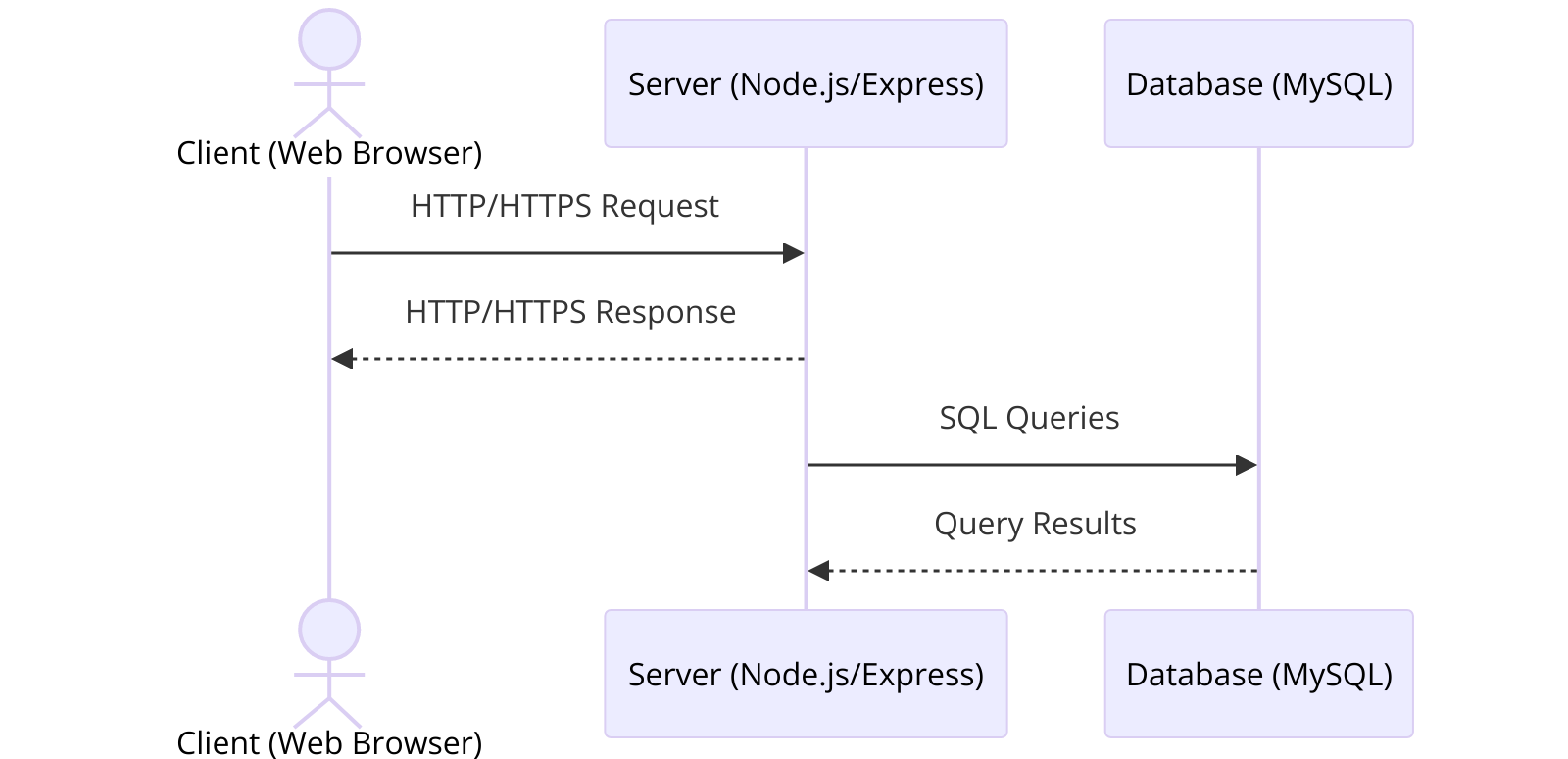
### **Outline**

Scholarly Library is a web application designed to manage and search educational materials. It allows users to view, add, edit, and delete materials, authors, and categories. Built using Node.js, Express, and MySQL, the application employs EJS as the templating engine and Bootstrap for styling. Users can perform CRUD operations on materials, authors, and categories, and utilize a search feature to find specific materials. The application is designed to be user-friendly and efficient, providing a seamless experience for managing educational resources.

