



A Full-Stack Library Web Application with Server-Side Rendering

Nabil Nasreldeen Mohammed

4th year student,

Computer science Department

Supervised By

Dr / Amal Ahmed Mohammed

Mathematics Department, Faculty of Science, South Valley University

2022 / 2023

Contents

Chapter 1: Web Development Technologies	4
1.1 Introduction	5
1.2 Front-End Development	11
1.2.1 HTML	11
1.2.2 CSS	12
1.2.3 JavaScript	13
1.2.4 EJS(Embedded JavaScript)	14
1.3 Responsive Web Design	15
1.4 Back-End Development	18
1.4.1 Node.js	18
1.4.2 MongoDB	19
1.4.3 Express.js	20
1.5 Pages Rendering Approaches	21
1.5.1 SSR(Server-Side Rendering)	21
1.5.2 CSR(Client-Side Rendering)	22
1.5.3 SSG(Static-Site Generation)	22
1.5.4 PWA(Progressive Web Apps)	23
1.5.5 Hybrid Approaches	24
1.5.6 Conclusion	25
1.6 Database Management	26

1.7 Web-Security	29
1.7.1 Protecting Against Web Vulnerabilities	30
1.7.2 Authentication and Authorization	30
1.7.3 Protecting Against Attacks on Third-Party Dependencies	31
1.7.4 Best Practices for Secure Coding	32
1.7.5 Conclusion	33
Chapter 2: HTML	34
2.1 Introduction	35
2.2 Getting Started	35
2.3 HTML Document Structure	36
2.4 HTML Tags	37
2.5 HTML Element	39
2.6 HTML Attributes	41
Chapter 3: CSS	43
3.1 Introduction	44
3.2 Getting Started	44
3.3 CSS Selectors	45
3.4 CSS Properties and Values	47
3.5 CSS Box Model	48
Chapter 4: JavaScript	50
4.1 Introduction	51

4.2 Getting Started	51
4.3 Variables	52
4.4 Data Types	53
4.5 Functions	53
4.6 Events	54
4.7 Conditional Statements	55
4.8 Loops	56
Chapter 5: A Full-Stack Web Application	57
5.1 Introduction	58
5.2 The Website Features	60
5.2.1 Authors and Books Visualization	60
5.2.2 Authors Search and Books Advanced Search	62
5.2.3 Responsive Page Sizes	63
5.2.4 External Images Server	65
5.2.5 Images Lazy Loading	65
5.2.6 Pagination	67
5.2.7 Admin Account	68
5.2.8 Input Validations and Error Messages	70
5.2.9 Details About Input Images	73
5.2.10 Security	74
5.3 The Website Links	75
References	76

Chapter 1

Web Development Technologies

1.1 Introduction:

Web development is the process of creating websites and web applications.

It involves designing, building, and maintaining websites using various technologies, programming languages, and tools.

Here are some key aspects of web development:

- **Front-End Development:** Front-end development focuses on the user interface and user experience of a website. It involves writing HTML, CSS, and JavaScript code to create the visual and interactive elements that users see and interact with in their web browsers.

- **Back-End Development:** Back-end development involves creating the server-side logic and functionality of a website.

It includes writing code that handles database operations, user authentication, server communication, and other server-side processes.

Common back-end languages include PHP, Python, Ruby, Node.js and C#.

- **Full-Stack Development:** Full-stack development combines both front-end and back-end development.

Full-stack developers are proficient in both client-side and server-side technologies, allowing them to handle all aspects of web development.

- **Web Development Frameworks:**
Frameworks provide a structured way to build web applications.
They offer pre-defined libraries, tools, and components that simplify development tasks and enhance productivity.
Popular frameworks include React.js, Angular, Vue.js (front-end), Django, Ruby on Rails, Laravel, Express.js (back-end), and many more.
- **Responsive Web Design:** With the growing use of mobile devices, it's crucial to create websites that adapt to different screen sizes.
Responsive web design ensures that websites look and function well on various devices, such as desktops, tablets, and smartphones.

- Database Management: Web applications often rely on databases to store and retrieve data.

Developers work with database management systems like MySQL, PostgreSQL, MongoDB, or SQLite to manage data effectively.

- Version Control: Version control systems, such as Git, are essential for tracking changes in code, collaborating with other developers, and rolling back to previous versions if needed.

They help manage codebase and ensure smooth development workflows.

- Deployment and Hosting: After development, web applications need to be

deployed to a server and made available on the internet.

Various hosting options exist, including shared hosting, virtual private servers (VPS), cloud hosting platforms (e.g., AWS, Azure), and specialized hosting services.

- Security: Web developers need to be mindful of security best practices to protect websites from vulnerabilities and attacks.

This includes input validation, proper authentication and authorization mechanisms, secure communication (HTTPS), and protection against common web vulnerabilities like cross-site scripting (XSS) and SQL injection.

- Continuous Learning: Web development is a dynamic field, and technologies evolve rapidly.

Staying updated with the latest trends, frameworks, and best practices is crucial for web developers to maintain their skills and deliver modern, efficient, and secure web applications.

....These are just some of the fundamental aspects of web development.

The field is vast and continuously evolving, offering many opportunities for developers to specialize in specific areas and technologies.

1.2 Front-End Development:

1.2.1 HTML:

HTML (Hypertext Markup Language) is the standard markup language used to create web pages.

It provides the structure of the web page and defines the content of the page.

HTML is a simple language that uses tags to define elements such as headings, paragraphs, and images.

Example:

```
<!DOCTYPE html>
<html>
<head>
  <title>Example</title>
</head>
<body>
  <h1>Hello World!</h1>
  <p>This is an example of HTML code.</p>
  
</body>
</html>
```

1.2.2 CSS:

CSS (Cascading Style Sheets) is a style sheet language used to describe the presentation of a web page.

It is used to define the layout, colors, fonts, and other visual elements of a web page.

CSS can be used to create responsive designs that adapt to different screen sizes and devices.

Example:

```
body {  
    background-color: #f2f2f2;  
    font-family: Arial, sans-serif;  
}  
h1 {  
    color: #333;  
    font-size: 36px;  
}  
p {  
    color: #666;  
    font-size: 18px;  
}  
img {  
    max-width: 100%;  
    height: auto;  
}
```

1.2.3 JavaScript:

JavaScript is a programming language used to create interactive and dynamic web pages.

It is used to add functionality to web pages, such as form validation, animations, and dynamic content.

JavaScript can be used on both the client-side and server-side of web development.

Example:

```
const button = document.getElementById("button");
const text = document.getElementById("text");
button.addEventListener("click", function () {
    if (text.innerHTML == "This is some text that will be
manipulated with JavaScript.") {
        text.innerHTML = "The text has been changed!";
        button.innerHTML = "Click me to change the text
back!";
    } else {
        text.innerHTML = "This is some text that will be
manipulated with JavaScript.";
        button.innerHTML = "Click me to change the text!";
    }
});
```

1.2.4 EJS:

EJS (Embedded JavaScript) is a templating engine used to generate HTML markup with plain JavaScript.

It allows developers to create reusable templates that can be used across multiple pages.

EJS can be used with Node.js and Express.js to create dynamic web pages.

Example:

```
<!DOCTYPE html>
<html>
<head>
  <title><%= title %></title>
</head>
<body>
  <h1><%= heading %></h1>
  <p><%= content %></p>
</body>
</html>
```

We can see that the code above is similar to HTML.

1.3 Responsive Web Design:

Responsive web design is an approach to designing websites that aims to provide an optimal viewing experience across a wide range of devices, including desktops, laptops, tablets, and smartphones.

The goal of responsive web design is to ensure that the website looks and functions well on any device, regardless of its screen size or resolution.

Responsive web design involves using a combination of flexible grids, images, and CSS media queries to adjust the layout and content of the website based on the device being used to access it.

This allows the website to adapt to different screen sizes and orientations, ensuring that the user has a consistent and user-friendly

experience, regardless of the device being used.

One of the key benefits of responsive web design is that it eliminates the need for separate mobile and desktop versions of the website. Instead, the website is designed to be responsive from the outset, ensuring that it looks and functions well on any device.

This not only saves time and money, but it also provides a more consistent user experience, as users can access the same content and features regardless of the device they are using.

Responsive web design also has significant benefits for search engine optimization (SEO).

