Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

Кафедра Програмного забезпечення комп'ютерних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *Терешонка Володимира Сергійовича*
(ПІБ)

академічної групи *122-18-2*
(шифр)

спеціальності *122 Комп'ютерні науки*
(код і назва спеціальності)

освітньої програм *Комп'ютерні науки*
(назва освітньої програми)

на тему: *Розробка веб-додатку для бібліотеки електронних книжок на основі фреймворка React*

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	*доц. Мещеряков Л.І.*			
розділів:				
спеціальний	*доц. Мещеряков Л.І.*			
економічний	*доц. Касьяненко Л.В*			
Рецензент				
Нормоконтролер	*доц.Гуліна І.Г.*			

Дніпро
2022

Міністерство освіти і науки України
НТУ «Дніпровська політехніка»

ЗАТВЕРДЖЕНО:
завідувач кафедри
програмного забезпечення комп'ютерних систем
(повна назва)
_____ І.М. Удовик
(підпис) (прізвище, ініціали)
« » _____ 2022 року

ЗАВДАННЯ
на кваліфікаційну роботу
бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *122-18-2* *Терешонка Володимира Сергійовича*
(група) (прізвище та ініціали)
тема кваліфікаційної роботи *Розробка веб-додатку для бібліотеки електронних книжок на основі фреймворка React*

затверджена наказом ректора НТУ «ДП» від *07.06.2022р.* № *317-с*

Розділ	Зміст виконання	Термін виконання
Спеціальний	***На основі матеріалів виробничої практики та их науково-технічних джерел провести аналіз у рішення проблеми та постановку задачі. Обґрунтувати вибір та здійснити реалізацію одів вирішення проблеми***	***13.05.2022 р.***
Економічний	***Провести розрахунок трудомісткості обки програмного забезпечення, витрат на рення ПЗ й тривалості його розробки***	***27.05.2022 р.***

Завдання видав _____ *доц. Мещеряков Л.І.*
(підпис) (посада, прізвище, ініціали)
Завдання прийняв до виконання _____ *Терешонок В.С.*
(підпис) (прізвище, ініціали)

Дата видачі завдання: *14.01.2022 р.*

Термін подання кваліфікаційної роботи до ЕК: *11.06.2022 р.*

РЕФЕРАТ

Пояснювальна записка: 65 с., 8 рис., 3 дод., 20 джерел.

Об'єкт дослідження: сайт електронної бібліотеки. Метою кваліфікаційної роботи є розробка Full-Stack додадтку електронних книг на основі фреймворку React.

У вступі обговорюється поточний стан проблеми, визначається мета кваліфікаційної роботи, актуальність та сфера застосування, а також уточнюється проблема.

У першому розділі описується предметна область, визначається актуальність розробки, формулюється проблема, вказуються вимоги до програмної реалізації, технології та програмні засоби.

У другому розділі порівнюються існуючі рішення, вибирається платформа для розробки, завершується проектування та розробка програми, описується функціонування програми, алгоритм та структура її роботи, визначаються вхідні та вихідні дані, характеризуються параметри технічних засобів.

В економічній частині визначається обсяг робіт для розробленої інформаційної системи, розраховуються витрати на створення програми, розраховується час створення програми.

Практична користь полягає у створенні веб-сайту, який дозволяє замовляти книги.

Актуальність інформаційної системи визначається високим попитом на електронні книги та можливістю оцифрування інформації.

Список ключових слів: КНИГА, ДОДАТОК, БІБЛІОТЕКА, ЕЛЕКТРОННА БІБЛІОТЕКА, КОМП'ЮТЕР.

ABSTRACT

Explanatory note: 65 pages, 8 figures, 3 appendices, 20 sources.

Object of research: electronic library site. The aim of the qualification work is to develop web applications for e-book libraries based on the Spring framework.

The introduction discusses the current state of the problem, determines the purpose of the qualification work, relevance and scope, as well as clarifies the problem.

The first section describes the subject area, determines the relevance of development, formulates the problem, specifies the requirements for software implementation, technology and software.

The second section compares existing solutions, selects a platform for development, completes the design and development of the program, describes the operation of the program, algorithm and structure of its work, determines the input and output data, characterizes the parameters of hardware.

In the economic part, the amount of work for the developed information system is determined, the costs of creating the program are calculated, the time of creating the program is calculated.

The practical benefit is to create a website that allows you to view and order books.

The relevance of the information system is determined by the high demand for e-books and the ability to digitize information.

List of key words: BOOK, APPENDIX, LIBRARY, ELECTRONIC LIBRARY, COMPUTER.

ЗМІСТ

СПИСОК УМОВНИХ ПОЗНАЧЕНЬ

БД – База даних

ІС – Інформаційна система

ВСТУП

Термін «ІС» відноситься до класу програмних продуктів, здатних виконувати роботу будь-якої установи, наприклад, В. бібліотека для автоматизації. Така система обробляє, збирає та шукає інформацію для зручнішого управління звичайним сайтом або навіть компанією. Я вибрав цю тему для своєї дисертації, тому що електронна бібліотека ІС дуже тісно пов'язана з моєю предметною областю «Інформатика».

Автоматизована ІС є сукупністю інформації, технологій та програмного забезпечення. Таким чином, за допомогою такої системи можна створити зв'язок між усіма частинами сайту електронної бібліотеки, що підвищує керованість процесів. ІС, що використовує принцип зворотного зв'язку на всіх рівнях управління та сучасні інформаційно-комунікаційні технології, забезпечує зв'язок між елементами системи управління та користувальницькими елементами, а також забезпечує можливість збирання, обробки та аналізу даних.

Щоб створити власний IP, ви повинні, по-перше, створити базу даних, оскільки база даних необхідна для використання запитів, а по-друге, підготувати веб-сайт для зручного використання користувачами та адміністраторами.

Мета фінального проекту — створити сайт, який дозволить читачам переглядати книги у зручному вигляді. Нарешті було сформовано тему дипломної роботи: «Проектування електронної бібліотеки об'єктів інтелектуальної власності». З одного боку, електронна бібліотека виконує функції традиційної бібліотеки: надання інформації читачеві, з іншого боку, вона відіграє типову роль для автоматизованої бібліотечної ІС – організації та зберігання локальних та віддалених електронних ресурсів та доступу до них. їх.

РОЗДІЛ 1

АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ І ПОСТАНОВКА ЗАДАЧІ

1.1. Загальні відомості з предметної галузі

Інформація зараз доступна не тільки в друкованому, але і в електронному вигляді. Інформаційні технології з кожним роком проникають практично у всі сфери діяльності. Інформація в електронному вигляді забезпечує загальну доступність та швидке поширення інформації по всьому світу. В даний час доступ користувачів до електронних ресурсів є одним із пріоритетів науки, освіти та культури. Розвиток інформаційних технологій стимулює швидке зростання обсягу інформаційних ресурсів, що знаходяться в ІС, наприклад, в електронних бібліотеках. При створенні електронної бібліотеки треба розуміти, що вона створюється насамперед для читачів, а вже потім для бібліотекарів. Прийнято вважати, що найбільш зручним та ефективним способом зберігання інформації є електронний, а не друкований вигляд.

Натомість старих бібліотек на службі освіти, культури та науки створюються мережі інтернет-бібліотек, які вже зберігають інформацію в електронному вигляді. Така ІС буде зручніша за традиційну бібліотеку, оскільки зможе самостійно обробляти запити користувачів без залучення персоналу. Доступ до такого ресурсу буде доступний будь-якому користувачеві через Інтернет.

В основі, що поєднує традиційні та електронні бібліотеки, лежить принцип зручності використання. Функції електронної бібліотеки відрізняються від функцій традиційної бібліотеки. З формального погляду значна частина електронних ресурсів складається з копій друкованих версій, й у сенсі електронна колекція складається з копій, а чи не оригіналів з

першоджерел. Електронна бібліотека (ЕБ) тепер підпорядковується традиційній класичній бібліотеці.

Через фінансові вузькі місця рівень вітчизняних бібліотек на один-два порядки нижчий, ніж у Європі. Найближчими роками швидко змінити ситуацію не вдасться, але за рахунок посилення інформаційного забезпечення вітчизняних фахівців це є можливим. Це може пояснити, чому держава виявляє великий інтерес до цифровізації тих сфер життя, де це можливо.

