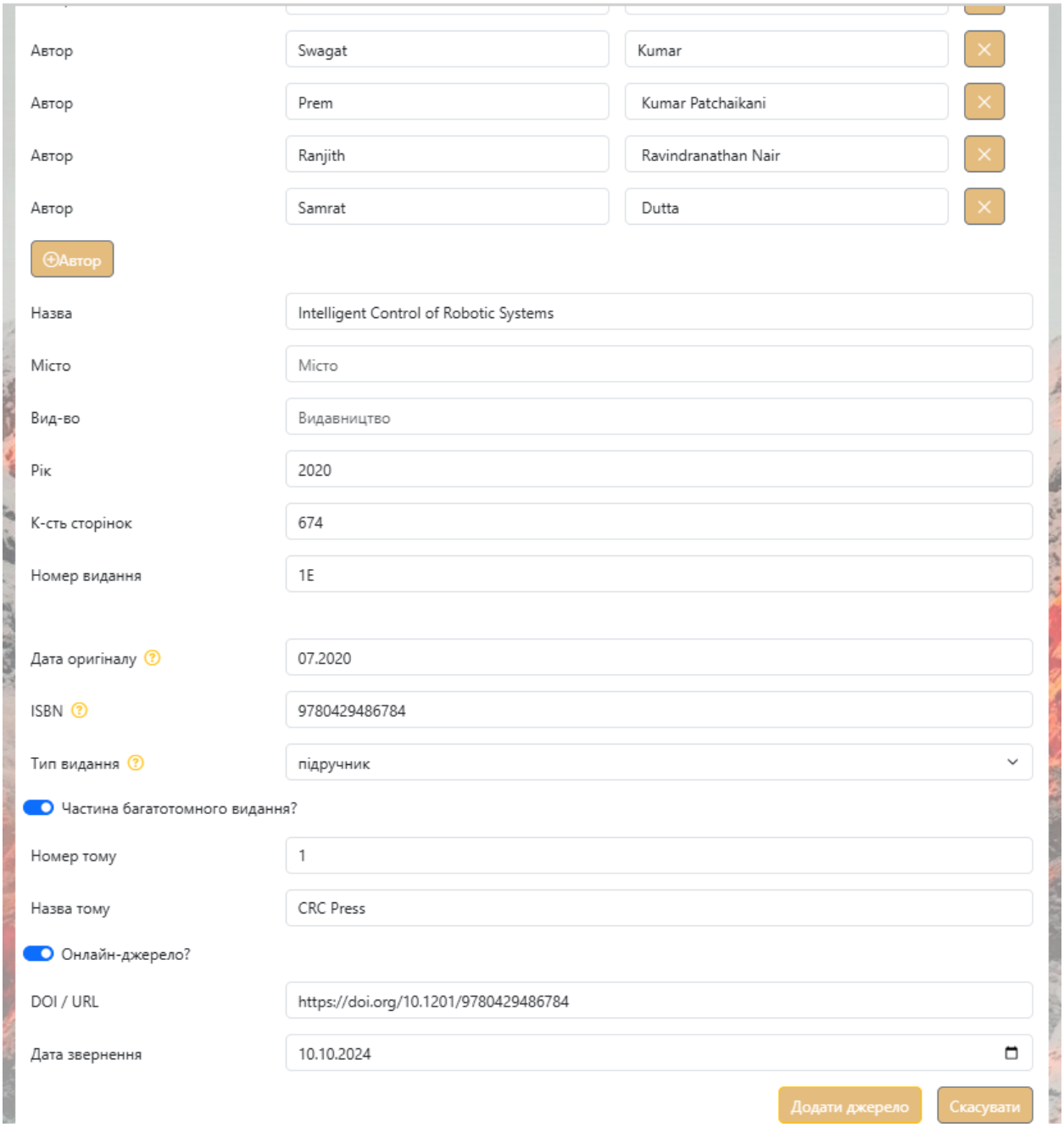

Рисунок 3.9 – Додавання бібліографічного джерела «Підручник»

Реалізовано перевірку правильності введення даних у поля з датами. Під час заповнення полів, таких як рік видання чи дата публікації, система автоматично перевіряє відповідність введеного значення формату (наприклад, лише числові значення для років або повний формат «день.місяць.рік»). У разі помилки користувач отримає підказку з поясненням, як правильно заповнити поле. Це гарантує коректність оформлення бібліографічного посилання відповідно до стандартів (рис. 3.10).

