Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра програмного забезпечення

Бакалаврська дипломна робота

на тему: «Розробка методу та засобів реалізації автоматизованої вебсистеми управління бібліотекою»

Виконав: студент IV курсу
групи ІПІ-20б
спеціальності
121 – Інженерія програмного забезпечення
(шифр і назва напряму підготовки, спеціальності)
Булавко І. О.
(прізвище та ініціали)

Керівник: к.т.н., доц. каф. ПЗ Войтко В. В.
(прізвище та ініціали)

Рецензент: к.т.н., доц. каф. ОТ Тарновський М. Г.
(прізвище та ініціали)

Допущено до захисту
Зав. кафедри _____
«10» червня 2024 р.

ВНТУ – 2024

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення
Рівень вищої освіти перший (бакалаврський)
Галузь знань 12 – «Інформаційні технології»
Спеціальність 121 – «Інженерія програмного забезпечення»
Освітньо-професійна програма – «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
Романюк О. Н.
«12» березня 2024 р.

З А В Д А Н Н Я
НА БАКАЛАВРСЬКУ ДИПЛОМНУ РОБОТУ СТУДЕНТУ

Булавку Івану Олександровичу

1. Тема роботи – «Розробка методу та засобів реалізації автоматизованої вебсистеми управління бібліотекою».

Керівник роботи: Войтко Вікторія Володимирівна, к.т.н., доцент кафедри ПЗ, затверджені наказом вищого навчального закладу від 11 березня 2024 р. №80.

2. Термін подання студентом роботи: 3 червня 2024 року.

3. Вихідні дані до роботи: середовище розробки – Visual Studio Code, система управління базами даних – Firebase; мова запитів – NoSQL; мови розробки – HTML, CSS, JavaScript/TypeScript; фреймворк SvelteKit; операційна система – Windows 8/10/11.

4. Зміст розрахунково-пояснювальної записки: вступ; аналіз розвитку автоматизованих вебсистем управління бібліотекою та постановка задач дослідження; розробка методу, моделі та алгоритмів роботи вебсистеми; розробка програмних модулів вебсистеми; тестування програми; висновки; список використаних джерел; додатки; ілюстративна частина.

5. Перелік ілюстративного матеріалу: титульний слайд – автор, наук керівник бакалаврської дипломної роботи, тема; мета, об'єкт та пре дослідження; задачі дослідження; наукова новизна; практична цінність отрим результатів; блок-схема загального алгоритму роботи вебсистеми; блок-с алгоритму керування обліковими записами; блок-схема алгоритму тестува uml-діаграми; тестування вебсистеми; апробація та публікації результатів роб впровадження; фінальний слайд.

6. Консультанти розділів бакалаврської дипломної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдан прийня
1 – 4	Войтко В. В., к.т.н., доцент кафедри ПЗ	13.03.24	03.06.2

7. Дата видачі завдання 13 березня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської дипломної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз розвитку автоматизованих вебсистем управління бібліотекою та постановка задач дослідження	14.03.24 – 22.03.24	виконано
2	Розробка методу, моделі та алгоритмів роботи вебсистеми	22.03.24 – 20.04.24	виконано
3	Розробка програмних модулів вебсистеми	20.04.24 – 03.05.24	виконано
4	Тестування програми	03.05.24 – 23.05.24	виконано
5	Оформлення матеріалів до захисту БДР	23.05.24 – 30.05.24	виконано

Студент (підпис) **Булавко І. О.** (прізвище та ініціали)

Керівник бакалаврської дипломної роботи (підпис) **Войтко В. В.** (прізвище та ініціали)

АНОТАЦІЯ

Бакалаврська дипломна робота складається зі 60 сторінок формату А4, на яких є 44 рисунків, 3 таблиці, список використаних джерел містить 21 найменування.

У бакалаврській дипломній роботі було проведено аналіз сфери автоматизованих вебсистем управління бібліотекою. Проведено аналіз аналогів, визначено їх переваги і недоліки та доведено актуальність розробки власного вебсервісу.

Розроблена вебсистема дозволяє налаштувати управління бібліотекою шляхом автоматизації процесу бронювання обраних користувачем творів та керування підбором літератури. Розроблено метод формування і відображення звітності вебсистеми.

Вебсистему було реалізовано на мові програмування JavaScript/TypeScript, каскадних стилях CSS та мові гіпертекстової розмітки HTML. Середовищем розробки було обрано Visual Studio Code Для відображення інформативної звітності використано бібліотеку Google Charts.

У результаті виконання бакалаврської дипломної роботи отримано автоматизовану вебсистему, що підвищить ефективність процесу керування бібліотекою та сприятиме зацікавленості відвідувачів користуватися послугами бібліотеки.

Отримані в бакалаврській дипломній роботі результати можна використати для покращення процесу автоматизації бібліотеки.

Ключові слова: вебсистеми, бронювання літератури, онлайн бібліотека.

ABSTRACT

The bachelor’s thesis consists of 60 pages of A4 format, which have 44 drawings, 3 tables, the list of used sources contains 21 denominations

In the bachelor's thesis, an analysis of the field of automated web-based library management systems was carried out. An analysis of analogues was carried out, their advantages and disadvantages were determined, and the relevance of developing one's own web service was proven.

The developed web system allows you to configure library management by automating the process of booking works selected by the user and managing the literature status. A method of graphical display of web system reporting has been developed.

The web system was written in the JavaScript/TypeScript programming language, CSS cascading styles, and HTML hypertext markup language. Visual Studio Code was chosen as the development environment. The Google Charts library was used to display informative reporting.

As a result of the bachelor thesis, an automated web system was obtained, which will increase the efficiency of the library management process and promote the interest of visitors in using the library services.

The results obtained in the bachelor thesis can be used to improve the library automation process.

Keywords: web systems, literature reservation, online library.

.

ЗМІСТ

ВСТУП 3
1 АНАЛІЗ РОЗВИТКУ АВТОМАТИЗОВАНИХ ВЕБСИСТЕМ УПРАВЛІННЯ БІБЛІОТЕКОЮ ТА ПОСТАНОВКА ЗАДАЧ ДОСЛІДЖЕННЯ 7
– 1.1 Аналіз стану автоматизованих вебсистем управління бібліотекою 7
– 1.2 Порівняльний аналіз аналогів 8
– 1.3 Аналіз методів розв’язання задачі 12
– 1.4 Постановка задач для розробки автоматизованої вебсистеми управління бібліотекою 13
– 1.5 Висновки 13
2 РОЗРОБКА МЕТОДУ, МОДЕЛІ ТА АЛГОРИТМІВ РОБОТИ ВЕБСИСТЕМИ 14
– 2.1 Аналіз вхідних даних системи 14
– 2.2 Розробка моделі системи 16
– 2.3 Розробка методу формування і відображення звітності 18
– 2.4 Розробка загального алгоритму роботи застосунку 21
– 2.5 Висновки 23
3. РОЗРОБКА ПРОГРАМНИХ МОДУЛІВ ВЕБСИСТЕМИ 24
– 3.1 Варіантний аналіз і обґрунтування вибору засобів для реалізації програми 24
– 3.2 Розробка засобів відображення графічного інтерфейсу 25
– 3.3 Розробка модуля створення звітності 32
– 3.4 Розробка модуля бронювання літератури 36
– 3.5 Розробка інтерфейсу програмного додатку 39
– 3.6 Висновки 45
4 ТЕСТУВАННЯ ПРОГРАМИ 46
– 4.1 Тестування системи 46
– 4.2 Розробка інструкції користувача 52
– 4.3 Висновки 56
ВИСНОВКИ 57
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 59
ДОДАТКИ 61
– Додаток А – Технічне завдання 62
– Додаток Б –Протокол перевірки бакалаврської дипломної роботи на наявність текстових запозичень 66
– Додаток В – Лістинг програми 67
– Додаток Г – Ілюстративна частина 90

ВСТУП

Обґрунтування вибору теми дослідження. У сучасному інформаційному суспільстві бібліотеки відіграють ключову роль у зберіганні, організації та поширенні знань. Вони є не лише місцем зберігання книг, а й важливим соціальним та культурним інститутом. Із ростом обсягів інформації та потреб користувачів зростає і необхідність в ефективному управлінні бібліотечними ресурсами та послугами [1].

З розвитком обсягів споживаної інформації та зростаючим попитом на інноваційні технології, популярною тенденцією останніх років стала розробка автоматизованих вебсистем. Це зумовило потребу в програмних модулях, які можуть бути використані для реалізації таких систем.

Спеціалізовані програмні модулі можуть допомогти розробникам створювати вебсистеми, які дозволяють користувачам спросити досягнення поставлених цілей [2].

Автоматизовані системи управління бібліотеками значно полегшують процес обліку та видачі книг, а також управління каталогами та резервуванням літератури.

Це дозволяє зменшити навантаження на бібліотекарів та підвищити загальну ефективність роботи бібліотеки.

Наявні аналогічні системи з використанням автоматизованого підходу не надають достатньо досить потужного та корисного функціоналу. Бронювання літератури з допомогою онлайн запитів є менш поширеними навіть серед потужних засобів. До того ж, зазвичай потужні засоби є платними.

Тому актуальною є розробка власної автоматизованої вебсистеми для управління бібліотекою.

Зв’язок роботи з науковими програмами, планами, темами. Робота виконувалася згідно плану виконання наукових досліджень на кафедрі програмного забезпечення.

Мета та завдання дослідження. Метою дослідження є покращення обслуговування користувачів бібліотеки та ведення обліку за рахунок розробки та використання спеціалізованої вебсистеми, яка забезпечує підвищення рівня автоматизації керування наявною літературою, спрощення процесів комунікації користувачів при бронюванні книг та візуалізацію звітності, що полегшує процеси аналізу та проведення статистичних досліджень і формування звітів.

Задачі дослідження:

– розробити модель автоматизованої вебсистеми управління бібліотекою, яка міститиме основний функціонал системи;

– розробити метод формування і відображення звітності;

– розробити алгоритм бронювання доступної літератури з допомогою онлайн запиту;

– розробити зручний та адаптивний графічний інтерфейс вебсистеми;

– розробити програмне забезпечення вебсистеми;

– реалізувати модуль статичних досліджень для створення аналітичної статистики вебсистеми;

– здійснити тестування вебсистеми відповідно до поставлених задач.

Об’єкт дослідження – процес розробки автоматизованої вебсистеми управління бібліотекою.

Предмет дослідження – методи та засоби реалізації автоматизованої вебсистеми управління бібліотекою.

Методи дослідження. У дослідженні було використано методи:

– методи і технології модульної розробки вебсистем для програмної реалізації модулів системи;

– методи візуалізації даних результатів звітності для забезпечення отримання інформативної аналітики роботи вебсистеми;

– методи UI/UX для створення та перевірки інтерфейсів вебсистеми;

– методи тестування для перевірки працездатності роботи програми.

Новизна отриманих результатів:

1. Подальшого розвитку отримав метод формування і відображення звітності про роботу бібліотеки, який, на відміну від існуючих, базується на графічній візуалізації даних у вигляді діаграм, що спрощує порівняльний аналіз статистичної інформації за обраний період та сприяє своєчасному прийняттю управлінських рішень і покращенню процесів управління бібліотекою.

2. Подальшого розвитку набула модель автоматизованої вебсистеми управління бібліотекою, яка, на відміну від існуючих, базується на використанні розширеного функціоналу і реалізації візуальних методів графічного подання звітності, що забезпечує можливість проводити статистичний аналіз даних та забезпечує автоматизацію процесів систематизації обліку літератури.

Практична цінність отриманих результатів. Практична цінність отриманих результатів полягає у можливості використання розробленої вебсистеми для автоматизації процесів управління бібліотекою, зокрема, дозволяє користувачам отримати всю необхідну інформацію щодо шуканої літератури, бронювання літератури, відстеження успішності процесів управління.

Особистий внесок здобувача. Усі наукові результати, викладені у бакалаврській дипломній роботі, отримані автором особисто. У науковій праці [3], опублікованій у співавторстві, автору належать такі результати: таблиця порівняння функціональних характеристик аналогів, алгоритм роботи програми.

Апробація матеріалів бакалаврської дипломної роботи. Матеріали бакалаврської дипломної роботи доповідалися на конференції: «LIII Науково-технічна конференція факультету інформаційних технологій та комп'ютерної інженерії (2024)».

Публікації. Результати роботи були опубліковані в матеріалах конференції[3] – LIII Науково-технічна конференція факультету інформаційних технологій та комп'ютерної інженерії (2024).

Структура та обсяг роботи. Бакалаврська дипломна робота складається зі вступу, чотирьох розділів, висновків, списку використаних джерел, що містить 21 найменування, 4 додатків. Робота містить 44 ілюстрації та 3 таблиці.

1 АНАЛІЗ РОЗВИТКУ АВТОМАТИЗОВАНИХ ВЕБСИСТЕМ УПРАВЛІННЯ БІБЛІОТЕКОЮ ТА ПОСТАНОВКА ЗАДАЧ ДОСЛІДЖЕННЯ

1.1 Аналіз стану автоматизованих вебсистем управління бібліотекою

Зі стрімким розвитком технологій вебсистеми стають дедалі популярнішими в різних галузях. Автоматизація цих технологій наразі має найвищий пріоритет у цій галузі, що надає нові можливості для швидкої та зручної взаємодії. Розробка програмних модулів для реалізації вебистеми з використанням автоматизації є ключовим завданням у цьому напрямі, адже дозволяє створювати інноваційні рішення, які дозволяють користувачам більш ефективно використовувати бібліотечні ресурси та послуги.

Однією з основних переваг використання автоматизації у вебсистемах управління бібліотекою є можливість включати в себе різноманітні функції, такі як пошук та бронювання літератури, отримання рейтингу від інших користувачів, віртуальні візуалізація аналітики. Ця технологія також успішно використовується і в різних сферах, таких як виробництво, логістика, освіта, торгівля тощо. Однак, процес автоматизації є надзвичайно важким і вимагає багато часу разом із відточеними навиками [4].

Для вирішення цього завдання багато вебсистем звертаються до спеціалізованих розробників програмного забезпечення, які мають досвід у розробці вебсистем для бібліотек та інших культурних установ. Вони можуть надати власні рішення, що відповідають конкретним потребам бібліотеки та її аудиторії. Такий підхід дозволяє заощадити час та ресурси, забезпечуючи при цьому якість та надійність вебсистеми управління бібліотекою.

Інтеграція інтерактивних елементів у вебсистеми управління бібліотекою має ряд переваг. Окрім покращення сервісу для користувачів, вона дозволяє бібліотеці збільшити свою привабливість для аудиторії та підвищити її задоволеність від використання послуг [5]. Такий підхід також може сприяти залученню нових

користувачів та підвищенню інтересу до читання та культурно-освітньої діяльності.

Отже, інтеграція інтерактивних елементів у вебсистеми управління бібліотекою є важливим напрямком розвитку сучасних бібліотечних технологій. Використання досвіду та компетенції спеціалізованих розробників дозволяє створювати інноваційні рішення, які покращують якість обслуговування та надають конкурентні переваги у сфері культурних послуг. Для досягнення успіху у цьому напрямку важливо враховувати тенденції розвитку вебтехнологій та потреби користувачів, щоб забезпечити актуальність та конкурентоспроможність вебсистеми управління бібліотекою.

1.2 Порівняльний аналіз аналогів

Автоматизація стає все більш поширеною у сучасних вебсистемах. Розробка програмних модулів для автоматизації процесів управління бібліотекою може значно спростити рутинні завдання бібліотекарів, покращити обслуговування користувачів та збільшити продуктивність бібліотеки в цілому. Наразі найбільш використовуваними рішеннями є:

1. ALEPH.
2. UniLib.
3. KOHA.

ALEPH є однією з провідних систем управління бібліотекою, яка надає широкий функціонал для автоматизації бібліотечних процесів [6]. Вона включає в себе налаштовувані модулі для каталогізації книг, управління позичанням та поверненням книг, а також забезпечує доступ до каталогу для користувачів через вебінтерфейс (рис. 1.1). Повна підтримка Юнікоду забезпечує багатонаправлені та багатострокові текстові можливості, з допомогою котрих користувачі можуть взаємодіяти з системою на обраній мові. Використання ALEPH може бути досить дорогим, особливо для невеликих бібліотек. Витрати на ліцензію та обслуговування можуть бути значними.

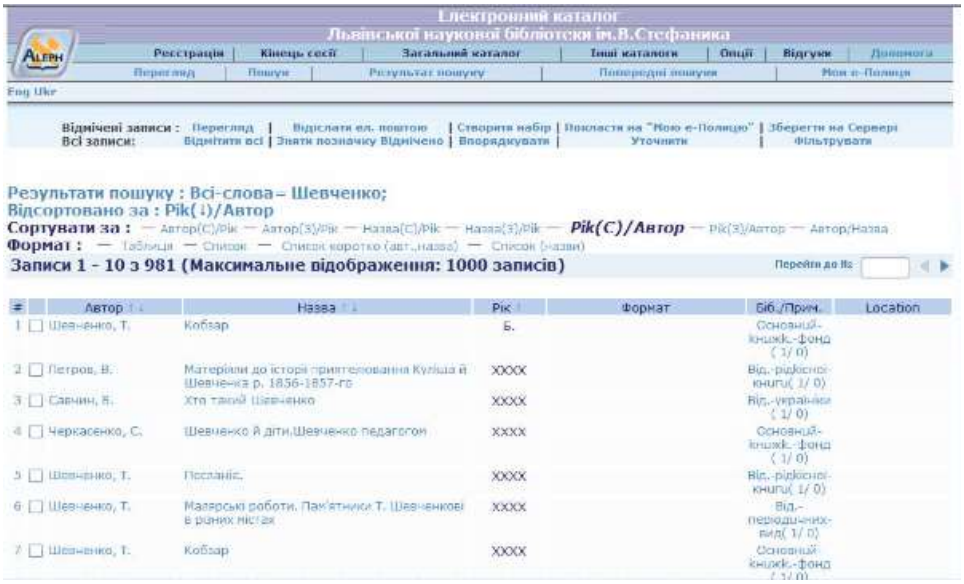

Рисунок 1.1 – Графічний інтерфейс ALEPH

Ще однією популярною автоматизованою системою управління бібліотекою є UniLib – система управління бібліотекою з різноманітними функціями [7]. Вона може включати у себе модулі для електронного каталогу, управління позичанням та поверненням книг, а також аналізу використання ресурсів бібліотеки. Можлива інтеграція з іншими системами (рис. 1.2). Вебсистема UniLib передбачає можливість видачі читачам літератури, якої ще немає в базі даних, через замовлення з іншої бібліотеки. Також наявний спеціалізований пошук літератури.

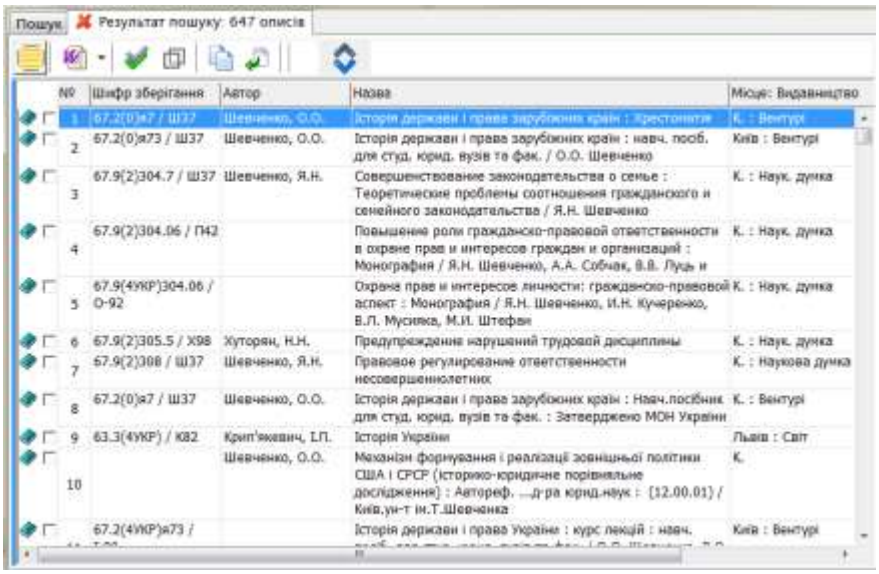

Рисунок 1.2 – Графічний інтерфейс UniLib

KOHA – це відкрите програмне забезпечення для автоматизації бібліотечних процесів [8]. Завдяки KOHA користувачі можуть каталогізувати книги, періодичні видання та інші матеріали, включаючи введення бібліографічних даних, класифікацію, визначення позицій та інші атрибути (рис. 1.3). Система надає засоби для аналізу статистичних даних та формування звітів про роботу бібліотеки, включаючи звіти про використання книг, активність користувачів та інші аспекти.

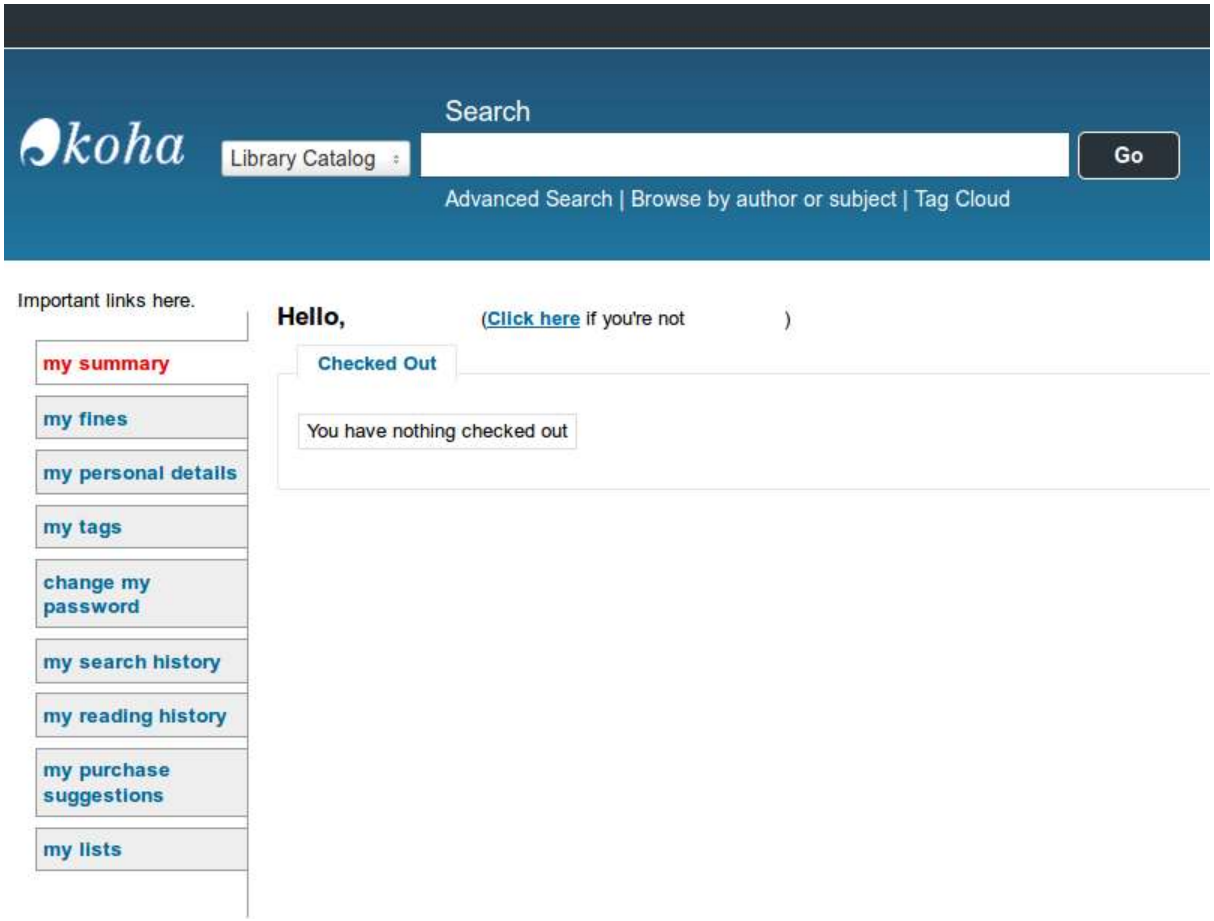

Рисунок 1.3 – Графічний інтерфейс KOHA

Розробка програмних модулів для впровадження системи автоматизації бібліотеки вимагає досвіду в галузі розробки програмного забезпечення та дизайну користувацького інтерфейсу. Це передбачає створення вебсистеми, яка може ефективно керувати каталогами, обліком запозичень та іншими аспектами роботи бібліотеки. Забезпечення зручного інтерфейсу передбачає розробку відповідних

користувацьких інтерфейсів, які забезпечують інтуїтивно зрозумілий та приємний досвід користувачів.

Результати порівняння аналогів зведено в табл. 1.1.

Таблиця 1.1 – Порівняльні характеристика автоматизованих вебсистем

Критерії	ALEPH	UniLib	KOHA	Власна розробка
Функціонал для налаштування аналітики бібліотеки	1	0	1	1
Бронювання наявної літератури	1	1	0	1
Підтримка Unicode	1	1	0	1
Генерація звітності	0	1	1	1
Інтеграція з іншими системами	1	1	1	1
Загальна оцінка	80%	80%	60%	100%

Враховуючи переваги й недоліки систем-аналогів, було визначено, що розробка власної автоматизованої вебсистеми керування бібліотекою є доцільною. Отриманий продукт зможе покрити недоліки існуючих рішень та забезпечити новий досвід та можливості застосування технології автоматизації.

Отже, розробка програмних модулів для впровадження автоматизованої вебсистеми управління бібліотекою відкриває перед бібліотечними установами широкі можливості для оптимізації робочих процесів, покращення обслуговування читачів та збільшення ефективності управління ресурсами. Розуміючи потенційні переваги застосування автоматизованої системи управління бібліотекою та вимоги до розробки програмного забезпечення, бібліотеки можуть створювати інноваційні

та зручні вебсистеми, які сприяють покращенню роботи бібліотеки та задоволенню потреб користувачів.

1.3 Аналіз методів розв’язання задачі

Для розробки програмних модулів управління бібліотекою вебсистеми важливо розглянути найкращі підходи до вирішення питання досягнення результатів завдання. Різні методи мають свої переваги та недоліки, які варто врахувати при виборі. У цьому розділі опишемо найпопулярніші методи для розробки необхідної автоматизованої вебсистеми.

Одним з найпоширеніших способів розробки необхідних модулів є метод створення мікросервісів [9]. Створення мікросервіса - це підхід до розробки програмного забезпечення, при якому програма розбивається на невеликі незалежні сервіси, які взаємодіють між собою за допомогою мережі. Кожен мікросервіс відповідає за виконання однієї конкретної функції і має свій власний стек технологій та базу коду. Хоча створення мікросервісів має свої переваги, воно також пов'язане з певними недоліками, а саме: складність управління, затратність розгортання та інфраструктура, системні вимоги та складність тестування та налагодження. Тому такий метод не підходить для задачі.

Найоптимальнішим методом буде розробка власної автоматизованої вебсистеми. У цьому випадку складність для адаптації, затратність та інші недоліки, які є у переважної більшості інших методів, нівелюються. Кінцевий продукт є у такому методі є унікальним і може бути розгорнутим на будь-якій платформі з необхідними налаштуваннями.

Розглянувши усі переваги та недоліки методів, було вирішено використати метод розробки власної автоматизованої вебсистеми. Такий підхід дозволить отримати повноцінний програмний продукт без необхідності використання потужного апаратного забезпечення з гнучкими налаштуваннями.

