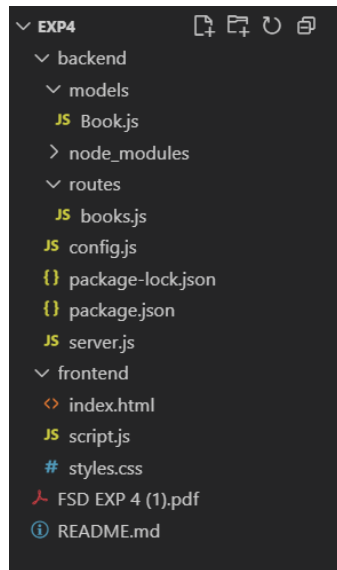


Experiment No. 4: REST API Design with MongoDB + Mongoose Integration

Directory Structure:



1. Code:


a. Index.html:

```
<? index.html >
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4  <meta charset="UTF-8">
5  <meta name="viewport" content="width=device-width, initial-scale=1.0">
6  <title>Book Library</title>
7  <link rel="stylesheet" href="styles.css">
8  </head>
9  <body>
10 <div class="container">
11 <div class="header">
12 <h1>Book Library</h1>
13 <p>Manage your book collection</p>
14 </div>
15 <div class="main">
16 <div class="add-book-form">
17 <h2>Add New Book</h2>
18 <form id="book-form">
19 <div class="form-group">
20 <label for="title">Title </label>
21 <input type="text" id="title" name="title" required>
22 </div>
23 <div class="form-group">
24 <label for="author">Author </label>
25 <input type="text" id="author" name="author" required>
26 </div>
27 <div class="form-group">
28 <label for="publishedYear">Published Year </label>
29 <input type="number" id="publishedYear" name="publishedYear" min="1800" max="2024" required>
30 </div>
31 <div class="form-group">
32 <label for="genre">Genre </label>
33 <input type="text" id="genre" name="genre" required>
34 </div>
35 <div class="form-actions">
36 <button type="submit" id="submit-btn">Add Book</button>
37 <button type="button" id="cancel-btn" style="display: none;">Cancel</button>
38 </div>
39 </form>
40 </div>
41 <div class="books-list">
42 <h2>Your Books</h2>
43 <div class="search-container">
44 <input type="text" id="search-input" placeholder="Search books...">
45 <button id="search-btn">Search</button>
46 </div>
47 <div id="loading" class="loading">Loading books...</div>
48 <div id="error-message" class="error-message" style="display: none;"></div>
49 <div id="books-list" class="books-list"></div>
50 </div>
51 </div>
52 <script src="script.js"></script>
53 </body>
54 </html>
```

b. Frontend script.js:

```
JS script.js X
frontend > JS script.js > BookLibrary > constructor
1 class BookLibrary {
2   constructor() {
3     this.books = [];
4     this.editingBookId = null;
5     this.apiUrl = 'http://localhost:3000/api/books';
6
7     this.initializeElements();
8     this.bindEvents();
9     this.loadBooks();
10  }
11
12  initializeElements() {
13    this.bookForm = document.getElementById('book-form');
14    this.formTitle = document.getElementById('form-title');
15    this.submitBtn = document.getElementById('submit-btn');
16    this.cancelBtn = document.getElementById('cancel-btn');
17    this.booksList = document.getElementById('books-list');
18    this.loading = document.getElementById('loading');
19    this.errorMessage = document.getElementById('error-message');
20    this.searchInput = document.getElementById('search-input');
21    this.searchBtn = document.getElementById('search-btn');
22  }
23
24  bindEvents() {
25    this.bookForm.addEventListener('submit', (e) => this.handleFormSubmit(e));
26    this.cancelBtn.addEventListener('click', () => this.cancelEdit());
27    this.searchBtn.addEventListener('click', () => this.searchBooks());
28    this.searchInput.addEventListener('input', () => this.searchBooks());
29  }
30
31  async loadBooks() {
32    try {
33      this.showLoading(true);
34      this.hideError();
35
36      const response = await fetch(this.apiUrl);
37      const result = await response.json();
38
39      if (result.success) {
40        this.books = result.data;
41        this.displayBooks(this.books);
42      } else {
43        this.showError('Failed to load books: ' + result.message);
44      }
45    } catch (error) {
46      this.showError('Error connecting to server. Make sure the backend is running. ');
47      console.error('Error loading books:', error);
48    } finally {
49      this.showLoading(false);
50    }
51  }
52
53  async handleFormSubmit(e) {
54    e.preventDefault();
55
56    const formData = new FormData(this.bookForm);
57    const bookData = {
58      title: formData.get('title').trim(),
59      author: formData.get('author').trim(),
60      publishedYear: parseInt(formData.get('publishedYear')),
61      genre: formData.get('genre').trim()
62    };
63  }
```

