Lab 9: Simple Book Management system

Objectives:

In this lab exercise, you will create a simple book management system where users can add books through a form, and the information about the books will be stored in a MySQL database. Additionally, users will be able to view all the books stored in the database.

Lab instruction:

- The LAB09 instruction and lab resources are posted on MS Teams channel LAB09 – Simple Book Management system of subject 953262 (your section).
- Download the zipped file of resources-lab09. Unzip the file and rename it as "se262_studentID_labname" on your local drive. (For example, se262_6521150xx_lab8)
- There are 4 steps according to the LAB09 sheet posted on the channel.
- The LAB09 is worth 26 points in total.
- Score criteria: full point (for output correct); -1 (for output does not correct); -1 (for not follow problem constraint)

Assignment Submission:

- Upload your solutions to MS Team assignments. The submission later than the 'due date' will get 50% off your score. At the 'close date', you cannot submit your assignment to the system.
- After you upload your works to MS teams, inform TA to verify your work on your computer.

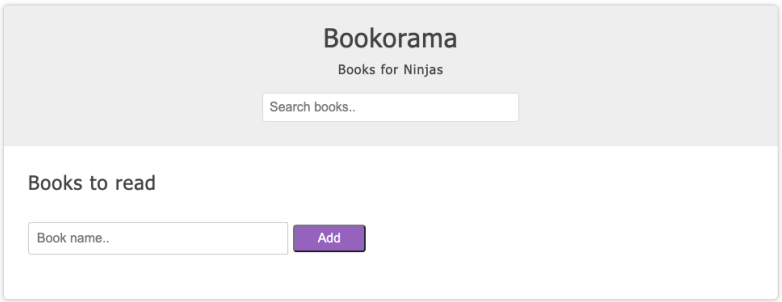
Step 1: Set Up the Database with MySQL Workbench or MySQL Shell (5 points)

- Create a MySQL database named `**book_management**`.
- Create a table named `**books**` with the following columns:
    - `**bookNo**` (integer, auto-increment, primary key)
    - `**bookName**` (varchar)
- Insert 4 new book records with the following titles:
    - Game of Thrones
    - Clash of kings
    - Name of Wind
    - The Wise Man's Fear

Step 2: Set Up the Node.js Backend (12 points)

- Initialize a Node.js project using `npm init -y`
- Install the required dependencies (express, mysql2 and body-parser).
- Create a server file (`**server.js**`) to handle HTTP requests
    - Setup express
    - Create a root route (**GET** method) with `**app.get('/')**`.
    - Send the words "Hello World" from the root route as the response.
    - Spin up your server on port 3000 with `**app.listen()**`.
    - Run the server with nodemon and access the root of your server in the browser (`**localhost:3000**`) to check whether the "Hello World" text is sent.
- Set up routes to handle:
    - Serving an HTML page with a form for adding books (**GET** /)
        - Download a lab_resource. You will see an `**index.html**` and `**styles.css**` in the lab_resource folder.
        - Create a new folder named `**public**` and put both the `**index.html**` and `**styles.css**` in it.

- Use the `**express.static**` middleware function to serve static files in a directory named `**public**` using `app.use(express.static('public'));`
- Modify the route **GET /** to send the `**index.html**` file as the response.
- Test your `**index.html**` on the server by typing http://localhost:3000 on the browser. You should have the UI like this.



- Handling form submission to add a book to the database (**POST /books/add**)
  - Sets up a connection to a MySQL database using `createConnection(config)` from the mysql2 module and specify the configuration options such as the host, user, password (if require) and database name.
  - Create a new route (`**/books/add**`) with `app.post()` to handle form submission.
  - Implement the callback function to handle the form submission and add a book to the database:
    - Parse a data named `**bookName**` from the HTML form using the `**body-parser**` of the request object
    - Create the SQL prepared statement using place holder (?) to insert a new book record the `**books**` table in the database.
  - Execute the SQL statement and `**bookName**` data with `**execute()**` of a MySQL 2 connection object. If there is an error, log the error and return `**res.status(500).send('Internal Server Error')**`. Otherwise, log the result and redirect the response to the route `**GET /**` using `**res.redirect('/')**`.
- Retrieving all books from the database (`**GET /books**`)
  - Create a new route (`**/books**`) with `app.get()`