Google has stated that responsive web design is its recommended approach to mobile-friendly websites, and websites that are not mobile-friendly may not rank as well in search results. By using responsive web design, website

owners can ensure that their website is optimized for search engines, regardless of the device being used to access it.

...In **conclusion**, responsive web design is a crucial approach to designing websites in today's digital age.

With more and more users accessing websites on a wide range of devices, it is essential that websites are designed to be responsive and adaptable.

By using flexible grids, images, and CSS media queries, website owners can ensure that their website provides a consistent and user-friendly experience, regardless of the device being used.

Additionally, responsive web design has significant benefits for SEO, ensuring that the website is optimized for search engines and can rank well in search results.

1.4 Back-End Development:

1.4.1 Node.js:

Node.js is a server-side JavaScript runtime environment used to create scalable and high-performance web applications.

It provides a non-blocking I/O model that allows for efficient handling of multiple requests.

Node.js can be used with various frameworks, including Express.js and Hapi.js.

Example:

```
const http = require('http');
const server = http.createServer((req, res) => {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');
  res.end('Hello World\n');
});
server.listen(port, hostname, () => {
  console.log(`Server running at
http://${hostname}:${port}/`);
});
```

1.4.2 MongoDB:

MongoDB is a NoSQL database used to store and retrieve data for web applications.

It provides a flexible schema that allows for easy scaling and performance.

MongoDB can be used with Node.js and Mongoose library to create robust web applications.

Example(in node.js):

```
const mongoose = require('mongoose');
mongoose.connect('mongodb://localhost/test',
{ useNewUrlParser: true });
const Schema = mongoose.Schema;
const userSchema = new Schema({
  name: String,
  age: Number,
  email: String
});
const User = mongoose.model('User', userSchema);
const user = new User({
  name: 'John Doe',
  age: 30,
  email: 'john.doe@example.com'
});
```

```
user.save(err, user) => {  
  if (err) return console.error(err);  
  console.log(user);  
});
```

1.4.3 Express.js:

Express.js is a web application framework used to create server-side applications with Node.js. It provides a simple and flexible API for handling HTTP requests and responses.

Express.js can be used with various middleware to add functionality to web applications.

Example:

```
const express = require('express');  
const app = express();  
app.get('/', (req, res) => {  
  res.send('Hello World!');  
});  
app.listen(3000, () => {  
  console.log('Example app listening on port  
3000!');  
});
```

1.5 Approaches to Render Web pages or Web Applications:

There are several approaches to rendering web pages or web applications, depending on the technology stack and architectural choices.

Here are some common approaches:

1.5.1 Server-Side Rendering (SSR):

In SSR, the web server processes the request and generates the HTML on the server-side. The fully rendered HTML is then sent to the client, which can display it immediately.

This approach allows search engines to easily crawl and index the content, as well as provides better initial page load times.

Popular server-side rendering frameworks include Next.js (with React) and Next.js (with Vue.js) or Node.js(with ejs).

1.5.2 Client-Side Rendering (CSR):

In CSR, the web server sends a minimal HTML file and the necessary JavaScript files to the client.

The client-side JavaScript framework then takes over and fetches data from APIs and dynamically renders the content in the browser.

This approach provides a more interactive user experience but may result in slower initial page load times and can have SEO implications.

Examples of CSR frameworks are React (with libraries like React Router) and Vue.js (with Vue Router).

1.5.3 Static Site Generation (SSG):

SSG pre-generates HTML files for each page at build time, which are then served directly to the client.

This approach offers the benefits of fast load times and improved SEO, as the content is readily available.

It's suitable for websites with relatively static content that doesn't require real-time data.

Frameworks like Gatsby (with React) and Gridsome (with Vue.js) support static site generation.

1.5.4 Progressive Web Apps (PWA):

PWAs are web applications that can function offline and offer a native-like experience.

They use a combination of techniques like service workers, caching, and manifest files to provide offline capabilities and push notifications.

PWAs can be built using any of the above rendering approaches, but they often employ CSR to enable dynamic updates and interactivity.

1.5.5 Hybrid Approaches:

Some frameworks and libraries, such as Next.js and Angular Universal, support hybrid rendering approaches. These approaches combine SSR and CSR to get the benefits of both. For example, the initial page load can be rendered on the server-side, and subsequent interactions can be handled client-side.

1.5.6 Conclusion:

It's worth noting that the choice of rendering approach depends on various factors like project requirements, scalability, performance goals, and development team expertise.

Each approach has its trade-offs, and the best choice often depends on the specific use case and priorities.

1.6 Database Management and MongoDB:

Database management is a crucial aspect of modern business operations, and MongoDB is one of the most popular database management systems in use today.

MongoDB is a document-based database that is designed to be highly scalable and flexible, making it an ideal choice for businesses and organizations of all sizes.

One of the key benefits of MongoDB is its flexibility.

Unlike traditional relational databases, which require data to be organized into tables with predefined schemas, MongoDB allows data to be stored in flexible, JSON-like documents.

This makes it easy to store and retrieve data in a way that is natural and intuitive, without the need for complex joins or other operations.

MongoDB is designed to be highly scalable, allowing businesses and organizations to easily add or remove servers as needed to meet changing demand.

This makes it an ideal choice for businesses and organizations that need to handle large amounts of data or that experience rapid growth.

MongoDB also has a number of features that make it ideal for use in modern web applications.

For example, MongoDB supports dynamic schema, which allows developers to easily add or remove fields from documents as needed. This makes it easy to adapt to changing

requirements and to quickly iterate on new features.

Additionally, MongoDB has a number of built-in features that make it easy to work with data in a web application context.

It also includes support for full-text search, making it easy to search for data across multiple fields and documents.

....**In conclusion**, MongoDB is a powerful and flexible database management system that is ideal for use in modern web applications.

Its flexibility and scalability make it an ideal choice for businesses and organizations of all sizes, while its built-in features make it easy to work with data in a web application context.

Whether you are building a small web application or a large-scale enterprise system, MongoDB is a powerful tool that can help you manage your data effectively and efficiently.

1.7 Web-Security with Node.js and Express.js:

Web security is a critical aspect of modern web development.

With the increasing number of cyberattacks and data breaches, it is more important than ever to ensure that web applications are secure.

Node.js and Express.js are two powerful tools that can help developers build secure web applications.

Here, we will explore some of the key security considerations and best practices when building web applications with Node.js and Express.js.

1.7.1 Protecting Against Common Web Vulnerabilities:

One of the most important considerations when building web applications is to ensure that the application is protected against common web vulnerabilities, such as cross-site scripting (XSS) and SQL injection.

This can be achieved by using middleware and other security features provided by Express.js, such as the helmet middleware, which adds a number of security headers to HTTP responses to help protect against common attacks.

1.7.2 Authentication and Authorization:

Another important consideration is to ensure that the application is protected against unauthorized access.

This can be achieved by implementing authentication and authorization features, such

as using JSON Web Tokens (JWT) to authenticate users and restrict access to certain parts of the application.

Express.js provides a number of middleware and other features that make it easy to implement these types of security features.

1.7.3 Protecting Against Attacks on Third-Party Dependencies:

It is also important to ensure that the application is protected against attacks that exploit vulnerabilities in third-party dependencies.

This can be achieved by using tools such as npm audit to identify and fix vulnerabilities in third-party dependencies, as well as by keeping dependencies up-to-date with the latest security patches.

1.7.4 Best Practices for Secure Coding:

In addition to these security considerations, there are several best practices that developers should follow when building web applications with Node.js and Express.js.

These include using secure coding practices, such as input validation and output encoding, as well as using SSL/TLS encryption to protect data in transit.

Developers should also follow the principle of least privilege, which involves limiting access to sensitive data and functions to only those users and processes that need them.

1.7.5 Conclusion:

Web security is a critical aspect of modern web development, and Node.js and Express.js provide developers with powerful tools for building secure web applications.

By following best practices and implementing security features such as authentication, authorization, and middleware, developers can help protect their applications against common web vulnerabilities and attacks.

Whether building a small web application or a large-scale enterprise system, Node.js and Express.js are powerful tools that can help developers build secure and reliable web applications.