### **Architecture**

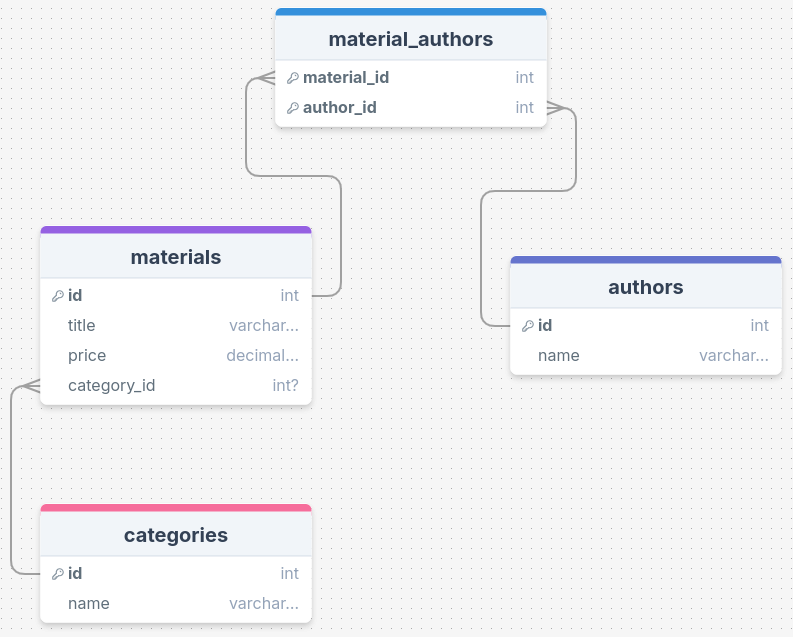
The application follows a client-server architecture. The server-side is built with Node.js and Express, handling HTTP requests and serving EJS templates. MySQL is used as the database to store information about materials, authors, and categories. The application tier consists of Express routes and middleware, while the data tier is managed by MySQL. The architecture ensures scalability and maintainability.



*Figure 1: Architecture*

### **Data Model**

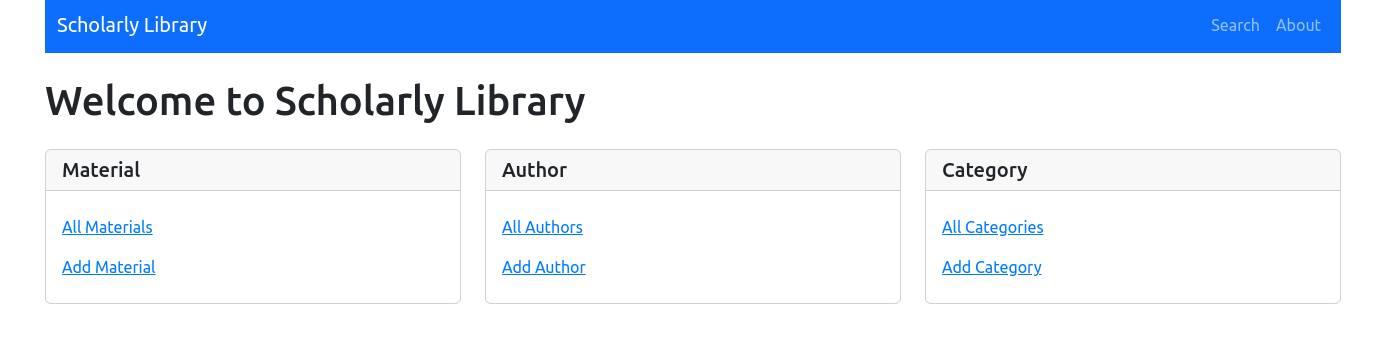
The data model consists of four main tables: materials, authors, categories, and material\_authors. The materials table stores information about educational materials, linked to categories and authors through foreign keys. The material\_authors table manages the many-to-many relationship between materials and authors.