- Implement the callback function of the `**app.get()**` to retrieve all books from the database
    - Create the SQL statement to retrieve all books from table books in the database
    - Execute the SQL statement with `**query()**` of an mySQL 2 connection object. If there is err then log the err and return `**res.status(500).send('Internal Server Error');**` Otherwise, log the result and send back the result in json format using `**res.json(rows);**`

Step 3: Create dynamic parts in the Frontend using JavaScript. (5 points)

- Open the `**index.html**`
- Use JavaScript to send form data to the backend when the form is submitted.
    - Add attributes `**action="/books/add"**` and `**method="post"**` of the form
- Display a list of books retrieved from the backend on the same page.
    - The `**fetch(Url)**` sends a HTTP request to the `**/books**` endpoint on the server to retrieve a list of books. Once the response is received, it parses the response body as JSON and then handles the parsed data by executing the callback function specified in the second `**.then()**` block. This allows you to work with the retrieved book data in your JavaScript code.

```
fetch('/books')
        .then(response => response.json())
        .then(books => {
        // Complete the codes to dynamically display a list of books retrieved from the backend.
          // Hint: you can create HTML elements on the fly according to the following commented codes in div id= 'book-list' element.
        }
```

If you log the JSON books, you will see a list of 4 books in JSON in the console

Frontend like below.

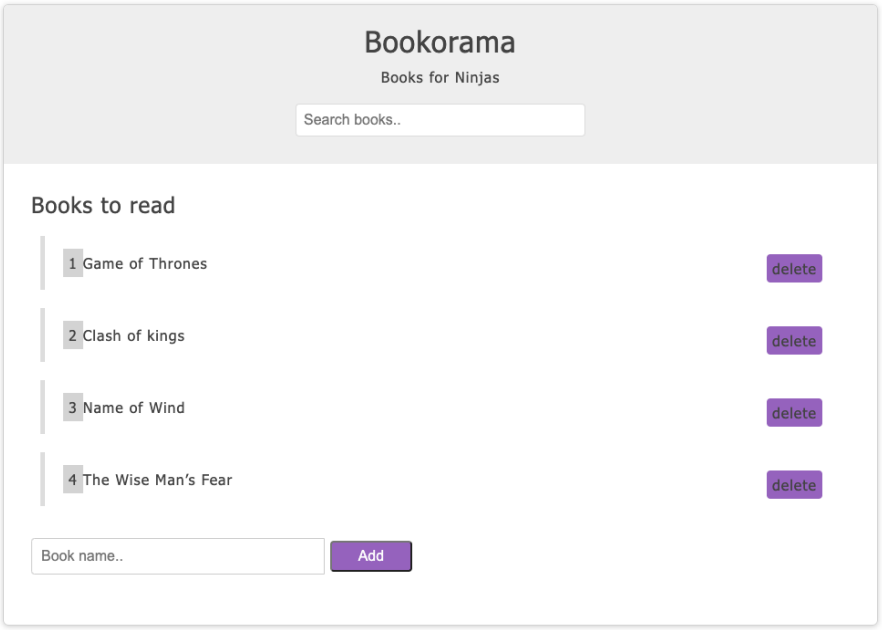
```
                                                    (index):61
▼ (4) [{…}, {…}, {…}, {…}] i
  ▶ 0: {userNo: 1, bookName: 'Game of Thrones'}
  ▶ 1: {userNo: 2, bookName: 'Clash of kings'}
  ▶ 2: {userNo: 3, bookName: 'Name of Wind'}
  ▶ 3: {userNo: 4, bookName: 'The Wise Man's Fea
    length: 4
  ▶ [[Prototype]]: Array(0)
```

Step 4: Testing (4 points)

- Test the application by adding new books through the form and verifying they are stored in the database.
- Test retrieving all books from the database and displaying them on the front end.