Основою будь-якої електронної бібліотеки буде доступ до книжкової бази даних, список нових надходжень, інформація про бібліотечні заходи: конференції, книжкові виставки. Тобто, необхідно враховувати важливу роль технологій баз даних в електронних бібліотеках. p align="justify"> При створенні електронних бібліотек використовуються системи управління базами даних, в основі яких лежать різні моделі даних - реляційні, об'єктно-орієнтовані, об'єктно-орієнтовані. Такі бази даних підтримують різні набори структурованих даних в електронних бібліотеках та забезпечують ефективний доступ до них.

1.2. Призначення розробки та галузь застосування

Призначення розробки та галузь застосування – це розробка електронної бібліотеки:

- корегування таблиць БД;
- зручний інтерфейс;
- редагування та видалення книг.

1.3. Підстава для розробки

Отже, підставами для розробки (виконання кваліфікаційної роботи) є:

- Графік навчального процесу та навчальний план;

- наказ ректора Національного технічного університету «Дніпровська політехніка» № 317-с від 07.06.2022 р;
- завдання на кваліфікаційну роботу проект на тему «Розробка веб-додатку для бібліотеки електронних книжок на основі фреймворка React».

1.4. Постановка завдання

Завданням проекту є проектування електронної бібліотеки. Програмне забезпечення призначене для надання користувачам можливість перегляду книг.

Програма повинна реалізувати наступні функції:

- можливість користувачів доступу до приватной бібліотеки книг;
- надання информации
- можливість додавання, редагування та видалення книг.

1.5. Вимоги до програми або програмного виробу

1.5.1. Вимоги до функціональних характеристик

Вимоги до програми – це проектування автоматизованої ІС електронної бібліотеки:

- читання таблиць БД з даними про книги;
- швидка навігація по сайту;
- UI/UX дизайн
- коректна візуалізація елементів веб-додатку.

1.5.2 Вимоги до інформаційної безпеки.

Головне вікно програми повинно давати змогу для входу користувача.

1.5.3 Вимоги до складу та параметрів технічних засобів.

Для забезпечення надійного функціонування програмного забезпечення необхідно, щоб обчислювальна машина, на якій буде експлуатуватися веб-додаток, мала такі характеристики:

- Маніпулятор “миша”.
- Клавіатура.
- Доступ до онлайн мережі.
- Процесор Intel Core i3-2348 з тактовою частотою 2.3 ГГц.
- Не менше ніж 4 Гб оперативної пам’яті.
- Рідкокристалічний монітор з діагоналлю 17".

Вище наведені характеристики являють собою рекомендовані. Це означає, що при наявності характеристик не нижче зазначених, розроблений додаток буде функціонувати відповідно до вимог щодо надійності, безпеки та швидкості обробки даних.

1.5.4 Вимоги інформаційної та програмної сумісності.

Для коректного функціонування програми необхідно, щоб програмне забезпечення обчислювальної машини, на якій буде експлуатуватися веб-додаток, відповідало наступним вимогам:

- Веб браузер Google Chrome, FireFox, IE10+.
- Операційна система.

Веб-орієнтована підсистема має бути реалізована на мові програмування JavaScript. Для візуалізації було використано CSS який описує зовнішній вигляд сторінки.

РОЗДІЛ 2

ПРОЕКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ

2.1. Функціональне призначення системи

Призначення програми складається з автоматизації ІС електронної бібліотеки. Головна сторінка сайту дозволяє користувачеві мати повну навігацію по сайту. З цієї сторінки можна побачити меню сайту, де можна перейти в розділ з книгами, отримати інформацію про сайт та зареєструвати нову книгу. При переході в розділ з книгами, можна тільки побачити перелік книг, для того, щоб їх змінювати або видаляти в свою бібліотеку.

Переглянути свої придбані книги можна в розділі Особистий кабінет. Для редагування книг потрібно перейти на сторінку Books де можливо додавати / змінювати / видаляти інформацію в БД через сайт.

2.2. Опис застосованих математичних методів

Оскільки особливості предметної області ІС електронної бібліотеки не передбачають застосування математичних методів при розробці, математичні методи не використовувалися.

2.3. Опис використаних технологій та мов програмування

При розробці даного сайту використовувалося IDE Visual Studio Code. Для обробки даних сайту я використовував інтерфейс MongoDB та програму Postman.

MongoDB - це документна система управління базами даних, яка не вимагає опису схеми таблиці. Вважається одним із класичних прикладів систем NoSQL, що використовують JSON-подібні документи та схеми баз даних.

Postman – це клієнт API, який дозволяє розробникам легко створювати, спільно використовувати, тестувати та документувати API. Це досягається за рахунок того, що користувачі можуть складати та зберігати прості та складні запити HTTP/s та читати відповіді.

JavaScript – мультипарадигмальна мова програмування. Підтримує об'єктно-орієнтований, імперативний та функціональний стилі. Це реалізація специфікації ECMAScript. JavaScript часто використовується як вбудована мова для програмного доступу до об'єктів програми.

React — це бібліотека JavaScript з відкритим вихідним кодом для розробки інтерфейсів користувача. React розробляється та підтримується Facebook, Instagram та спільнотою окремих розробників та компаній. React можна використовувати для розробки односторінкових та мобільних програм.

Программа розподіляється на на Backend-серверну частину та Frontend інтерфейсну.

Найважливіші частини програми:

Backend

Динамічна структура(схема) розробки книжок яка має прості методи інтегрування та розширення функціоналу від потреб замовника :

```
const bookSchema = new Schema({
  name: {
    type: String,
    required: true
```

```
    },
    author: {
      type: String,
      required: true
    },
    description: {
      type: String,
      required: true
    },
    price: {
      type: Number,
      required: true
    },
    avaliable: {
      type: Boolean,
    },
    image: {
      type: String,
      required: true
    }
  })
```

Контроллер для callback обробки данних з сервера та при налаштування помилок при роботі з ним:

```
const getAllBooks = async (req, res, next) => {
 let books
 try {
   books = await Book.find()
 } catch (err) {
   console.log(err)
 }


 if(!books) {
   return res.status(404).json({message:"No products found"})
 }
 return res.status(200).json({ books })
```

```
}
const getById = async (req, res, next) => {
  const id = req.params.id
  let book
  try {
    book = await Book.findById(id)
  } catch (err) {
    console.log(err)
  }
  if(!book) {
    return res.status(404).json({message:"No Book found"})
  }
  return res.status(200).json({ book })
}
const addBook = async (req, res, next) => {
  const {name, author, description, price, avaliable, image} = req.body
  let book
  try {
    book = new Book({
      name,
      author,
      description,
      price,
      avaliable,
      image
    })
    await book.save()
  } catch (err) {
    console.log(err)
  }
  if (!book) {
    return res.status(500).json({message:'Unable To Add'})
  }
  return res.status(201).json({ book })
}
const updateBook = async (req, res, next) => {
```

```
  const id = req.params.id
  const {name, author, description, price, avaliable, image} = req.body
  let book
  try {
    book = await Book.findByIdAndUpdate(id, {
      name,
      author,
      description,
      price,
      avaliable,
      image,
    })
    book = await book.save()
  } catch (err) {
    console.log(err)
  }
  if (!book) {
    return res.status(404).json({message:'Unable To Update By this ID'})
  }
  return res.status(200).json({ book })
}
const deleteBook = async (req, res, next) => {
  const id = req.params.id
  let book
  try {
    book = await Book.findByIdAndRemove(id)
  } catch {
    console.log(err)
  }
  if(!book) {
    return res.status(404).json({message:"Unable To Delete by this ID"})
  }
  return res.status(200).json({ message:"Product Successfully Deleted"})
}
```