1.4 Постановка задач для розробки автоматизованої вебсистеми управління бібліотекою

Провівши аналіз методів розв’язання поставленої задачі, аналіз стану автоматизованих вебсистем управління бібліотекою, порівняння аналогів було визначено такий функціонал для власної розробки:

– розробити модель автоматизованої вебсистеми управління бібліотекою, яка міститиме основний функціонал системи;

- розробити метод формування і відображення звітності;

– розробити алгоритм бронювання доступної літератури з допомогою онлайн запиту;

– розробити зручний та адаптивний графічний інтерфейс вебсистеми;

- розробити програмне забезпечення вебсистеми;

– реалізувати модуль статичних досліджень для створення аналітичної статистики вебсистеми;

– здійснити тестування вебсистеми згідно поставлених задач.

1.5 Висновки

У першому розділі було розглянуто стан автоматизованих вебсистем керування бібліотекою. Було проведено порівняння аналогів, таких як: ALEPH, UniLib, KOHA.

У результаті порівняння було виявлено переваги та недоліки досліджуваних ресурсів, а також функціонал, завдяки якому вони набули поширення серед автоматизованих вебсистем керування бібліотекою. Було виявлено, що вищезгадані аналоги не надають усіх можливостей бронювання та аналітики зі звітністю. Таким чином було доведено доцільність розробки власного програмного рішення. На основі отриманої інформації було сформовано перелік задач, які необхідно виконати для розробки автоматизованої вебсистеми управління бібліотекою.

2 РОЗРОБКА МЕТОДУ, МОДЕЛІ ТА АЛГОРИТМІВ РОБОТИ ВЕБСИСТЕМИ

2.1 Аналіз вхідних даних системи

Для реалізації програмних засобів автоматизованої вебсистеми управління бібліотекою будуть використані мова стандартизована мова розмітки HTML, мова стилю сторінок CSS, мова програмування JavaScript, фреймворк SvelteKit, бібліотека Google Charts та середовище розробки Microsoft Visual Studio Code.

HTML (HyperText Markup Language) використовується для створення структури вебсторінок. Він використовує теги для визначення різних елементів, таких як заголовки, абзаци, списки, зображення та інші. HTML дозволяє вбудовувати вебелементи, такі як відео, аудіо та форми, і встановлювати зв'язки між сторінками за допомогою гіперпосилань [10].

CSS (Cascading Style Sheets) використовується для стилізації вебсторінок і надання їм зовнішнього вигляду [11]. За допомогою CSS можна задавати кольори, шрифти, розміри елементів, відступи, рамки і анімацію. CSS також дозволяє створювати адаптивний дизайн, що підлаштовується під різні пристрої і розміри екранів.

JavaScript (JS) - це мова програмування, яка використовується для створення динамічних вебсайтів і вебдодатків. Вона є однією з найпоширеніших мов програмування у веброзробці і дозволяє створювати інтерактивні елементи, анімацію, змінювати вміст сторінки під час завантаження або після взаємодії користувача. JavaScript є скриптовою мовою, що означає, що вона виконується на стороні клієнта (у браузері користувача), що дозволяє змінювати вміст сторінки без необхідності перезавантаження [12].

SvelteKit - це фреймворк для створення вебдодатків, який базується на компіляції. Він є еволюцією фреймворку Svelte і надає розширені можливості для розробки вебдодатків. SvelteKit надає структуру проекту з багатьма корисними функціями, такими як маршрутизація, підтримка статичного рендерингу,

передзавантаження на стороні сервера (SSR) і багато іншого. Однією з ключових особливостей SvelteKit є його швидкодія, оскільки компілює код до чистого JavaScript без використання віртуального DOM [13].

Google Charts - це безкоштовна бібліотека для створення і візуалізації даних у вебдодатках. Вона надає можливості для створення різних типів діаграм, графіків і карт, які можна вставляти в вебсторінки і додатки для відображення даних у зручному форматі [14].

Microsoft Visual Studio Code - це безкоштовний і відкритий редактор коду, розроблений Microsoft. Він підтримує багато мов програмування і відзначається високою продуктивністю і багатими функціями. Microsoft Visual Studio Code має інтегровану підтримку Git для контролю версій, розширювану систему за допомогою додатків і плагінів, можливості налаштування робочого середовища, такі як теми, шрифти і розташування панелей.

Розробка програмних засобів автоматизованої вебсистеми управління бібліотекою передбачає створення вебзастосунку, який відображатиме необхідні дані для користувачів у зручному інтрефейсі. Також, є необхідність створення бази даних, яка зберігатиме усю необхідну інформацію щодо літератури, рейтингу, статистики тощо. Розробка вебзастосунку буде на основні поєднання окремих вебкомпонент.

База даних також повинна буде зберігати інформацію щодо користувачів та адміністраторів автоматизованої вебсистеми. Для зручності буде розроблено модуль реєстрації нових користувачів, а також вхід для вже створених. Адміністратори матимуть додатковий доступ до статистики та користувачів.

Розробка цих програмних засобів вимагає глибокого розуміння баз даних, створення графічного інтерфейсу та програмування мовою JavaScript з використанням фреймворку SvelteKit. Для реалізації програмних засобів буде використано середовище розробки Microsoft Visual Studio Code. Програмні засоби будуть протестовані та доопрацьовані, щоб забезпечити безперебійну роботу вебсистеми управління бібліотекою.

Отже, розробка програмних засобів автоматизованої вебсистеми управління бібліотекою з використанням HTML, CSS, JavaScript, SvelteKit та Microfost Visual Studio Code є комплексним завданням, яке включає кілька етапів. Щоб забезпечити ефективне тестування, необхідно ретельно спроєктувати, реалізувати та протестувати засоби. Проте за допомогою вірних інструментів та досвіду можна створити систему, яка змінить підхід до автоматизації вебсистеми управління бібліотекою.

2.2 Розробка моделі системи

Для кращого розуміння роботи програмних засобів автоматизованої вебсистеми управління бібліотекою було вирішено розробити модель роботи системи на прикладі UML діаграми діяльності (рис. 2.1) [17]. UML діаграми допомагають представити структуру та поведінку системи в наочному вигляді, що полегшує розуміння складних систем і процесів.

Згідно вказаної діаграми користувач може використати пошук та відвідати сторінку з рейтингом без авторизації, але для використання іншого функціоналу реєстрація чи авторизація є обов’язковими. На сторінці пошуку користувач може знайти необхідну книгу. Задля цього є поле уведення, яке приймає як назву твору, так і автора. Також, є можливість обрання фільтрів, які будуть застосовані до пошуку.

Наступним кроком користувач може забронювати необхідну літературу. Для цього він знову ж таки може скористатися сторінкою пошуку. Обравши необхідний твір, користувач потрапляє на сторінку літератури, де він може її забронювати з допомогою кнопки “Забронювати”. Натиснувши на неї, на екрані з’явиться модальне вікно з полями, які необхідно заповнити. Після введення, користувач може надіслати запит на бронювання

Також, якщо користувач вже бронював певну книгу, то у нього є можливість написання відгуку, який допоможе іншим відвідувачам з їхнім вибором. Кожен

відгук містить рейтинг та текст, який обмежено розміром у одну тисячу символів. У автора є можливість залишитися анонімним при написанні відгуку, що збільшує вірогідність чесного відгуку.

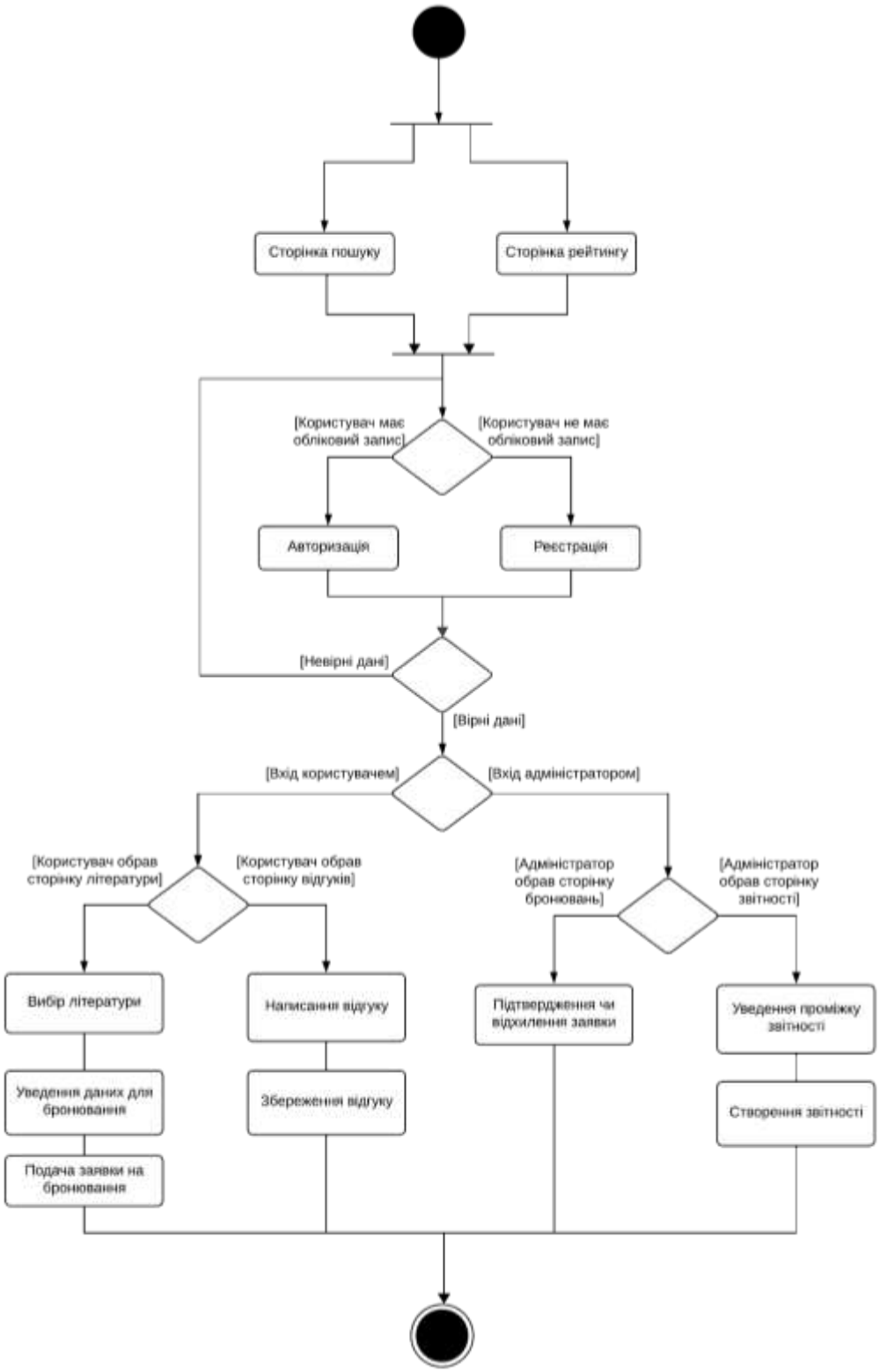

Рисунок 2.1 – Модель системи у вигляді діаграми діяльності

Діаграма діяльності системи ілюструє загальний процес роботи програми, забезпечує високий рівень абстракції, спрощує розуміння робочих процесів.

2.3 Розробка методу формування і відображення звітності

Адміністратори мають змогу переглянути аналітику популярності літератури у бібліотеці у будь-який момент. Звіти дозволяють відстежувати, які книги є найбільш популярними серед користувачів бібліотеки. Це може бути корисно для планування закупівель нових книг, оновлення колекцій та визначення тенденцій у читацьких вподобаннях. Також, вони можуть допомогти у виявленні тенденцій та патернів у користуванні бібліотечними ресурсами. Наприклад, можна виявити певні жанри або авторів, які є найбільш популярними серед користувачів, і відповідно пристосувати асортимент книг.

Аналіз звітів може допомогти бібліотеці покращити обслуговування користувачів шляхом забезпечення доступності найпопулярніших книг або покращенням політики видачі книг у певних жанрах. Такого роду аналітика про популярність літератури можуть служити основою для планування різноманітних програм, заходів та подій, які можуть бути цікавими для користувачів бібліотеки.

Звіти про популярність літератури також можуть допомогти бібліотеці в рекламі та маркетингу. З їх допомогою можна підвищити відомість про нові або маловідомі книги, що є популярними серед користувачів, та сприяти їх більш активному використанню. Крім того, звіти можуть бути корисні для бюджетного планування, допомагаючи бібліотеці ефективно розподіляти ресурси на найбільш популярні книги та авторів.

Метод формування і відображення звітності щодо популярності літератури дозволяє створити вибірку літератури за певний проміжок часу з сортуванням за популярністю. Завдяки можливості змінити період, надано можливість зручно порівняти результати.

Метод включає такий функціонал:

1. Авторизований користувач переходить на сторінку звітності.

2. Користувач обирає період з допомогою полів введення, що дають можливість ввести початкову та кінцеву дати. Після цього натискаємо кнопку «Сформувати звіт».

3. Користувач бачить перед собою індикатор завантаження.

4. При натисканні кнопки запускається функція, яка відповідає за отримання даних літератури з Firebase та викликає подальші функції обробки.

5. Метод getLiteratureFromDB виконує з’єднання з базою даних Firebase Firestore.

6. За допомогою Firebase SDK до бази даних виконується HTTP запит для отримання списку літератури, який задовільняє уведений проміжок дат. Якщо ж користувач не задав цей проміжок, то система поверне список з даними за останні 365 днів.

7. Метод getLiteratureFromDB повертає JSON об’єкт, який містить список літератури, яка була забронювана за вказаний проміжок часу.

8. Отримані дані використовуються в методі formateData, який форматує дані для відображення:

8.1. Отримані дані заносяться у колекцію літератури. Після цього створюється колекція жанрів, яка містить жанр ключем, а кількість бронювань значенням.

8.2. Сформовані колекції передаються методу sortCollectionsByAmount, котрий сортує з допомогою використання алгоритму сортування злиттям.

8.3. Метод transformCollectionToArray отримує відсортовані колекції, котрі потім перетворює на масиви даних.

9. Після цього запускається метод createAnalytics, який надсилає отримані дані з допомогою HTTP запиту на сервер сервісу Google Charts, котрий, у свою чергу, повертає HTML зі сформованими графіками.

10. Індикатор завантаження зникає і користувачеві відображується сформований інтерактивний звіт.

Описаний метод включає кроки авторизації користувача, візуалізацію індикатора завантаження, що покращує досвід користувача, отримання необхідних даних з Firebase Firestore з допомогою HTTP запиту, форматування та сортування даних для правильного та інтерактивного відображення з допомогою Google Charts, завершення формування звіту з прибирання індикатора завантаження і відображення результату користувачу. Алгоритм роботи цього методу наведено на рисунку 2.2.

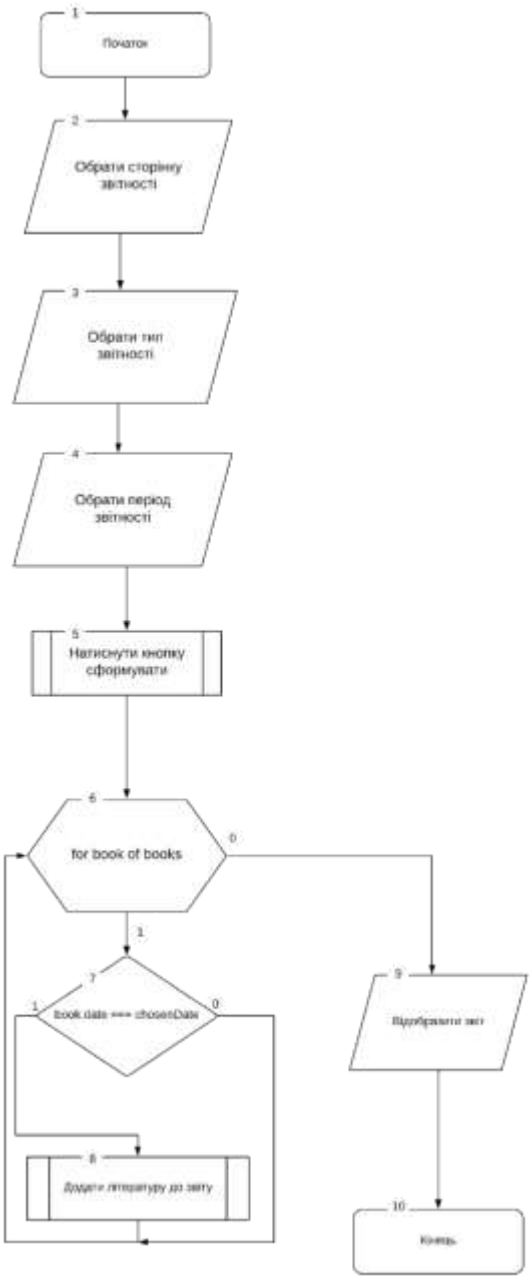

Рисунок 2.2 – Алгоритм методу формування і відображення звітності

На рисунку 2.2. зображено алгоритм створення звітності адміністратором задля отримання інформації про ефективність вебзастосунку.

2.4 Розробка загального алгоритму роботи застосунку

Для розробки засобів автоматизованої вебсистеми управління бібліотекою необхідно розробити загальний алгоритм, згідно якого буде працювати вебзастосунок (рис. 2.3). За цим алгоритмом спершу користувачеві варто обрати бажаний підпункт меню. Якщо ж він авторизований адміністратором, то може перейти у панель адміністратора, з котрої може потрапити до панелі літератури з доступом до редагування даних літератури, її оновлення чи видалення. Важливою функцією для адміністратора є можливість формування звітності, яка допомагає у проведенні аналітики щодо стану вебсистеми. За умови, що користувач попередньо забронював книгу, адміністратор може перейти на спеціальну сторінку з підтвердження бронювання, або ж його скасуванням, якщо вийшла помилка.

Авторизувавшись користувачем вебсистеми, користувач може потрапити до списку із літературою і обрати необхідну для бронювання. Також користувач може відвідати сторінку з детальною інформацією про книгу. Вона містить зображення, короткий опис, рейтинг та відгуки інших користувачів. Авторизований користувач може залишити відгук про літературу, якщо він вже прочитав її. Є можливість залишення анонімного відгуку, який допомагає користувачам написати саме чесний відгук про книгу. Відгук також містить оцінку у 5-ти бальній системі та містить обмеження за довжиною у 5000 символів.

Якщо ж користувач не авторизувався, він може переглянути список найпопулярніших книг на головній сторінці. Також, є можливість скористуватися пошуком необхідної літератури. Для цього надається спеціальне поле уведення, яке приймає як назву так і автора. Запит пошуку можна комбінувати з наданою фільтрацією, яка дозволяє відформатувати дані згідно доступності для бронювання, загального рейтингу серед користувачів та бажаного жанру.

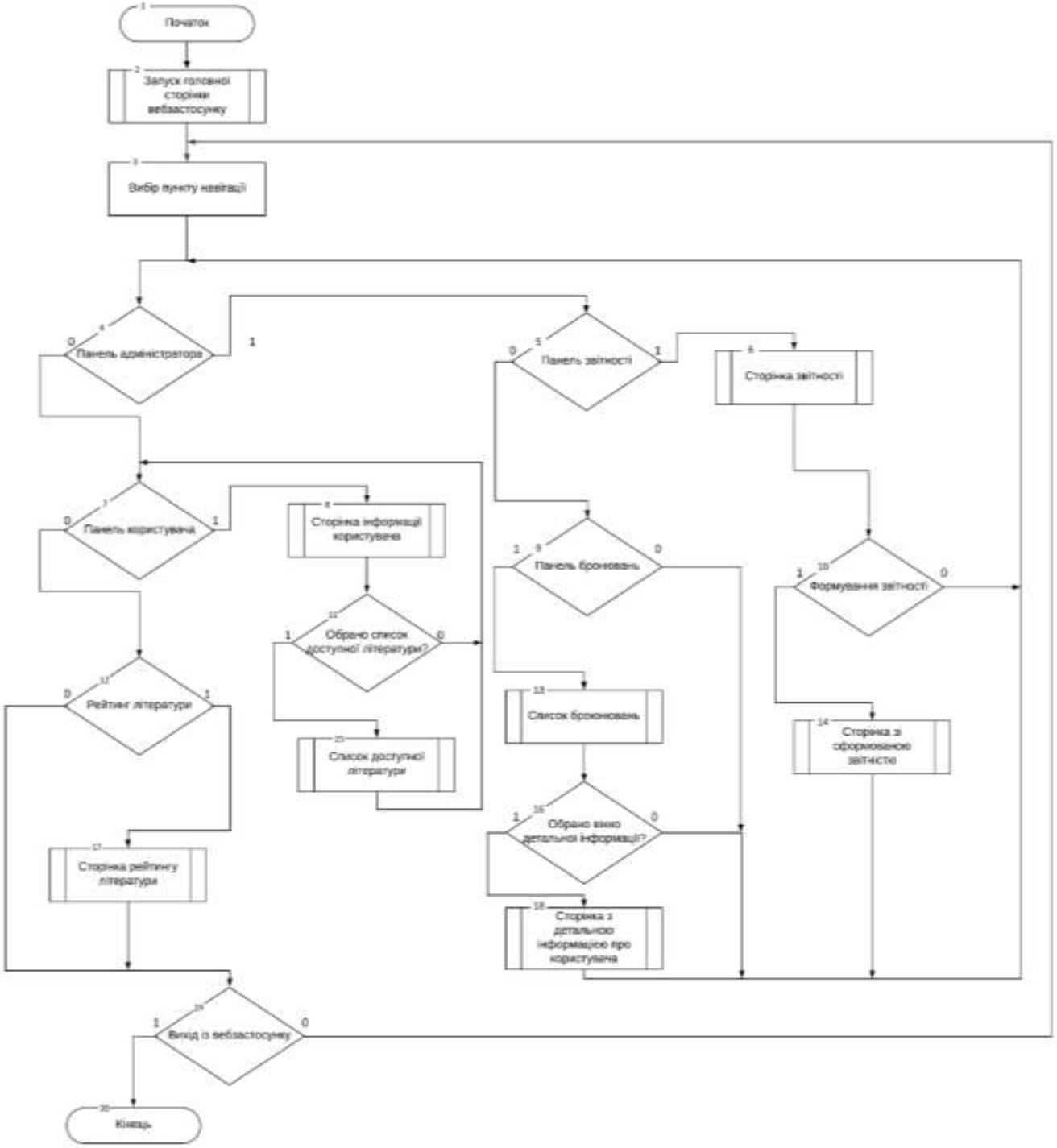

Рисунок 2.3 – Загальний алгоритм роботи вебсистеми

На рисунку 2.3 зображено загальний алгоритм роботи автоматизованої вебсистеми керування бібліотекою. З його допомогою можна зрозуміти загальну структуру автоматизованої вебсистеми й переглянути загальний функціонал, який надається адміністраторам, зареєстрованим та незареєстрованим користувачам.

2.5 Висновки

У другому розділі було проведено аналіз вхідних та вихідних даних вебсистеми. Описано розроблений метод формування і відображення звітності. Було розроблено основний алгоритм роботи системи та алгоритм звітності. Також розроблено модель роботи програмного продукту за допомогою UML діаграм.

3. РОЗРОБКА ПРОГРАМНИХ МОДУЛІВ ВЕБСИСТЕМИ

3.1 Варіантний аналіз і обґрунтування вибору засобів для реалізації програми

При розробці автоматизованої вебсистеми управління бібліотекою, необхідно відповідально підійти до обрання використовуваних технологій. JavaScript – це мова програмування, яка використовується для створення динамічних інтерактивних вебсайтів [12]. Дана мова програмування має широку екосистему бібліотек і фреймворків, які спрощують розробку вебзастосунків. Наприклад, для створення SPA (односторінкових додатків) ви можете скористатися фреймворками, такими як React, Svelte або Vue. JavaScript є досить легким для вивчення мовою програмування, особливо для тих, хто вже знайомий з HTML і CSS. Це робить його доступним для багатьох розробників, які хочуть створювати вебзастосунки.

SvelteKit – це фреймворк для розробки вебдодатків, який базується на ідеї компіляції компонентів [13]. SvelteKit генерує оптимальний вихідний код, що дозволяє створювати швидкі вебдодатки з мінімальними витратами на ресурси. Синтаксис дуже зрозумілий і схожий на звичайний HTML, CSS і JavaScript, що дозволяє легко розуміти і модифікувати код. Цей фреймворк дозволяє використовувати звичайні JavaScript-бібліотеки та компоненти, що робить його дуже гнучким для розвитку проектів будь-якої складності. А також має вбудовану підтримку для різних функцій, таких як маршрутизація, передача даних між сторінками, підтримка статичних і динамічних маршрутів і багато іншого.

Google Charts – це бібліотека JavaScript, яка дозволяє легко створювати різноманітні графіки та діаграми для вебсайтів і додатків [14]. Google Charts надає простий у використанні API, який дозволяє швидко створювати графіки та діаграми з мінімальними зусиллями, підтримує багато типів графіків і діаграм, включаючи лінійні графіки, стовпчасті діаграми, кругові діаграми, графіки розсіювання, гістограми та інші. Google Charts підтримує можливість додавання взаємодії до

графіків, такої як наведення, клікання та перетягування, що робить їх більш інтерактивними для користувачів. Узагальнюючи, Google Charts - це потужний і простий у використанні інструмент для створення графіків та діаграм для вебсайтів і додатків, який дозволяє створювати візуально привабливі та інтерактивні представлення даних.

Firebase – це платформа для розробки мобільних та вебдодатків, яка надає ряд інструментів для зберігання та обробки даних в реальному часі [18]. Основною особливістю Firebase є можливість зберігання даних в хмарі, що дозволяє отримувати доступ до них з будь-якого пристрою та з будь-якої точки світу. Firebase підтримує різні типи даних, включаючи структуровані дані, файли, аутентифікацію користувачів, аналітику та інші функції. Firebase забезпечує можливість миттєвого оновлення даних у реальному часі без необхідності оновлення сторінки.

Це поєднання технологій дозволяє створити інноваційну систему, яка забезпечує зручність та ефективність у управління бібліотекою. SvelteKit разом з Firebase забезпечують високу швидкодію автоматизованої вебсистеми, у той час як Google Charts забезпечує оптимізоване та швидке створення гріфіків.

Отже, для розробки програмних модулів для автоматизованої вебсистеми керування бібліотекою обрано наступні технології: JavaScript, SvelteKit, Firebase, Google Charts. Поєднання цих технологій надає необхідні інструменти та готові рішення для швидкої та ефективної розробки надійну вебсистему зі високою швидкодією та простотою у підтримці.

3.2 Розробка засобів відображення графічного інтерфейсу

Графічний інтерфейс є невід'ємною частиною будь-якої вебсистеми, в тому числі і автоматизованої системи управління бібліотекою [19]. Важливість графічного інтерфейсу для такої системи можна пояснити наступними аспектами:

1. Зручність користування. Графічний інтерфейс дозволяє зробити взаємодію з системою більш зручною та інтуїтивно зрозумілою для користувачів.

Він дозволяє швидко знаходити необхідні функції та виконувати потрібні дії без додаткових зусиль.

2. Ефективність використання. Гарний графічний інтерфейс допомагає зробити роботу з системою більш продуктивною. Він може оптимізувати робочий процес, допомагаючи користувачам швидше знаходити та обробляти необхідну інформацію.

3. Привабливість. Привабливий та сучасний дизайн графічного інтерфейсу може позитивно вплинути на сприйняття системи користувачами. Це може підвищити їхню зацікавленість та бажання використовувати систему.

4. Підтримка функціональності. Графічний інтерфейс дозволяє ефективно відображати різноманітну інформацію та функціонал системи. Він дозволяє відобразити структуру бібліотеки, доступні послуги, можливості пошуку та фільтрації книг та інші важливі елементи системи.

5. Конкурентоспроможність. Графічний інтерфейс може стати важливим конкурентним перевагою системи, особливо в сучасному світі, де важливе значення має зовнішній вигляд та користувацький досвід.

Створення графічного інтерфейсу з SvelteKit має кілька переваг. Перш за все, цей фреймворк відзначається простотою використання: він надає зрозумілий синтаксис, що дозволяє швидко створювати інтерактивні вебінтерфейси. Крім того, завдяки компіляції на стадії збірки, SvelteKit генерує легкий і ефективний JavaScript-код, що поліпшує швидкодію та продуктивність вебдодатків.