Chapter 2

HTML

2.1 Introduction:

HTML, or HyperText Markup Language, is a markup language used to create and structure content on the web. It is the foundation of web development, and every web developer should have a good understanding of HTML. In this tutorial, we will cover the basics of HTML, including tags, elements, attributes, and more.

2.2 Getting Started:

To create an HTML document, we need a text editor.

We can use any text editor you like, such as Notepad, Sublime Text, or Visual Studio Code. Open our text editor and create a new file.

Save the file with a .html extension.

2.3 HTML Document Structure:

An HTML document consists of two main sections: the head section and the body section. The head section contains information about the document, such as the title and any links to external stylesheets or scripts.

The body section contains the content of the document.

Here is an example of a basic HTML document:

```
<!DOCTYPE html>
<html>

<head>
  <title>My First HTML Document</title>
</head>

<body>
  <h1>Welcome to my website!</h1>
  <p>This is my first HTML document.</p>
</body>

</html>
```

The `<!DOCTYPE html>` declaration at the beginning of the document tells the browser that this is an HTML5 document.

The `<html>` tag marks the beginning and end of the HTML document.

The `<head>` tag contains information about the document, such as the title.

The `<body>` tag contains the content of the document, including headings, paragraphs, links, and more.

2.4 HTML Tags:

HTML tags are used to mark up content on the web.

Tags are enclosed in angle brackets, like this: `<tag>` .

The most basic tag is the `<html>` tag, which marks the beginning and end of an HTML document.

Other common tags include:

- <head> - marks the beginning and end of the head section
- <body> - marks the beginning and end of the body section
- <h1> - marks a level 1 heading
- <p> - marks a paragraph
- <a> - marks a link
- - marks an image

Here is an example of how to use these tags:

```
<!DOCTYPE html>
<html>
<head>
  <title>My First HTML Document</title>
</head>
<body>
  <h1>Welcome to my website!</h1>
  <p>This is my first HTML document.</p>
  <a href="https://www.example.com">Click here
to visit Example.com</a>
  
</body>
</html>
```

In this example, we have used the `<h1>` tag to create a level 1 heading that says "Welcome to my website!" We have also used the `<p>` tag to create a paragraph that says "This is my first HTML document."

The `<a>` tag is used to create a link to Example.com, and the `` tag is used to display an image of a cat.

2.5 HTML Elements:

HTML elements are made up of tags and content. For example, a paragraph element would look like this:

```
<p>This is a paragraph.</p>
```

The opening tag (`<p>`) marks the beginning of the paragraph element, and the closing tag (`</p>`) marks the end. The content of the element is "This is a paragraph."

Here are some more examples of HTML elements:

- Heading element: `<h1>Heading 1</h1>`
- Paragraph element: `<p>This is a paragraph.</p>`
- Link element: `https://www.example.com`
Click here to visit Example.com
- Image element: ``
- List element: `Item 1Item 2`
- Table element: `<table><tr><th>Header 1</th><th>Header 2</th></tr><tr><td>Row 1, Column 1</td><td>Row 1, Column 2</td></tr></table>`

2.6 HTML Attributes:

HTML attributes provide additional information about an element. Attributes are added to the opening tag of an element, like this:

```
<a href="https://www.example.com">This is a  
link</a>
```

The href attribute specifies the URL that the link should point to.

Here are some more examples of HTML attributes:

- src - specifies the URL of an image or other media file
- alt - specifies alternative text for an image
- class - specifies one or more class names for an element
- id - specifies a unique id for an element
- style - specifies inline CSS styles for an element

Here is an example of how to use HTML attributes:

```
<!DOCTYPE html>
<html>
<head>
  <title>My First HTML Document</title>
  <style>
    .red-text {
      color: red;
    }
  </style>
</head>
<body>
  <h1>Welcome to my website!</h1>
  <p class="red-text">This text is red.</p>
  
</body>
</html>
```

In this example, we have used the `class` attribute to give the paragraph element a class of `red-text`. We have also used inline CSS styles to make the text red.

Chapter 3

CSS

3.1 Introduction:

CSS, or Cascading Style Sheets, is a style sheet language used to describe the presentation of a document written in HTML or XML. CSS allows you to separate the presentation of a document from its content, making it easier to maintain and update. In this tutorial, we will cover the basics of CSS, including selectors, properties, values, and more.

3.2 Getting Started:

To add CSS to an HTML document, we can use the `<style>` tag. The `<style>` tag should be placed in the `<head>` section of the HTML document or we can place the content of `<style>` tag in external `.css` file.

Here is an example of how to add CSS to an HTML document:

```
<!DOCTYPE html>  
<html>  
<head>
```

```
<title>My First CSS Document</title>
<style>
  h1 {
    color: red;
  }
</style>
</head>
<body>
  <h1>Welcome to my website!</h1>
  <p>This is my first CSS document.</p>
</body>
</html>
```

In this example, we have used the `color` property to make the text of the `<h1>` element red.

3.3 CSS Selectors:

CSS selectors are used to select and style HTML elements. There are several types of selectors, including:

- Element selectors: select elements based on their tag name. For example, `h1` selects all `<h1>` elements.

- Class selectors: select elements based on their class attribute. For example, `.red-text` selects all elements with a class of `red-text`.
- ID selectors: select elements based on their ID attribute. For example, `#header` selects the element with an ID of `header`.
- Attribute selectors: select elements based on their attributes. For example, `[type="text"]` selects all input elements with a `type` attribute of `text`.

Here are some examples of CSS selectors:

```
/* Element selector */
h1 {
  color: red;
}
/* Class selector */
.red-text {
  color: red;
}
/* ID selector */
#header {
  background-color: blue;
}
/* Attribute selector */
[type="text"] {
  border: 1px solid black;}
```

3.4 CSS Properties and Values:

CSS properties are used to style HTML elements. Properties are added to selectors to specify how the element should be styled. Here are some common CSS properties:

- color : sets the color of the text
- background-color : sets the background color of the element
- font-family : sets the font family of the text
- font-size : sets the font size of the text
- text-align : sets the alignment of the text

Here are some examples of CSS properties and values:

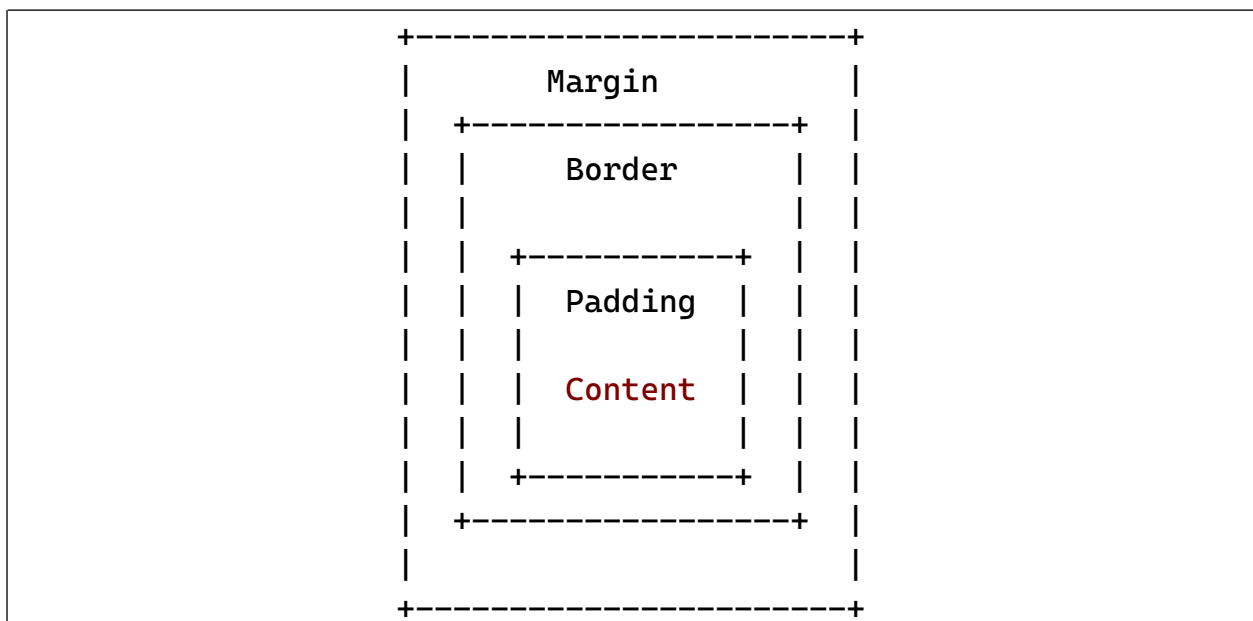
```
/* Set the color of the text to red */  
h1 {  
    color: red;  
}  
/* Set the background color of the element to blue */  
#header {  
    background-color: blue;  
}  
/* Set the font family of the text to Arial */
```



```
body {  
    font-family: Arial, sans-serif;  
}  
/* Set the font size of the text to 16 pixels */  
p {  
    font-size: 16px;  
}  
/* Set the alignment of the text to center */  
h1, h2, h3 {  
    text-align: center;  
}
```