Frontend

Опис функції для редагування книжок:

```
const BookDetail = () => {
  const [inputs, setInputs] = useState()
  const id = useParams().id
  const [checked, setChecked] = useState(false)
  const history = useNavigate()

  useEffect(() => {
    const fetchHandler = async () => {
      await axios
        .get(`http://localhost:5000/books/${id}`)
        .then((res) => res.data)
        .then((data) => setInputs(data.book));
    }
    fetchHandler()
  }, [id])

  const sendRequest = async () => {
    await axios
      .put(`http://localhost:5000/books/${id}`, {
        name: String(inputs.name),
        author: String(inputs.author),
        description: String(inputs.description),
        price: Number(inputs.price),
        image: String(inputs.image),
        available: Boolean(checked),
      }).then((res) => res.data)
  };
  const handleSubmit = (e) => {
    e.preventDefault()
    sendRequest().then(() => history("/books"))
  }
  const handleChange = (e) => {
    setInputs((prevState) => ({
      ...prevState,
```

```
    [e.target.name]: e.target.value,
  }))
}

return (
  <div>
    {inputs && (
      <form onSubmit={handleSubmit}>
        <Box
          display="flex"
          flexDirection="column"
          justifyContent={"center"}
          maxWidth={700}
          alignContent={"center"}
          alignSelf="center"
          marginLeft={"auto"}
          marginRight="auto"
          marginTop={10}
        >
          <FormLabel>Name</FormLabel>
          <TextField
            value={inputs.name}
            onChange={handleChange}
            margin="normal"
            fullWidth
            variant="outlined"
            name="name"
          />
          <FormLabel>Author</FormLabel>
          <TextField
            value={inputs.author}
            onChange={handleChange}
            margin="normal"
            fullWidth
            variant="outlined"
```

```
  name="author"
/>
<FormLabel>Description</FormLabel>
<TextField
  value={inputs.description}
  onChange={handleChange}
  margin="normal"
  fullWidth
  variant="outlined"
  name="description"
/>
<FormLabel>Price</FormLabel>
<TextField
  value={inputs.price}
  onChange={handleChange}
  type="number"
  margin="normal"
  fullWidth
  variant="outlined"
  name="price"
/>
<FormLabel>Image</FormLabel>
<TextField
  value={inputs.image}
  onChange={handleChange}
  margin="normal"
  fullWidth
  variant="outlined"
  name="image"
/>
<FormControlLabel
  control={
    <Checkbox
      checked={checked}
      onChange={() => setChecked(!checked)}
    />
```

```
            }
            label="Available"
          />
          <Button variant="contained" type="submit">
            Update Book
          </Button>
        </Box>
      </form>
    )}
  </div>
 )
}
```

Опис функції для додавання нових книжок:

```
const AddBook = () => {
 const history = useNavigate();
 const [inputs, setInputs] = useState({
  name: "",
  description: "",
  price: "",
  author: "",
  image: "",
 });
 const [checked, setChecked] = useState(false);
 const handleChange = (e) => {
  setInputs((prevState) => ({
   ...prevState,
   [e.target.name]: e.target.value,
  }));
  // console.log(e.target.name, "Value", e.target.value);
 };


 const sendRequest = async () => {
  await axios
```

```
    .post("http://localhost:5000/books", {
      name: String(inputs.name),
      author: String(inputs.author),
      description: String(inputs.description),
      price: Number(inputs.price),
      image: String(inputs.image),
      available: Boolean(checked),
    })
    .then((res) => res.data);
};
const handleSubmit = (e) => {
  e.preventDefault();
  console.log(inputs, checked);
  sendRequest().then(() => history("/books"));
};
return (
  <form onSubmit={handleSubmit}>
    <Box
      display="flex"
      flexDirection="column"
      justifyContent={"center"}
      maxWidth={700}
      alignContent={"center"}
      alignSelf="center"
      marginLeft={"auto"}
      marginRight="auto"
      marginTop={10}
    >
      <FormLabel>Name</FormLabel>
      <TextField
        value={inputs.name}
        onChange={handleChange}
        margin="normal"
        fullWidth
        variant="outlined"
        name="name"
```

```
/>
<FormLabel>Author</FormLabel>
<TextField
 value={inputs.author}
 onChange={handleChange}
 margin="normal"
 fullWidth
 variant="outlined"
 name="author"
/>
<FormLabel>Description</FormLabel>
<TextField
 value={inputs.description}
 onChange={handleChange}
 margin="normal"
 fullWidth
 variant="outlined"
 name="description"
/>
<FormLabel>Price</FormLabel>
<TextField
 value={inputs.price}
 onChange={handleChange}
 type="number"
 margin="normal"
 fullWidth
 variant="outlined"
 name="price"
/>
<FormLabel>Image</FormLabel>
<TextField
 value={inputs.image}
 onChange={handleChange}
 margin="normal"
 fullWidth
 variant="outlined"
```

```
          name="image"
        />
        <FormControlLabel
          control={
            <Checkbox checked={checked} onChange={() => setChecked(!checked)} />
          }
          label="Available"
        />
        <Button variant="contained" type="submit">
          Add Book
        </Button>
      </Box>
    </form>
  )
}
```

2.4. Опис структури системи та алгоритмів її функціонування.

Структура мого проекту складається з великої кількості файлів, зараз ми їх розглянемо детальніше. На рис. 2.1., 2.2., 2.3., 2.4. наведена структура файлів проекту.

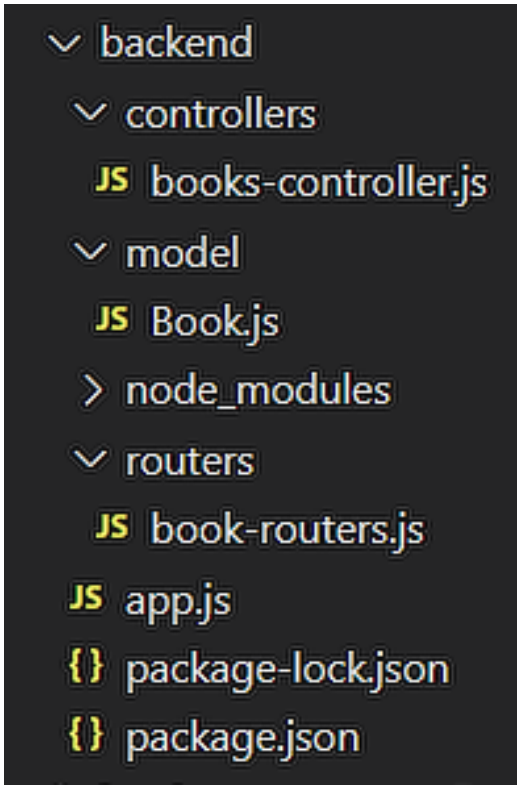

Рис. 2.1. Структура файлів backend

Рис. 2.2. Структура файлів проекту Frontend

Опис файлів структури:

1. Controllers – файли, які дозволяють працювати з конфигами сервера

2. Model – папка, де зберігаються схеми розробки.
3. Routers - файли де описується логіка навигації по сайту
4. Components - папка з усіма интерфейсами сайту.
5. Book - файли, де описується основна логіка роботи книжок.
6. Exceptions – винятки та їх обробка.
7. Package.json - файлы для опису пакетів розробки
8. *.css - файли опису стилів интерфейсів
9. App.js - файли збірки проєкту
10. Gitignore - файли для роботи з GIT

Дуже важливою частиною моєї ІС є БД, яка зроблена в програмі MongoDB, це видно на Рис.2.3.

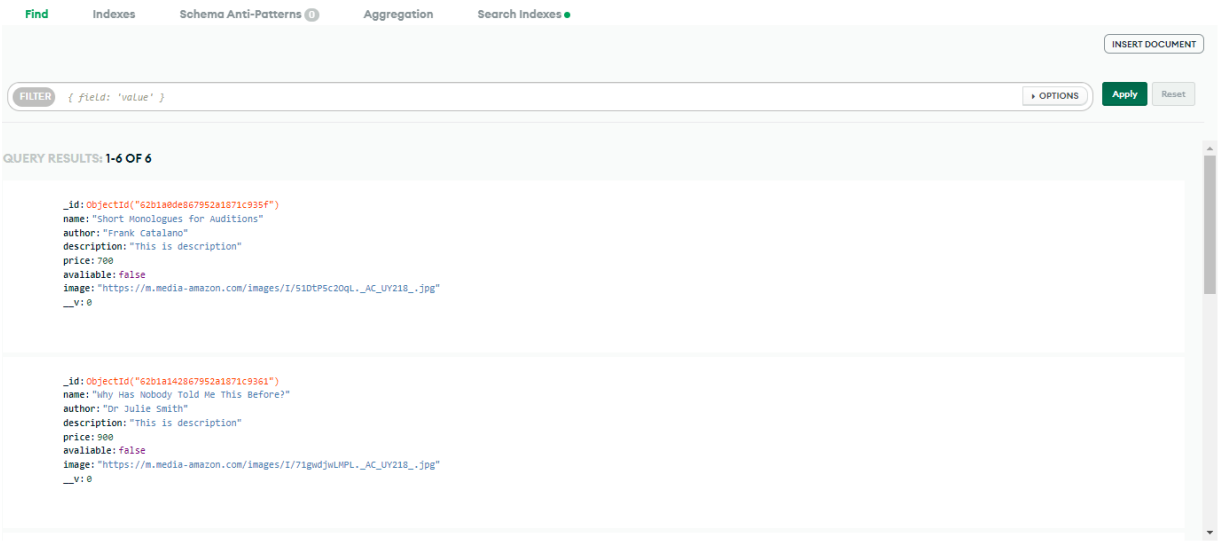

Рис. 2.3. опис БД

2.5 Обґрунтування та організація вхідних та вихідних даних програми

Згідно задачам, які вирішує даний програмний засіб, організація вхідних та вихідних даних програми має такий вигляд.