Розробка з використанням функціоналу SvelteKit відбувається досить швидко завдяки тому, що при збереженні файлу в середовищі розробки, усі зміни одразу відображаються у браузерному середовищі. Використавши Split Screen – усталену функцію Windows, можна писати код і одразу бачити результат веброзробки (рис. 3.1).

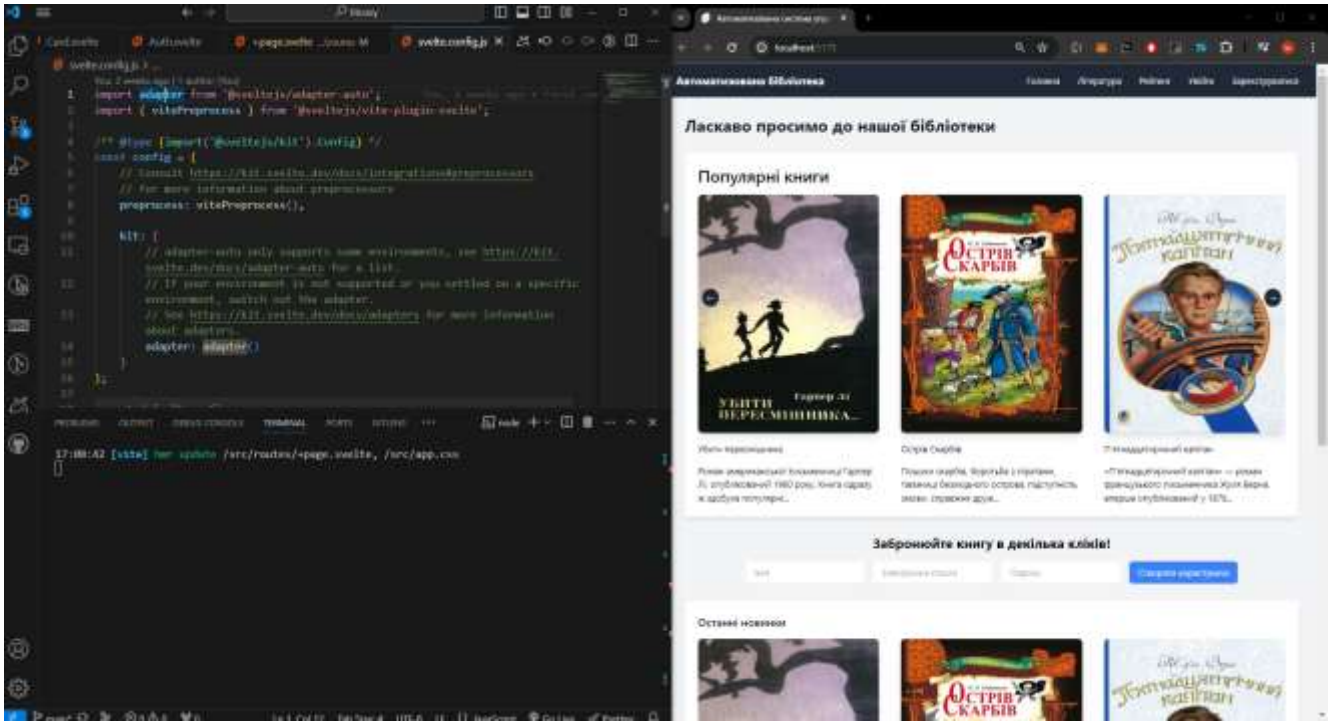

Рисунок 3.1 – Розробка графічного інтерфейсу

У SvelteKit можна писати HTML та CSS, як у звичайному вебпроєкті, а також створювати компоненти, які забезпечують реактивність та перевикористання коду. Для створення розмітки з HTML можна використати HTML-теги, атрибути та структуру для організації контенту.

CSS є можливість вбудувати безпосередньо в компоненти за допомогою блоку <style>, або завантажити CSS з зовнішніх файлів. Також є можливість використання SCSS.

SvelteKit дозволяє створювати перевикористовувані компоненти, які можна використовувати в різних частинах вебзастосунку. SvelteKit включає в себе компілятор Svelte, який перетворює ваші компоненти Svelte на ефективний JavaScript-код під час розробки. Компоненти мають свою внутрішню логіку та можуть приймати параметри. Приклад використання цього функціоналу зображено на рисунку 3.2. Крім цього, SvelteKit надає додаткові інструменти для розробки універсальних вебдодатків, такі як система маршрутизації, обробка HTTP-запитів на сервері та попереднє завантаження сторінок для поліпшення швидкодії додатка.

```
        <div class="bg-white p-6 rounded-lg shadow-md">
            <h2 class="test text-xl font-semibold mb-4">Останні новинки</h2>
            <Slider />
        </div>

        <Auth />
    </div>
</main>

<style>
    You, 1 second ago | 1 author (You)
    .test {
        background-color: aqua;
    }
</style>
```

Рисунок 3.2 – Використання HTML, CSS та власної компоненти

Форми реєстрації та авторизації є візуально подібними, адже так користувачеві легше сприймати інформацію, а також легко перемкнутися з одного типу на інший. Для авторизації достатньо вказати власну пошту чи ім’я корситувача та пароль (рис. 3.3). Для реєстрації потрібно придумати ім’я користувача, написати пошту та увести пароль двічі, аби мінімізувати шанс помилки при його уведенні (рис. 3.4).

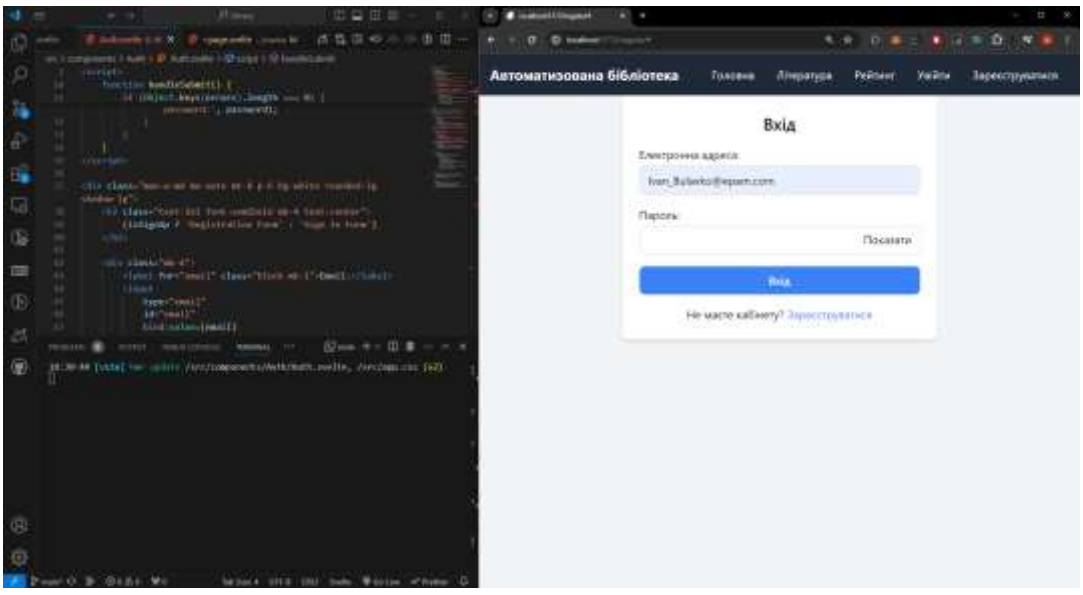

Рисунок 3.3 – Створення вікна авторизації

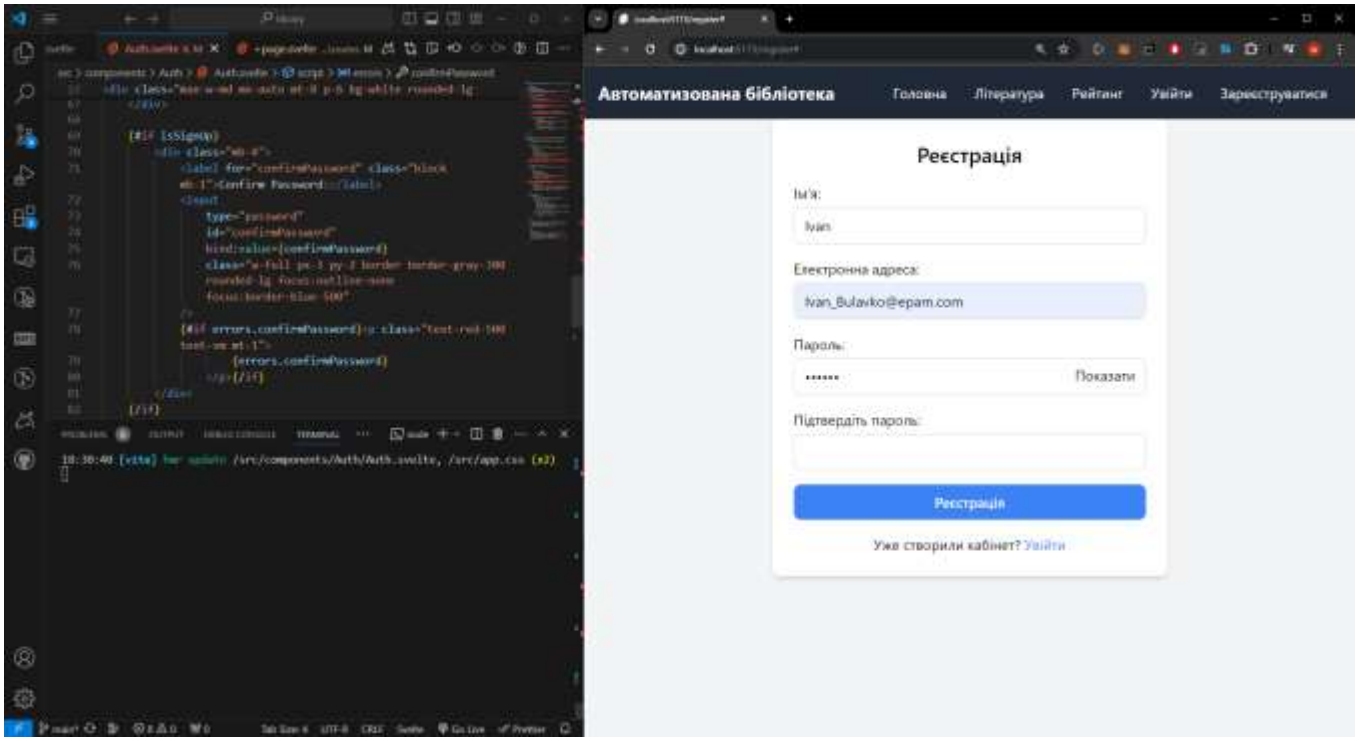

Рисунок 3.4 – Створення вікна реєстрації

Головна сторінка забезпечує базову взаємодію користувача з автоматизованою вебсистемою (рис. 3.5).

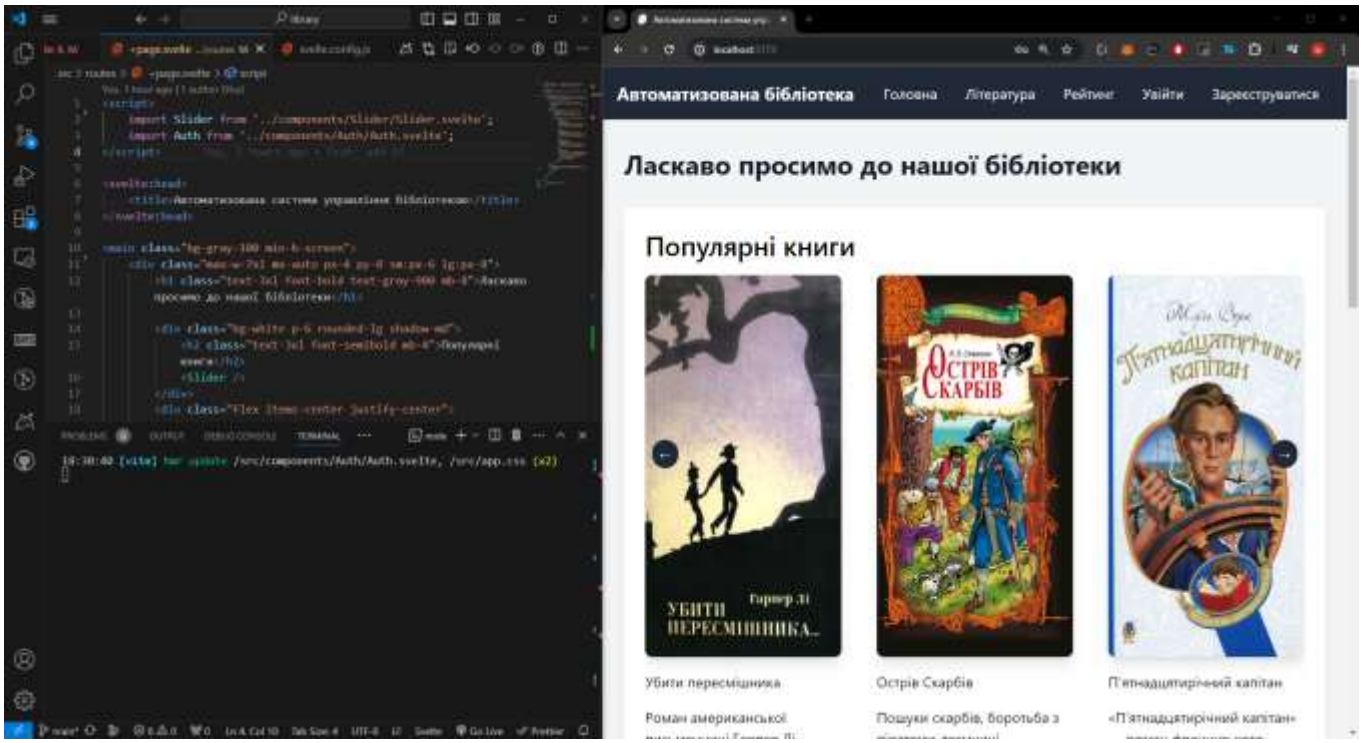

Рисунок 3.5 – Створення головної сторінки вебсистеми

На головній сторінці користувачеві доступний перехід на усі інші сторінки з допомогою рядка навігації. Однак, не увесь функціонал доступний усім користувачам. Задля використання усіх функцій необхідно авторизуватися. Якщо ж користувач не авторизований, то він побачить відповідний текст і матиме можливість авторизуватися (рис. 3.6).

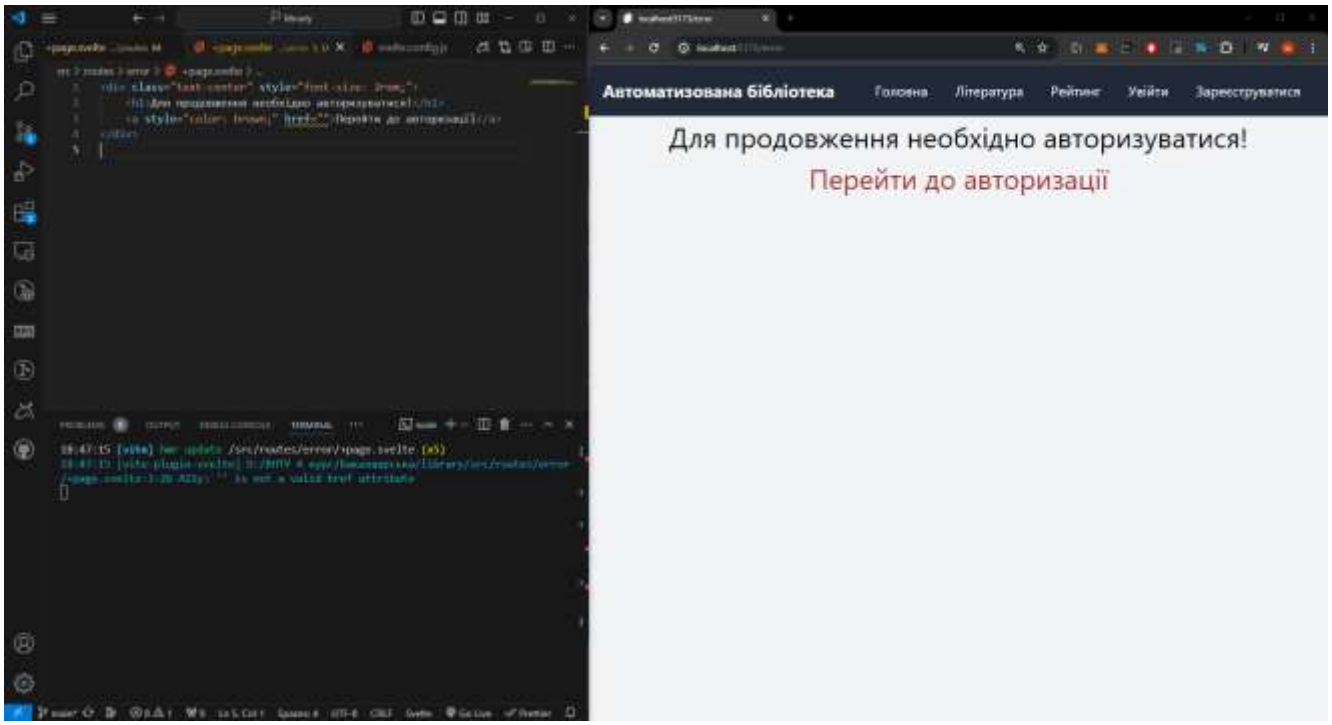

Рисунок 3.6 – Створення сторінки з повідомленням про авторизацію

Задля зручності використання автоматизованої вебсистеми, користувачам надано можливість перегляду загальної кількість літератури, яка існує в бібліотеці. Користувач може скористатися пошуком задля того, аби знайти необхідну книгу. Також в елемент уведення може бути ім’я та прізвище автора, якщо є потреба у пошукові саме за письменником (рис. 3.7). Окрім цього, існує можливість застосування фільтрів разом з пошуком. Фільтр рейтингу дозволяє обрати необхідний рейтинг літератури. Якщо ж користувача цікавить спеціальний жанр, то з допомогою відповідного фільтру, він може відсортувати усі необхідні книги. Важливою є і фільтрація за наявністю літератури. Якщо користувача цікавить саме література, яку він може забронювати, то необхідно скористуватися цією

фільтрацією. Усталено, фільтрація не застосовується без явного використання користувачем шляхом натискання на відповідні поля.

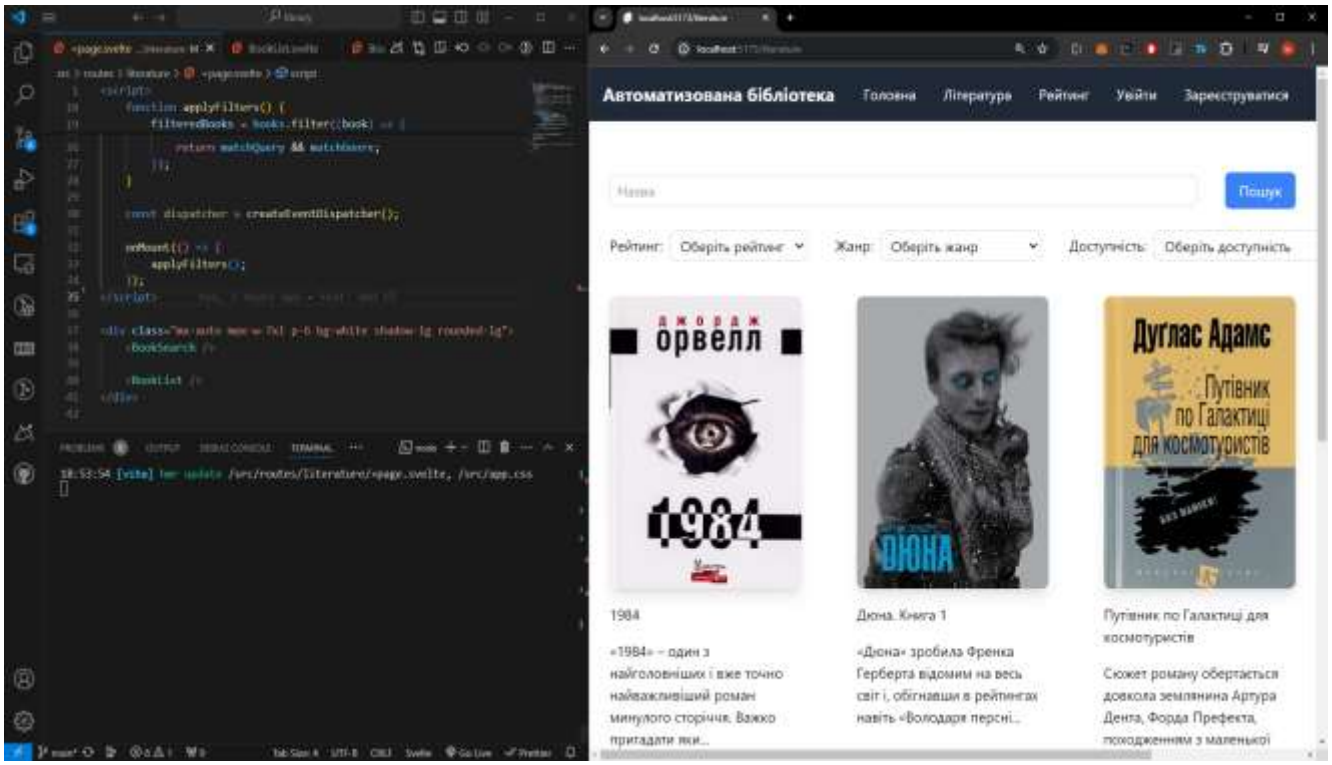

Рисунок 3.7 – Створення сторінки зі списком літератури

За потреби бронювання книги, користувачеві необхідно потрапити на сторінку обраної літератури. Для цього він може просто натиснути на необхідну картку книги на сторінці списку. На сторінці обраної книги користувач може ознайомитися назвою, автором та коротким описом (рис. 3.8). Також зазначено жанр, рейтинг і доступність для броні. Також можна прочитати відгуки інших користувачів, якщо такі є наявними, або написати власний. У відгукові користувач може обрати рейтинг, написати рецензію, яка не може перевищувати відмітки у 750 символів, а також увімкнути функцію анонімності. У такому випадку відгук буде від імені анонімної особи. Якщо ж потреби у цьому не має, то над відгуком буде зазначено ім’я користувача, який його залишив. Задля бронювання користувач може натиснути на кнопку “Забронювати”. Увести власні дані і очікування на підтвердження бронювання.

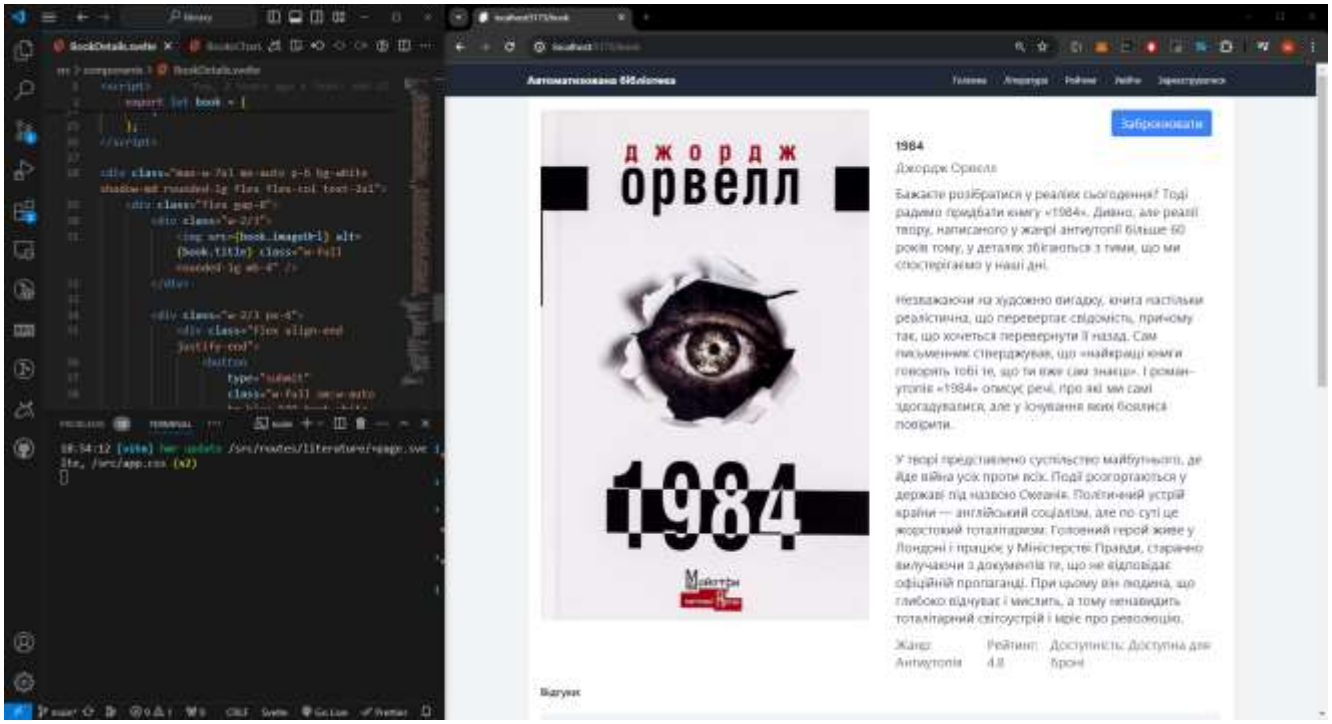

Рисунок 3.8 – Створення сторінки обраної літератури

Отже, саме тому розробка засобів відображення графічного інтерфейсу користувача є однією з найпріоритетніших етапів розробки програмного застосунку.

3.3 Розробка модуля створення звітності

Одним з ключових компонентів будь-якої вебсистеми є можливість створення звітності задля проведення аналітики по загальній роботі вебзастосунку [20]. Звіти про популярність допомагають визначити найбільш популярні серед користувачів книги та жанри. Ця інформація дозволяє бібліотеці планувати закупівлі нових видань, організовувати заходи та промоакції, спрямовані на підвищення інтересу до читання. Також допомагають виявляти слабкі місця у роботі бібліотеки. Це може бути низька популярність певних ресурсів, недостатня кількість примірників популярних книг або неефективність певних процесів. Аналіз цих даних дозволяє вживати заходів для покращення роботи бібліотеки.

Задля створення звітності для початку необхідно зібрати усі дані про книги та їх популярність.

У даній автоматизованій вебсистемі використовується Firebase, який дозволяє отримати усі дані літератури (рис. 3.9).