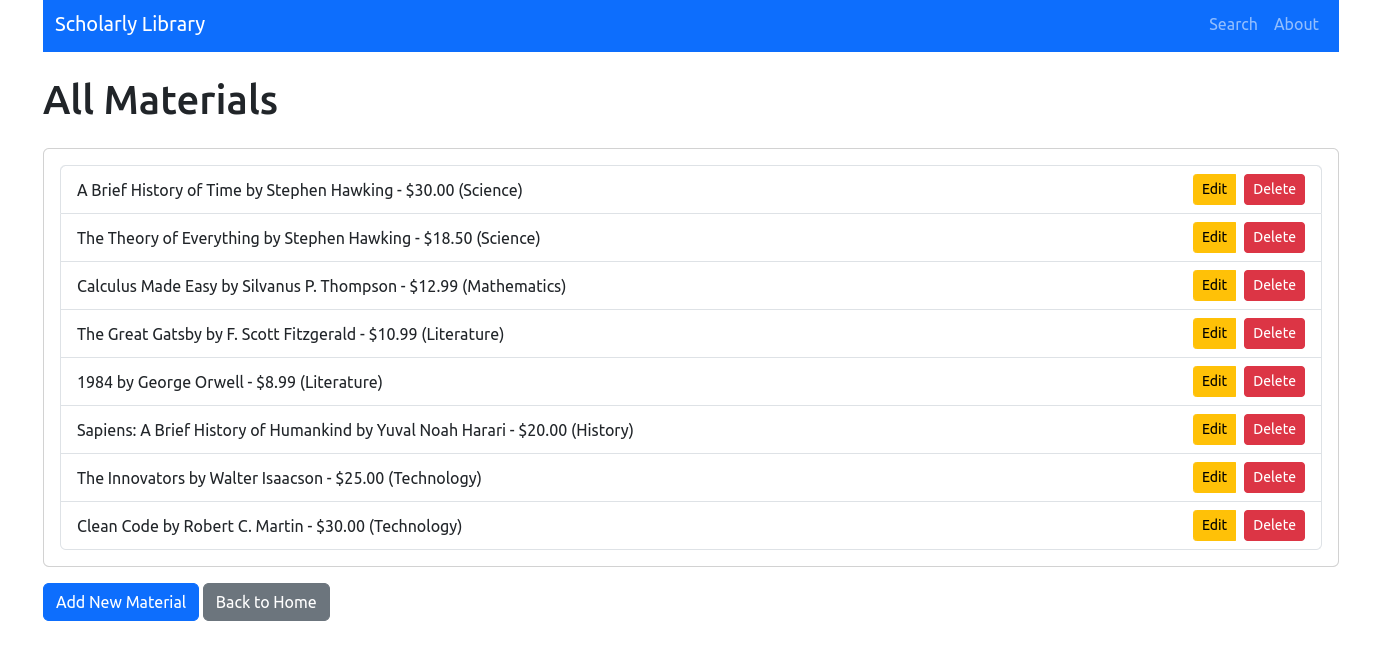


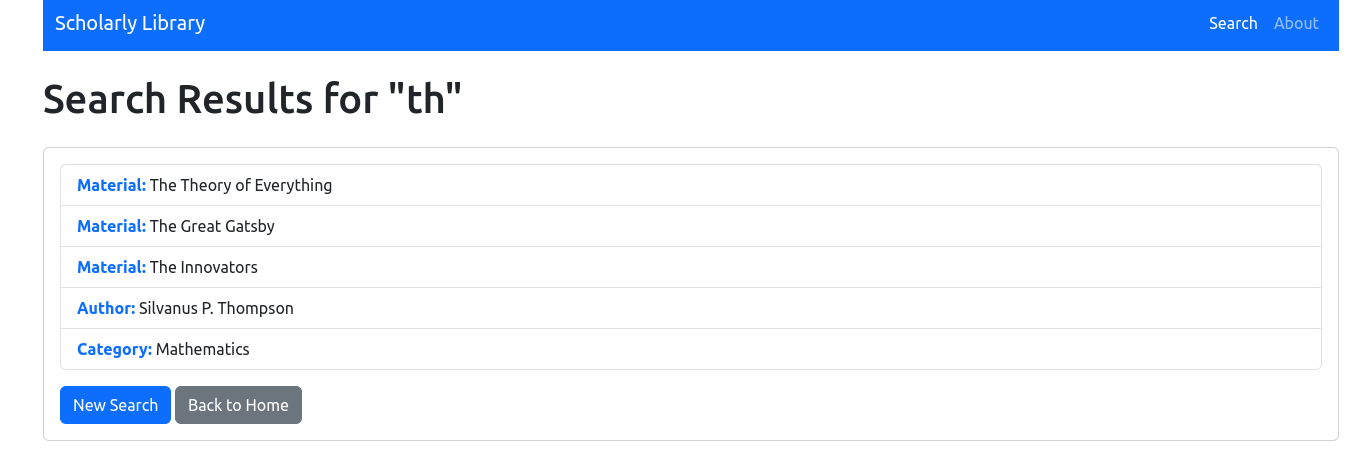
*Figure 2: Data Model*

### **User Functionality**

Users can navigate to the home page to access various features. The application allows users to view all materials, authors, and categories. Users can add new entries through forms, edit existing entries, and delete them if necessary. The search functionality enables users to find materials by title, author, or category. Each page is styled with Bootstrap for a responsive design.



*Figure 3: Home Page**Figure 4: Materials Listing*

*Figure 5: Search Results*

### **Advanced Techniques**

The application demonstrates advanced techniques such as using EJS for server-side rendering and handling complex SQL queries for data retrieval. For example, the SQL query to display all materials with their authors and categories is optimized using LEFT JOIN and GROUP\_CONCAT:

router.get("/materials", (req, res) => {

*const* sqlQuery = `

SELECT

materials.id,

materials.title,

materials.price,

categories.name AS category,

GROUP\_CONCAT(authors.name SEPARATOR ', ') AS authors

FROM materials

LEFT JOIN categories ON materials.category\_id = categories.id

LEFT JOIN material\_authors ON materials.id = material\_authors.material\_id

LEFT JOIN authors ON material\_authors.author\_id = authors.id

GROUP BY materials.id

`;

db.query(sqlQuery, (err, result) => {

if (err) throw err;

res.render("materials/materials", { materials: result });

});

});

Additionally, the application uses environment variables for database configuration, enhancing security and flexibility:

DB\_HOST=localhost

DB\_USER=<your\_mysql\_username>

DB\_PASSWORD=<your\_mysql\_password>

DB\_DATABASE=scholarly\_library

These techniques showcase the ability to build a robust and secure web application with efficient data handling.