Вхідними даними для даного програмного комплексу є:

– JSON-файли MongoDB.

Вихідними даними комплексу є:

– таблиця книг.

2.6. Опис розробленої системи.

2.6.1. Використані технічні засоби.

Для функціонування системи необхідна клієнтська персональна ЕОМ з наступними мінімальними характеристиками:

– процесор класу Intel Core i3 2 ядра 3,9Ггц;

– монітор;

– не менше 4Гб ОЗУ;

– 500Мб вільного місця на диску;

– клавіатура;

– маніпулятор «миша».

2.6.2. Використані програмні засоби.

При створенні мого сайту була використана середовище розробки Visual Studio Code, оскільки саме ця програма має велику кількість плюсів, а саме:

- VS Code дозволяє розробляти як консольні програми, так і програми з графічним інтерфейсом, у тому числі з підтримкою технології Windows Forms, а також веб-сайти, веб-програми, веб-служби як у рідному, так і в керованому кодах для всіх платформ.
- У редакторі присутні вбудований налагоджувач, інструменти для роботи з Git та засоби рефакторингу, навігації за кодом, автодоповнення типових конструкцій та контекстної підказки.
- Продукт підтримує розробку для платформ ASP.NET і Node.js, і вважається легковажним рішенням, яке дозволяє обійтися без повного інтегрованого середовища розробки.
- Великим плюсом редактора є підтримка великої кількості мов, таких як C++, C#, Python, PHP, JavaScript та інші.

Також для розробки БД було обрано MongoDB оскільки, як і MySQL, MongoDB також надає багаті функції, має власну мову запитів, високодоступні вторинні індекси (включаючи текстовий пошук та географічне розташування), потужне, високоагреговане середовище аналізу даних та багатший тип даних, ніж реляційні бази даних. І масштабованість

2.6.3. Виклик та завантаження програми

Сайт не потребує інсталювання, може бути використаний з будь-якого носія стандартними засобами браузера.

Сайт розроблений для будь-яких комп'ютерів або телефонів з інтернетом. Для використання сайт не потребує додаткових програмних засобів.

2.6.4. Опис інтерфейсу користувача

На головній сторінці, тобто Homepage, можна перейти до своей бібліотеки та потрапити на ней за допомогою натискання на Icon у ливому верхньому кутку, це видно на Рис. 2.4.

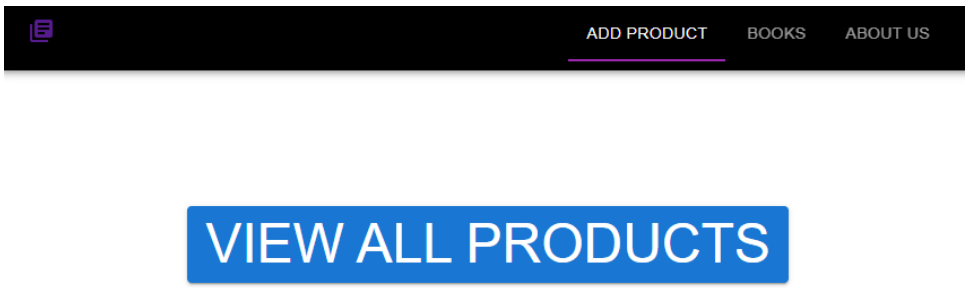

Рис. 2.4. Головна сторінка сайту

В розділі Books можна побачити саму бібліотеку, яка представляє собою таблицю з книгами. Користувач, що не зайшов під своїми особистими даними в акаунт, не може взаємодіяти з таблицею (Рис. 2.5.)

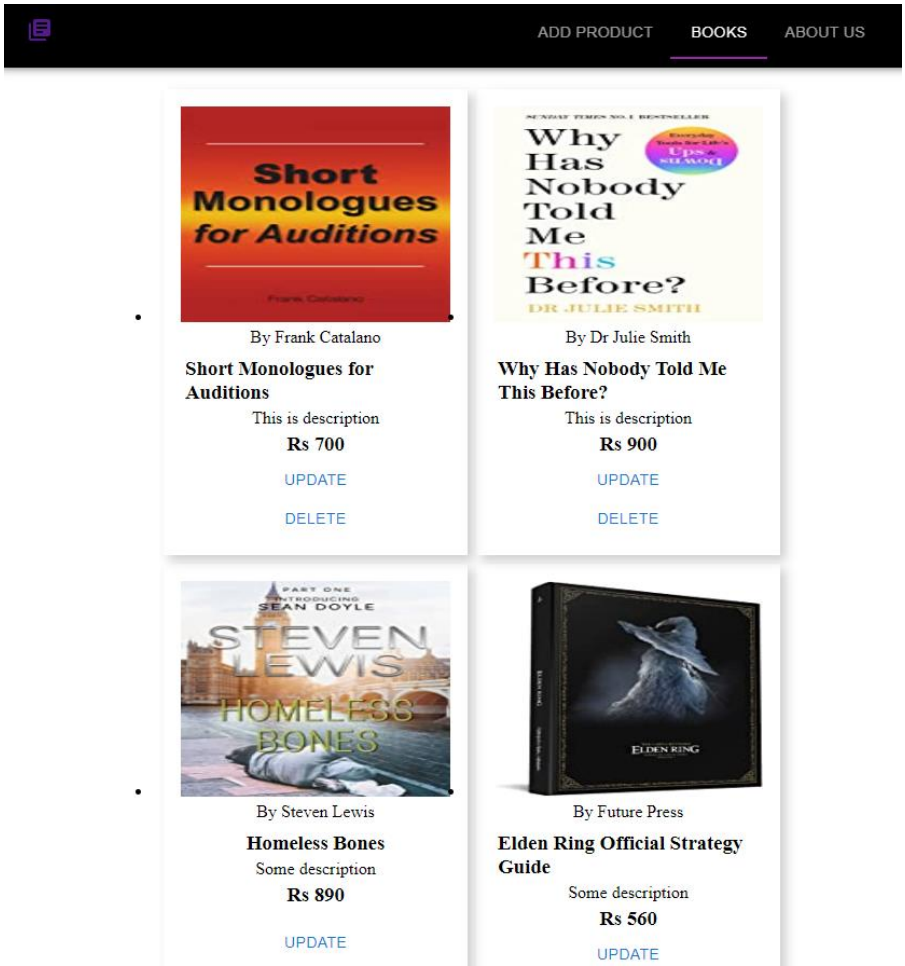

Рис. 2.5. Сторінка списку книг

Наступний розділ, Add Product, надає можливість додавати свої книжки у бібліотеку (Рис. 2.6.).