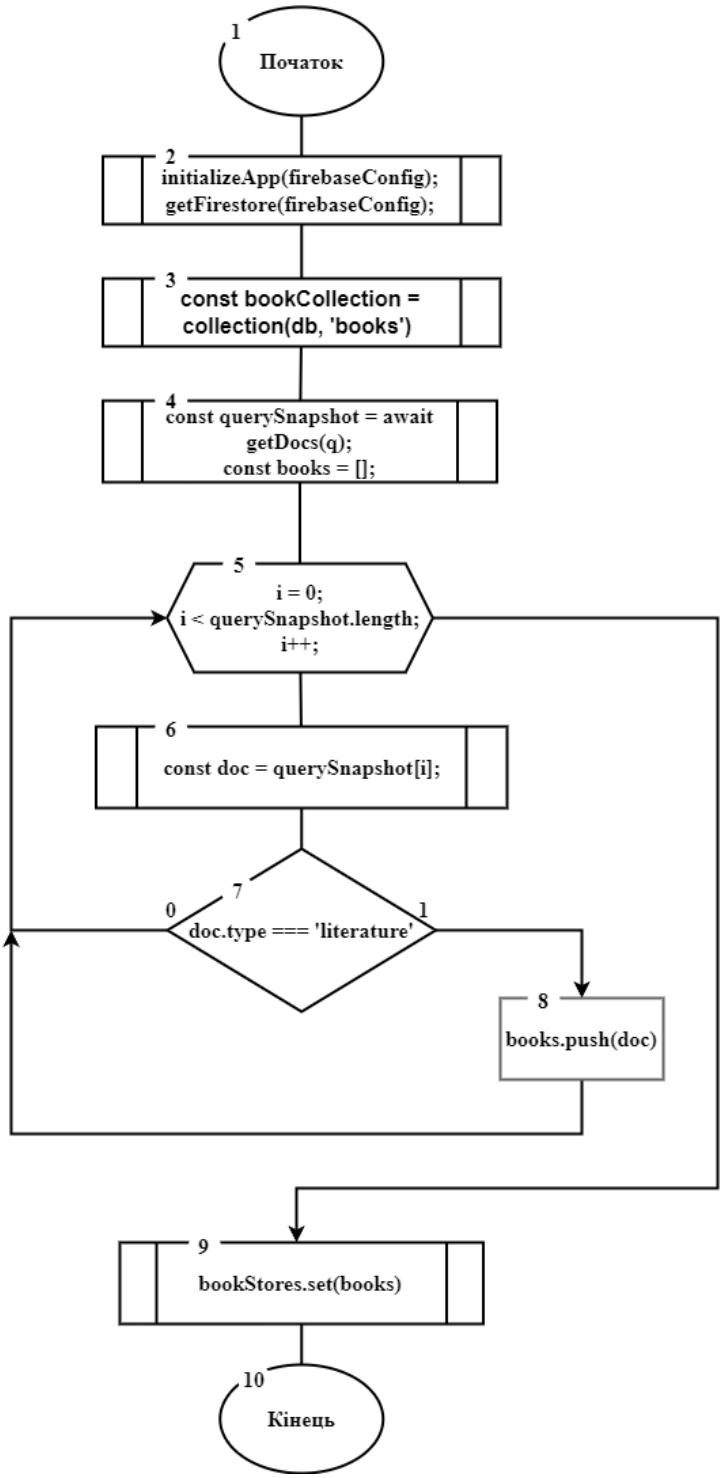

Рисунок 3.9 – Отримання даних для звітності

Наступним етап є агрегація даних. У автоматизованій вебсистемі існує два типи звітності: популярність за книгою та за жанром. Для цього для отриманих даних потрібно викликати певні функції обробки. Так як у адміністратора є змога обрання періоду звітності, то ці дані враховуються під час агрегації. Є змога обрати дані за день, тиждень, місяць, рік та увесь час (рис. 3.10). Відповідно до цих даних фільтрується список книг і формується звітність згідно 5 найпопулярніших екземплярів.

```
Ivan Bulavko, 4 hours ago | 1 author (Ivan Bulavko)
export const getBookingByDate = (
    bookings: string[],
    startDate: string = '2022-01-01',
    endDate: string = '2022-12-31'
): number => {
    let count = 0;

    for (const booking of bookings) {
        if (booking >= startDate && booking <= endDate) {
            count++;
        }
    }

    return count;
};
```

Рисунок 3.10 – Фільтрування даних згідно обраного періоду

Для звіту по жанрах перевикористовується отримана інформація популярності літератури. Для цього кількість бронювань кожної книги записується у спеціально створений об’єкт жанрів. Якщо жанр книги вже існує, то до нього додається кількість бронювань. Отримавши усі необхідні дані, вебсистема розпочинає візуалізацію даних з допомогою бібліотека Google Charts. Для відображення жанрів використовується кругова діаграма.

Задля відображення п’яти найпопулярніших книг використовується стовпчаста діаграма з допомогою котрої зручно ілюструється кореляція даних згідно бронювань (рис. 3.14).

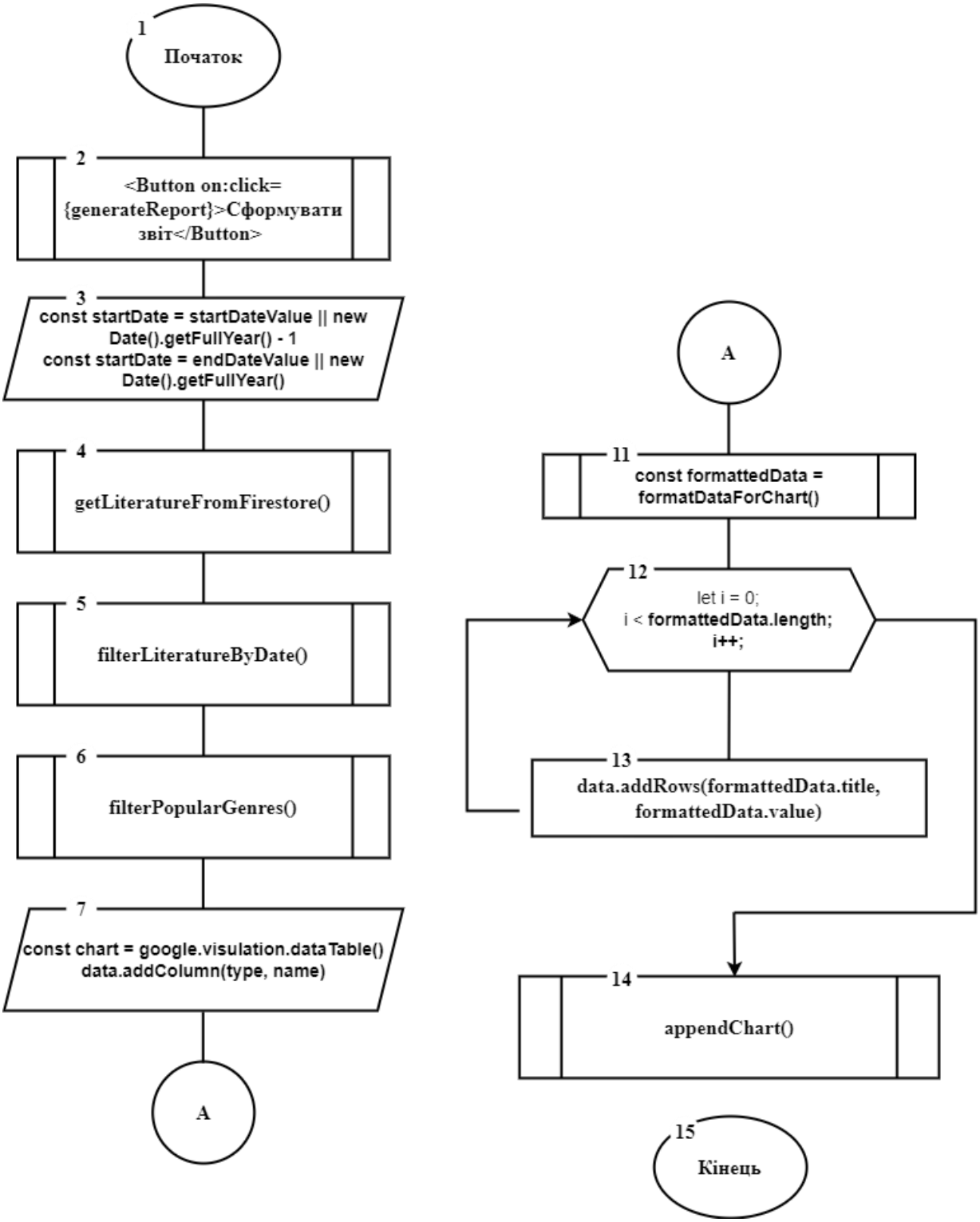

Рисунок 3.11 – Блок-схема роботи модуля звітності

З допомогою спеціальних методів, які надає Google Charts та переданих даних звітності у спеціальному форматі, у адміністратора відображається інтерактивні діаграми з аналітичною інформацією щодо бронювання бібліотечної літератури.

За допомогою описаних методів відбувається збір необхідних даних автоматизованої вебсистеми, потім аналіз внутрішніх даних з допомогою агрегації усіх даних з послідовними визначенням найпопулярніших книг та жанрів літератури. На кінець, відбувається перетворення отриманих даних у потрібно відформатований об’єкт JavaScript задля візуалізації даних з допомогою Google Charts. Для передавання даних використовується HTTP запит на спеціальний сервер Google.

3.4 Розробка модуля бронювання літератури

Модуль бронювання літератури відіграє важливу роль у вебсистеми управління бібліотекою. Розробка цього модуля має на меті значно спростити процес бронювання літератури, котрий вимагає вживу займає багато часу та зайвих дій. Основний зміст у розробці даного модуля бронювання полягає в тому, щоб дозволити користувачеві увести усі необхідні дані й просто очікувати на підтвердження зі сторони бібліотеки .

Цей модуль приймає наступні вхідні параметри:

– userData: об’єкт, що містить дані щодо користувача.

Об’єкт містить наступні поля:

– userName: ім’я користувача, яке він вказав при реєстрації;

– userFIO: справжні ім’я та прізвище користувача;

– phoneNumber: номер телефону з допомогою котрого можна зв’язатися з користувачем;

– periodOfBooking: період на котрий користувач бажає забронювати книгу.

Основні функції модуля бронювання літератури включають:

1. Завантаження сторінки літератури, яку бажає завантажити користувач. З цієї сторінки у користувача є можливість натиснути на кнопку “Забронювати”, після чого з’явиться модальне вікно бронювання (рис. 3.12).

2. Уведення даних необхідних для бронювання книги. Так як користувач при реєстрації у автоматизованій вебсистемі уводить необхідні дані для себе, то увести потрібно буде лише два поля. Перше стосується номеру телефону, з допомогою котрого є можливість зв’язатися із користувачем. Друге поле передбачає уведення даних щодо періоду бронювання літератури. Після цього надсилається заявка на бронювання

3. Адміністратор отримує заявку на бронювання літератури і може переглянути необхідну інформацію. Якщо ж заявка влаштовує усі умови, то заявку можна підтвердити. Завдяки вказаних користувачем даних адміністрація має можливість комунікації, яка може бути необхідною у непередбачуваних випадках. У випадку неможливості надання бронювання адміністрація вебсистеми може відхилити заявку.

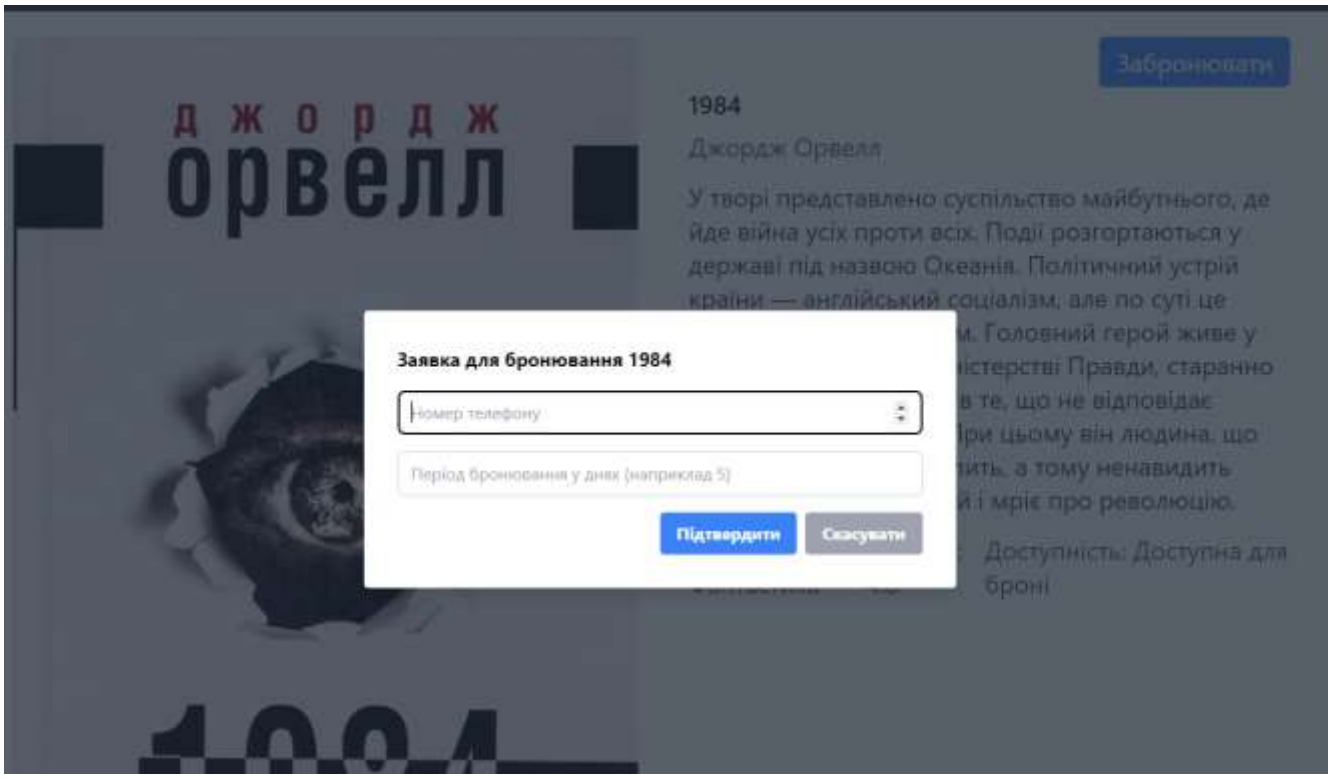

Рисунок 3.12 – Модальне вікно бронювання

Алгоритм роботи модуля наведено на рисунку 3.13.

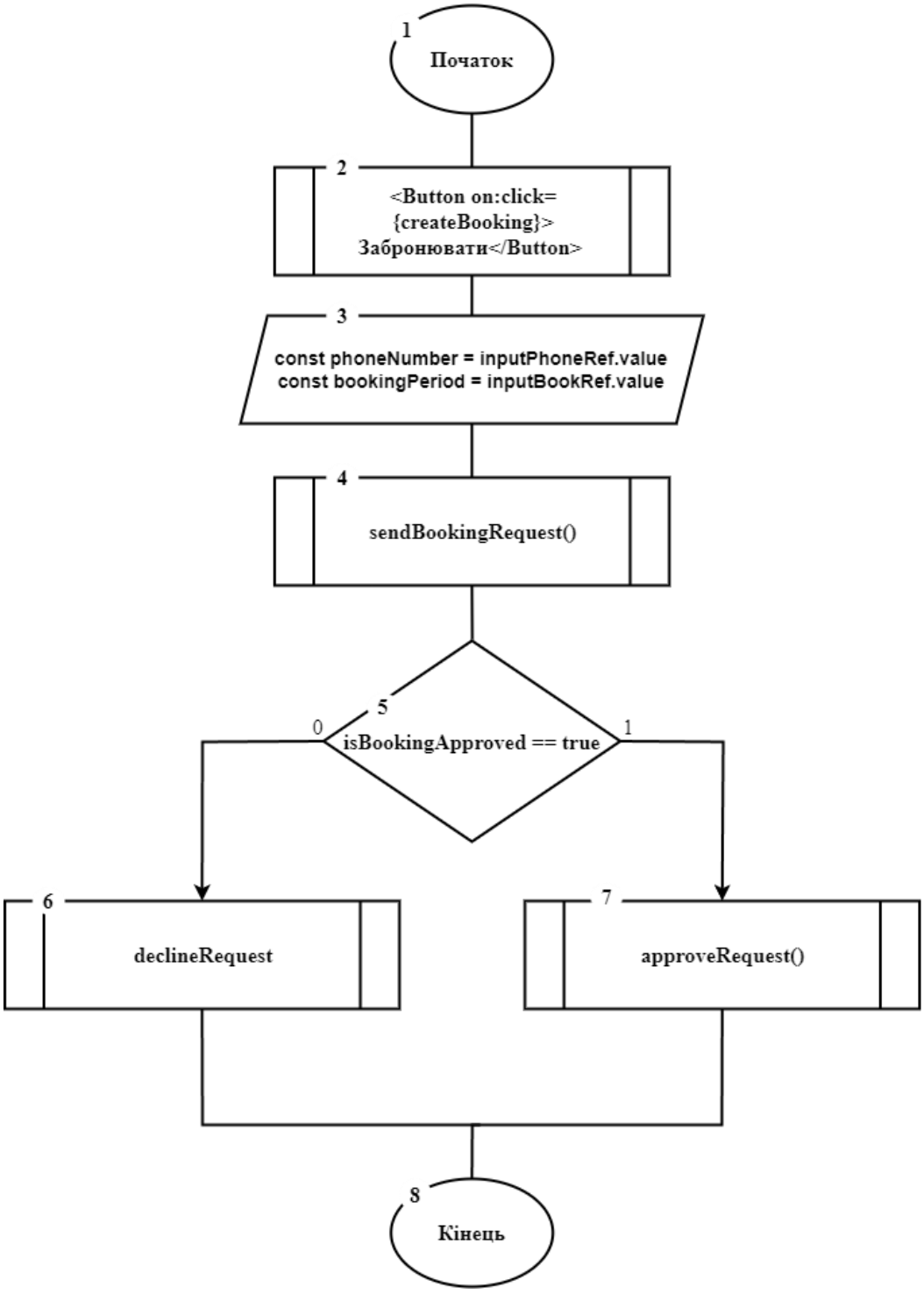

Рисунок 3.13 – Блок-схема алгоритму роботи модуля

Згідно розробленого алгоритму, користувач має зможу створити запит бронювання, у той час як адміністрація розглянути його і підтвердити чи відхилити.

3.5 Розробка інтерфейсу програмного додатку

Враховуючи, що вебсистемою будуть користуватися люди усіх вікових категорій, було вирішено розробити його інтуїтивно зрозумілим та простим, щоб не перевантажувати користувачів.

Основним кольором інтерфейсу обрано темно-сірий, оскільки даний колір є нейтральним. Обравши базовий колір, необхідно сформувати палітру. Вирішено скористатися Paletton, щоб вибрати інші необхідні кольори. Paletton надав блакитний та білі кольори допоміжними, адже вони найкраще контрастують з основним кольором, при цьому не дратуючи око користувача [15].

Після відкриття головної сторінки вебзастосунку, перед користувачем відображається головна сторінка (рис. 3.14). На ній знаходиться зручне меню навігації, найпопулярніші книги та форма реєстрації нового користувача, щоб забронювати необхідну книгу.

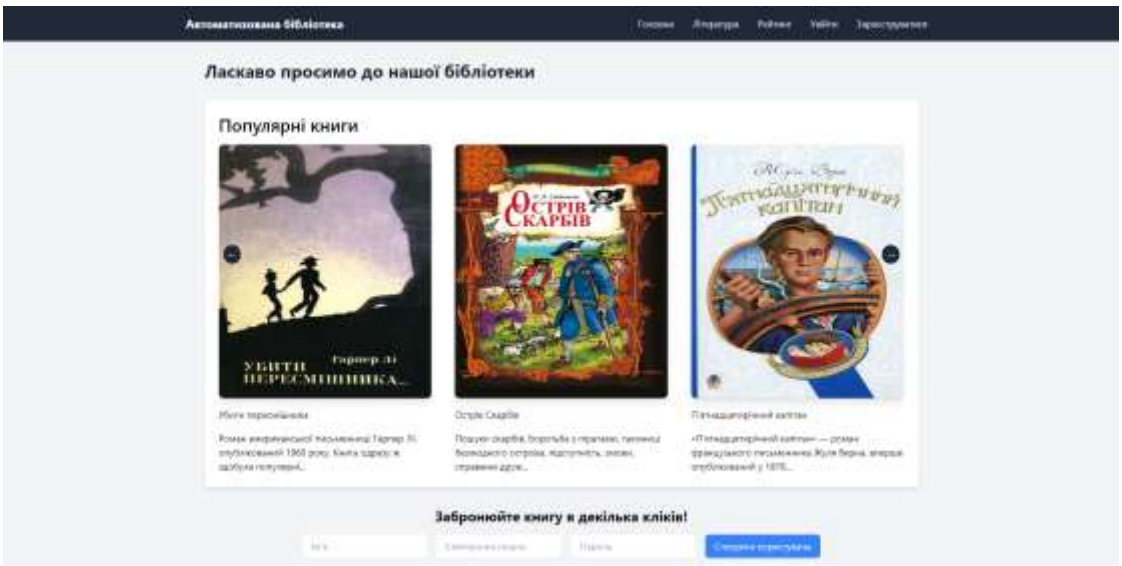

Рисунок 3.14 – Головна сторінка

Щоб мати можливість використовувати увесь функціонал, користувачеві необхідно авторизуватися. Вебсистема передбачає реєстрацію нових користувачів з допомогою форми на окремо створеній вебсторінці (рис. 3.15).

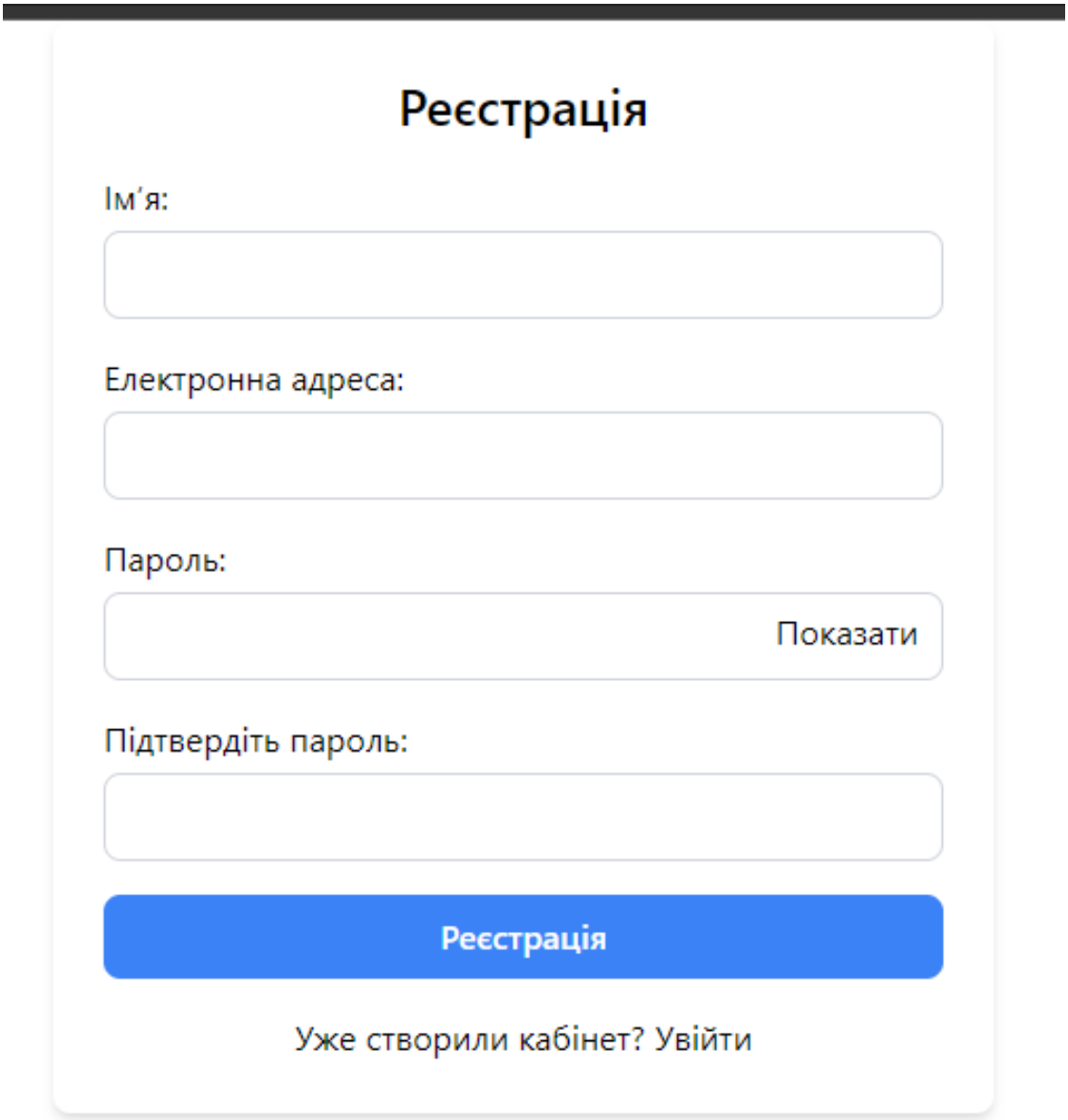

Рисунок 3.15– Форма реєстрації користувача

На цій сторінці користувач повинен увести унікальні дані імені, а також персональну пошту з паролем задля створення даних з допомогою котрих відбуватиметься вхід у вебсистему. Так як дані будь-якого формату надзвичайно важко опрацьовувати, форма використовує валідацію полів введення [16]. Ім'я користувача може бути від 4 до 20 англійських символів, електронна адреса повинна відповідати стандартам, а пароль повинен бути завдовшки від 6 до 22 символів, містити великі та малі літери, а також цифри зі спеціальними знаками (рис 3.16).

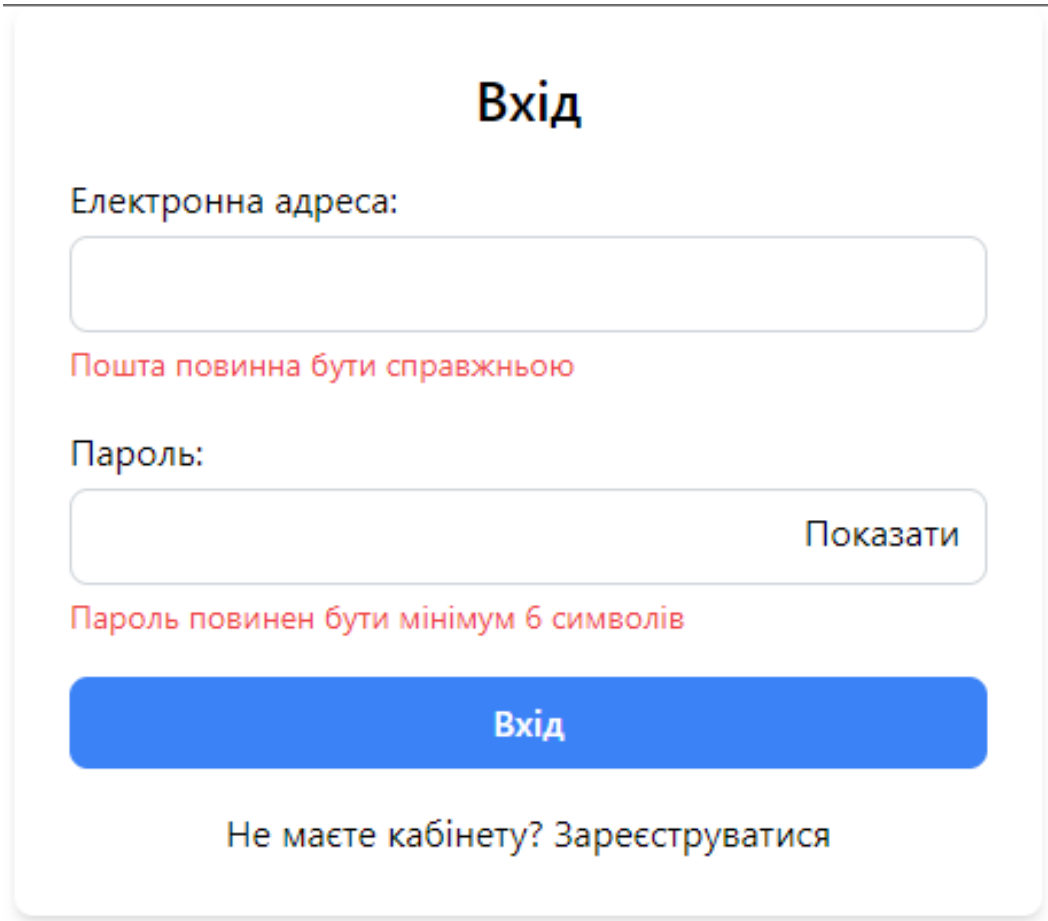

Рисунок 3.16 – Валідація полів

У вікнах авторизації та реєстрації поле для пароля усталено приховує символи, які уводить користувач (рис. 3.17).