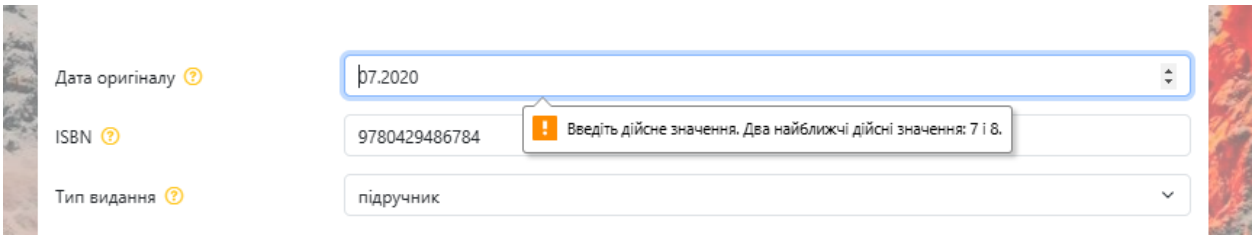

Рисунок 3.10 – Перевірка введення даних

На головній сторінці системи відображається список усіх доданих користувачем бібліографічних джерел. Кожне джерело представлено у вигляді окремого запису з короткою інформацією, такою як назва, автори, рік публікації та тип джерела та інше (рис. 3.11).

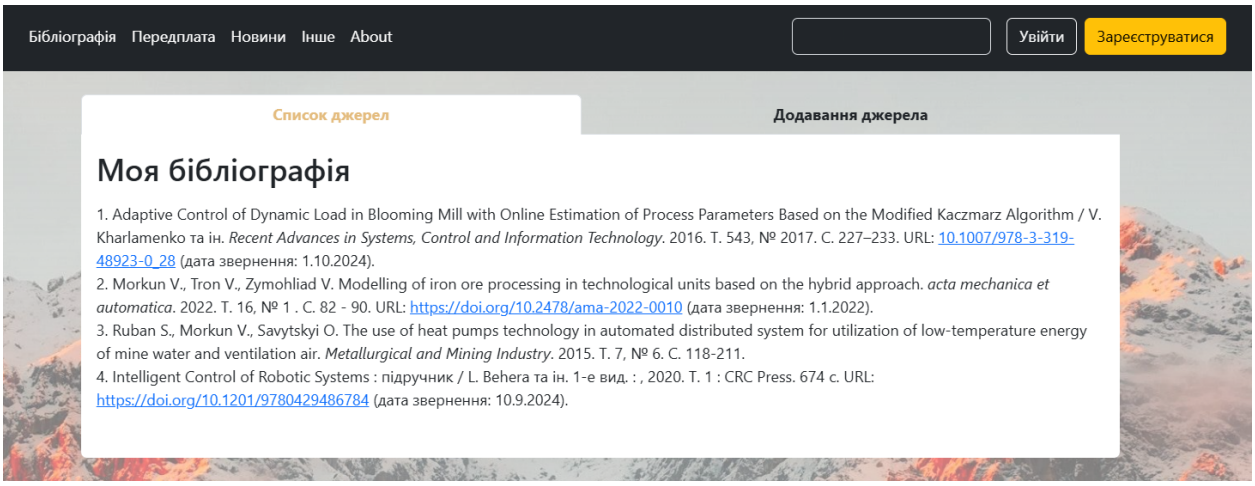

Рисунок 3.11 – Головна сторінка веб-сервісу

Висновки до розділу:

У даному розділі випускної роботи проведено перевірку розробленого веб-сервісу для оформлення бібліографічних посилань у відповідності до ДСТУ 8302:2015.

Проведена апробація показала, що веб-сервіс успішно вирішує основні завдання, пов’язані з автоматизацією процесу створення бібліографічних посилань. Методика використання системи включає зручний покроковий процес: реєстрація, створення нового джерела, заповнення даних, перевірка правильності оформлення. Реалізація таких можливостей, як автоматичне додавання джерел через DOI/URL, підтримка роботи з кількома авторами, а також функція редагування й управління списком джерел, забезпечує високу функціональність і гнучкість системи.

Веб-сервіс дозволяє не лише автоматизувати рутинні операції, а й значно підвищити точність оформлення посилань відповідно до вимог ДСТУ 8302:2015. Усі процеси виконуються у зрозумілому інтерфейсі, що забезпечує легкість освоєння навіть для новачків.