3.5 CSS Box Model:

The CSS box model is used to describe the layout and sizing of HTML elements. The box model consists of four parts: content, padding, border, and margin. Here is an example of the CSS box model:



Here are some CSS properties related to the box model:

- width : sets the width of the content box
- height : sets the height of the content box
- padding : sets the padding between the content and the border
- border : sets the border around the element
- margin : sets the margin outside the border

Here are some examples of how to use CSS properties related to the box model:

```
/* Set the width of the element to 300 pixels */
div {
    width: 300px;
}
/* Set the padding of the element to 10 pixels */
p {
    padding: 10px;
}
/* Set the border of the element to 1 pixel solid black */
img {
    border: 1px solid black;
}
/* Set the margin of the element to 20 pixels */
h1 {
    margin: 20px;}
```

Chapter 4

JavaScript

4.1 Introduction:

JavaScript is a high-level, interpreted programming language that is used to create interactive web pages and dynamic web applications. It is often used in combination with HTML and CSS to add functionality and interactivity to web pages. In this tutorial, we will cover the basics of JavaScript, including variables, data types, functions, and more.

4.2 Getting Started:

To add JavaScript to an HTML document, we can use the `<script>` tag. The `<script>` tag should be placed in the `<head>` or `<body>` section of the HTML document or its content will be placed in external .js file. Here is an example of how to add JavaScript to an HTML document:

```
<!DOCTYPE html>  
<html>  
<head>
```

```
<title>My First JavaScript Document</title>
</head>
<body>
  <h1>Welcome to my website!</h1>
  <script>
    alert("Hello, World!");
  </script>
</body>
</html>
```

In this example, we have used the `alert()` function to display a message box with the text "Hello, World!".

4.3 Variables:

Variables are used to store data in JavaScript. To declare a variable, you can use the `var`, `let`, or `const` keyword. Here is an example of how to declare a variable:

```
var name = "John";    // string
let age = 30;          // number
const isMale = true;   // boolean
```

In this example, we have declared a variable called `name` and assigned it the value "John". We have also declared variables called `age`

and `isMale` and assigned them the values 30 and `true`, respectively.

4.4 Data Types:

JavaScript has several data types, including strings, numbers, booleans, arrays, and objects. Here are some examples of how to use data types in JavaScript:

```
let name = "John";           // string
let age = 30;                 // number
let isMale = true;           // boolean
let fruits = ["apple", "banana", "orange"]; // array
let person = {                // object
  name: "John",
  age: 30,
  isMale: true
};
```

4.5 Functions:

Functions are used to perform a specific task in JavaScript. To declare a function, we can use the `function` keyword. Here is an example of how to declare a function:

```
function addNumbers(num1, num2) {  
    return num1 + num2;  
}
```

In this example, we have declared a function called `addNumbers` that takes two parameters and returns their sum.

4.6 Events:

Events are actions that occur on a web page, such as a user clicking a button. JavaScript can be used to handle these events and perform a specific task when they occur. Here is an example of how to handle a button click event:

```
<!DOCTYPE html>  
<html>  
<head>  
    <title>Button Click Event</title>  
</head>  
<body>  
    <h1>Welcome to my website!</h1>  
    <button onclick="alert('Button  
Clicked!')">Click Me</button>  
</body>  
</html>
```

In this example, we have used the `onclick` attribute to handle the button click event and

display a message box with the text "Button Clicked!".

4.7 Conditional Statements:

Conditional statements are used to execute different actions based on different conditions. Here is an example of how to use conditional statements in JavaScript:

```
let age = 30;
if (age < 18) {
  console.log("You are a minor.");
} else if (age ≥ 18 && age < 60) {
  console.log("You are an adult.");
} else {
  console.log("You are a senior citizen.");
}
```

In this example, we have used the `if`, `else if`, and `else` statements to check the value of the `age` variable and execute different actions based on the condition.

4.8 Loops:

Loops are used to execute a block of code multiple times. Here is an example of how to use loops in JavaScript:

```
let fruits = ["apple", "banana", "orange"];  
for (let i = 0; i < fruits.length; i++) {  
    console.log(fruits[i]);  
}
```

In this example, we have used a for loop to iterate through the `fruits` array and print each element to the console.

Chapter 5

A Full-Stack Web Application

5.1 Introduction:

In this chapter we will discuss a Full-Stack web application which is a library website called (Mybrary) that provides an alternative solution for library stores that display their books on WhatsApp or Facebook groups etc...

Mybrary displays data about books and writers, and provides search for authors or advanced search for books.

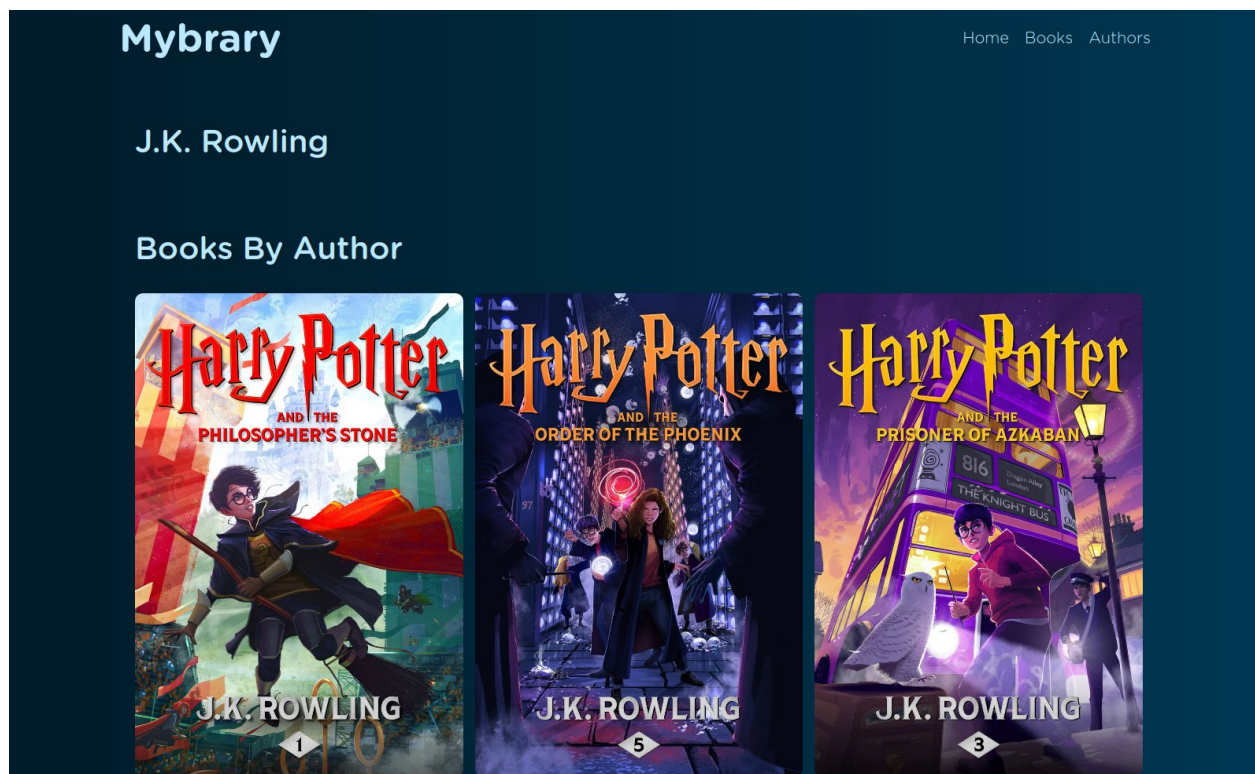
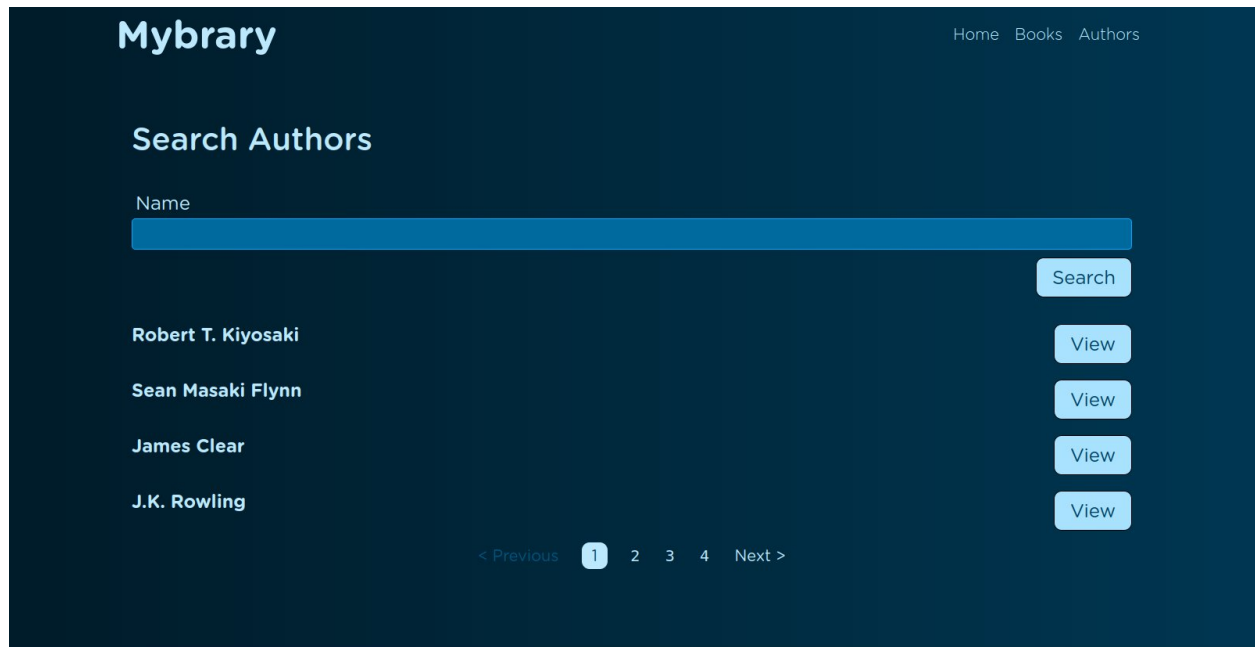
Used technologies:

1. Front-End development:
 1. CSS(Cascading Style Sheets)
 2. JavaScript
 3. EJS(Embedded JavaScript template engine)
2. Back-End development:
 1. Node.js

2. Express.js(Node.js framework)
3. MongoDB(NoSQL database management system)
3. Rendering pages approach:
 - Server-Side rendering(SSR) with EJS
4. Security:
 1. Input validations
 2. Authentication with JWT(JsonWebToken) and cookies
 3. Authorization with JWT and cookies
 4. HTTP-Only cookies(It can't be manipulated in client-side)
 5. Limited request size
 6. Rate-Limiter for requests per IP address
 7. HPP(HTTP-Parameter Pollutions) attack safe
 8. HTTPS with SSL certificate hosting

5.2 The Website Features:

5.2.1 Authors and Books Visualization:





5.2.2 Authors Search and Books Advanced Search:

Mybrary Home Books Authors

Search Books

Title
o

Author
▼

Created After
mm/dd/yyyy

Created Before
06/28/2023

Published After
mm/dd/yyyy

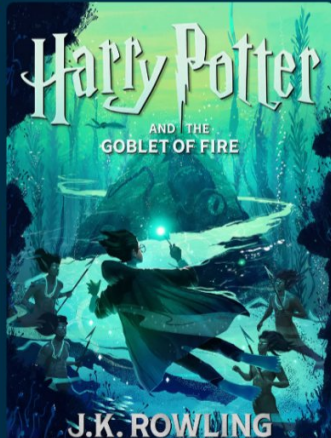

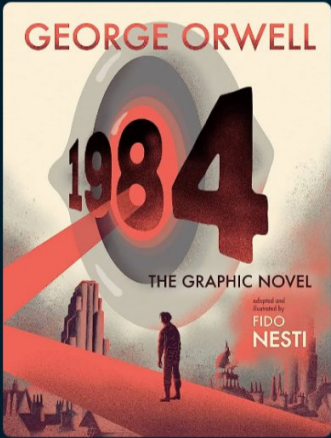
Published Before
mm/dd/yyyy

Minimum Pages Count

Maximum Pages Count
900

[Hide Advanced Search...](#)

[Reset](#) [Search](#)



Search Authors

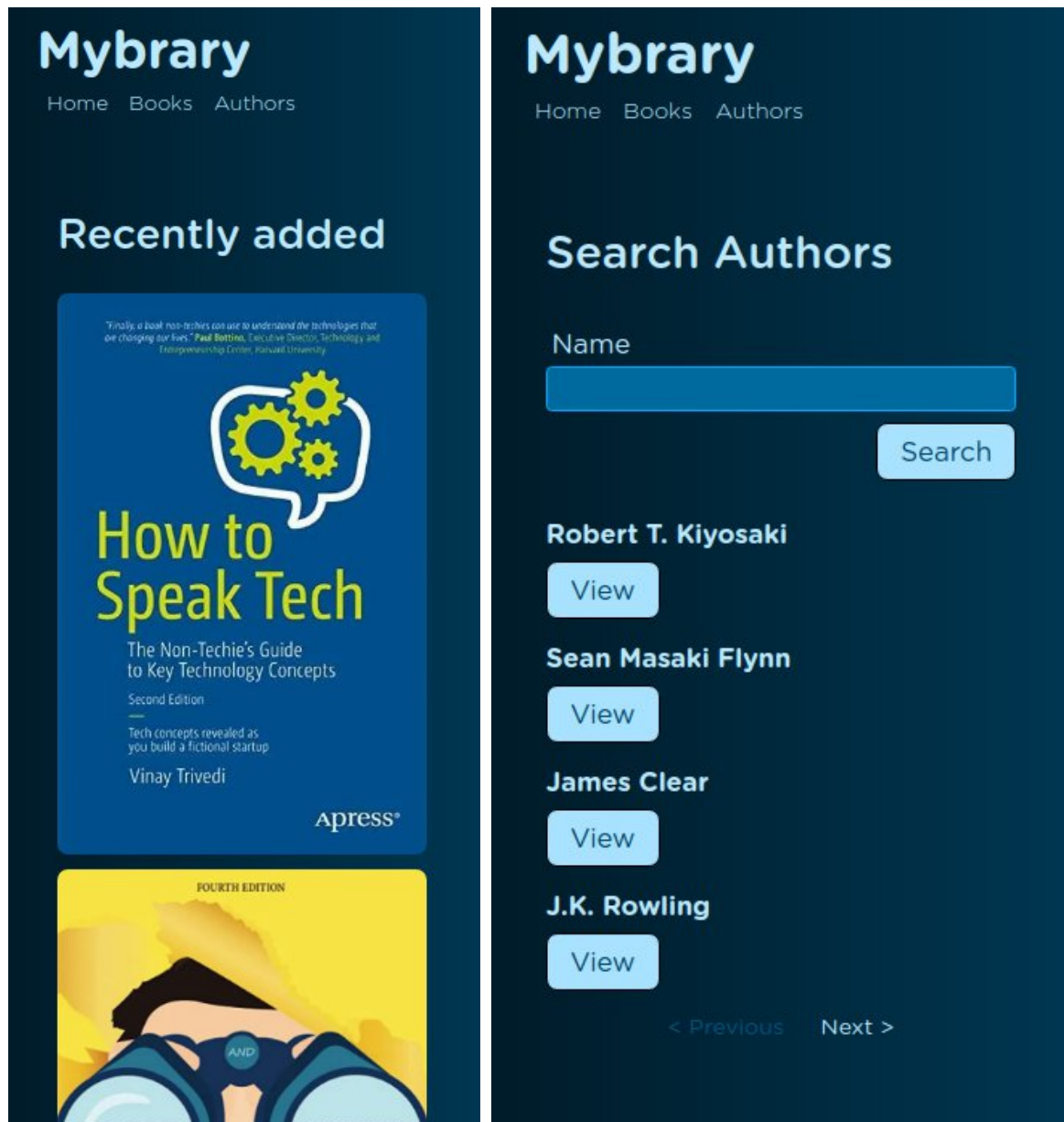
Name
ea

[Search](#)