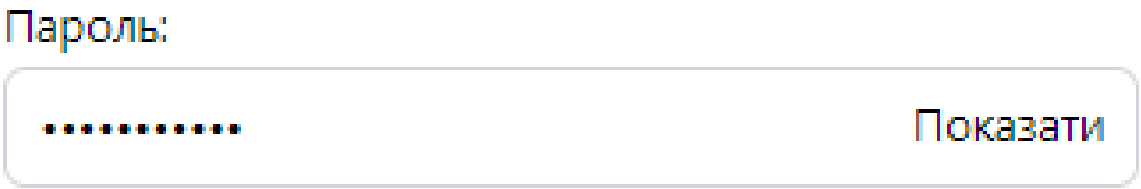

Рисунок 3.17– Усталене відображення паролю

Користувач не матиме можливість авторизуватися чи зареєструватися, якщо паролі будуть різними. Задля перевірки вірнсоті введення, можна скористуватись функцією «показати пароль», натиснувши на відповідний текст (рис. 3.18).

Рисунок 3.18 – Показ паролю

Після успішної авторизації, користувач потрапляє вікно, з котрого перейшов у вікно авторизації, для того, аби не доводилося знову відтворювати попередні дії користувача. Після авторизації користувач має можливість перейти на сторінку зі списком літератури (рис 3.19), де у нього буде можливість знайти необхідну інформацію з допомогою тексту пошука, а також фільтрами рейтингу, жанру, доступності. Також, користувач може відсортувати отриманий список згідно назви за алфавітом, або ж згідно рейтингу від найкращої до найгіршої книги. Після отримання списку літератури, які задовільнятимуть умови, які обрав користувач, він зможе обрати необхідну літературу і перейти на вебсторінку з детальною інформацією (рис 3.20). Також, якщо обрана література є доступною, то у користувача буде можливість її забронювати. Для цього у правому верхньому кутку потрібно буде натиснути на кнопку “Забронювати”.

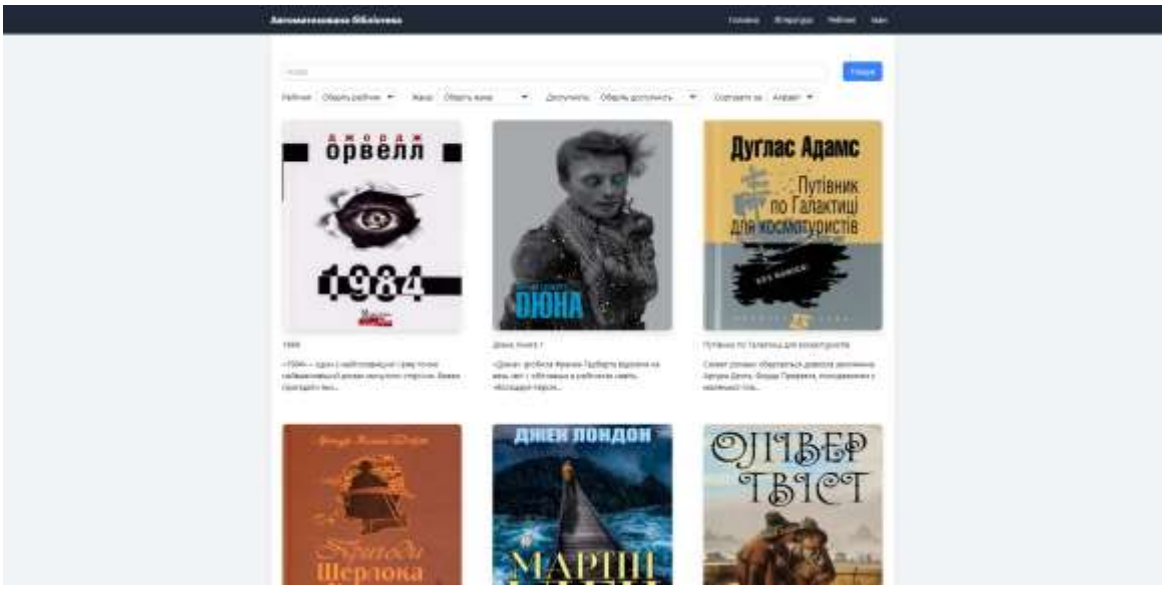

Рисунок 3.19– Сторінка з пошуком та списком літератури

Рисунок 3.20– Сторінка з детальною інформацією

Для адміністраторів передбачено можливість редагування інформації щодо літератури з метою оновлення актуальності чи доступності. Для цього у адміністраторів на місці кнопки “Забронювати” знаходиться кнопка з надписом “Редагувати” (рис. 3.21).

Рисунок 3.21 – Кнопка “Редагувати”

Окрім інформації щодо самої літератури, на сторінці також наявні відгуки інших користувачів (рис. 3.22). Завдяки цьому, є можливість одразу ознайомитися з літературою і мати розуміння про переваги і недоліки з точки зору інших читачів.

Кожен відгук містить рейтинг від 1 до 5, саму рецензію, а також, ім’я користувача, котрий залишив відгук, якщо він цього забажає, адже користувачі можуть залишати ці відгуки в анонімному режимі.

Рисунок 3.22 – Відгуки до книги

Для адміністраторів також передбачено можливість формування звітів задля формування аналітики популярних жанрів/книг (рисунок 3.23). Статистика може бути налаштованою за періодами, а саме: за день, тиждень, місяць, рік та увесь час. Також є можливість додавання та вилучення певних показників, як-от вилучення певного жанру літератури, або певного автору, чи навіть певної книги. Така статистика допоможе дослідити актуальні та загальні потреби користувачів у літературі і таким чином допоможе спрогнозувати потреби.

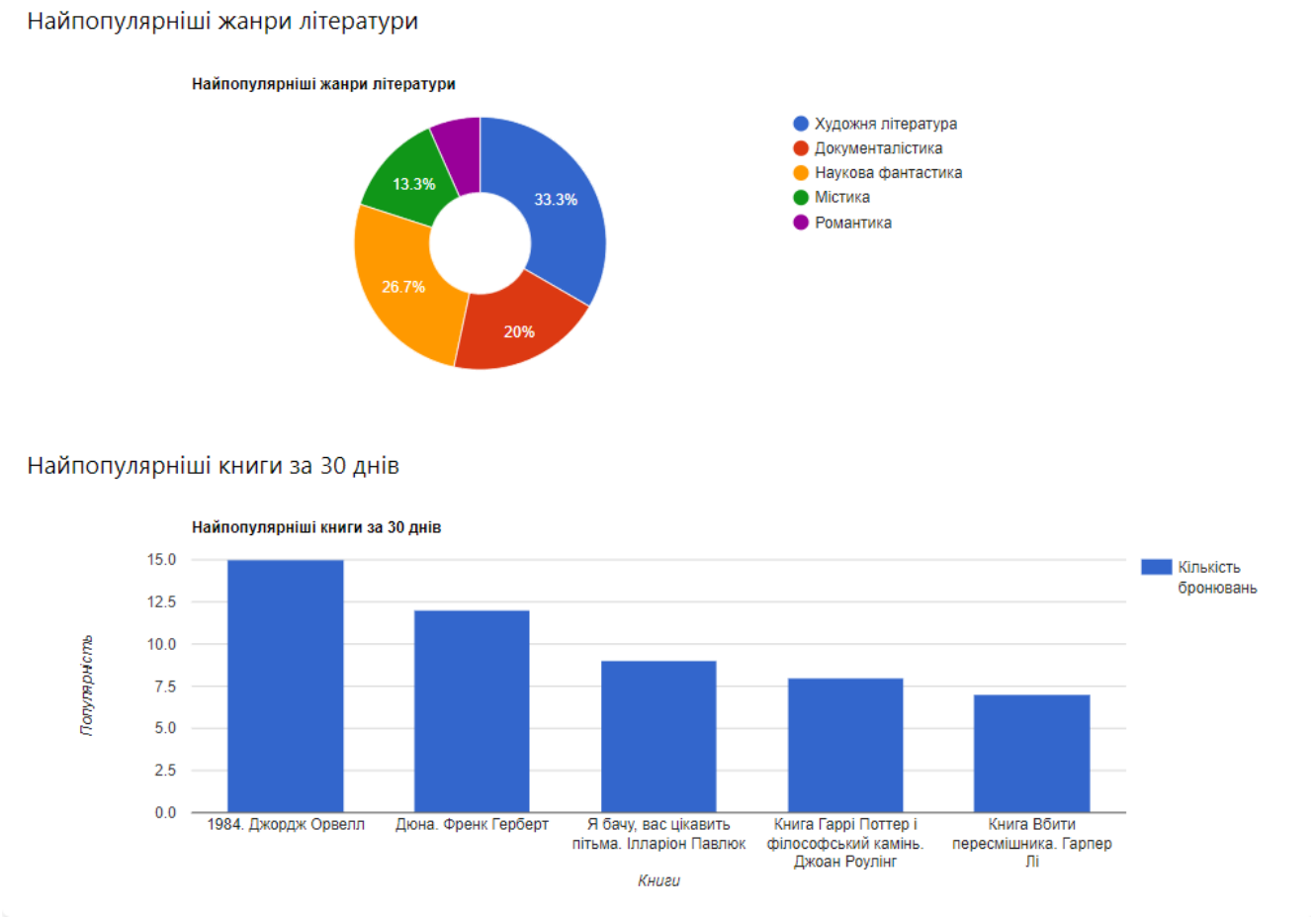

Рисунок 3.23 – Сформована статистика

Розробка графічного інтерфейсу платформи була проведена у середовищі розробки Microsoft Visual Studio Code з використанням технологій HTML, CSS, JavaScript, SvelteKit та Google Charts.

3.6 Висновки

У третьому розділі було обґрунтовано вибір мови програмування та технологій для розробки системи. Проаналізувавши потреби програмного продукту, було обрано мову програмування JavaScript з фреймворком SvelteKit. Для зручності збереження даних бібліотеки обрано Firebase. В якості бібліотеки відображення інформації у форматі діаграм було використано Google Charts. Також було описано створення ключового компоненту системи.

4 ТЕСТУВАННЯ ПРОГРАМИ

4.1 Тестування системи

Перевірка функціональності, безпеки та правильності роботи вебзастосунку є ключовим етапом у тестуванні системи. Надзвичайно важливим критерієм успішного тестування є впевненість у тому, що програма працює коректно у різних версіях браузерів.

Під час тестування проводяться функціональні тести для перевірки правильності роботи програми, що включають тести на введення та виведення даних, тести на реакцію програми на негативні вхідні дані, а також тести на відповідність виконання задач. Подальшим етапом є тести на безпеку програми, зокрема перевірка на проникнення, тести на збереження конфіденційної інформації та тести на злам програми. Для цього можуть використовуватись спеціальні програмні засоби, які дозволяють моделювати атаки на програму та виявляти її вразливості.

Браузер Chrome надає наступні інструменти для перевірки швидкодії та проблем вебсайту, як: Network, Performance та Lighthouse.

Performance дозволяє виміряти різні аспекти продуктивності вебзастосунку, такі як час завантаження, час рендерингу, час реакції на події та інші метрики . Це допомагає виявляти місця, де сайт можна оптимізувати для покращення продуктивності (рис. 4.2). Містить інформацію, які ресурси (зображення, скрипти, стилі і т.д.) завантажуються на вебсайт та як це впливає на час завантаження сторінки, а також надає інструменти для виявлення проблем, які можуть сповільнювати вебзастосунок, такі як довгі цикли JavaScript або блокування рендерингу.

Lighthouse – це інструмент, що надається у складі Chrome DevTools та який допомагає в аналізі та вдосконаленні якості вебсторінок. Він використовується для проведення аудиту вебсторінок з різних поглядів, включаючи продуктивність,

доступність, SEO та інші аспекти (рис. 4.2). Lighthouse аналізує швидкість завантаження сторінки, виявляє можливість покращення її продуктивності, такі як зменшення часу першого контенту, оптимізація величини зображень та інші оптимізації .

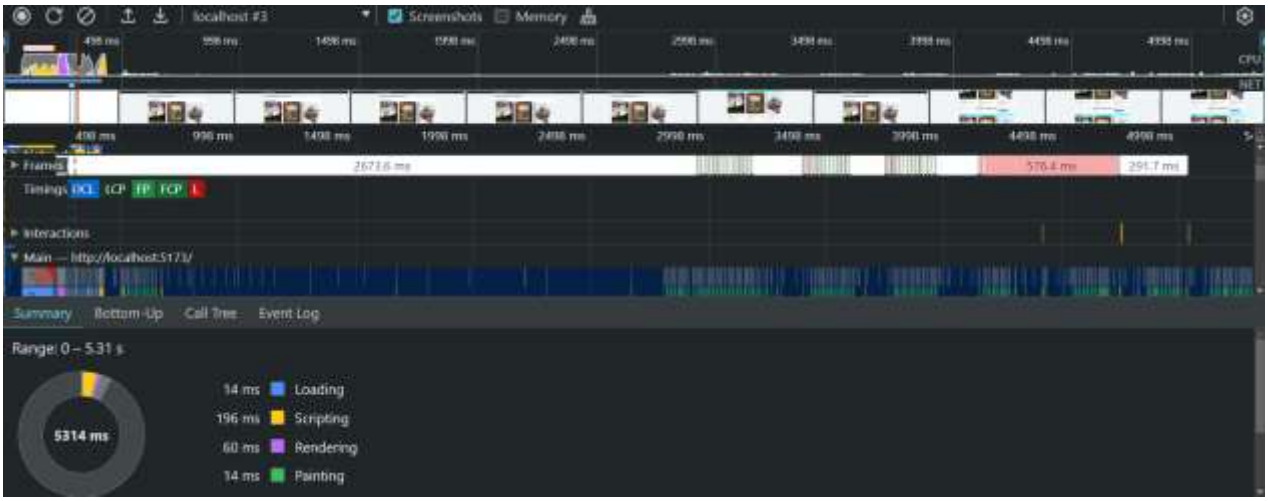

Рисунок 4.1 – Вікно тестування Performance

Інструмент перевіряє вебсторінку на відповідність стандартам доступності, виявляючи проблеми, такі як недостатньо визначені атрибути, неявність фокусу елементів та інші. Lighthouse проводить аудит вебсторінки з точки зору SEO, виявляючи можливості для покращення метатегів, швидкості завантаження та інші аспекти, що впливають на рейтинг вебсайту у пошукових системах. Для прогресивних вебдодатків Lighthouse проводить аудит на відповідність стандартам PWA та виявляє можливості для їх прокращення.

Рисунок 4.2 – Вікно тестування Lighthouse

Інструмент Network в DevTools дозволяє аналізувати мережевий трафік вебсайту (рис. 4.3). Він надає детальну інформацію про всі запити, які відправляються серверу для завантаження сторінки, включаючи ресурси, такі як HTML, CSS, JavaScript, зображення та інші.

Показує список всіх запитів, які відбуваються під час завантаження сторінки, разом з даними про час, метод, URL та статус кожного запиту. Для кожного запиту можна переглянути різні деталі, такі як заголовки запиту та відповіді, параметри запиту, тіло відповіді та інші. Фільтрація та пошук: Інструмент Network дозволяє фільтрувати запити за різними критеріями, такими як тип ресурсу (наприклад, зображення або скрипт), метод запиту (GET, POST і т.д.) та інші параметри. Також є можливість шукати конкретні запити за URL або іншими параметрами.

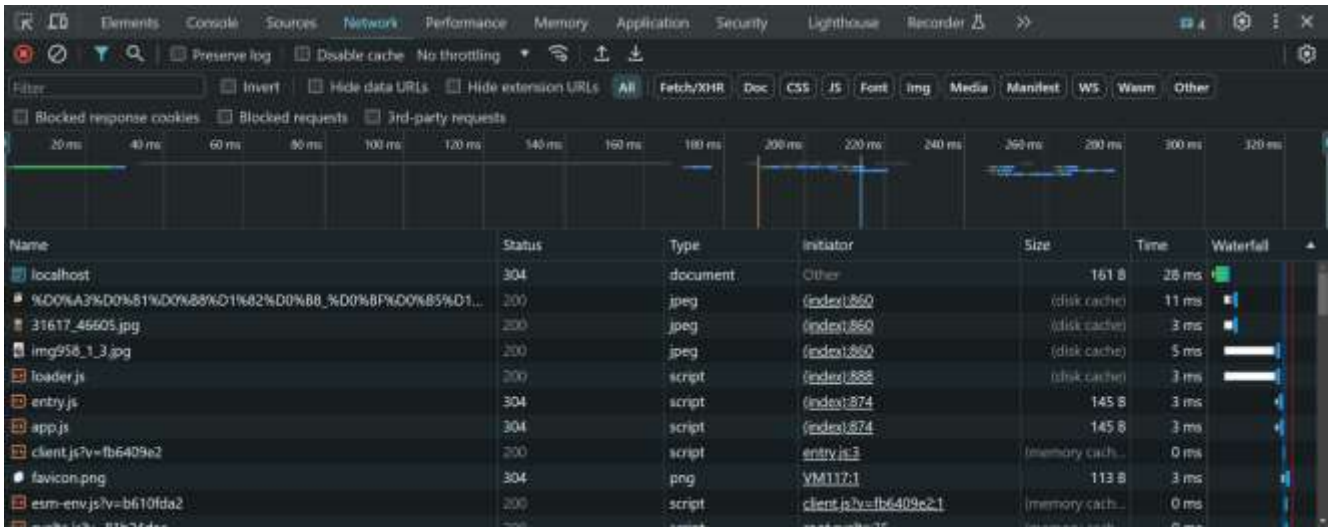

Рисунок 4.3 – Вікно тестування Network

Наступним кроком є перевірка функціоналу вебзастосунку. У першу варто перевірити вікно авторизації та реєстрації, адже проходження цього вікна є обов’язковим для усіх користувачів. А тому варто забезпечити правильну роботу цього вікна.

Якщо користувач спробує авторизуватися з неправильними даними, то побачить текст із повідомленням про помилку (рис. 4.4).

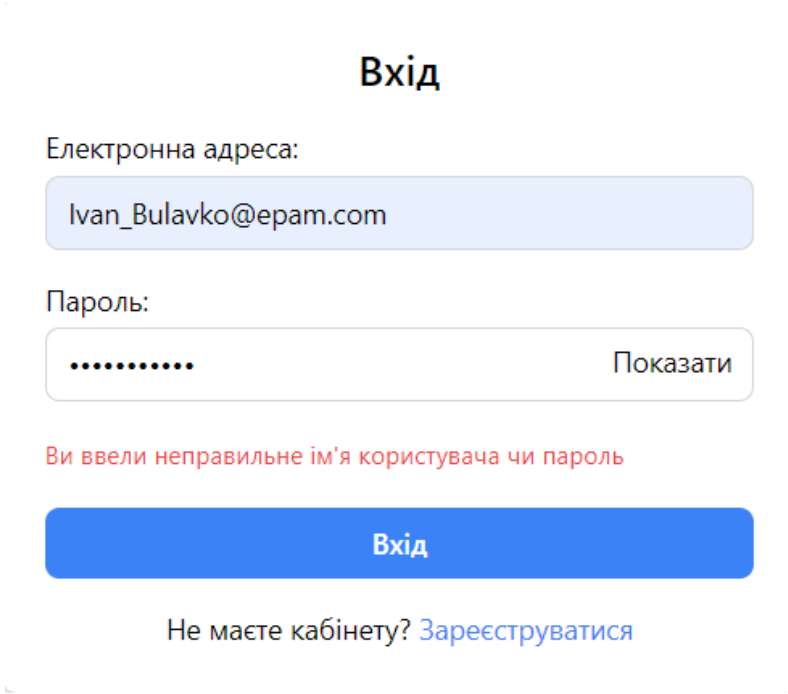

Рисунок 4.4 – Повідомлення про неправильні дані

Після перевірки правильності введених даних є можливість заповнити всі поля правильно і перевірити можливість входу в систему у випадку відсутності підключення до серверу чи мережі користувача. Після завершення перевірки система повідомить користувача про помилку та порадить перевірити правильність введених даних або з’єднання з Інтернетом (рис. 4.5).

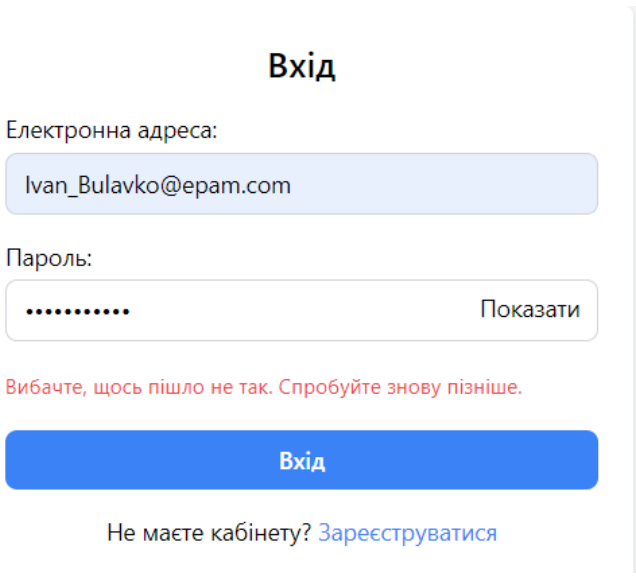

Рисунок 4.5 – Повідомлення про проблеми із під’єднанням

За умови заповнення усіх полів коректними даними та стабільним з’єднанням, автоматизована вебсистема авторизує користувача.

Валідація відіграє важливу роль і під час реєстрації. Задля безпеки користувачів, безперервності роботи додатку, та зручності використання її було додано до сторінки реєстрації. Першим полем є ім’я користувача, яке у вебсистемі є унікальним, а тому не може існувати два однакових імені (рис. 4.6).

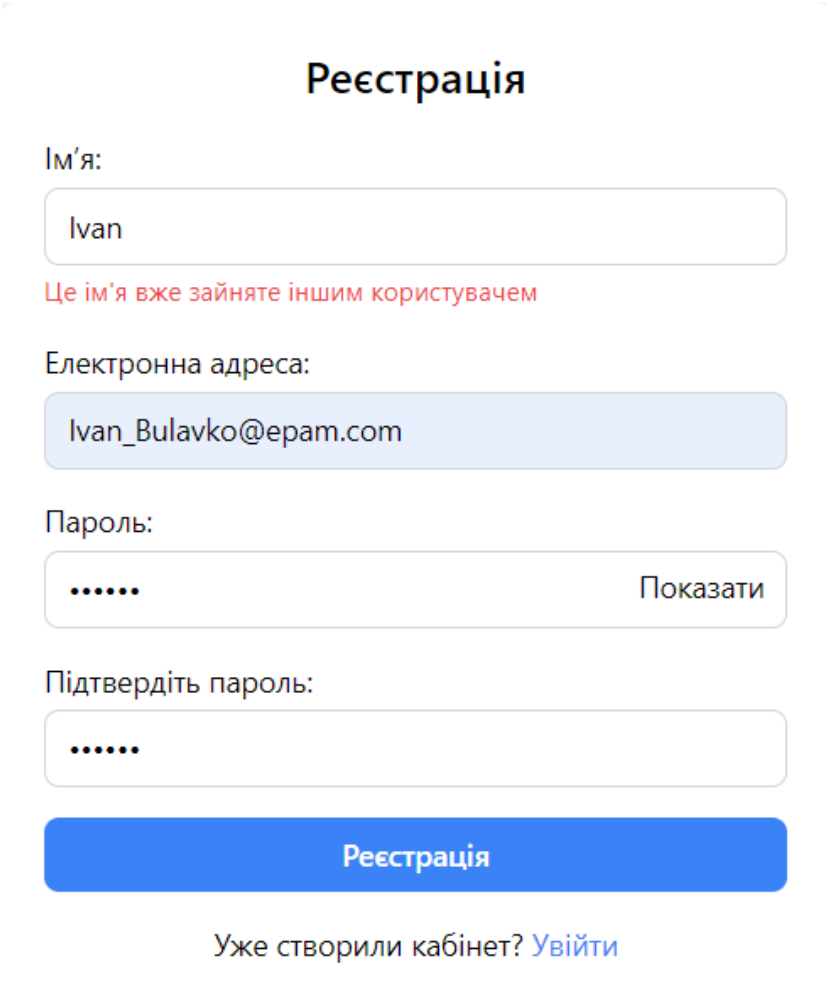

Рисунок 4.6 – Повідомлення про зайняте ім’я користувача

Важливу роль відіграє і електронна адреса. З її допомогою користувачі можуть з легкістю відновити доступ до вебсистеми, а тому необхідно, щоб при реєстрації вона була уведена правильно (рис. 4.7).

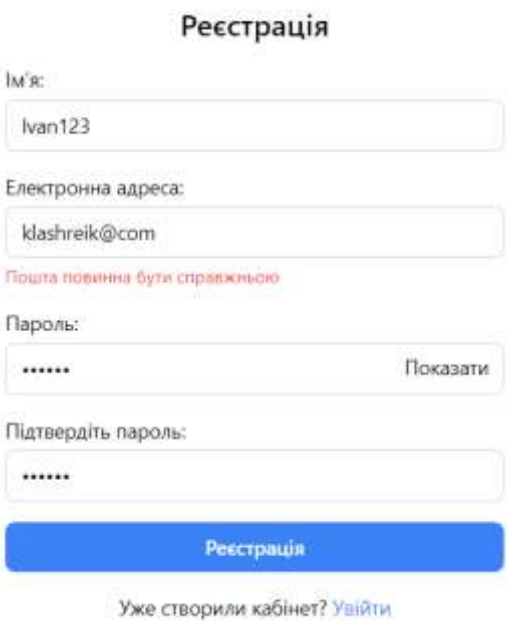

Рисунок 4.7 – Повідомлення про неправильну електронну адресу

Також важливо, щоб користувач увів пароль, який інші користувачі не зможуть легко вгадати. Тому відбувається перевірка на складність паролю. І, аби при реєстрації мінімізувати шанс на помилку з паролем, користувач повинен ввести його двічі. Якщо паролі відрізнятимуться, то буде показано відповідне повідомлення (рис. 4.8).

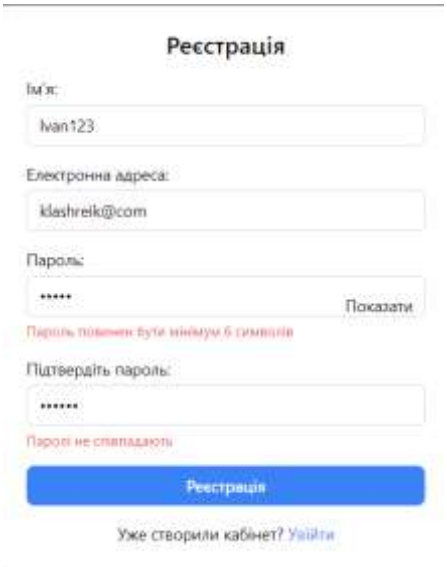

Рисунок 4.8 – Повідомлення про помилки з паролем

Отже, проведено тестування атоматизованої вебсистеми управління бібліотекою. У результаті тестування отримано гарні показники продуктивності інструментом Performance, оптимізованості кількості завантажуваних ресурсів з допомгою інструмента Network та загальний огляд вебзастосунку інструментом Lighthouse. Вебзастосунок забезпечує безперебійну роботу з допомогою проведення валідації задля попередження помилок у системі та для зручності користувачів.

4.2 Розробка інструкції користувача

Інструкція користувача включає в себе встановлення технічних вимог для запуску програмного продукту. Інформацію про мінімальну та рекомендовану конфігурацію серверного комп'ютера можна знайти у таблицях 4.1 та 4.2.