Результати тестування підтвердили стабільність роботи сервісу, його відповідність стандартам і здатність забезпечити потреби широкого кола користувачів – від студентів і викладачів до науковців і фахівців, які працюють із великими обсягами бібліографічних даних. Система дозволяє:

- значно скоротити час на створення бібліографічних списків;
- забезпечити коректне оформлення згідно з державними стандартами;
- зменшити ризик помилок за рахунок автоматизації перевірки даних.

Впровадження розробленого веб-сервісу в навчальну, наукову та професійну практику сприятиме підвищенню якості роботи з бібліографічними джерелами, спрощенню підготовки академічних та професійних документів, а також покращенню загальної культури оформлення бібліографії. Веб-сервіс є сучасним і ефективним інструментом, що відповідає актуальним вимогам у сфері інформаційних технологій і роботи з джерелами.

ВИСНОВКИ

У дипломній роботі на тему «Розробка веб-сервісу для оформлення бібліографічних посилань у відповідності до ДСТУ 8302:2015» виконано комплексне дослідження, розробку та апробацію інформаційної системи, що забезпечує зручне та точне формування бібліографічних посилань.

У першому розділі було проведено огляд існуючих сервісів для оформлення бібліографічних посилань, які показали обмеженість функціональності та складність роботи з деякими з них. На основі аналізу було визначено сучасні технології, які найкраще підходять для реалізації веб-сервісу, та сформульовано функціональні і нефункціональні вимоги до системи. Це забезпечило ґрунтовну основу для подальшої розробки платформи.

Другий розділ присвячено проєктуванню та реалізації основних компонентів сервісу. Було обґрунтовано вибір технологій, таких як React для клієнтської частини, Firebase для роботи з базою даних, і Bootstrap для верстки. Розроблено інтерфейс користувача та здійснено верстку основних сторінок, включаючи головну сторінку та форми для додавання бібліографічних посилань. Реалізовано обробку форм, що забезпечує автоматичну перевірку введених даних, синхронізацію інформації між компонентами та відповідність стандарту ДСТУ 8302:2015. Управління станом додатку було реалізовано за допомогою Redux Toolkit, що забезпечує ефективну та надійну роботу системи навіть за значного навантаження.

У третьому розділі проведено практичну апробацію розробленого веб-сервісу. Було продемонстровано його функціональні можливості та описано методику використання, яка включає процес реєстрації, створення джерел, редагування даних та отримання результатів у потрібному форматі. Тестування системи підтвердило її відповідність функціональним і нефункціональним вимогам, зокрема, зручність використання, швидкодію та коректність роботи.

Результати дослідження показали, що розроблений веб-сервіс дозволяє суттєво полегшити процес створення бібліографічних посилань, забезпечує високу точність і відповідність вимогам стандарту. Впровадження такої системи може бути корисним для широкого кола користувачів, зокрема студентів, викладачів, науковців і редакторів. Робота створює основу для подальшого вдосконалення платформи, зокрема, розширення функціоналу, інтеграції з іншими системами та впровадження нових інструментів для автоматизації роботи.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

-

1. Приклади оформлення бібліографічного опису відповідно до ДСТУ 8302:2015 – Бібліотека К-ПНУ. *Бібліотека К-ПНУ – Кам'янець-Подільський національний університет ім. Івана Огієнка*. URL: https://library.kpnu.edu.ua/?p=989 (дата звернення: 09.09.2024).

2. Генератор посилань за ДСТУ 8302:2015 – grafiati. Grafiati: *Оформити списки використаних джерел онлайн*. URL: https://www.grafiati.com/uk/blogs/dstu-8302-2015-referencing-generator (дата звернення: 10.09.2024).

3. Cite this for me: harvard, APA, MLA reference generator. *Cite This For Me: Harvard, APA, MLA Reference Generator*. URL: https://www.citethisforme.com (дата звернення: 15.09.2024).

4. Zotero | Your personal research assistant. *Zotero | Your personal research assistant*. URL: https://www.zotero.org/ (дата звернення: 15.09.2024).

5. Mendeley - reference management software. *Mendeley - Reference Management Softw*are. URL: https://www.mendeley.com/ (дата звернення: 15.09.2024).

6. BibGuru - A new FREE APA, harvard, & MLA citation generator. *Bibguru*. URL: https://www.bibguru.com (дата звернення: 15.09.2024).

7. Teresius J. API vs MVC–which one to choose?. *Medium*. URL: https://medium.com/@justinast/api-vs-mvc-which-one-to-choose-69570c51785a (дата звернення: 20.09.2024).