James Clear [View](#)

Sean Masaki Flynn [View](#)

5.2.3 Responsive Page Sizes:





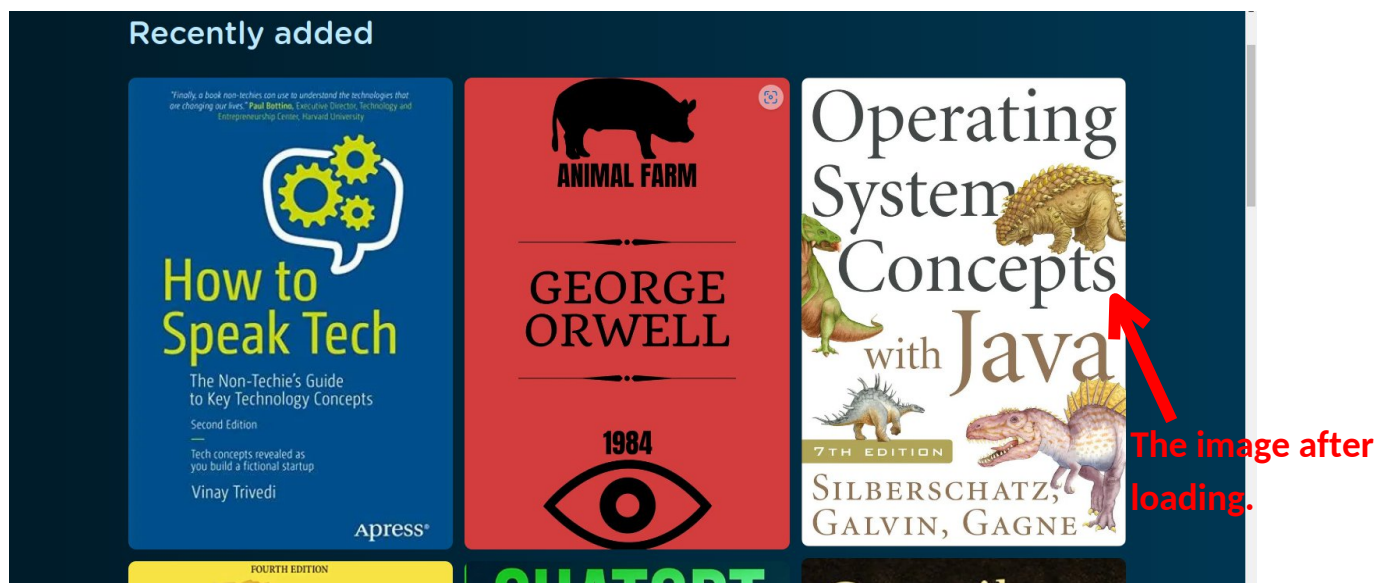
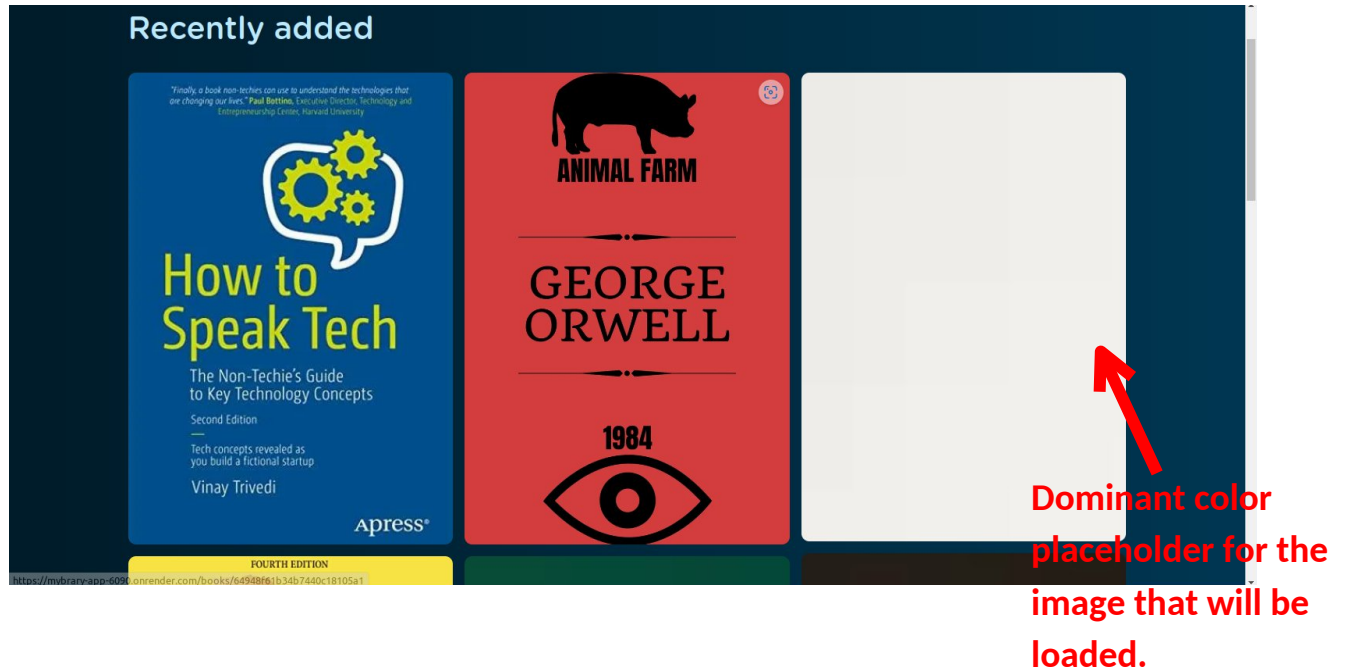
5.2.4 External Server for Storing Images:

Using a service called (Imagekit) which is server that storing images and gives the ability for optimizing images sizes depending on the user screen size which gives a huge saving in bandwidth for any user who has poor internet connection.