Таблиця 4.1 – Мінімальні конфігураційні вимоги до апаратного забезпечення

Процесор	Intel Core i5-4460S 2.9 GHz або AMD Athlon X4 870K 3.9 GHz
Оперативна пам'ять	4 GB
Пропускна здатність мережі	100 Mbps
Вільного місця на диску	2ГБ

Таблиця 4.2 – Рекомендовані конфігураційні вимоги до апаратного забезпечення

Процесор	Intel Core i7-6700K 4.0 GHz або AMD Ryzen 5 1600 3.2 GHz
Оперативна пам'ять	8 GB
Пропускна здатність мережі	300 Mbps
Вільного місця на диску	4ГБ

Для початку роботи користувачеві варто потрапити на головну сторінку вебсистеми, на котрій зображено популярні книги бібілотеки, а також можна перейти на різні сторінки з допомогою навігаційного меню (рис. 4.9).

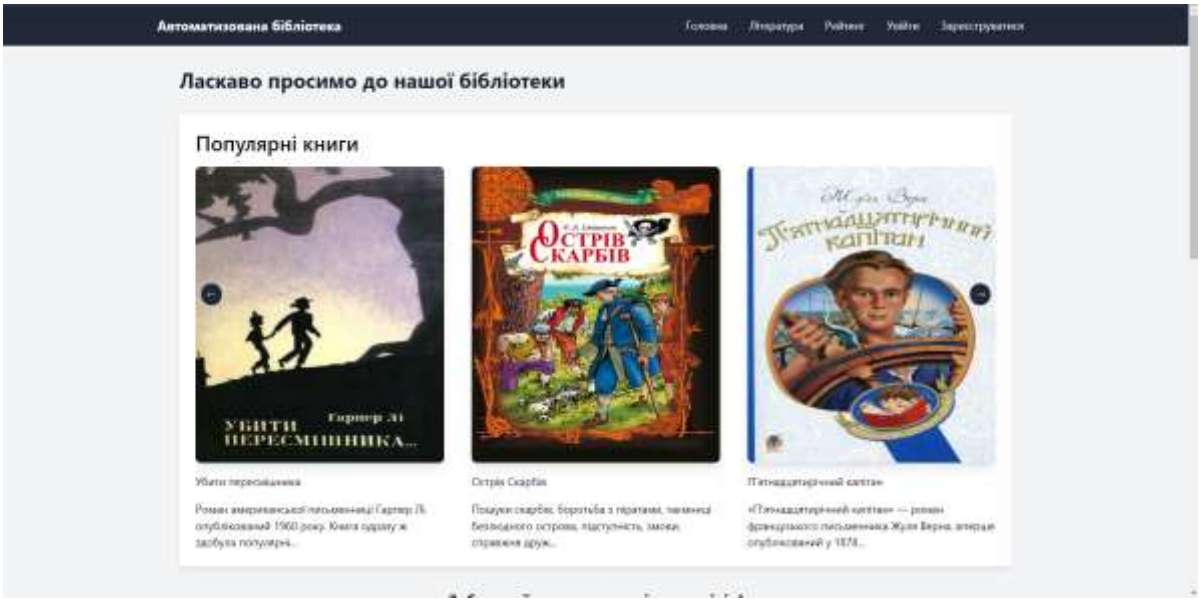

Рисунок 4.9 – Головна сторінка вебзастосунку

Для продовження безперервної роботи із вебсистемою користувач повинен авторизуватися (рис. 4.10).

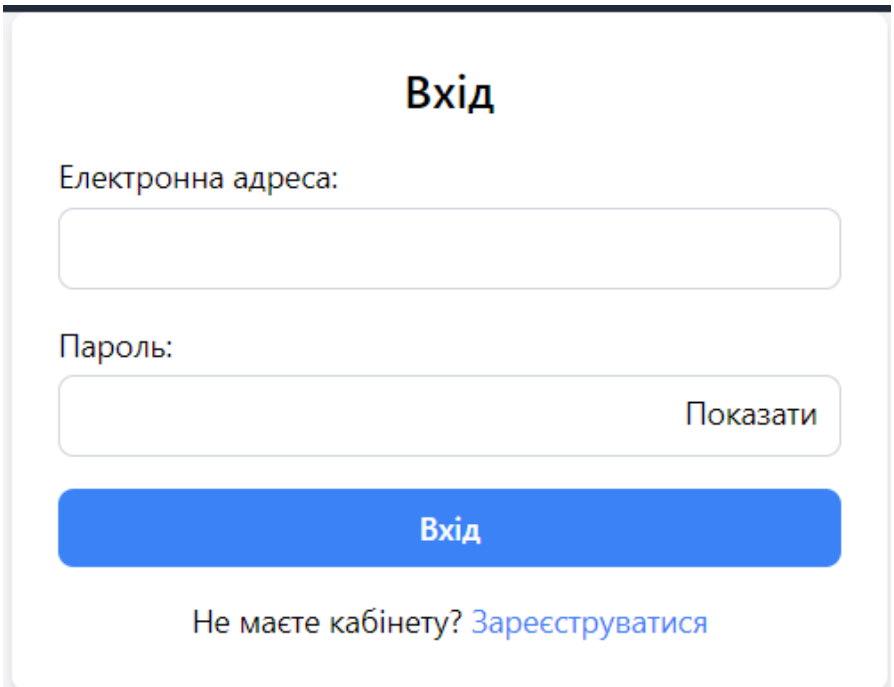

Рисунок 4.10 – Сторінка авторизації

Якщо ж користувач не має власного облікового запису, то він може створити його на сторінці реєстрації (рис. 4.11).

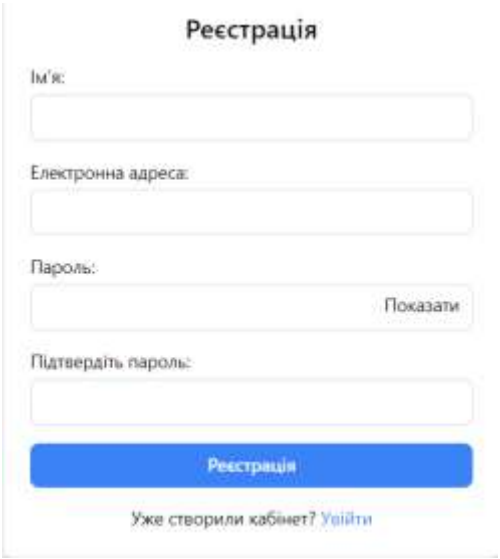

Рисунок 4.11 – Сторінка реєстрації

Повністю авторизувавшись, користувач може переглянути список літератури на створеній для цього сторінці (рис. 4.12). У користувача є можливість скористатися фільтрами та елементом уведення пошукового запиту задля знаходження необхідної літератури.

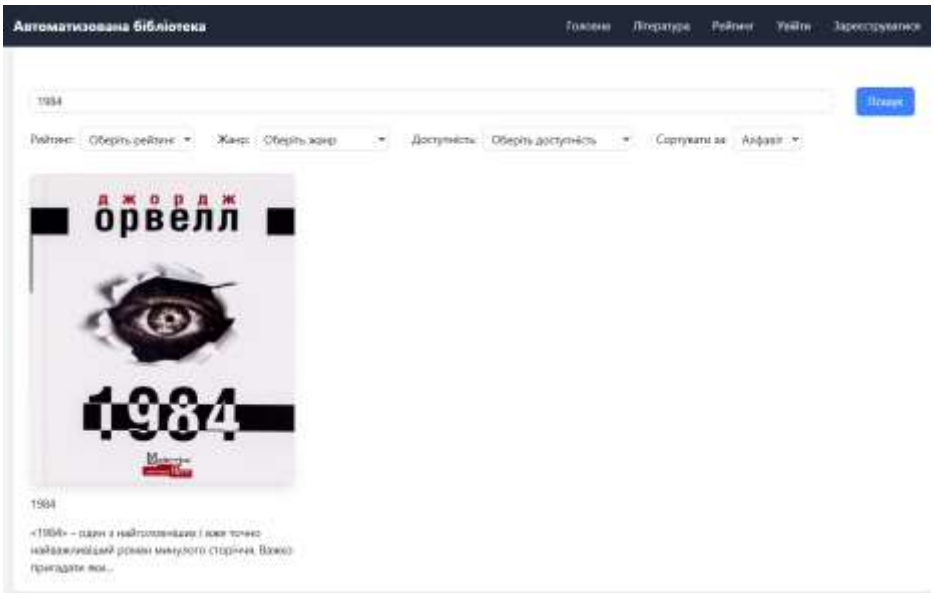

Рисунок 4.12 – Сторінка літератури

Якщо користувача зацікавила якась книга, то він може відвідати її сторінку для отримання повної інформації з відгуками інших користувачів, а також забронювати літературу, якщо вона є в наявності (рис. 4.13).

Рисунок 4.13 – Сторінка обраної книги

Якщо користувач авторизувався як адміністратор, то в нього є можливість створення графіків аналітики популярності літератури за датою для жанрів та кожної окремої книги. Звіт з популярності жанрів є звичайною круговою діаграмою з інформацією про кількість бронювань за вказаний період (рис. 4.14).

Звіт найпопулярнішої літератури демонструє найбільш популярні книги у форматі стовпчастої діаграми з інформацією про кількість бронювань за вказаний період (рис. 4.15).

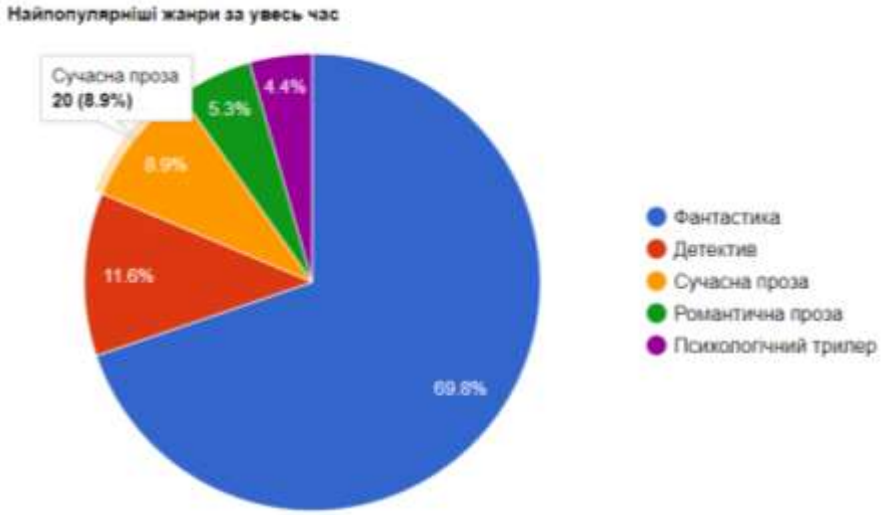

Рисунок 4.14 – Кругова діаграма популярності жанрів

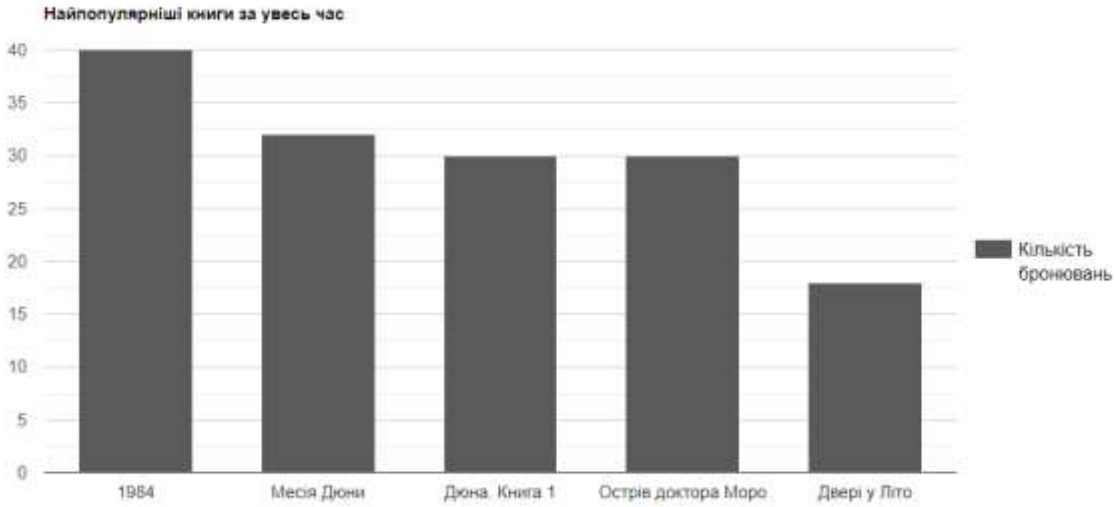

Рисунок 4.15 – Стовпчаста діаграма популярності літератури

Таким чином відбувається взаємодія із автоматизовано вебсистемою керування бібліотекою.

4.3 Висновки

У четвертому розділі відбулося тестування програмних засобів системи, включаючи аналіз ресурсів вебзастосунку, а також перевірку валідації полів під час реєстрації та авторизації користувачів.

ВИСНОВКИ

У бакалаврській дипломній роботі було розроблено програмні засоби реалізації автоматизованої вебсистеми керування бібліотекою.

Для розробки було використано середовище програмної розробки Microsoft Visual Studio Code.

Бакалаврська дипломна робота оформлена згідно методичних вказівок [21].

Під час виконання бакалаврської дипломної роботи було проаналізовано стан розвитку систем управління бібліотекою на сьогоднішній день. Було розглянуто основні аналоги програмного продукту. Було визначено їхні недоліки та порівняно з власним програмним застосунком. Базуючись на результатах порівняльного аналізу, було встановлено основні задачі бакалаврської дипломної роботи.

Під час аналізу технологій розробки було обґрунтовано вибір мови програмування JavaScript та фреймворк SvelteKit. Також було розглянуто переваги використання Google Charts для візуалізації даних.

Відповідно до поставлених задач було:

– розроблено модель автоматизованої вебсистеми управління бібліотекою, яка міститиме основний функціонал системи;

- розроблено метод формування і відображення звітності;

– розроблено алгоритм бронювання доступної літератури з допомогою онлайн запиту;

– розроблено зручний та адаптивний графічний інтерфейс вебсистеми;

- розроблено програмне забезпечення вебсистеми;

– реалізовано модуль статичних досліджень для створення аналітичної статистики вебсистеми;

– здійснено тестування вебсистеми згідно поставлених задач.

Було розроблено схеми загального алгоритму роботи системи та модуля тестування. Також було розроблено модель системи з використанням UML діаграм.

Тестування програми довело повну працездатність даного програмного продукту та відповідність поставленому технічному завданню. Також було розроблено інструкцію користувача.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Що таке електронна бібліотека? Її основні задачі та особливості [Електронний ресурс] – Режим доступу до ресурсу: https://elect-library.blogspot.com/2017/03/blog-post_10.html.

2. Бібліотека як потужний інформаційний центр сучасності [Електронний ресурс] – Режим доступу до ресурсу: http://conference.nbuv.gov.ua/report/view/id/422

3. Булавко І. О. Автоматизована вебсистема управління бібліотекою / І. О. Булавко, В. В. Войтко, Г. О. Черноволик, Л. М. Круподьорова, А. В. Денисюк // Матеріали LIII Науково-технічної конференції підрозділів Вінницького національного технічного університету, Вінниця, 2024. [Електронний ресурс] – Режим доступу до ресурсу: https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2024/paper/view/20419/16966.

4. A Library Management System is important for small and big libraries [Електронний ресурс] – Режим доступу до ресурсу: https://slimkm.com/blog/6-reasons-a-library-management-system-is-important-for-small-and-big-libraries.

5. What are some interactive features you can add to a library website for better user experience? [Електронний ресурс] – Режим доступу до ресурсу: https://www.linkedin.com/advice/0/what-some-interactive-features-you-can-add-library-am3mc.

6. Aleph [Електронний ресурс] – Режим доступу до ресурсу: https://exlibrisgroup.com/products/aleph-integrated-library-system.

7. Бібліотечна система UniLib [Електронний ресурс] – Режим доступу до ресурсу: http://unilib.com.ua/web/uk.

8. Koha Library Software [Електронний ресурс] – Режим доступу до ресурсу: https://koha-community.org.

9. What are Microservices? [Електронний ресурс] – Режим доступу до ресурсу: https://aws.amazon.com/microservices.

10. HTML Introduction [Електронний ресурс] – Режим доступу до ресурсу: https://www.w3schools.com/html/html_intro.asp

11. CSS specifications [Електронний ресурс] – Режим доступу до ресурсу: https://www.w3.org/Style/CSS/specs.en.html.

12. JavaScript [Електронний ресурс] – Режим доступу до ресурсу: https://developer.mozilla.org/en-US/docs/Web/JavaScript.

13. SvelteKit [Електронний ресурс] – Режим доступу до ресурсу: https://kit.svelte.dev.

14. Charts - Google for Developers [Електронний ресурс] – Режим доступу до ресурсу: https://developers.google.com/chart.

15. Paletton - The Color Scheme Designer [Електронний ресурс] – Режим доступу до ресурсу: https://paletton.com.

16. Client-side form validation [Електронний ресурс] – Режим доступу до ресурсу: https://developer.mozilla.org/en-US/docs/Learn/Forms/Form_validation.

17. What is a UML diagram? [Електронний ресурс] – Режим доступу до ресурсу: https://miro.com/diagramming/what-is-a-uml-diagram.

18. Firebase [Електронний ресурс] – Режим доступу до ресурсу: https://firebase.google.com/.

19. The Importance of UI/UX Design [Електронний ресурс] – Режим доступу до ресурсу: https://www.igexsolutions.com/blog/the-importance-of-ui-ux-design.

20. Web analytics Електронний ресурс] – Режим доступу до ресурсу: https://www.hotjar.com/web-analytics.

21. Методичні вказівки до виконання бакалаврських кваліфікаційних робіт для студентів студентів спеціальності 121 "Інженерія програмного забезпечення" / Уклад. О. Н. Романюк, Г. О. Черноволик – Вінниця: ВНТУ, 2022. – 52с.

ДОДАТКИ

62

Додаток А – Технічне завдання

Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
Романюк О. Н.
«12» березня 2024 р.

Технічне завдання
на бакалаврську дипломну роботу
«Розробка методу та засобів реалізації автоматизованої вебсистеми управління бібліотекою»
за спеціальністю 121 – Інженерія програмного забезпечення

Керівник бакалаврської дипломної роботи:
_____ к.т.н., доц. Войтко В. В.
«12» березня 2024 р.

Виконав:
_____ студент гр. 1ПІ-20б Булавко І. О.
«12» березня 2024 р.

Вінниця – 2024 року

1. Найменування та галузь застосування

Бакалаврська дипломна робота: «Розробка методу та засобів реалізації автоматизованої вебсистеми управління бібліотекою».

Галузь застосування – автоматизовані вебсистеми.

2. Підстава для розробки.

Завдання на роботу, яке затверджене на засіданні кафедри програмного забезпечення – наказ №80 від «11» березня 2024 р по ВНТУ про закріплення тем БДР.

3. Мета та призначення розробки.

Метою дослідження є покращення обслуговування користувачів бібліотеки та ведення обліку за рахунок розробки та використання спеціалізованої вебсистеми, яка забезпечує підвищення рівня автоматизації керування наявною літературою, спрощення процесів комунікації користувачів при бронюванні книг та візуалізацію звітності, що полегшує процеси аналізу та проведення статистичних досліджень і формування звітів.

Призначення роботи – розробка програмного продукту для автоматизації управління бібліотекою.

4. Вихідні дані для проведення НДР

1. A Library Management System is important for small and big libraries [Електронний ресурс] – Режим доступу до ресурсу: https://slimkm.com/blog/6-reasons-a-library-management-system-is-important-for-small-and-big-libraries.

2. What is a UML diagram? [Електронний ресурс] – Режим доступу до ресурсу: https://miro.com/diagramming/what-is-a-uml-diagram.

3. Бібліотека як потужний інформаційний центр сучасності [Електронний ресурс] – Режим доступу до ресурсу: http://conference.nbuv.gov.ua/report/view/id/422

4. JavaScript [Електронний ресурс] – Режим доступу до ресурсу: https://developer.mozilla.org/en-US/docs/Web/JavaScript.

5. Технічні вимоги

Серверний комп'ютер з 64-розрядним (x64) процесором та тактовою частотою 2 ГГц. Об'єм оперативної пам'яті 4 ГБ, розмір жорсткого диску 128 ГБ. Операційна система Windows 10.

З клієнтського боку: ОС Windows, Android, IOS та будь-який браузер.

6. Конструктивні вимоги.

Вебзастосунок має відповідати ергономічним та технічним вимогам. Текстова та графічна документація повинна відповідати стандартам України.

7. Перелік технічної документації, що пред'являється по закінченню робіт:

- пояснювальна записка до бакалаврської дипломної роботи;
- технічне завдання;
- лістинги програми.

8. Вимоги до рівня уніфікації та стандартизації

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ.

9. Стадії та етапи розробки

№ з/п	Назва етапів бакалаврської дипломної роботи	Строк виконання етапів роботи
1	Аналіз розвитку автоматизованих вебсистем управління бібліотекою та постановка задач дослідження	14.03.24 – 22.03.24
2	Розробка методу, моделі та алгоритмів роботи вебсистеми	22.03.24 – 20.04.24
3	Розробка програмних модулів вебсистеми	20.04.24 – 03.05.24
4	Тестування програми	03.05.24 – 23.05.24
5	Оформлення матеріалів до захисту БДР	23.05.24 – 30.05.24

10 Порядок контролю та прийняття.

Всі етапи бакалаврської роботи контролюються науковим керівником згідно плану по виконанню роботи. Прийняття бакалаврської кваліфікаційної роботи здійснюється ДЕК, затвердженою зав. кафедрою згідно з графіком захисту.

66

Додаток Б – ПРОТОКОЛ ПЕРЕВІРКИ БАКАЛАВРСЬКОЇ ДИПЛОМНОЇ РОБОТИ НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: «Розробка методу та засобів реалізації автоматизованої вебсистеми управління бібліотекою»

Тип роботи: БДР

Підрозділ: кафедра програмного забезпечення, ФІТКІ

Науковий керівник: Войтко В. В.

Оригінальність	94,2%
Схожість	5,8%

Аналіз звіту подібності

■ **Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.**

□ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її автора. Роботу направити на доопрацювання.

□ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку Черноволик Г. О.

Ознайомлені з повним звітом подібності, який був згенерований системою Unichek

Автор роботи Булавко І. О.

Керівник роботи Войтко В. В.

Додаток В – Лістинг програми

```
<script>
	import Slider from '../components/Slider/Slider.svelte';
	import Auth from '../components/Auth/Auth.svelte';
</script>

<svelte:head>
	<title>Автоматизована система управління бібліотекою</title>
</svelte:head>

<main class="bg-gray-100 min-h-screen">
	<div class="max-w-7xl mx-auto px-4 py-8 sm:px-6 lg:px-8">
		<h1 class="text-3xl font-bold text-gray-900 mb-8">Ласкаво просимо до нашої бібліотеки</h1>

		<div class="bg-white p-6 rounded-lg shadow-md">
			<h2 class="text-3xl font-semibold mb-4">Популярні книги</h2>
			<Slider />
		</div>
		<div class="flex items-center justify-center">
			<div class="my-8">
				<h2 class="text-2xl font-bold mb-4 flex items-center justify-center">
					Забронюйте книгу в декілька кліків!
				</h2>
				<form class="flex flex-col sm:flex-row gap-4">
					<input
						type="text"
						placeholder="Ім'я"
						class="w-full sm:w-auto px-4 py-2 rounded-md shadow-sm focus:outline-none focus:ring-2 focus:ring-blue-500"
					/>
					<input
						type="email"
						placeholder="Електронна пошта"
						class="w-full sm:w-auto px-4 py-2 rounded-md shadow-sm focus:outline-none focus:ring-2 focus:ring-blue-500"
					/>
					<input
```

```
                                    type="password"
                                    placeholder="Пароль"
                                    class="w-full sm:w-auto px-4 py-2 rounded-md
shadow-sm focus:outline-none focus:ring-2 focus:ring-blue-500"
                              />
                              <button
                                    type="submit"
                                    class="w-full  sm:w-auto  bg-blue-500  text-
white px-4 py-2 rounded-md shadow-sm hover:bg-blue-600 transition-colors focus:outline-none
focus:ring-2 focus:ring-blue-500"
                                    >Створити користувача
                              </button>
                        </form>
                  </div>
            </div>
            <div class="bg-white p-6 rounded-lg shadow-md">
                  <h2 class="test text-xl font-semibold mb-4">Останні новинки</h2>
                  <Slider />
            </div>

            <Auth />
      </div>
</main>

<style>
      .test {
            background-color: aqua;
      }
</style>

<script>
      import '../app.css';
</script>

<main class="bg-gray-100 min-h-screen">
      <nav class="bg-gray-800 text-white p-4">
            <div class="max-w-7xl mx-auto flex justify-between items-center">
                  <div class="flex items-center">
                        <a   href="/"   class="text-xl   font-bold">Автоматизована
бібліотека</a>
```

```
                </div>
                <div class="flex items-center">
                        <a href="/" class="mx-4">Головна</a>
                        <a href="/about" class="mx-4">Література</a>
                        <a href="/contact" class="mx-4">Рейтинг</a>
                        <a href="/login" class="mx-4">Увійти</a>
                        <a href="/register" class="mx-4">Зареєструватися</a>
                </div>
        </div>
    </nav>
    <slot />
</main>


<script>
    let isRegistering = false;
    let username = '';
    let email = '';
    let password = '';
    let confirmPassword = '';
    let showPassword = false;

    let errors = {};

    function handleSubmit() {
        errors = {};

        if (!username) {
            errors.username = 'Ім’я користувача повинно бути заповненим';
        }

        if (!email.includes('@')) {
            errors.email = 'Пошта повинна бути справжньою';
        }

        if (password.length < 6) {
            errors.password = 'Пароль повинен бути мінімум 6 символів';
        }

        if (isRegistering && password !== confirmPassword) {
```

```
                errors.confirmPassword = 'Паролі не співпадають';
            }

            if (Object.keys(errors).length === 0) {
                if (isRegistering) {
                    // Handle registration logic
                    console.log(
                        'Registering with username:',
                        username,
                        'email:',
                        email,
                        'and password:',
                        password
                    );
                } else {
                    // Handle login logic
                    console.log('Logging in with email:', email, 'and
password:', password);
                }
            }
        }
    </script>

    <div class="max-w-md mx-auto p-6 bg-white shadow-md rounded-lg">
        <h2 class="text-2xl font-semibold mb-4 text-center">{isRegistering ?
'Реєстрація' : 'Вхід'}</h2>

        <form on:submit|preventDefault={handleSubmit}>
            {#if isRegistering}
                <div class="mb-4">
                    <label for="username" class="block mb-1">Ім’я:</label>
                    <input
                        type="text"
                        id="username"
                        bind:value={username}
                        class="w-full px-3 py-2 border border-gray-300
rounded-lg focus:outline-none focus:border-blue-500"
                    />
                    {#if errors.username}<p class="text-red-500 text-sm mt-
1">{errors.username}</p>{/if}
```

```
                </div>
            {/if}

            <div class="mb-4">
                <label for="email" class="block mb-1">Електронна адреса:</label>
                <input
                    type="email"
                    id="email"
                    bind:value={email}
                    class="w-full px-3 py-2 border border-gray-300 rounded-lg
focus:outline-none focus:border-blue-500"
                />
                {#if    errors.email}<p    class="text-red-500    text-sm    mt-
1">{errors.email}</p>{/if}
            </div>

            <div class="mb-4 relative">
                <label for="password" class="block mb-1">Пароль:</label>
                {#if showPassword}
                    <input
                        type="text"
                        id="password"
                        bind:value={password}
                        class="w-full  px-3  py-2  border  border-gray-300
rounded-lg focus:outline-none focus:border-blue-500"
                    />
                {:else}
                    <input
                        type="password"
                        id="password"
                        bind:value={password}
                        class="w-full  px-3  py-2  border  border-gray-300
rounded-lg focus:outline-none focus:border-blue-500"
                    />
                {/if}
                <button
                    type="button"
                    class="absolute inset-y-0 right-0 top-6 px-3 py-2"
                    on:click={() => (showPassword = !showPassword)}
                    >{showPassword ? 'Приховати' : 'Показати'}</button
```

```
                        >
                        {#if    errors.password}<p    class="text-red-500    text-sm    mt-
1">{errors.password}</p>{/if}
                    </div>

                    {#if isRegistering}
                        <div class="mb-4">
                            <label     for="confirmPassword"     class="block     mb-
1">Підтвердіть пароль:</label>
                            <input
                                type="password"
                                id="confirmPassword"
                                bind:value={confirmPassword}
                                class="w-full   px-3   py-2   border   border-gray-300
rounded-lg focus:outline-none focus:border-blue-500"
                            />
                            {#if errors.confirmPassword}<p class="text-red-500 text-sm
mt-1">
                                    {errors.confirmPassword}
                                </p>{/if}
                        </div>
                    {/if}
                    <button
                        type="submit"
                        class="bg-blue-500 hover:bg-blue-700 text-white font-semibold px-
4 py-2 rounded-lg w-full focus:outline-none focus:shadow-outline"
                        >{isRegistering ? 'Реєстрація' : 'Вхід'}</button
                    >
                </form>

                <p class="mt-4 text-center">
                    {isRegistering ? 'Уже створили кабінет? ' : 'Не маєте кабінету? '}
                    <a
                        href="#"
                        on:click={() => (isRegistering = !isRegistering)}
                        class="text-blue-500 hover:underline">{isRegistering ? 'Увійти' :
'Зареєструватися'}</a
                    >
                </p>
            </div>
```

```
<script lang="ts">
	import { onMount } from 'svelte';

	export let dataArray = [];
	export let chartOptions = {};

	onMount(() => {
		google.charts.load('current', { packages: ['corechart'] });

		google.charts.setOnLoadCallback(drawChart);

		function drawChart() {
			var data = new google.visualization.DataTable();
			data.addColumn('string', 'Книга');
			data.addColumn('number', 'Кількість бронювань');
			data.addRows(dataArray.map((item) => [item[0].title, item[1]]));

			var chart = new google.visualization.ColumnChart(document.getElementById('chart_div'));

			chart.draw(data, chartOptions);
		}
	});
</script>

<div id="chart_div"></div>


<script>
	import Review from './Review.svelte';
	export let book = {
		title: 'Book Title',
		author: 'Author Name',
		description: 'Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla nec velit urna.',
		imageUrl: 'https://via.placeholder.com/300',
		genre: 'Антиутопія',
		rating: 4.5,
		availability: true,
```