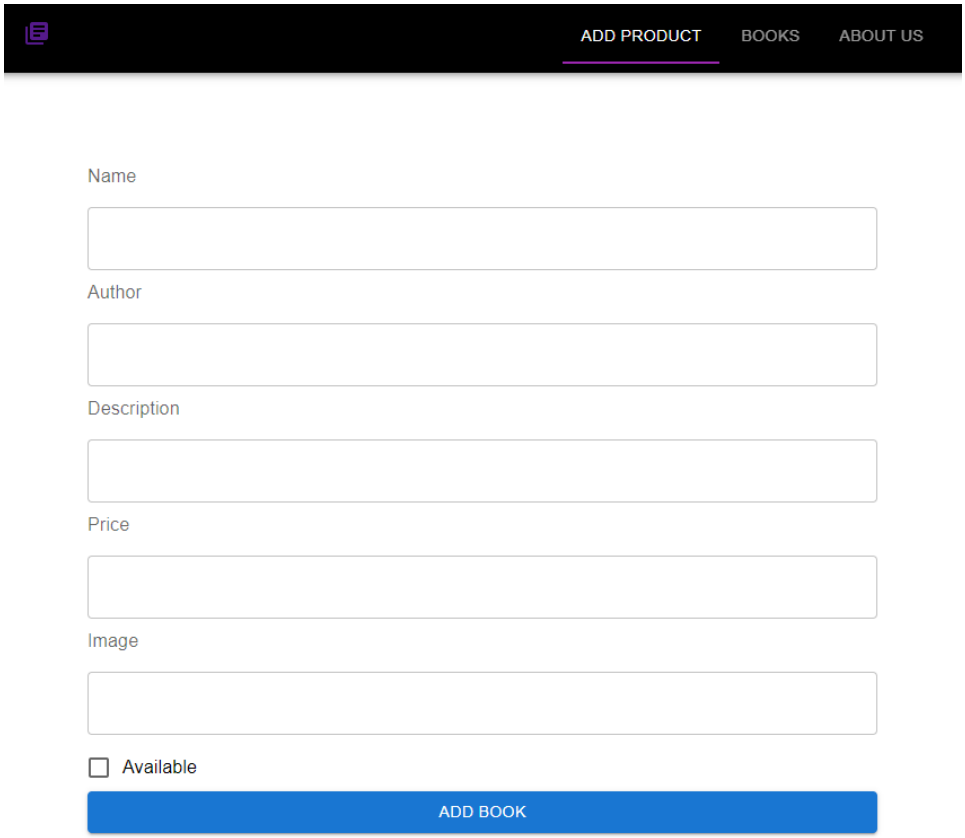

Рис. 2.6. Вікно додавання книжок

На інтерфейсі самих книжок є можливість перейти до їх редагування та видалення з бібліотеки (Рис.2.7) , (Рис.2.8) та (Рис.2.9)

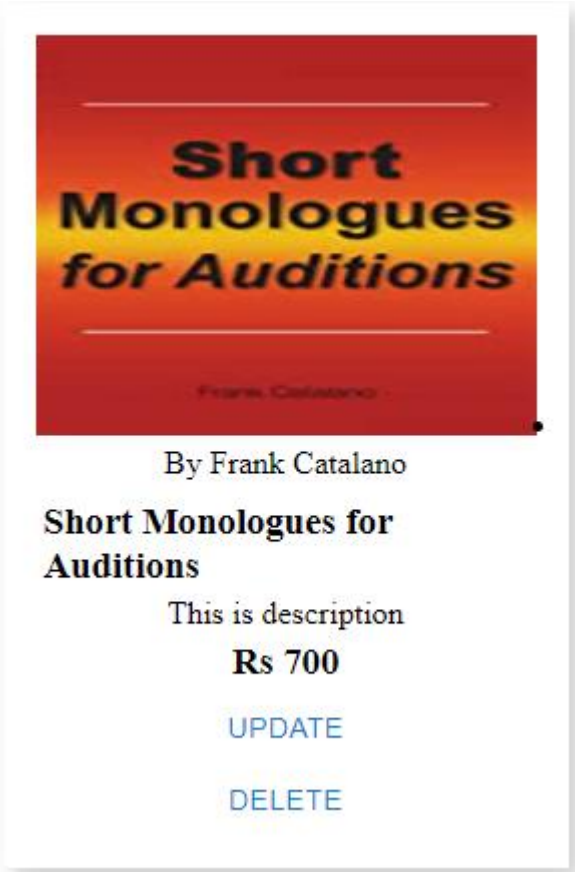

Рис. 2.7. Інтерфейс книжки

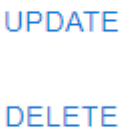

Рис. 2.8. Інтерфейс кнопок

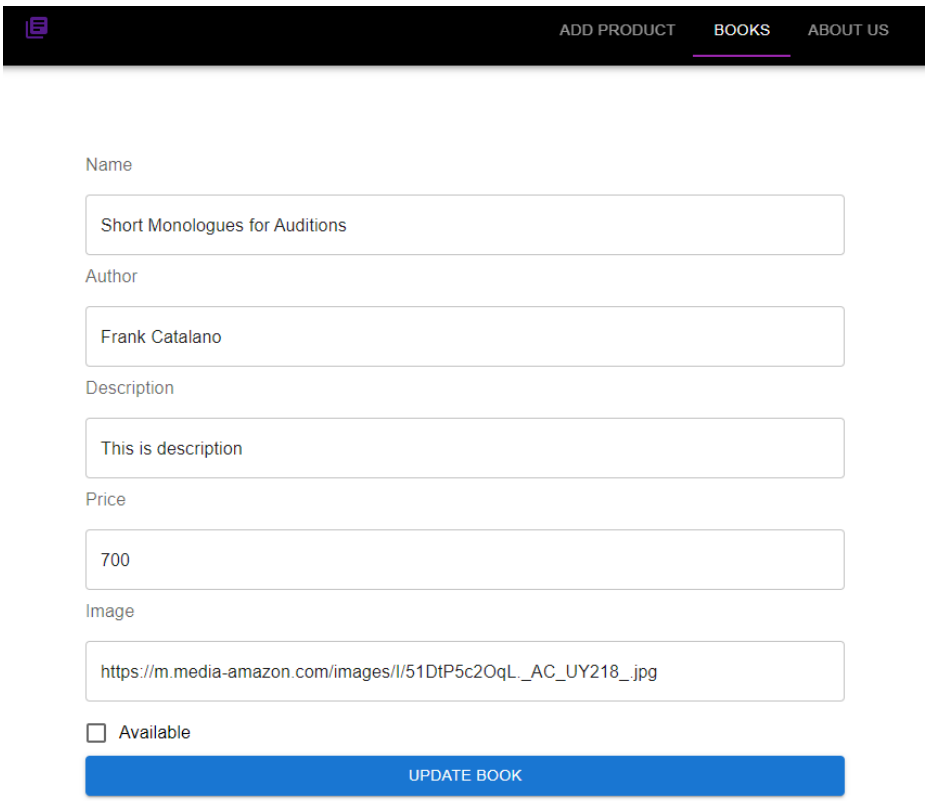

Рис. 2.9. Інтерфейс редагування

РОЗДІЛ 3
ЕКОНОМІЧНИЙ РОЗДІЛ

Вихідні дані розробки програмного забезпечення:

а) передбачуване число операторів – 1283;

б) коефіцієнт складності програми – 1,31;

в) коефіцієнт кореляції програми в ході її розробки – 0,2;

г) середня годинна заробітна плата програміста, грн/год – 100[19];

д) коефіцієнт кваліфікації програміста, обумовлений від стажу – 0,8;

е) вартість машино-години ЕОМ, грн/год – 8.

3.1. Визначення трудомісткості розробки програмного забезпечення

Трудомісткість розробки ПЗ можна розрахувати за формулою:

$$t = t_o + t_u + t_a + t_n + t_{отл} + t_д \text{, людино-годин,} \qquad (3.1)$$

де t_o – витрати праці на підготовку й опис поставленої задачі (приймається 60 людино-годин);

t_u – витрати праці на дослідження алгоритму рішення задачі,

t_a – витрати праці на розробку блок-схеми алгоритму,

t_n – витрати праці на програмування по готовій блок-схемі,

$t_{отл}$ – витрати праці на налагодження програми на ЕОМ,

$t_д$ – витрати праці на підготовку документації.

Складові витрати праці визначаються через умовне число операторів у ПЗ, яке розробляється.