c. Frontend style.css:

```
# styles.css X
frontend > # styles.css > 
1  * {
2    margin: 0;
3    padding: 0;
4    box-sizing: border-box;
5  }
6
7  body {
8    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
9    background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
10   min-height: 100vh;
11   color: #333;
12 }
13
14 .container {
15   max-width: 1200px;
16   margin: 0 auto;
17   padding: 20px;
18 }
19
20 header {
21   text-align: center;
22   margin-bottom: 40px;
23   color: white;
24 }
25
26 header h1 {
27   font-size: 3rem;
28   margin-bottom: 10px;
29   text-shadow: 2px 2px 4px rgba(0,0,0,0.3);
30 }
31
32 header p {
33   font-size: 1.2rem;
34   opacity: 0.9;
35 }
36
37 main {
38   display: grid;
39   grid-template-columns: 1fr 2fr;
40   gap: 30px;
41   align-items: start;
42 }
43
44 .form-section {
45   background: white;
46   padding: 30px;
47   border-radius: 15px;
48   box-shadow: 0 10px 30px rgba(0,0,0,0.2);
49 }
50
51 .form-section h2 {
52   margin-bottom: 25px;
53   color: #333;
54   border-bottom: 3px solid #667eea;
55   padding-bottom: 10px;
```

d. Backend Configuration:

```
JS config.js X
backend > JS config.js > [0] <unknown>
1  module.exports = {
2    PORT: process.env.PORT || 3000,
3    MONGODB_URI: process.env.MONGODB_URI || 'mongodb+srv://bpkuser:book123@cluster0.ggdegby.mongodb.net/bpklibrary?retryWrites=true&w=majority&appName=Cluster0'
4  };
5
```

e. BookModel Mongoose Schema:


```
JS Book.js X
backend > models > JS Book.js > ...
1  const mongoose = require('mongoose');
2
3  const bookSchema = new mongoose.Schema({
4    title: {
5      type: String,
6      required: [true, 'Title is required'],
7      trim: true
8    },
9    author: {
10     type: String,
11     required: [true, 'Author is required'],
12     trim: true
13   },
14   publishedYear: {
15     type: Number,
16     required: [true, 'Published year is required'],
17     min: [1000, 'Published year must be valid'],
18     max: [new Date().getFullYear(), 'Published year cannot be in the future']
19   },
20   genre: {
21     type: String,
22     required: [true, 'Genre is required'],
23     trim: true
24   }
25 }, {
26   timestamps: true
27 });
28
29 module.exports = mongoose.model('Book', bookSchema);
30
```

f. Routes book.js:

```
JS books.js X
backend > routes > JS books.js > router.post("/") callback
1  const express = require('express');
2  const Book = require('../models/Book');
3  const router = express.Router();
4
5  // GET /books - Fetch all books
6  router.get('/', async (req, res) => {
7    try {
8      const books = await Book.find().sort({ createdAt: -1 });
9      res.json({
10        success: true,
11        data: books
12      });
13    } catch (error) {
14      res.status(500).json({
15        success: false,
16        message: 'Error fetching books',
17        error: error.message
18      });
19    }
20  });
21
22  // GET /books/:id - Fetch a single book by ID
23  router.get('/:id', async (req, res) => {
24    try {
25      const book = await Book.findById(req.params.id);
26
27      if (!book) {
28        return res.status(404).json({
29          success: false,
30          message: 'Book not found'
31        });
32      }
33
34      res.json({
35        success: true,
36        data: book
37      });
38    } catch (error) {
39      res.status(500).json({
40        success: false,
41        message: 'Error fetching book',
42        error: error.message
43      });
44    }
45  });