reviews: [
{
author: 'Анонімний користувач',
content:
'Неймовірна пророчість Завжди, коли чуєш про якісь справді важливі для читання і розуміння книжки, спадає на думку лише одне: читати чи ні. Я не та людина, яка біжить читати все, про що говорять. Але колись все ж вирішила прочитати "1984" і це була дуже захоплива книжкова подорож. Хтось в коментарях писав, що з перших рядків поринув у читання, що це було дуже круто. Для мене перші декілька сторінок були пеклом: нічого не зрозуміло, тому що дія всередині святу, який вже існує в книжці як самостійний елемент. Було важкувато спочатку, декілька разів я поверталась до початку, але потім почалась якась магія. Після того, як розумієш світ, що описав Джордж Орвелл у своїй книжці, читання йде набагато швидше. І ось тоді починається те саме захоплення, про яке кажуть майже всі читачі. Мені сподобався головний герой, який намагався думати, писати, змінити свою рутину на краще. У світі, де немає ніякої правди, де не існує історії, бо вони переписується кожного дня, він знайшов у собі сили бути кимось іншим. Та мені більше подобається не момент з тим, як він до цього приходить, а момент, коли він вже у руках влади і його намагаються зломити. Для мене це було найцікавішим для читання, мені хотілось думати, аналізувати, складати власті картинки того, що відбувалось. Але все ж у книжки є один суттєвий мінус - вона виявилась до неймовірної точності пророчою. І це дуже лякає.',
rating: 5
},
{
author: 'Анонімний користувач',
content:
'Неймовірна антиутопія, що захопила світ Джордж Орвелл першим приходить в голову, якщо хтось заговорить про антиутопії. Цей видатний автор неймовірно точно передає гнітючу атмосферу, прогнивше суспільство і керуючу верхівку тоталітаризму. Його антиутопія "1984" є однією з найкращих книг цього жанру і, можу сказати, не дарма. Це творіння наповнене злим і темним, брудним і огидним, атмосфера ніби світ раптово втратив кольори. Головний герой Уінстон - простий робітник з проблемами зі здоров’ям, жертва системи Старшого брата. Він стирає старі газети, переписує звіти... Але як це жити в брехні? Коли один рік повторювався скільки разів, що точно не можеш сказати свій вік. У Лондоні, столиці Океанії, це звична справа - не знати правди. Коли тобі повторюють одне й те ж тисячі а то і сотні тисяч разів, справді починаєш вірити, що 2+2=5. Океанія - це місце без думок, бо за думкозлочини тебе зітруть з лиця землі. Не так поглянеш, скажеш хоч слово чи виразиш недостатньо ненависті до ворогів режиму... Та від цього не сховатися. Дітей з малечку навчають здавати та прослуховувати будь-кого, навіть батьків. У квартирах стоять екрани з камерами та звукозаписуючими пристроями, що породжує постійний страх бути поміченим кимось з наглядачів. Але налякала мене зовсім не атмосфера, а розуміння, що подібні сценарії можливі

```
і у 21 столітті і деякі країни вже до цього наближаються... Хеппі енду не варто чекати. "1984" це не про веселий чи заплюючий сюжет. Це про жорстку реальність тоталітаризму, яка змусить вас моментами закривати книгу від гнітючості та відрази, але точно залишить слід у розумі',
                                rating: 4.5
                        }
                ]
        };
    </script>

    <div class="max-w-7xl mx-auto p-6 bg-white shadow-md rounded-lg flex flex-col text-2xl">
        <div class="flex gap-8">
            <div class="w-2/3">
                <img src={book.imageUrl} alt={book.title} class="w-full rounded-lg mb-4" />
            </div>

            <div class="w-2/3 px-4">
                <div class="flex align-end justify-end">
                    <button
                        type="submit"
                        class="w-full sm:w-auto bg-blue-500 text-white px-4 py-2 rounded-md shadow-sm hover:bg-blue-600 transition-colors focus:outline-none focus:ring-2 focus:ring-blue-500"
                        >Забронювати
                    </button>
                </div>

                <h2 class="text-2xl font-semibold mb-2">{book.title}</h2>
                <p class="text-gray-600 mb-4">{book.author}</p>

                <p class="text-gray-800 whitespace-pre-line">{book.description}</p>

                <div class="mt-4 flex items-center">
                    <span class="text-gray-600 mr-2">Жанр: {book.genre}</span>
                    <span class="text-gray-600 mr-2">Рейтинг: {book.rating}</span>
                    <span class="text-gray-600"
```

```
                                    >Доступність: {book.availability ? 'Доступна для броні' : 'Не доступна для броні'}</span
                                    >
                              </div>
                        </div>
                  </div>
                  <h3 class="text-xl font-semibold my-6">Відгуки:</h3>
                  {#each book.reviews as review}
                        <Review {review} />
                  {/each}
            </div>

            <script>
                  export let filters = {
                        name: '',
                        rating: '',
                        genre: '',
                        availability: '',
                        sort: 'alphabet'
                  };

                  const ratings = ['1 зірка', '2 зірки', '3 зірки', '4 зірки', '5 зірок'];
                  const genres = [
                        'Художня література',
                        'Документалістика',
                        'Наукова фантастика',
                        'Містика',
                        'Романтика'
                  ];
                  const availabilities = ['Наявні', 'Заброньовані', 'Очікують надходження'];

                  function handleChange(event, key) {
                        filters = { ...filters, [key]: event.target.value };
                  }
            </script>

            <div class="flex flex-col space-y-4 my-8">
                  <div class="flex items-center w-full gap-8">
                        <input
                              type="text"
```

```
                        id="name"
                        placeholder="Назва"
                        bind:value={filters.name}
                        class="px-2 py-1 border border-gray-300 rounded-lg focus:outline-
none focus:border-blue-500 w-full"
                        on:input={(e) => handleChange(e, 'name')}
                    />
                    <button
                        type="submit"
                        class="w-full sm:w-auto bg-blue-500 text-white px-4 py-2 rounded-
md shadow-sm hover:bg-blue-600 transition-colors focus:outline-none focus:ring-2
focus:ring-blue-500"
                        >Пошук
                    </button>
                </div>
                <div class="flex align-center gap-8">
                    <div class="flex items-center">
                        <label for="rating" class="mr-2">Рейтинг:</label>
                        <select
                            id="rating"
                            bind:value={filters.rating}
                            class="px-2 py-1 border border-gray-300 rounded-lg
focus:outline-none focus:border-blue-500"
                            on:change={(e) => handleChange(e, 'rating')}
                        >
                            <option value="">Оберіть рейтинг</option>
                            {#each ratings as rating}
                                <option value={rating}>{rating}</option>
                            {/each}
                        </select>
                    </div>

                    <div class="flex items-center">
                        <label for="genre" class="mr-2">Жанр:</label>
                        <select
                            id="genre"
                            bind:value={filters.genre}
                            class="px-2 py-1 border border-gray-300 rounded-lg
focus:outline-none focus:border-blue-500"
                            on:change={(e) => handleChange(e, 'genre')}
```

```
                        >
                                <option value="">Оберіть жанр</option>
                                {#each genres as genre}
                                        <option value={genre}>{genre}</option>
                                {/each}
                        </select>
                </div>

                <div class="flex items-center">
                        <label for="availability" class="mr-2">Доступність:</label>
                        <select
                                id="availability"
                                bind:value={filters.availability}
                                class="px-2 py-1 border border-gray-300 rounded-lg
focus:outline-none focus:border-blue-500"
                                on:change={(e) => handleChange(e, 'availability')}
                        >
                                <option value="">Оберіть доступність</option>
                                {#each availabilities as availability}
                                        <option
value={availability}>{availability}</option>
                                {/each}
                        </select>
                </div>

                <div class="flex items-center">
                        <label for="sort" class="mr-2">Сортувати за:</label>
                        <select
                                id="sort"
                                bind:value={filters.sort}
                                class="px-2 py-1 border border-gray-300 rounded-lg
focus:outline-none focus:border-blue-500"
                                on:change={(e) => handleChange(e, 'sort')}
                        >
                                <option value="alphabet">Алфавіт</option>
                                <option value="rating">Рейтинг</option>
                        </select>
                </div>
            </div>
        </div>
```

```
<script lang="ts">
	import { onMount } from 'svelte';
	import { createEventDispatcher } from 'svelte';
	import BookSearch from '../../components/BookSearch.svelte';
	import BookList from '../../components/BookList.svelte';
	import { goto } from '$app/navigation';
	export let data;
	console.log(data);

	let searchQuery = '';
	let genreFilter = '';

	const books = data.books;

	let filteredBooks = books;

	function applyFilters() {
		filteredBooks = books.filter((book) => {
			const matchQuery = searchQuery
				? book.title.toLowerCase().includes(searchQuery) ||
					book.author.toLowerCase().includes(searchQuery)
				: true;
			const matchGenre = genreFilter ? book.genre === genreFilter :
true;

			return matchQuery && matchGenre;
		});
	}

	const dispatcher = createEventDispatcher();

	onMount(() => {
		applyFilters();
	});

	const handleBookSelection = (e: CustomEvent<{ id: number }>) => {
		goto(`/literature/${e.detail.id}`);
	};
```

```
        const handleSearch = (e: CustomEvent) => {
            const search = e.detail;

            const query = search.name.toLowerCase().trim();

            console.log(search);

            console.log(data.books);

            filteredBooks = data.books.filter(
                (book) =>
                    book.title.toLowerCase().includes(query) ||
book.author.toLowerCase().includes(query)
            );

            if (search.rating) {
                const rating = search.rating.match(/\d/)[0];
                filteredBooks = filteredBooks.filter((book) => book.rating >=
rating - 0.5);
            }

            if (search.availability) {
                const availability = search.availability;

                if (availability === 'Наявні') {
                    filteredBooks = filteredBooks.filter((book) =>
book.availability === true);
                }
            }
        };
    </script>

    <div class="mx-auto max-w-7xl p-6 bg-white shadow-lg rounded-lg">
        <BookSearch on:search={handleSearch} />

        <BookList bookList={filteredBooks} on:bookSelected={handleBookSelection} />
    </div>

    <script lang="ts">
```

```
	export let data;
	let bookings = data.approvals;

	function approveBooking(id) {
		const index = bookings.findIndex((booking) => booking.id === id);
		if (index !== -1) {
			bookings[index].approved = true;
		}
	}
</script>

<div class="max-w-7xl mx-auto p-6 bg-white shadow-md rounded-lg flex flex-col text-2xl">
	<table>
		<thead>
			<tr>
				<th>ID</th>
				<th>Назва</th>
				<th>Автор</th>
				<th>Номер телефону</th>
				<th>Протяжність бронювання (дні)</th>
				<th>Дія</th>
			</tr>
		</thead>
		<tbody>
			{#each bookings as booking}
				<tr class="text-center pt-5">
					<td>{booking.id}</td>
					<td>{booking.title}</td>
					<td>{booking.author}</td>
					<td>{booking.phoneNumber}</td>
					<td>{booking.duration} дні</td>
					<td>
						{#if !booking.approved}
							<button
								type="submit"
								class="w-full sm:w-auto bg-blue-500 text-white px-4 py-2 rounded-md shadow-sm hover:bg-blue-600 transition-colors focus:outline-none focus:ring-2 focus:ring-blue-500"
```

```
                                                on:click={() => approveBooking(booking.id)}
                                                >Погодити
                                            </button>
                                        {:else}
                                            <div
                                                class="w-full sm:w-auto bg-green-500 text-white px-4 py-2 rounded-md shadow-sm hover:bg-green-600 transition-colors focus:outline-none focus:ring-2 focus:ring-green-500"
                                            >
                                                Погодженно
                                            </div>
                                        {/if}
                                    </td>
                                </tr>
                            {/each}
                        </tbody>
                    </table>
                </div>

                <script lang="ts">
                    import { getBookingByDate } from '$lib/utils/getBookingByDate';
                    import { BOOKS } from '../../constants/books';
                    import ColumnChart from '../../components/analytics/ColumnChart.svelte';
                    import PieChart from '../../components/analytics/PieChart.svelte';
                    import { onMount } from 'svelte';

                    let bookingCountMap = new Map();

                    let fromDate = '';
                    let toDate = '';
                    let genresTop = [];
                    let top5 = [];

                    onMount(() => {
                        // Initialize fromDate and toDate with current date
                        const currentDate = new Date().toISOString().split('T')[0];
                        fromDate = currentDate;
                        toDate = currentDate;
                    });
```

```
function handleFromDateChange(event) {
	fromDate = event.target.value;
}

function handleToDateChange(event) {
	toDate = event.target.value;
}

const formGenresChart = () => {
	BOOKS.forEach((book) => {
		bookingCountMap.set(book, 	getBookingByDate(book.bookings, fromDate, toDate));
	});

	const sortedMap = new Map([...bookingCountMap.entries()].sort((a, b) => b[1] - a[1]));
	// console.log(sortedMap);
	let genreCountMap = new Map();
	sortedMap.forEach((value, key) => {
		const { genre } = key;

		if (genreCountMap.has(genre)) {
			genreCountMap.set(genre, 	genreCountMap.get(genre) 	+ value);
		} else {
			genreCountMap.set(genre, value);
		}
	});

	genresTop = Array.from(genreCountMap.entries());

	top5 = Array.from(sortedMap.entries()).slice(0, 5);
};

const columnChartOptions = {
	title: 'Найпопулярніші книги за увесь час',
	legend: { alignment: 'center', textStyle: { color: 'black', fontSize: 16 } },
	colors: ['#5A5A5A'],
```

```
                width: 1200,
                height: 600
        };

        const pieChartOptions = {
                title: 'Найпопулярніші жанри за увесь час',
                legend: { alignment: 'center', textStyle: { color: 'black', fontSize:
16 } },
                width: 1200,
                height: 600
        };
    </script>

    <div class="max-w-7xl mx-auto p-6 bg-white shadow-md rounded-lg flex flex-col text-
2xl">
        <div class="flex space-x-4 mx-auto">
                <div>
                        <label for="fromDate" class="block text-sm font-medium text-gray-
700">From</label>
                        <input
                                type="date"
                                id="fromDate"
                                name="fromDate"
                                class="mt-1 block w-full px-3 py-2 border border-gray-300
rounded-md shadow-sm focus:outline-none focus:ring-indigo-500 focus:border-indigo-500
sm:text-sm"
                                bind:value={fromDate}
                                on:change={handleFromDateChange}
                        />
                </div>
                <div>
                        <label for="toDate" class="block text-sm font-medium text-gray-
700">To</label>
                        <input
                                type="date"
                                id="toDate"
                                name="toDate"
                                class="mt-1 block w-full px-3 py-2 border border-gray-300
rounded-md shadow-sm focus:outline-none focus:ring-indigo-500 focus:border-indigo-500
sm:text-sm"
```

```
                        bind:value={toDate}
                        on:change={handleToDateChange}
                  />
            </div>
            <button
                  type="submit"
                  class="w-full sm:w-auto bg-blue-500 text-white px-4 py-2 rounded-md shadow-sm hover:bg-blue-600 transition-colors focus:outline-none focus:ring-2 focus:ring-blue-500"
                  on:click={formGenresChart}
                  >Сформувати звіти
            </button>
      </div>
      {#key genresTop && top5}
            <PieChart dataArray={genresTop} chartOptions={pieChartOptions} />
            <ColumnChart dataArray={top5} chartOptions={columnChartOptions} />
      {/key}
</div> export const getBookingByDate = (
      bookings: string[],
      startDate: string = '2022-01-01',
      endDate: string = '2022-12-31'
): number => {
      let count = 0;

      for (const booking of bookings) {
            if (booking >= startDate && booking <= endDate) {
                  count++;
            }
      }

      return count;
};
<script>
      import { createEventDispatcher } from 'svelte';

      const dispatch = createEventDispatcher();

      export let isOpen = false;

      function openModal() {
```

```
                isOpen = true;
        }

        function closeModal() {
                isOpen = false;
        }

        function handleConfirm() {
                dispatch('confirm');
                closeModal();
        }

        function handleCancel() {
                dispatch('cancel');
                closeModal();
        }
</script>

<button
        class="bg-blue-500 hover:bg-blue-700 text-white font-bold py-2 px-4 rounded"
        on:click={openModal}
>
        Open Modal
</button>

{#if isOpen}
        <div
                class="fixed top-0 left-0 w-full h-full flex items-center justify-center
bg-gray-800 bg-opacity-75"
        >
                <div class="bg-white p-8 rounded shadow-lg">
                        <slot />
                        <div class="flex justify-end">
                                <button
                                        class="bg-blue-500   hover:bg-blue-700   text-white
font-bold py-2 px-4 rounded mr-2"
                                        on:click={handleConfirm}
                                >
                                        Підтвердити
                                </button>
```

```
                                <button
                                      class="bg-gray-400   hover:bg-gray-600   text-white
font-bold py-2 px-4 rounded"
                                      on:click={handleCancel}
                                >
                                      Скасувати
                                </button>
                          </div>
                    </div>
              </div>
        {/if}

        <script>
              export let isSignUp = false;

              let email = '';
              let password = '';
              let confirmPassword = '';
              let showPassword = false;
              let errors = { confirmPassword: false };

              function handleSubmit() {
                    errors = {};

                    if (!email.includes('@')) {
                          errors.email = 'Email must be valid';
                    }

                    if (password.length < 6) {
                          errors.password = 'Password must be at least 6 characters long';
                    }

                    if (isSignUp && password !== confirmPassword) {
                          errors.confirmPassword = 'Passwords do not match';
                    }
                    if (Object.keys(errors).length === 0) {
                          if (isSignUp) {
                                // Handle sign up logic
                                console.log('Signing   up   with   email:',   email,   'and
password:', password);
```

```
                        } else {
                                // Handle sign in logic
                                console.log('Signing in with email:', email, 'and
password:', password);
                        }
                }
        }
    </script>

    <div class="max-w-md mx-auto mt-8 p-6 bg-white rounded-lg shadow-lg">
        <h2 class="text-2xl font-semibold mb-4 text-center">
            {isSignUp ? 'Registration Form' : 'Sign In Form'}
        </h2>

        <div class="mb-4">
            <label for="email" class="block mb-1">Email:</label>
            <input
                type="email"
                id="email"
                bind:value={email}
                class="w-full px-3 py-2 border border-gray-300 rounded-lg
focus:outline-none focus:border-blue-500"
            />
            {#if errors.email}<p class="text-red-500 text-sm mt-
1">{errors.email}</p>{/if}
        </div>

        <div class="mb-4 relative">
            <label for="password" class="block mb-1">Password:</label>
            <input
                type="password"
                id="password"
                bind:value={password}
                class="w-full px-3 py-2 border border-gray-300 rounded-lg
focus:outline-none focus:border-blue-500"
            />
            {#if errors.password}<p class="text-red-500 text-sm mt-
1">{errors.password}</p>{/if}
            <button
                type="button"
```

```
                        class="absolute inset-y-0 top-6 right-0 px-3 py-2"
                        on:click={() => (showPassword = !showPassword)}>{showPassword ?
'Hide' : 'Show'}</button
                    >
            </div>

            {#if isSignUp}
                <div class="mb-4">
                    <label    for="confirmPassword"    class="block    mb-1">Confirm
Password:</label>
                    <input
                        type="password"
                        id="confirmPassword"
                        bind:value={confirmPassword}
                        class="w-full px-3 py-2 border border-gray-300 rounded-lg
focus:outline-none focus:border-blue-500"
                    />
                    {#if errors.confirmPassword}<p class="text-red-500 text-sm mt-1">
                            {errors.confirmPassword}
                        </p>{/if}
                </div>
            {/if}

            <div class="flex items-center justify-center mb-4">
                <input type="checkbox" id="isSignUp" bind:checked={isSignUp} class="mr-
2" />
                <label for="isSignUp">Увійти</label>
            </div>

            <button
                type="submit"
                class="bg-blue-500 hover:bg-blue-700 text-white font-semibold px-4 py-2
rounded-lg w-full focus:outline-none focus:shadow-outline"
                on:click={handleSubmit}>{isSignUp ? 'Sign Up' : 'Sign In'}</button
            >
        </div>
```

Додаток Г – Ілюстративна частина

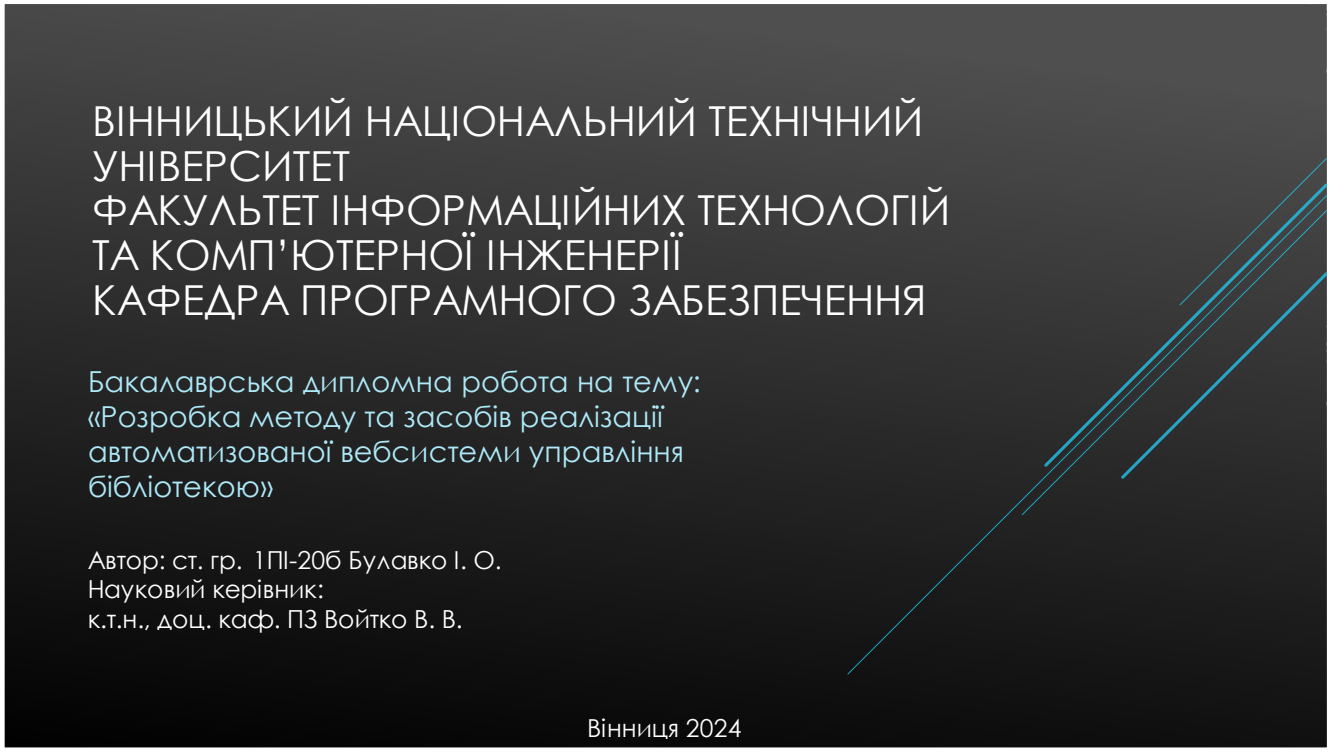

Рисунок Г.1 – Титульний слайд

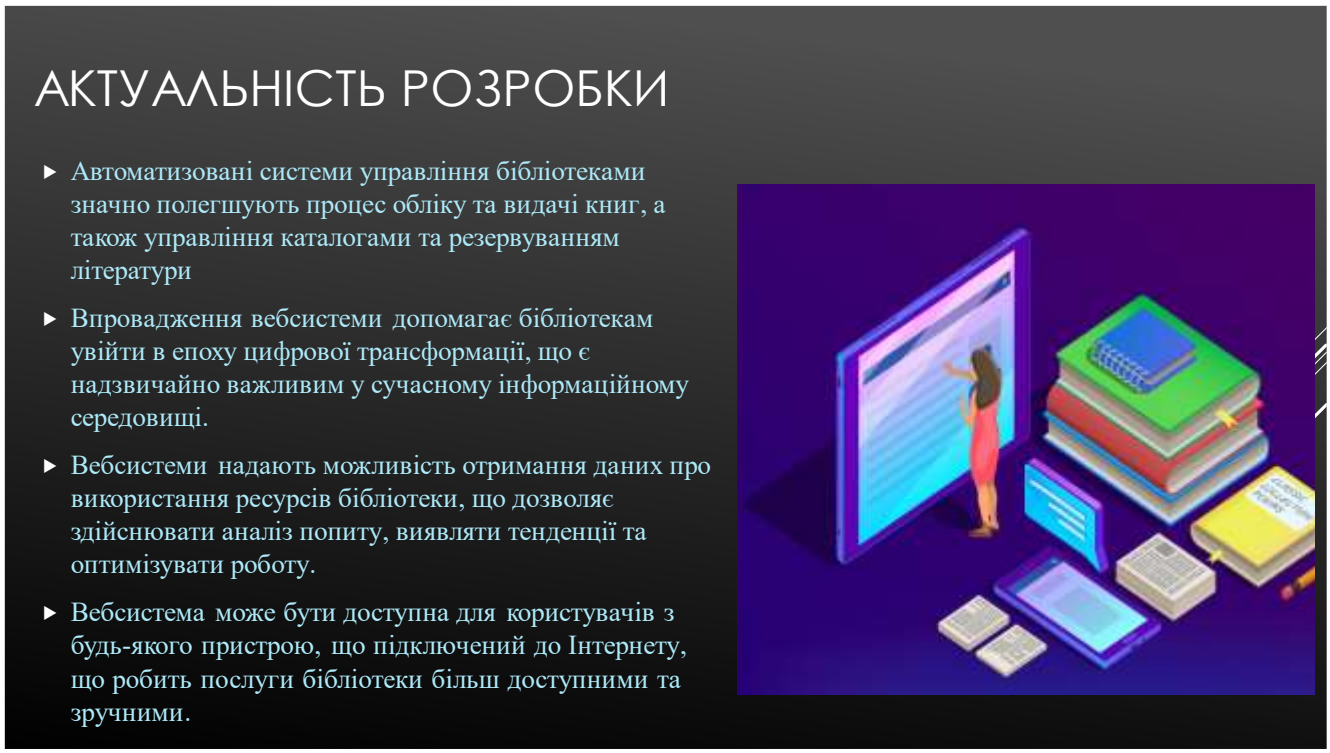

Рисунок Г.2 – Актуальність теми

МЕТА, ПРЕДМЕТ ТА ОБ'ЄКТ ДОСЛІДЖЕННЯ

Метою дослідження є покращення обслуговування користувачів бібліотеки та ведення обліку за рахунок розробки та використання спеціалізованої вебсистеми, яка забезпечує підвищення рівня автоматизації керування наявною літературою, спрощення процесів комунікації користувачів при бронюванні книг та візуалізацію звітності, що полегшує процеси аналізу та проведення статистичних досліджень і формування звітів.