Умовне число операторів (підпрограм):

$$Q = q \cdot C(1 + p), \text{ людино-годин,} \qquad (3.2)$$

де q – передбачуване число операторів,

C – коефіцієнт складності програми,

p – коефіцієнт кореляції програми в ході її розробки.

$$Q = 1283 \cdot 1{,}31 \cdot (1 + 0{,}2) = 2\,016{,}8$$

Витрати праці на вивчення опису задачі t_u визначається з урахуванням уточнення опису і кваліфікації програміста:

$$t_u = \frac{Q \cdot B}{(75..85) \cdot k}, \text{ людино-годин,} \qquad (3.3)$$

де B, яке дорівнює 1,4, – коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі,

k, яке дорівнює 0,8, – коефіцієнт кваліфікації програміста, обумовлений стажем роботи з даної спеціальності.

$$t_u = \frac{2\,016{,}8 \cdot 1{,}4}{82 \cdot 0{,}8} = 43{,}04, \text{ людино-годин.}$$

Витрати праці на розробку алгоритму рішення задачі:

$$t_a = \frac{Q}{(20..25)\cdot k}; \tag{3.4}$$

$$t_a = \frac{2016{,}8}{20\cdot 0{,}8} = 126{,}05, \text{ людино-годин.}$$

Витрати на складання програми по готовій блок-схемі:

$$t_n = \frac{Q}{(20..25)\cdot k}, \tag{3.5}$$

$$t_n = \frac{2016{,}8}{21\cdot 0{,}8} = 120, \text{ людино-годин.}$$

Витрати праці на налагодження програми на ЕОМ:

– за умови автономного налагодження одного завдання:

$$t_{отл} = \frac{Q}{(4..5)\cdot k}; \tag{3.6}$$

$$t_{отл} = \frac{2016{,}8}{5\cdot 0{,}8} = 504{,}2, \text{ людино-годин,}$$

– за умови комплексного налагодження завдання:

$$t_{отл}^{к} = 1{,}4 \cdot t_{отл};$$ (3.7)

$$t_{отл}^{к} = 1{,}4 \cdot 504{,}2 = 705.88$$, людино-годин.

Витрати праці на підготовку документації:

$$t_{д} = t_{др} + t_{до},$$ (3.8)

де $t_{др}$ – трудомісткість підготовки матеріалів і рукопису

$$t_{др} = \frac{Q}{(15..20) \cdot k};$$ (3.9)

$$t_{др} = \frac{2016{,}8}{15 \cdot 0{,}8} = 168{,}06$$, людино-годин.

де $t_{до}$ – трудомісткість редагування, печатки й оформлення документації

$$t_{до} = 0{,}75 \cdot t_{д};$$ (3.10)

$$t_{до} = 0{,}75 \cdot 168{,}06 = 126{,}04$$, людино-годин.

$$t_{д} = 168{,}06 + 126{,}04 = 294{,}1$$, людино-годин.

Отримаємо трудомісткість розробки програмного забезпечення:

$$t_д = 60 + 43{,}04 + 126{,}05 + 120 + 504{,}2 + 294{,}1 = 1147{,}39 \text{ , людино-годин.}$$

У результаті ми розрахували, що в загальній складності необхідно 1147,39 людино-годин для розробки даного програмного забезпечення.

3.2. Розрахунок витрат на створення програми

Витрати на створення ПЗ $K_{ПО}$ включають витрати на заробітну плату виконавця програми $З_{ЗП}$ і витрат машинного часу, необхідного для налагодження програми на ЕОМ.

$$K_{ПО} = З_{ЗП} + З_{МВ} \text{, грн,} \qquad (3.11)$$

де $З_{ЗП}$ – заробітна плата виконавців, яка визначається за формулою:

$$З_{ЗП} = t \cdot C_{ПР} \text{, грн,} \qquad (3.12)$$

де t – загальна трудомісткість, людино-годин,

$C_{ПР}$ – середня годинна заробітна плата програміста, грн/година.

$$З_{ЗП} = 1147{,}39 \cdot 100 = 114739 \text{ , грн.}$$

$З_{МВ}$ – вартість машинного часу, необхідного для налагодження програми на ЕОМ:

$$З_{МВ} = t_{отл} \cdot C_{МЧ} \text{, грн,} \qquad (3.13)$$

де $t_{отл}$ – трудомісткість налагодження програми на ЕОМ, год.

$C_{МЧ}$ – вартість машино-години ЕОМ, грн/год.

$$З_{МВ} = 504{,}2 \cdot 8 = 4033{,}6 \text{, грн,}$$

$$К_{ПО} = 114739 + 4033{,}6 = 118772{,}6 \text{, грн.}$$

Очікуваний період створення ПЗ:

$$T = \frac{t}{B_k \cdot F_p} \text{, мес,} \qquad (3.14)$$

де B_k – число виконавців,

F_p – місячний фонд робочого часу (при 40 годинному робочому тижні Fp = 176 годин).

$$T = \frac{1147{,}39}{1 \cdot 176} \approx 6{,}51 \text{ міс.}$$

Висновки: Час розробки цього програмного забезпечення становить 1147,39 людино-годин. Таким чином, очікуваний час розробки становить 6,51 місяця за 40-годинного робочого тижня (місячний фонд робочого часу 176 годин), а вартість створення програмного забезпечення становить 118 772,6 грн.

ВИСНОВКИ

Метою проекту було створення онлайн-бібліотеки, що дозволяє редагувати та купувати книги зручним та зрозумілим способом.

Програма є дуже корисною для тих, хто читає книги в Інтернеті. Перевага моєї програми в тому, що сайт працює практично на будь-якому пристрої з підключенням до інтернету.

Відповідно до метою даної роботи було вирішено всі завдання та вимоги до електронної бібліотеки, а саме:

- Зручний і зрозумілий графічний інтерфейс
- Можливість покупки книг
- Можливість редагування книг
- Визначено трудомісткість розробленої інформаційної системи (1147,39 людино-годин), проведений підрахунок вартості роботи по створенню програми (118772,6 грн) та розраховано час на його створення (6,51 міс).

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. About JavaScript [Електронний ресурс] : MDN // Режим доступу: https://developer.mozilla.org/en-US/docs/Web/JavaScript/About_JavaScript

2. Кеннеді Б., Мусіано Ч. HTML та XHTML. Повне керівництво/Б. Кеннеді, Ч. Мусіано. – М.: «Символ-Плюс», 2012. – 752 с.

3. Пасічник О. Г. Основи web-дизайну: навчальний посібник [Текст] / О.Г. Пасічник, О.В.Пасічник, І.В.Стеценко. - К.: Вид. група BHV, 2009. - 336 с.

4. Проектування інструментального програмного забезпечення. Методичні рекомендації для виконання курсової роботи / Уклад. В.О. Денисюк. – Вінниця : ВНТУ, 2014. - 22 с

5. Robson E. Head First HTML and CSS: A Learner’s Guide to Creating Standards-Based Web Pages / E. Robson, E. Freeman., 2012. – 768 с.

6. Database Concepts / D.Kroenke, D. Auer, S. Vandenberg, R. Yoder., 2019. – 552 с.

7. HTML & CSS – W3C [Електронний ресурс] // Режим доступу: http://www.w3.org/standards/webdesign/htmlcss

8. Нільсен Я. Web-дизайн: зручність використання Web-сайтів [Текст] / Якоб Нільсен, Хоа Лоранжер. - М.: Вільямс, 2007. - 368 с.

9. Coronel C. Database Systems: Design, Implementation, & Management 13th Edition / Carlos Coronel., 2018. – 816 с.

10. McKay E. UI is Communication: How to Design Intuitive, User Centered Interfaces by Focusing on Effective Communication 1st Edition / Everett McKay., 2013. – 378 с.

11. Farrington T. UX Design 2020: The Ultimate Beginner's Guide to User Experience / T. Farrington, T. Brooks, L. Ferrante., 2020.

12. Гончаров А. Ю. Web-дизайн: HTML, JavaScript та CSS. Кишеньковий довідник [Текст]/А. Ю.Гончаров. - КУДИЦЬ-ПРЕС, 2007. - 320 с.

13. Бородаев Д. В. Web-сайт как объект графического дизайна. Монография [Текст] / Д. В. Бородаев. – Х. : Септима ЛТД, 2006. – 288 с.