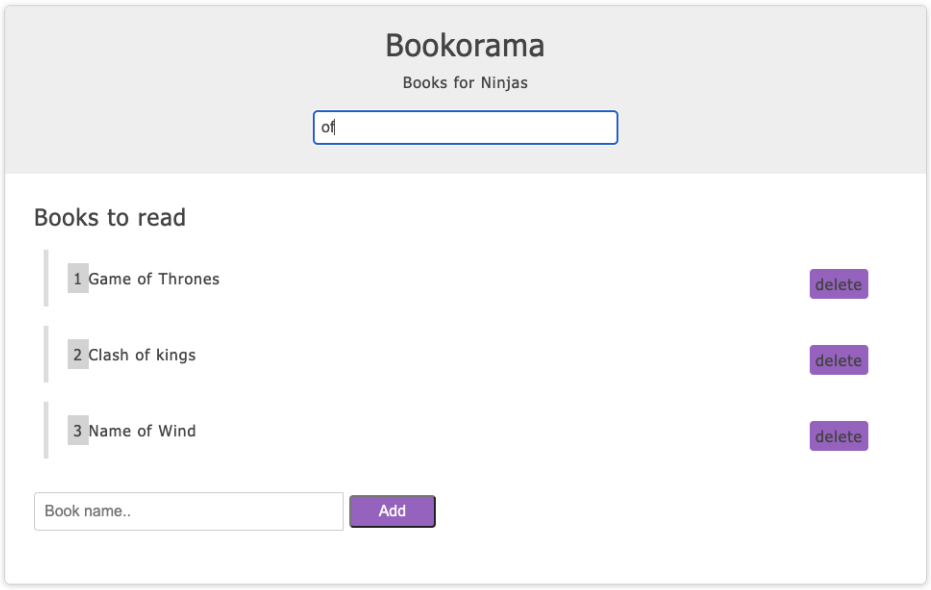
Challenge Tasks (Optional):

- Add functionality to delete books from the database. The UI will be something like below.

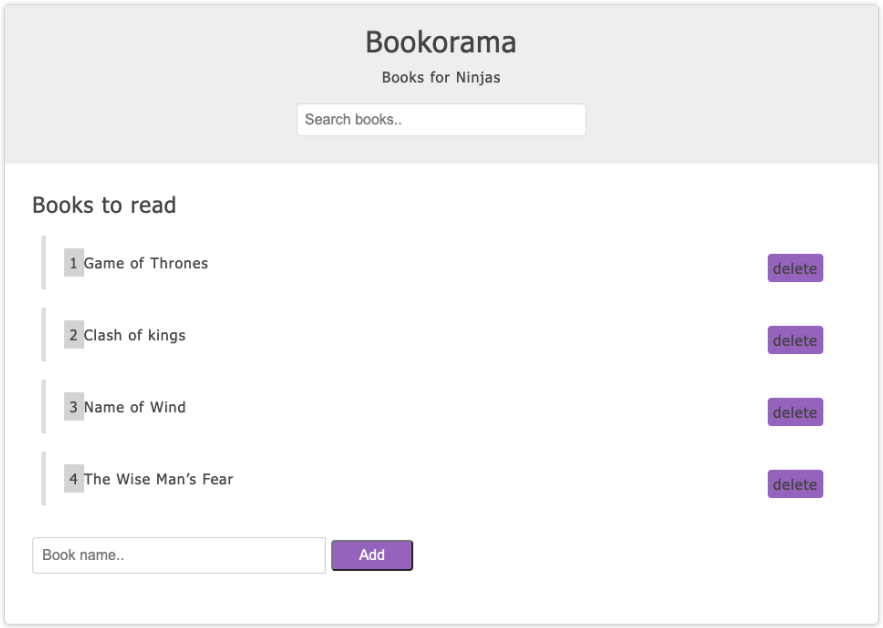


And after you click the delete button, the row of the delete book is removed from the UI and database.

- Add functionality to search books from the page. For example, when you type “o” in the search input, the page will display books having id 1, 2, and 3.



And when you remove “of” from the search input, the page brings those books back.



===== This is the end of today’s lab =====