```

2. Output:-

 **Book Library**
Manage your book collection

Add New Book

Title *


Author *



Published Year *


Genre *


Add Book

Your Books



Test Book
by Test Author
Year: 2023 Genre: Fiction
 Edit  Delete



 **Book Library**
Manage your book collection

Edit Book

Title *


Author *



Published Year *


Genre *

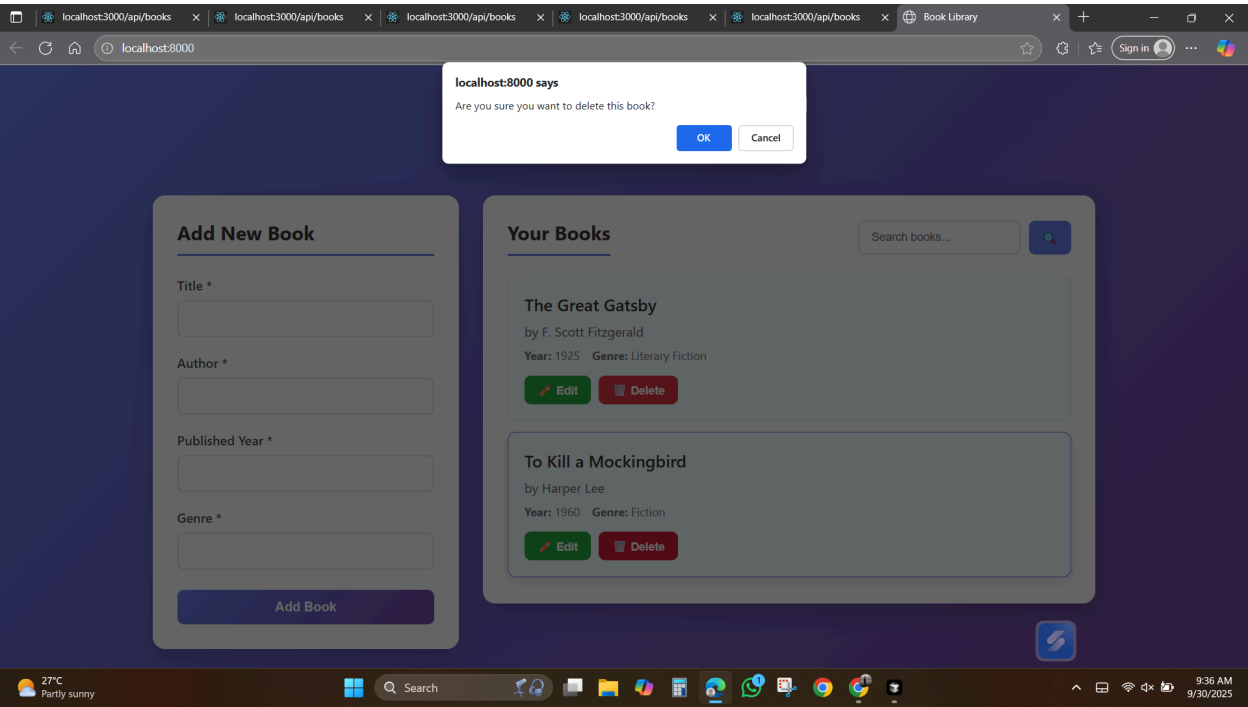
Update Book Cancel

Your Books



To Kill a Mockingbird
by Harper Lee
Year: 1960 Genre: Fiction
 Edit  Delete





Database Access

Database Users

Custom Roles

+ ADD NEW DATABASE USER

User	Description	Authentication Method	MongoDB Roles	Resources	Actions
<div><div></div><div>bookuser</div></div>		SCRAM	readWriteAnyDatabase@admin	All Resources	<div><div>EDIT</div><div>DELETE</div></div>

Atlas

Atharva's Or...

Access Manager

Billing

All ClustersGet HelpAtharva

Project 0

Data ServicesCharts

Overview

DATABASE

Clusters

SERVICES

SECURITY

+ Create Database

Search Namespaces

booklibrary

books

sample_mflix

booklibrary.books

STORAGE SIZE: 36KB LOGICAL DATA SIZE: 495B TOTAL DOCUMENTS: 3 INDEXES TOTAL SIZE: 36KB

FindIndexesSchema Anti-PatternsAggregationSearch Indexes

Generate queries from natural language in Compass

INSERT DOCUMENT

FilterType a query: { field: 'value' }ResetApplyOptions

QUERY RESULTS: 1-3 OF 3

```
_id: ObjectId('68db574ec74dd17f76228833')
title: "The Great Gatsby"
author: "F. Scott Fitzgerald"
publishedYear: 1925
genre: "Literary Fiction"
createdAt: 2025-09-30T04:06:38.691+00:00
updatedAt: 2025-09-30T04:06:38.691+00:00
...v: 0
```