14. The State of JavaScript [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://2019.stateofjs.com/testing/.

15. Abelson H. Structure and Interpretation of Computer Programs / H. Abelson, G. Sussman., 1996. – 883 с. – (Second edition)

16. Node.js - Market Share & Web Usage Statistics [Електронний ресурс] // SimilarTech – Режим доступу до ресурсу: https://www.similartech.com/technologies/nodejs

17. ECMAScript 2021 Language Specification [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://tc39.es/ecma262/.

18. Зельдман Д. Web-дизайн за стандартами [Текст]/Джеффрі Зельдман. - М.: НТ Прес, 2005. - 432 с.

19. Nixon R. Learning PHP, MySQL & JavaScript: With jQuery, CSS & HTML5 (Learning PHP, MYSQL, Javascript, CSS & HTML5) / Robin Nixon..

20. Web-программист: средняя зарплата в Украине [Електронний ресурс] – Режим доступу до ресурсу: www.work.ua/ru/salary-web-программист/.

ДОДАТОК А

КОД ПРОГРАМИ

```
const Book = require("../model/Book")



const getAllBooks = async (req, res, next) => {
  let books
  try {
    books = await Book.find()
  } catch (err) {
    console.log(err)
  }


  if(!books) {
    return res.status(404).json({message:"No products found"})
  }
  return res.status(200).json({ books })
}


const getById = async (req, res, next) => {
  const id = req.params.id
  let book
  try {
    book = await Book.findById(id)
  } catch (err) {
    console.log(err)
  }
  if(!book) {
    return res.status(404).json({message:"No Book found"})
  }
  return res.status(200).json({ book })
```

```
}


const addBook = async (req, res, next) => {
  const {name, author, description, price, avaliable, image} = req.body
  let book
  try {
    book = new Book({
      name,
      author,
      description,
      price,
      avaliable,
      image
    })
    await book.save()
  } catch (err) {
    console.log(err)
  }


  if (!book) {
    return res.status(500).json({message:'Unable To Add'})
  }
  return res.status(201).json({ book })
}


const updateBook = async (req, res, next) => {
  const id = req.params.id
  const {name, author, description, price, avaliable, image} = req.body
  let book
  try {
```

```
    book = await Book.findByIdAndUpdate(id, {
      name,
      author,
      description,
      price,
      avaliable,
      image,
    })
    book = await book.save()
  } catch (err) {
    console.log(err)
  }
  if (!book) {
    return res.status(404).json({message:'Unable To Update By this ID'})
  }
  return res.status(200).json({ book })
}
const deleteBook = async (req, res, next) => {
  const id = req.params.id
  let book
  try {
    book = await Book.findByIdAndRemove(id)
  } catch {
    console.log(err)
  }
  if(!book) {
    return res.status(404).json({message:"Unable To Delete by this ID"})
  }
  return res.status(200).json({ message:"Product Successfully Deleted"})
}


exports.getAllBooks = getAllBooks
exports.addBook = addBook
```

```
exports.getById = getById
exports.updateBook = updateBook
exports.deleteBook = deleteBook

const mongoose = require('mongoose')


const Schema = mongoose.Schema


const bookSchema = new Schema({
  name: {
    type: String,
    required: true
  },
  author: {
    type: String,
    required: true
  },
  description: {
    type: String,
    required: true
  },
  price: {
    type: Number,
    required: true
  },
  avaliable: {
    type: Boolean,
  },
  image: {
    type: String,
    required: true
  }
```

```
})



module.exports = mongoose.model("Book", bookSchema)


// books
const express = require("express")
const router = express.Router()
const Book = require("../model/Book")
const booksController = require("../controllers/books-controller")


router.get("/", booksController.getAllBooks)
router.post("/", booksController.addBook)
router.get("/:id", booksController.getById)
router.put("/:id", booksController.updateBook)
router.delete("/:id", booksController.deleteBook)


module.exports = router


const express = require('express')
const mongoose = require('mongoose')
const router = require("./routers/book-routers")
const cors = require('cors')
const app = express()


// Middlewares
app.use(express.json())
app.use(cors())
app.use("/books", router)// localhost:5000/books
```

```
mongoose.connect("mongodb+srv://admin:F191odtZtuGjjmTU@cluster0.pw6ix.mongodb.net/?retryWrites=true&w=majority")
  .then(() => console.log("Connected To Database"))
  .then(() => {
    app.listen(5000)
  }).catch((err) => console.log(err))
```

```
import React from "react";
import Header from "./components/Header";
import { Route, Routes } from "react-router-dom";
import Home from "./components/Home";
import AddBook from "./components/AddBook";
import Books from "./components/Book/Books";
import About from "./components/About";
import BookDetail from "./components/Book/BookDetail";
function App() {
  return (
    <React.Fragment>
      <header>
        <Header />
      </header>
      <main>
        <Routes>
          <Route path="/" element={<Home />} exact />
          <Route path="/add" element={<AddBook />} exact />
          <Route path="/books" element={<Books />} exact />
          <Route path="/about" element={<About />} exact />
          <Route path="/books/:id" element={<BookDetail />} exact />
        </Routes>
      </main>
    </React.Fragment>
  );
}
```

```
export default App;



import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import App from './App';
import { BrowserRouter } from 'react-router-dom';


const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <BrowserRouter>
    <React.StrictMode>
      <App />
    </React.StrictMode>
  </BrowserRouter>
  );
ul {
  display: flex;
  flex-direction: row;
  justify-content: center;
  align-content: center;
  flex-wrap: wrap;
}
li {
  margin: 10px;
  padding: 10px;
  width: 250px;
  max-width: 250px;
  max-height: 400px;
}
.card {
  margin: auto;
```

```
  padding: 1rem;
  width: 100%;
  height: 100%;
  box-shadow: 5px 5px 10px #ccc;
  align-items: center;
  display: flex;
  flex-direction: column;
  justify-content: center;
}
.card > img {
  width: 100%;
  height: 50%;
}
.card article {
  padding: 5px;
}
.card h3 {
  padding: 4px;
}
.card p {
  padding: 3xp;
}
import React from 'react'
import axios from 'axios'
import { Button } from '@mui/material'
import { Link, useNavigate } from "react-router-dom"
import "./Book.css"
const Book = (props) => {
  const history = useNavigate()
  const {_id, name, author, description, price, image} = props.book
  const deleteHandler = async() => {
    await axios
      .delete(`http://localhost:5000/books/${_id}`)
```

```
      .then(res => res.data)
      .then(() => history("/"))
      .then(() => history("/books"))
  }
  return (
    <div className='card'>
      <img src={ image } alt={ name }/>
      <article>By { author }</article>
      <h3>{ name }</h3>
      <p>{ description }</p>
      <h3>Rs { price } </h3>
      <Button LinkComponent={Link} to={`/books/${_id}`} sx={{mt:'auto'}} >Update</Button>
      <Button onClick={deleteHandler} sx={{mt:'auto'}} >Delete</Button>
    </div>
  )
}


export default Book


import {
  Box,
  Button,
  Checkbox,
  FormControlLabel,
  FormLabel,
  TextField,
} from "@mui/material";
import axios from "axios";
import React, { useEffect, useState } from "react";
import { useNavigate, useParams } from "react-router-dom";
```

```
const BookDetail = () => {
  const [inputs, setInputs] = useState()
  const id = useParams().id
  const [checked, setChecked] = useState(false)
  const history = useNavigate()


  useEffect(() => {
    const fetchHandler = async () => {
      await axios
        .get(`http://localhost:5000/books/${id}`)
        .then((res) => res.data)
        .then((data) => setInputs(data.book));
    }
    fetchHandler()
  }, [id])


  const sendRequest = async () => {
    await axios
      .put(`http://localhost:5000/books/${id}`, {
        name: String(inputs.name),
        author: String(inputs.author),
        description: String(inputs.description),
        price: Number(inputs.price),
        image: String(inputs.image),
        available: Boolean(checked),
      }).then((res) => res.data)
  };
  const handleSubmit = (e) => {
    e.preventDefault()
    sendRequest().then(() => history("/books"))
  }
  const handleChange = (e) => {
```

```
  setInputs((prevState) => ({
    ...prevState,
    [e.target.name]: e.target.value,
  }))
}


return (
  <div>
    {inputs && (
      <form onSubmit={handleSubmit}>
        <Box
          display="flex"
          flexDirection="column"