5.2.5 Images Lazy Loading:

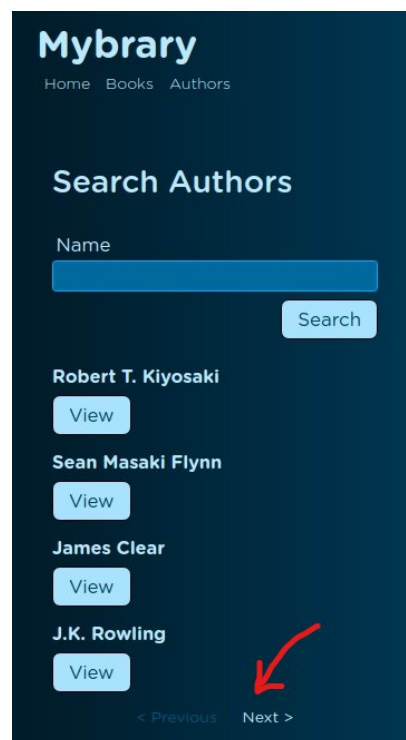
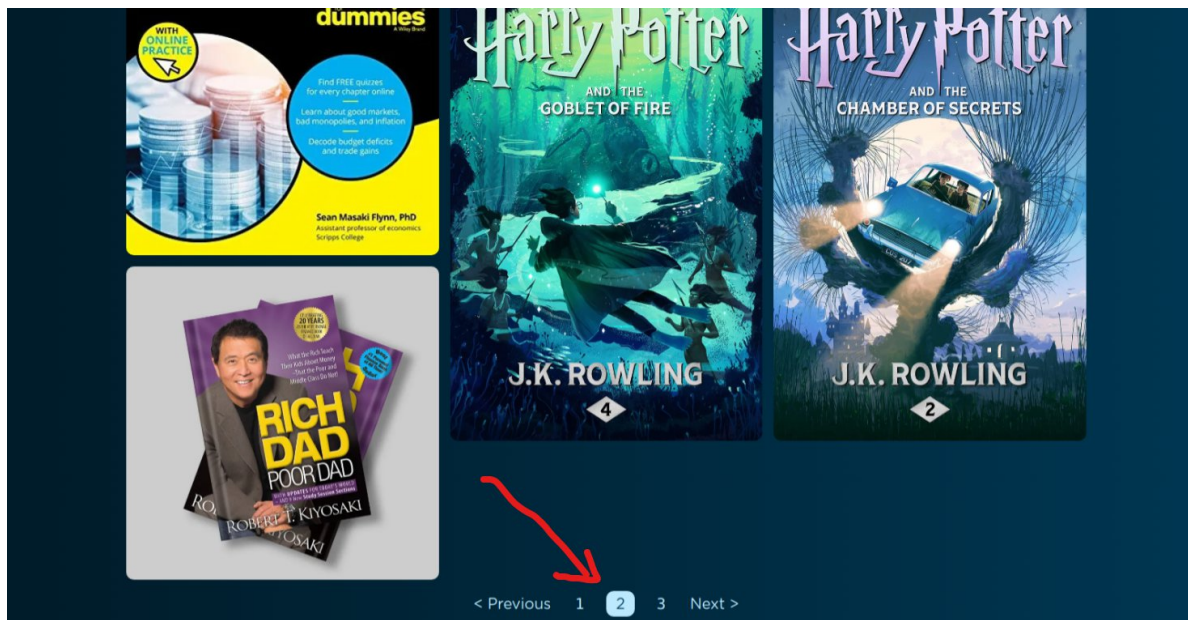
Images lazy loading is a front-end development approach for loading only the images that the user see in the page view and loading the other images only when the user scroll in the page which gives a huge bandwidth saving with performance boost.

The image that doesn't loaded has a very small image place holder with the original image dominant color (The placeholder generated on the image server).



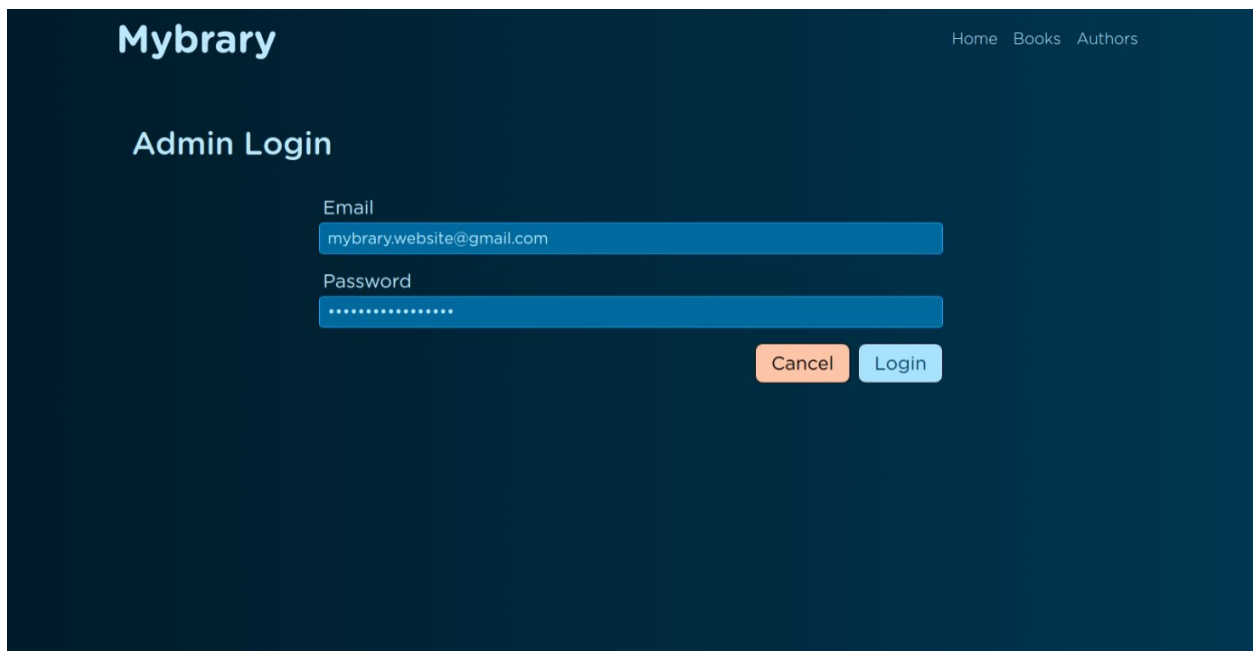
5.2.6 Pagination:

A pagination with the same implementation as github for any long page.



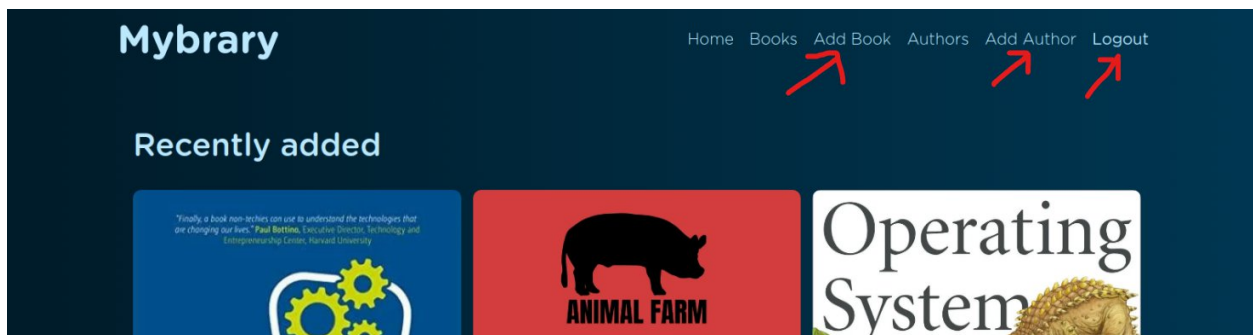
5.2.7 An Admin Management Account:

For creating , editing and deleting books and authors there is a hidden route for logging into an admin account with email and password.



The screenshot shows the 'Mybrary' website with a dark blue background. In the top right corner, there are links for 'Home', 'Books', and 'Authors'. The main heading is 'Admin Login'. Below it, there are two input fields: 'Email' with the text 'mybrary.website@gmail.com' and 'Password' with masked characters. At the bottom right of the form are two buttons: 'Cancel' (orange) and 'Login' (blue).

And after logging as an admin:



Mybrary Home Books Add Book Authors Add Author Logout

Search Authors

Name

Robert T. Kiyosaki	<input type="button" value="View"/>	<input type="button" value="Edit"/>	<input type="button" value="Delete"/>
Sean Masaki Flynn	<input type="button" value="View"/>	<input type="button" value="Edit"/>	<input type="button" value="Delete"/>
James Clear	<input type="button" value="View"/>	<input type="button" value="Edit"/>	<input type="button" value="Delete"/>
J.K. Rowling	<input type="button" value="View"/>	<input type="button" value="Edit"/>	<input type="button" value="Delete"/>

< Previous 1 2 3 4 Next >

Mybrary Home Books Add Book Authors Add Author Logout

Abraham Silberschatz

Books By Author




Mybrary Home Books Add Book Authors Add Author Logout

Rich Dad Poor Dad: What the Rich Teach Their Kids About Money That the Poor and Middle Class Do Not!



Author : Robert T. Kiyosaki

Uploaded : last month

Updated : last month

Publish Date : Mon, 7 Jan 2013

Pages Count : 336

Description :

Rich Dad Poor Dad is Robert's story of growing up with two dads — his real father and the father of his best friend, his rich dad — and the ways in which both men shaped his thoughts about money and investing. The book explodes the myth that you need to earn a high income to be rich and explains the difference between working for money and having your money work for you.

5.2.8 Strong Input Validations and Error Messages:

Every input field has validation layers on server-side and client side too with html or javascript with a detailed error messages for the user or the admin.

The validations can catch the manipulated input fields in the browser developer tools.