8. GeeksforGeeks. Differences between Web API and MVC - GeeksforGeeks. *GeeksforGeeks*. URL: https://www.geeksforgeeks.org/differences-between-web-api-and-mvc/ (дата звернення: 20.09.2024).

9. Plant UML. URL: https://plantuml.com// (дата звернення: 22.09.2024).

10. React. *React*. URL: https://react.dev/ (дата звернення: 24.09.2024).

11. Починаючи | Axios Docs. *Axios*. URL: https://axios-http.com/uk/docs/intro (дата звернення: 24.09.2024).

12. Get started with Bootstrap. *Bootstrap. The most popular HTML, CSS, and JS library in the world*. URL: https://getbootstrap.com/docs/5.3/getting-started/introduction/ (дата звернення: 24.09.2024).

13. React Router Official Documentation. React Router Official Documentation. URL: https://reactrouter.com/ (дата звернення: 24.09.2024).

14. GitHub - softchris/react-book: Free book on React. Beginner to intermediate. *GitHub*. URL: https://github.com/softchris/react-book (дата звернення: 24.09.2024).

15. Banks A., Porcello E. Learning React: Functional Web Development with React and Redux. O'Reilly Media, 2017. 350 p.

16. Gackenheimer C. Introduction to React. Berkeley, CA : Apress, 2015. URL: https://doi.org/10.1007/978-1-4842-1245-5 (дата звернення: 24.09.2024).

17. SurviveJS. URL: https://survivejs.com/books/react/getting-started/?utm_source=devfreebooks&utm_medium=medium&utm_campaign=DevFreeBooks (дата звернення: 07.10.2024).

18. Free React JS Book. Free Programming Books – *GoalKicker.com*. URL: https://books.goalkicker.com/ReactJSBook/ (дата звернення: 11.10.2024).

19. React : up and Running: Building Web Applications. O'Reilly Media, Incorporated, 2021. 222 p. URL: https://dl.ebooksworld.ir/books/React.Up.and.Running.2nd.Edition.Stoyan.Stefanov.OReilly.9781492051466.EBooksWorld.ir.pdf/ (дата звернення: 13.10.2024).

20. Redux - A JS library for predictable and maintainable global state management | Redux. *Redux - A JS library for predictable and maintainable global state management | Redux*. URL: https://redux.js.org (дата звернення: 16.10.2024).

21. 7 Best React State Management Libraries for Any Project Size | Relia Software. *Relia Software: Trusted Partner for Custom Software Development Services*. URL: https://reliasoftware.com/blog/react-state-management-libraries (дата звернення: 18.10.2024).

22. Phát H. React State Management in 2024. *DEV Community*. URL: https://dev.to/nguyenhongphat0/react-state-management-in-2024-5e7l (дата звернення: 18.10.2024).

23. 18 Best React State Management Libraries in 2025. *2025 Web Development Roadmap & Trends*. URL: https://fe-tool.com/awesome-react-state-management (дата звернення: 18.10.2024).

24. A Complete Guide to React State Management - eTatvaSoft. *eTatvaSoft*. URL: https://www.etatvasoft.com/blog/react-state-management/ (дата звернення: 21.10.2024).

25. Redux Toolkit | Redux Toolkit. Redux Toolkit | Redux Toolkit. URL: https://redux-toolkit.js.org/ (дата звернення: 05.11.2024).

26. Моркун Н. В., Маринич І. А. Методичні рекомендації до виконання кваліфікаційної роботи бакалавра для студентів спеціальності 122 - “Комп’ютерні науки”. Кривий Ріг, Видавничий центр ДВНЗ «КНУ». 2017.

27. ДСТУ 3008:2015. Інформація та документація. Звіти у сфері науки і техніки. Структура і правила оформлення. Київ, ДП «УкрННЦ», 2015. 26с. (Інформація та документація).

28. ДСТУ 8302:2015. Бібліографічне посилання. Загальні вимоги та правила складання Київ, ДП «УкрННЦ», 2016. 16 с. (Інформація та документація).

29. ДСТУ 3582:2013. Бібліографічний опис. Скорочення слів і словосполучень в українській мові. Загальні вимоги та правила. Київ, ДП «УкрННЦ», 2013. 23 с. (Інформація та документація).

30. ДСТУ 3651.0-97 Метрологія. Одиниці фізичних величин. Основні одиниці фізичних величин Міжнародної системи одиниць. Основні положення, назви та позначення Київ, Держстандарт України, 1998. 27 с. (Інформація та документація).