          justifyContent={"center"}
          maxWidth={700}
          alignContent={"center"}
          alignSelf="center"
          marginLeft={"auto"}
          marginRight="auto"
          marginTop={10}
        >
          <FormLabel>Name</FormLabel>
          <TextField
            value={inputs.name}
            onChange={handleChange}
            margin="normal"
            fullWidth
            variant="outlined"
            name="name"
          />
          <FormLabel>Author</FormLabel>
          <TextField
            value={inputs.author}
```

```
  onChange={handleChange}
  margin="normal"
  fullWidth
  variant="outlined"
  name="author"
/>
<FormLabel>Description</FormLabel>
<TextField
  value={inputs.description}
  onChange={handleChange}
  margin="normal"
  fullWidth
  variant="outlined"
  name="description"
/>
<FormLabel>Price</FormLabel>
<TextField
  value={inputs.price}
  onChange={handleChange}
  type="number"
  margin="normal"
  fullWidth
  variant="outlined"
  name="price"
/>
<FormLabel>Image</FormLabel>
<TextField
  value={inputs.image}
  onChange={handleChange}
  margin="normal"
  fullWidth
  variant="outlined"
  name="image"
```

```
          />
          <FormControlLabel
            control={
              <Checkbox
                checked={checked}
                onChange={() => setChecked(!checked)}
              />
            }
            label="Available"
          />


          <Button variant="contained" type="submit">
            Update Book
          </Button>
        </Box>
      </form>
    )}
  </div>
 )
}


export default BookDetail

import React, { useEffect, useState } from 'react'
import axios from 'axios'
import Book from './Book'
import "./Book.css"
const URL = "http://localhost:5000/books"


const fetchHandler = async() => {
 return await axios.get(URL).then((res) => res.data)
```

```
}
const Books = () => {
  const [books, setBooks] = useState()
  useEffect(() => {
    fetchHandler().then(data => setBooks(data.books))
  },[])
  console.log(books)
  return (
    <div>
      <ul>
        {books && books.map((book, i) =>(
          <li className='book' key={i}>
            <Book book={book}/>
          </li>
        ))}
      </ul>
    </div>
  )
}
export default Books
import React from 'react'
import { Box, Typography } from "@mui/material"
const About = () => {
  return (
    <div>
      <Box display="flex" flexDirection="column" alignItems="center">
        <Typography sx={{ fontFamily: "fantasy" }} variant="h2">Tereshonok Vladimir</Typography>
        <Typography variant="h3">By MERN STACK</Typography>
      </Box>
    </div>
  )
}
export default About
```

```
import {
  Button,
  Checkbox,
  FormControlLabel,
  FormLabel,
  TextField,
} from "@mui/material";
import { Box } from "@mui/system";
import axios from "axios";
import React, { useState } from "react";
import { useNavigate } from "react-router-dom";


const AddBook = () => {
  const history = useNavigate();
  const [inputs, setInputs] = useState({
    name: "",
    description: "",
    price: "",
    author: "",


    image: "",
  });
  const [checked, setChecked] = useState(false);
  const handleChange = (e) => {
    setInputs((prevState) => ({
      ...prevState,
      [e.target.name]: e.target.value,
    }));
    // console.log(e.target.name, "Value", e.target.value);
  };
```

```
const sendRequest = async () => {
  await axios
    .post("http://localhost:5000/books", {
      name: String(inputs.name),
      author: String(inputs.author),
      description: String(inputs.description),
      price: Number(inputs.price),
      image: String(inputs.image),
      available: Boolean(checked),
    })
    .then((res) => res.data);
};


const handleSubmit = (e) => {
  e.preventDefault();
  console.log(inputs, checked);
  sendRequest().then(() => history("/books"));
};


return (
  <form onSubmit={handleSubmit}>
    <Box
      display="flex"
      flexDirection="column"
      justifyContent={"center"}
      maxWidth={700}
      alignContent={"center"}
      alignSelf="center"
      marginLeft={"auto"}
      marginRight="auto"
      marginTop={10}
    >
```

```
<FormLabel>Name</FormLabel>
<TextField
  value={inputs.name}
  onChange={handleChange}
  margin="normal"
  fullWidth
  variant="outlined"
  name="name"
/>
<FormLabel>Author</FormLabel>
<TextField
  value={inputs.author}
  onChange={handleChange}
  margin="normal"
  fullWidth
  variant="outlined"
  name="author"
/>
<FormLabel>Description</FormLabel>
<TextField
  value={inputs.description}
  onChange={handleChange}
  margin="normal"
  fullWidth
  variant="outlined"
  name="description"
/>
<FormLabel>Price</FormLabel>
<TextField
  value={inputs.price}
  onChange={handleChange}
  type="number"
  margin="normal"
```

```
          fullWidth
          variant="outlined"
          name="price"
        />
        <FormLabel>Image</FormLabel>
        <TextField
          value={inputs.image}
          onChange={handleChange}
          margin="normal"
          fullWidth
          variant="outlined"
          name="image"
        />
        <FormControlLabel
          control={
            <Checkbox checked={checked} onChange={() => setChecked(!checked)} />
          }
          label="Available"
        />
        <Button variant="contained" type="submit">
          Add Book
        </Button>
      </Box>
    </form>
  )
}
export default AddBook

import React, { useState } from 'react'
import { AppBar, Tab, Tabs, Toolbar, Typography } from '@mui/material'
import LibraryBooksIcon from '@mui/icons-material/LibraryBooks'
import { NavLink } from 'react-router-dom'

const Header = () => {
```

```
  const [value, setValue] = useState()
  return (
    <div>
      <AppBar sx={{backgroundColor:"black"}} position="sticky">
        <Toolbar>
          <NavLink to="/" sx={{ color: "white"}}>
            <Typography>
              <LibraryBooksIcon/>
            </Typography>
          </NavLink>
            <Tabs
              sx={{ml: "auto"}}
              textColor="inherit" indicatorColor="secondary"
              value={value}
              onChange={(e,val) => setValue(val)}
              >
                <Tab LinkComponent={ NavLink } to="/add" label='Add Product'/>
                <Tab LinkComponent={ NavLink } to="/books" label='Books'/>
                <Tab LinkComponent={ NavLink } to="/about" label='About Us'/>
            </Tabs>
        </Toolbar>
        </AppBar>
    </div>
  )
}

export default Header
import { Typography, Button, Box } from '@mui/material'
import { Link } from 'react-router-dom'
import React from 'react'
const Home = () => {
  return (
    <div>
```

```
        <Box display="flex" flexDirection="column" alignItems="center">
          <Button LinkComponent={Link} to="/books" sx={{ marginTop: 15, backgroundColor: "grey"
}} variant="contained">
            <Typography variant="h3">View All products</Typography>
          </Button>
        </Box>
      </div>
    )
  }
  export default Home

  import React from "react";
  import Header from "./components/Header";
  import { Route, Routes } from "react-router-dom";
  import Home from "./components/Home";
  import AddBook from "./components/AddBook";
  import Books from "./components/Book/Books";
  import About from "./components/About";
  import BookDetail from "./components/Book/BookDetail";
  function App() {
    return (
      <React.Fragment>
        <header>
          <Header />
        </header>
        <main>
          <Routes>
            <Route path="/" element={<Home />} exact />
            <Route path="/add" element={<AddBook />} exact />
            <Route path="/books" element={<Books />} exact />
            <Route path="/about" element={<About />} exact />
            <Route path="/books/:id" element={<BookDetail />} exact />
          </Routes>
        </main>
```

```
    </React.Fragment>
  );
}
export default App;

import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import App from './App';
import { BrowserRouter } from 'react-router-dom';
const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <BrowserRouter>
    <React.StrictMode>
      <App />
    </React.StrictMode>
  </BrowserRouter>
  );
```

ДОДАТОК Б

ВІДГУК

керівника економічної частини

ДОДАТОК В

ПЕРЕЛІК ФАЙЛІВ НА ДИСКУ

Ім’я файлу	Опис
Пояснювальні документи	
	Пояснювальна записка кваліфікаційної роботи. Документ Word.
	Пояснювальна записка кваліфікаційної роботи в форматі PDF.
Програма	
	Архів. Містить коди програми.
Презентація	
	Презентація кваліфікаційної роботи.