Small samples:

Before pressing update the are manipulated fields and fields doesn't met the constraints:

The screenshot shows the 'Mybrary' application interface for editing a book. The form includes fields for Title, Author, Publish Date, Pages Count, Cover, and Description. The 'Cover' field displays a validation error: 'Wrong Image Type X' with details 'Image Name : MS SQL.2.jpg' and 'Image Size : 0 bytes'. The 'Description' field contains a long, repetitive text block. At the bottom right, there are 'Cancel' and 'Update' buttons.

Mybrary Home Books Add Book Authors Add Author Logout

Edit Book

*Title: Rich Dad Poor Dad is Robert's story of growing up with twRic

*Author: Robert T. Kiyosaki

*Publish Date: 01/07/2013

*Pages Count: 33633

Cover: Wrong Image Type X
Image Name : MS SQL.2.jpg
Image Size : 0 bytes

*Description: Difference between working for money and having your money work for you. Rich Dad Poor Dad is Robert's story of growing up with two dads — his real father and the father of his best friend, his rich dad — and the ways in which both men shaped his thoughts about money and investing. The book explodes the myth that you need to earn a high income to be rich and explains the difference between working for money and having your money work for you. Rich Dad Poor Dad is Robert's story of growing up with two dads — his real father and the father of his best friend, his rich dad — and the ways in which both men shaped his thoughts about money and investing. The book explodes the myth that you need to earn a high income to be rich and explains the difference between working for money and having your money work for you.

Cancel Update

After pressing update we can see the error messages:

Mybrary Home Books Add Book Authors Add Author Logout

Server Error : Wrong file type ✖

Server Error : Too long book title (100 characters maximum) ✖

Server Error : Invalid author id ✖

Server Error : The author id not belongs to any author ✖

Server Error : Maximum pages count is (5000) ✖

Server Error : Too long book description (1000 characters maximum) ✖

Edit Book

*Title: Rich Dad Poor Dad is Robert's story of growing up with twRic

*Author: Robert T. Kiyosaki

*Publish Date: 01/07/2013

*Pages Count: 33633

Cover: Choose File (No file chosen)

*Description: Rich Dad Poor Dad is Robert's story of growing up with two dads — his real father and the father of his best friend, his rich dad — and the ways in which both men shaped his thoughts about money and investing. The book explodes the myth that you need to earn a high income to be rich and explains the difference between working for money and having your money work for you. Rich Dad Poor Dad is Robert's story of growing up with two dads — his real father and the father of his best friend, his rich dad — and the ways in which both men shaped his thoughts about money and investing. The book explodes the myth that you need to earn a high income to be rich and explains the difference between working for money and having your money work for

Cancel Update

Another validation messages samples:

Mybrary Home Books Add Book Authors Add Author Logout

Server Error : The author (Sean Masaki Flynn) has books and can't be deleted ✖

Sean Masaki Flynn

Edit Delete

Books By Author

LEARNING MADE EASY
3rd Edition

Mybrary Home Books Add Book Authors Add Author Logout

Server Error : Author name (Sean Masaki Flynn) has been used before ✖

New Author

Name

Sean Masaki Flynn

Cancel Create

Mybrary Home Books Authors

Server Error : Invalid email address ✖

Admin Login

Email

mybrary.website@g

Password

.....

Cancel Login

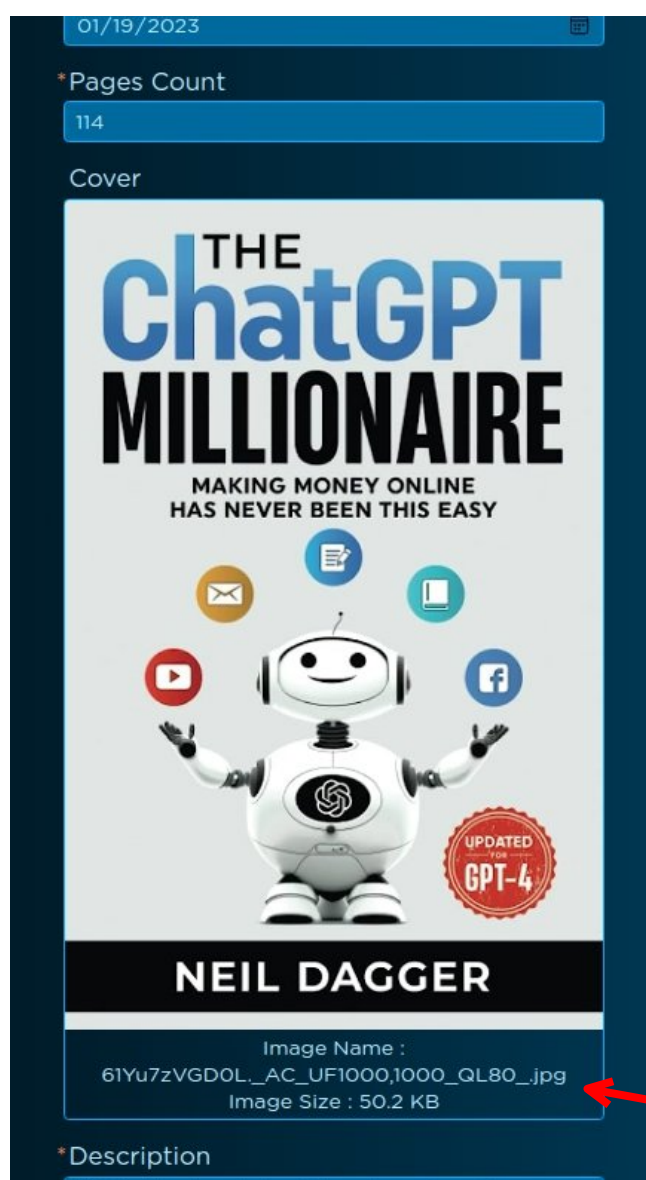
Mybrary Home Books Authors

Server Error : Wrong email or password ✖

Admin Login

5.2.9 Details About Input Images:

Details about input images with image magic number validation(which catches the real extension of the file if it was renamed from image.txt to image.jpg for example).



5.2.10 Security:

A) Authentication and authorization with JWT.

Which is a modern approach for storing the authorization token in the client-side without any worries.

B) Rate limiter by IP address for the user that makes more than 200 requests per 30 minutes.



Too many requests from this IP, please try again in
30 minutes X

C) Limiting book cover image size for 2MB.

D) Limiting request size to 20KB.

E) Preventing HTTP Parameter Pollution attacks (HPP).

5.3 The Website Links:

Source code link : <https://github.com/Nabil-Nasr/mybrary>

The website hosting link : <https://mybrary-app-6090.onrender.com>

References:

- "HTML Elements Reference" by Mozilla:
<https://developer.mozilla.org/en-US/docs/Web/HTML/Element>
- "CSS Reference" by Mozilla:
<https://developer.mozilla.org/en-US/docs/Web/CSS/Reference>
- "JavaScript Reference" by Mozilla:
<https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference>
- "EJS - Embedded JavaScript" by EJS:
<https://ejs.co/#docs>
- "Responsive Web Design Basics" by Google:
<https://developers.google.com/web/fundamentals/design-and-ux/responsive>
- "Node.js" by Node.js:
<https://nodejs.org/en/about/>
- "Getting Started with Node.js" by Mozilla:
<https://developer.mozilla.org/en->

US/docs/Learn/Server-side/Express_Nodejs/Introduction

- "MongoDB Manual" by MongoDB:
<https://docs.mongodb.com/manual/>
- "Getting Started with MongoDB" by Mozilla:
https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs/mongoose
- "Express.js" by Express.js:
<https://expressjs.com/>
- "Getting Started with Express.js" by Mozilla:
https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs/Introduction
- "Server-Side Rendering" by React:
<https://reactjs.org/docs/react-dom-server.html>
- "Client-Side Rendering" by React:
<https://reactjs.org/docs/react-dom.html>

- "Progressive Web Apps" by Google:
<https://web.dev/progressive-web-apps/>
- "Hybrid Rendering" by Next.js:
<https://nextjs.org/docs/basic-features/data-fetching#getserversideprops-server-side-rendering>
- "OWASP Top Ten" by OWASP:
<https://owasp.org/Top10/>
- "Authentication and Authorization" by Mozilla: https://developer.mozilla.org/en-US/docs/Learn/Server-side/First_steps/Website_security#Authentication_and_authorization