Об'єктом дослідження є процес розробки автоматизованої вебсистеми управління бібліотекою.

Предметом дослідження є методи та засоби реалізації автоматизованої вебсистеми управління бібліотекою.

Рисунок Г.3 – Мета, об'єкт та предмет дослідження

НАУКОВА НОВИЗНА

1. Подальшого розвитку отримав метод формування і відображення звітності про роботу бібліотеки, який, на відміну від існуючих, базується на графічній візуалізації даних у вигляді діаграм, що спрощує порівняльний аналіз статистичної інформації за обраний період та сприяє своєчасному прийняттю управлінських рішень і покращенню процесів управління бібліотекою.
2. Подальшого розвитку набула модель автоматизованої вебсистеми управління бібліотекою, яка, на відміну від існуючих, базується на використанні розширеного функціоналу і реалізації візуальних методів графічного подання звітності, що забезпечує можливість проводити статистичний аналіз даних та забезпечує автоматизацію процесів систематизації обліку літератури.

Рисунок Г.4 – Наукова новизна

Рисунок Г.5 – Постановка задачі

Рисунок Г.6 – Практична цінність отриманих результатів

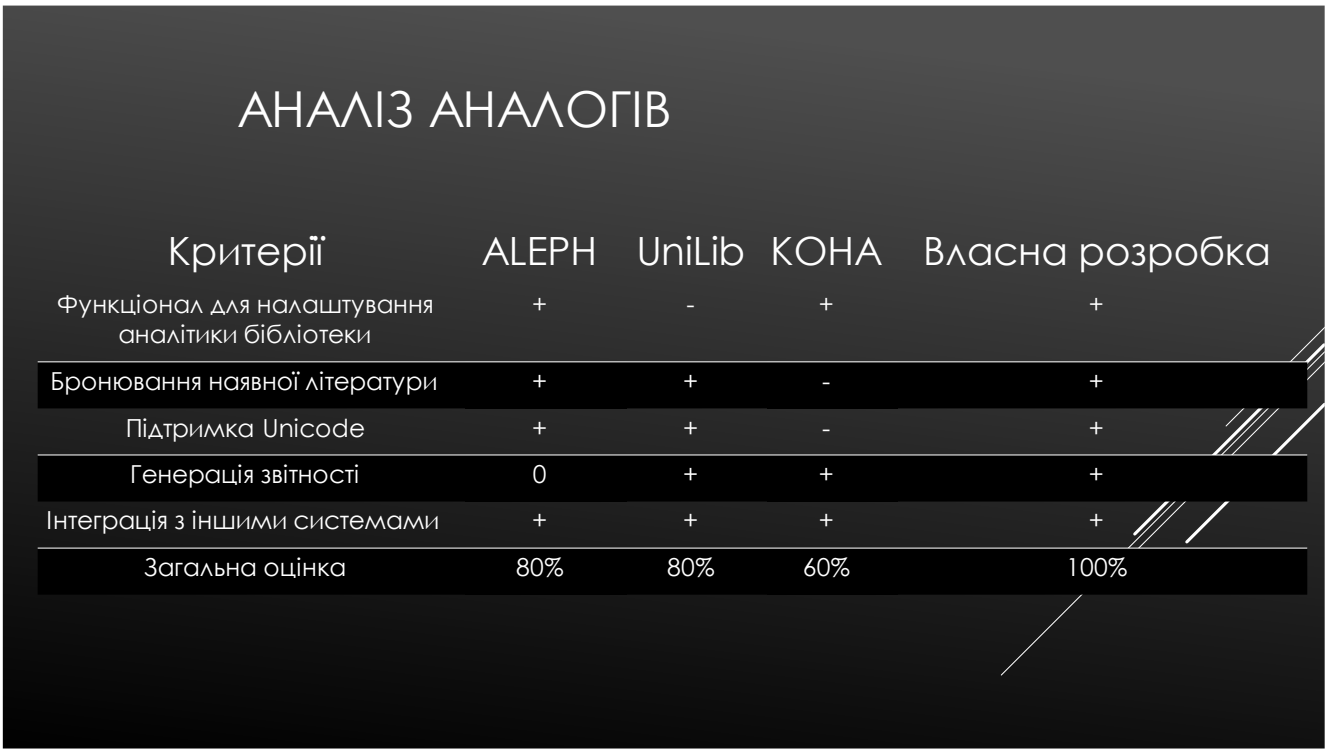

Рисунок Г.7 – Аналіз аналогів

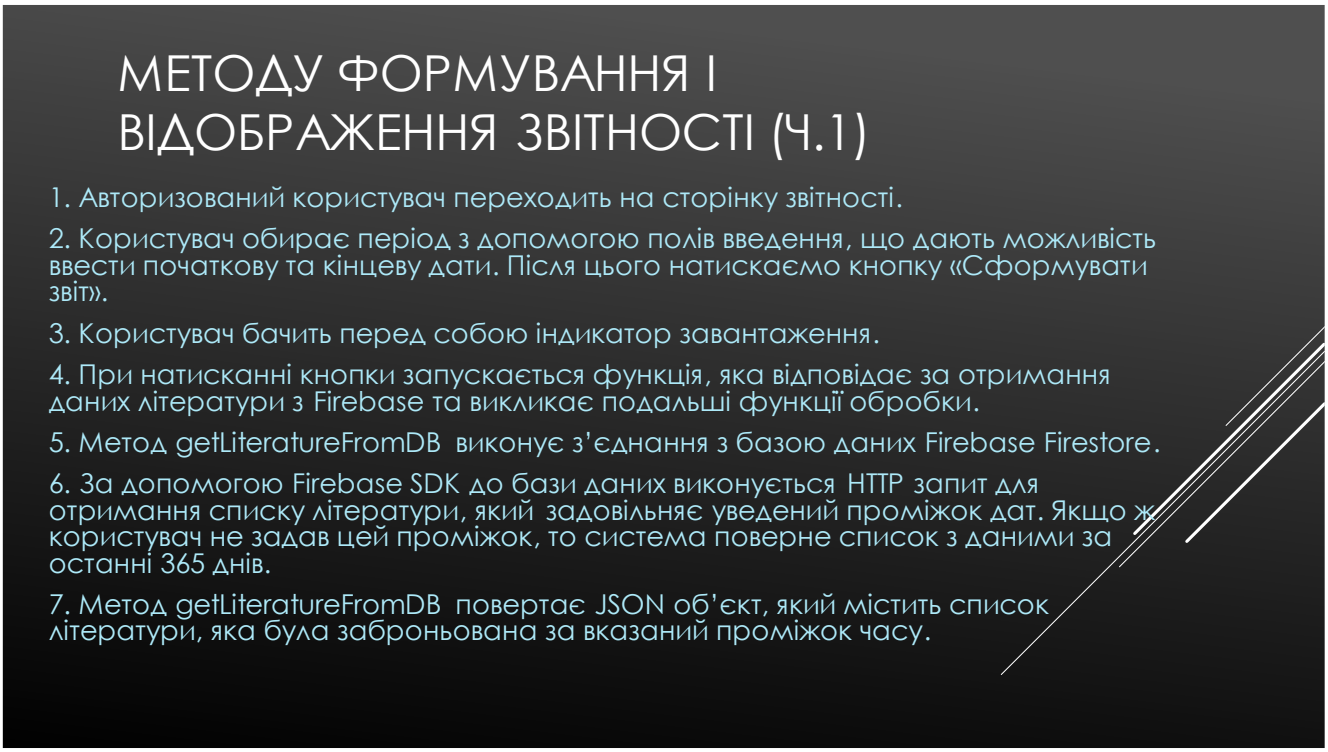

Рисунок Г.8 – Розробка методу формування і відображення звітності ч.1

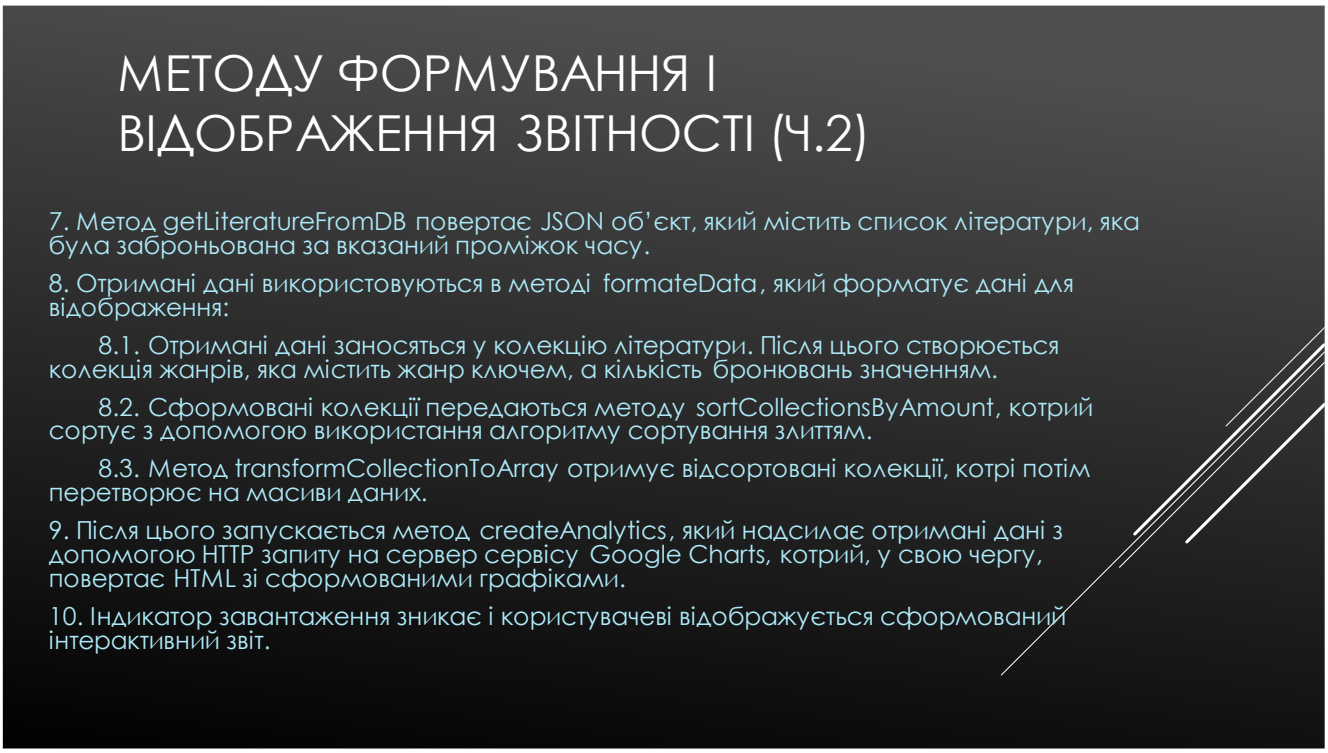

Рисунок Г.9 – Розробка методу формування і відображення звітності ч.2

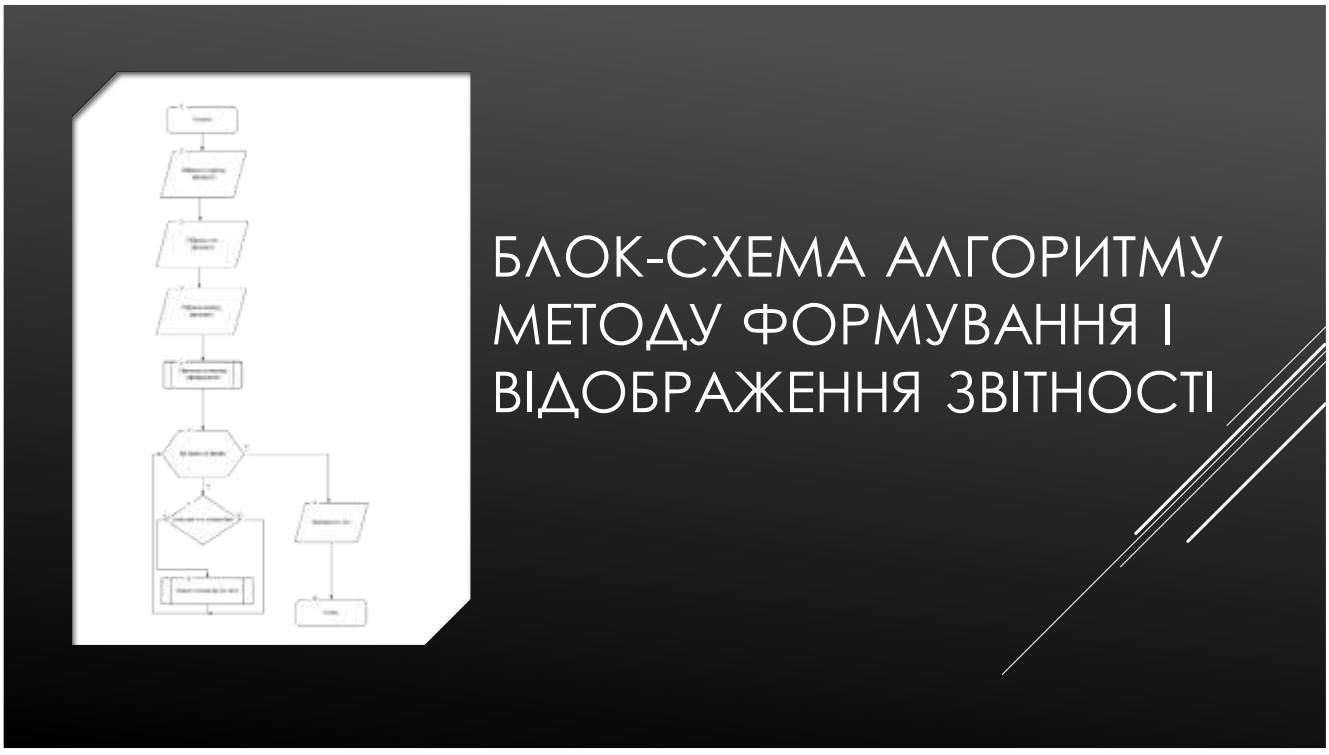

Рисунок Г.10 – Блок-схема алгоритму роботи методу формування і відображення звітності

Рисунок Г.11 – Модель системи на прикладі діаграми діяльності

Рисунок Г.12 – Модель системи на прикладі діаграми діяльності

Рисунок Г.13 – Блок-схема алгоритму бронювання літератури

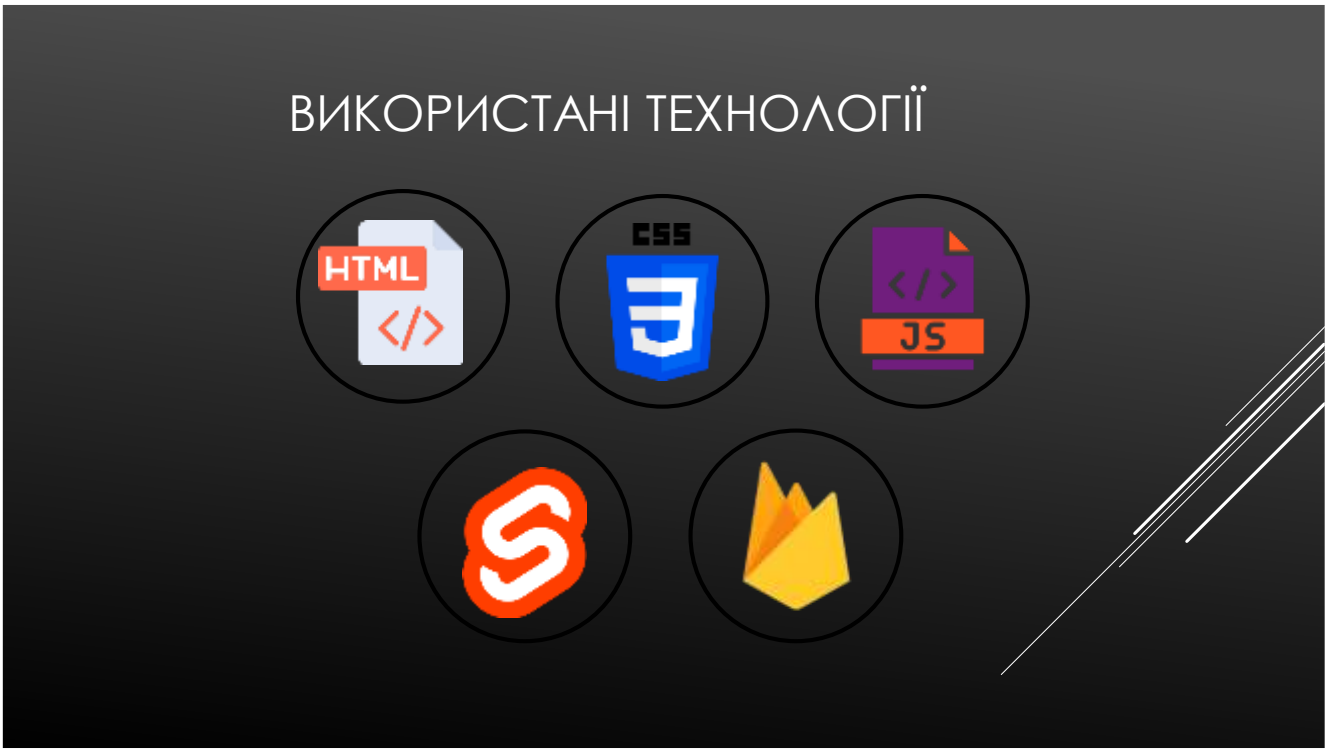

Рисунок Г.14 – Використані технології

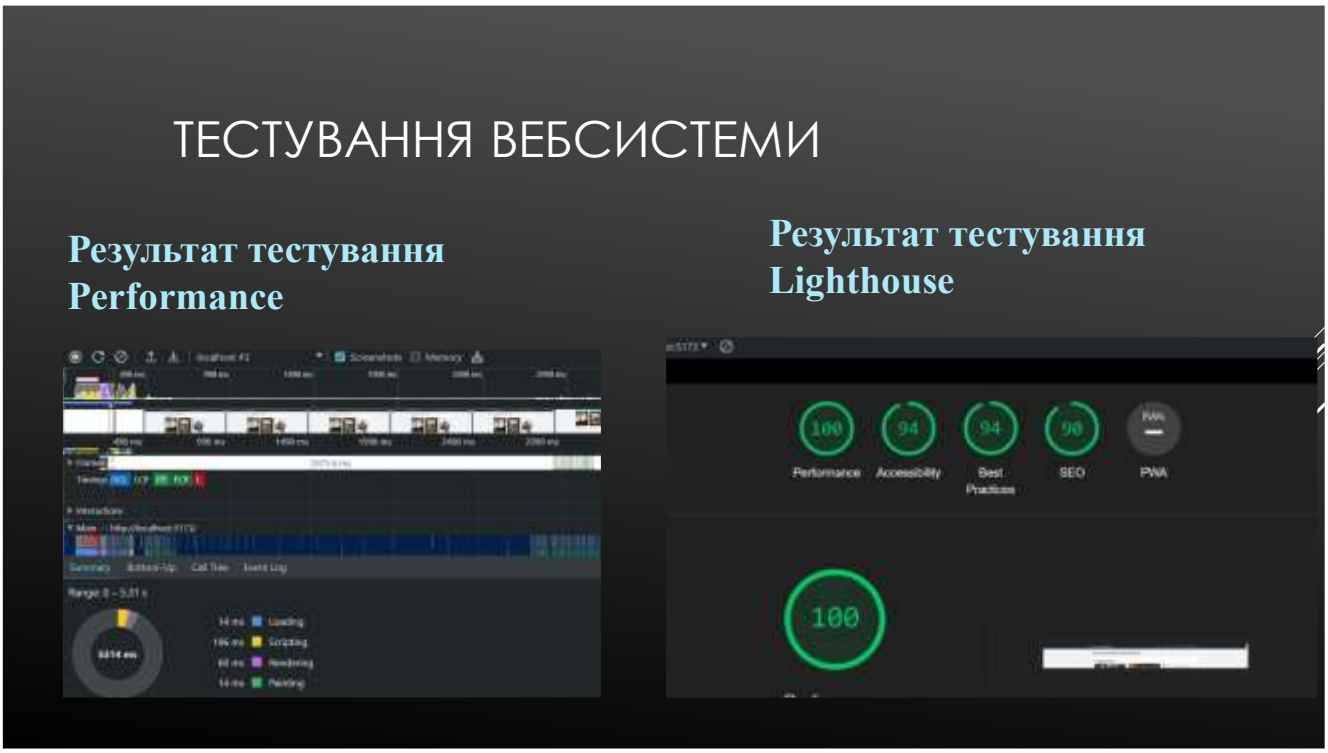

Рисунок Г.15 – Тестування вебсистеми

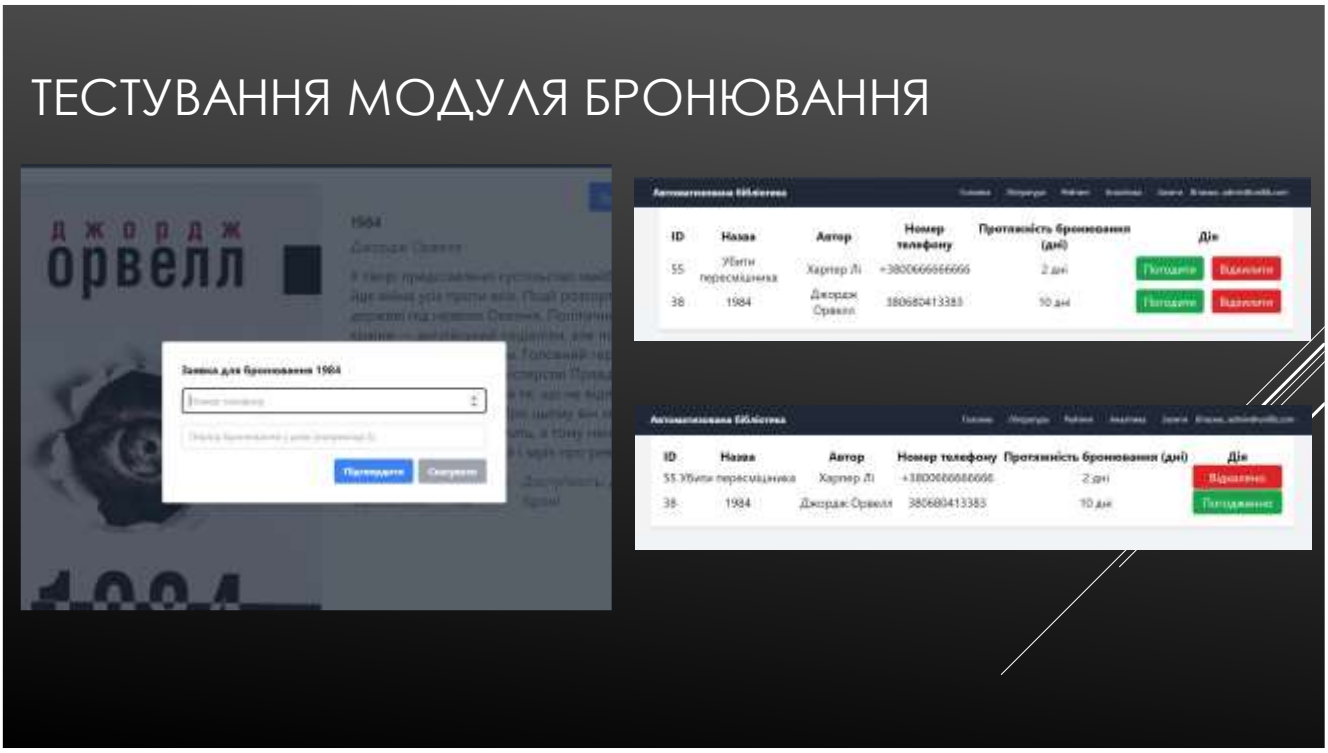

Рисунок Г.16 – Тестування модуля бронювання

АПРОБАЦІЯ ТА ПУБЛІКАЦІЯ РЕЗУЛЬТАТІВ РОБОТИ

Матеріали бакалаврської дипломної роботи доповідалися на:

LIII Науково-технічна конференція факультету інформаційних технологій та комп'ютерної інженерії (2024), Вінниця, 2024.

За тематикою дослідження опубліковано 1 наукову працю у збірниках матеріалів конференції.

УДК 681.3.07

В. В. Войтко
Г. О. Черновалюк
Л. М. Крушельорова
А. В. Денисюк
І. О. Буланов

АВТОМАТИЗОВАНА ВЕБСИСТЕМА УПРАВЛІННЯ БІБЛІОТЕКОЮ

Вінницький національний технічний університет

Анотація

Проведено порівняльний аналіз існуючих автоматизованих вебсистем управління бібліотеками. Визначено функціонал власної розробки. Побудовано алгоритм роботи системи.

Ключові слова: *вебсистема, автоматизація, бібліотека.*

Abstract

A comparative analysis of the automated web system for the library management was carried out. The functionality of own development is defined. An algorithm of the system's operation was built.

Keywords: *web system, automation, library.*

Вступ

У сучасному інформаційному суспільстві бібліотеки відіграють ключову роль у зберіганні, організації та поширенні знань. Автоматизовані веб-системи управління бібліотекою стають невід'ємною частиною діяльності бібліотек, спрощуючи процеси обліку та обробки документів, надання читацького обліку, організації доступу до ресурсів тощо [1,2,3].

Метою роботи є створення ефективної системи, що спростить та оптимізує процеси управління бібліотекою, забезпечивши зручний доступ до інформаційних ресурсів для користувачів та адміністраторів.

Програма орієнтована на створення ефективного і зручного інструменту для організації робочих процесів у бібліотеці. Створена система дозволяє користувачам систематизувати та оптимізувати

Рисунок Г.17 – Апробація та публікація результатів роботи

ВИСНОВКИ

- **розроблено модель автоматизованої вебсистеми управління бібліотекою, яка міститиме основний функціонал системи;**
- **розроблено метод формування і відображення звітності;**
- **розроблено алгоритм бронювання доступної літератури з допомогою онлайн запиту;**
- **розроблено зручний та адаптивний графічний інтерфейс вебсистеми;**
- **розроблено програмне забезпечення вебсистеми;**
- **реалізовано модуль статичних досліджень для створення аналітичної статистики вебсистеми;**
- **здійснено тестування вебсистеми згідно поставлених задач.**

Рисунок Г.18 – Висновки