Developing BookStore App - Backend

BSApp-Backend - Working Steps:

A)Setup MongoDB

1) Create the Database called **mybookstoredb**
2) Create the Collection called **mybooks** under **mybookstoredb**
3) Insert 5 Documents into **mybooks** collection as follows

```
[{
"bookId": 101,
"bookName": "Learn Angular",
"author": "Srinivas Dande",
"category": "Web",
"publications": "JLC",
"price":15000
}, {
"bookId": 102,
"bookName": "Learn React",
"author": "Srinivas Dande",
"category": "Web",
"publications": "JLC",
"price":15000
}, {
"bookId": 103,
"bookName": "Learn Spring",
"author": "Srinivas Dande",
"category": "Java",
"publications": "JLC",
"price":10000
},
{  "bookId": 104,
"bookName": "Learn Spring Boot",
"author": "Srinivas Dande",
"category": "Java",
"publications": "JLC",
"price":8000
}, {
"bookId": 105,
"bookName": "Learn Java8",
"author": "Srinivas Dande",
"category": "Java",
"publications": "JLC",
"price":8000
}
]
```

B)Setup the backend Server with Node and Express

4) Create the folder for your application

bookstore-app-backend

5) Open that folder

bookstore-app-backend>

6) Initialize the node project

bookstore-app-backend> npm init

Walk through the questions asked and answer them

finally package.json will be created as follows.

```
{
 "name": "bookstore-app-backend",
 "version": "1.1.1",
 "description": "This is  JLCBookStore App Backend ",
 "main": "myserver.js",
 "scripts": {
  "test": "echo \"Error: no test specified\" && exit 1"
 },
 "keywords": [
  "NodeJS",
  "ExpressJS",
  "MongoDB"
 ],
 "author": "Srinivas Dande",
 "license": "ISC"
}
```

7) Install the Dependencies required for your backend project.

```
npm install express
npm install body-parser
npm install mongoose
npm install nodemon
npm install dotenv
npm install cors
```

npm install express body-parser mongoose nodemon dotenv cors

8) Create .env.jlc file under the project folder

9) Place the following in .env.jlc file

```
BASE_URL=http://localhost:5500
MONGODB_URI=mongodb://localhost:27017/mybookstoredb
```

10) Create the file called myserver.js under the project folder

11) Place the following Code in myserver.js file

```
const express = require("express");
const dotenv = require("dotenv");

dotenv.config({ path: ".env.jlc" });

const app = express();
const PORT = process.env.PORT || 5500;

app.get("/hello", (req, res) => {
  console.log("Request for - /hello");
   return res.send("Hello Guys -- I am Ready");
});

/**
 * start Express server on port 5500
 */
app.listen(PORT, () => {
  console.log("Express server is running at http://localhost:%d", PORT);
  console.log("Press CTRL-C to stop\n");
});
```

12) Update the package.json

Before Updating:

```
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
```

After Updating:

```
 "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
    "server": "nodemon --watch src myserver.js"
  },
```

13) Start the Server

npm run server

14) Open the URL

http://localhost:5500/hello
You can see Response ----- Hello Guys -- I am Ready

C) Develop the REST API

15) Create the src folder under the project folder

16) Create the mymodels folder under src folder

17) Create the mycontrollers folder under src folder

18) Write the Book Model under src/mymodels

```
const mongoose = require("mongoose");
const Schema = mongoose.Schema;

let BookSchema = new Schema(
 {
 bookId: Number,
 bookName: String,
 author: String,
 category: String,
 publications: String,
 price: String,
 },
 {
 timestamps: true,
 collection: 'mybooks'
 }
);

const Book = mongoose.model('Book', BookSchema);

module.exports = Book;
```

19) Write BookController under src/mycontrollers folder

```
let Book = require("../models/Book");

// Get All Books
exports.getAllBooks = (req, res, next) => {
   console.log("/mybooks ---get()");
   Book.find((error, data) => {
      if (error) {
         return next(error);
      } else {
         res.json(data);
      }
   }).sort({
      bookId: 1
   });
};


// Get Book By Book Id
exports.findBookByBookId = (req, res, next) => {
   console.log("/mybook --- get()");
   Book.findOne({
      bookId: parseInt(req.params.bookId)
   }, (error, data) => {
      if (error) {
         return next(error);
      } else {
         console.log(data)
         res.json(data);
      }
   });
};
```

20) Update myserver.js

```
const express = require('express');
const dotenv = require('dotenv');

const bodyParser = require('body-parser');
const mongoose = require('mongoose');
const path = require('path');
const cors = require("cors");

const BookController = require('./src/controllers/BookController');
```

```
dotenv.config({    path: '.env.jlc'  });

const app = express();
const PORT = process.env.PORT || 5500;

/**
 * Connect to MongoDB.
 */
mongoose.connect(process.env.MONGODB_URI, {
  useUnifiedTopology: true,
  useFindAndModify: false,
  useNewUrlParser: true,
}).then(() => {
  console.log('MongoDB connected successfully.');
}, error => {
  console.error(error);
  console.log('MongoDB connection error. Please make sure MongoDB is
running.');
  process.exit();
});

/**
 * Add middlewares
 */
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({
  extended: false
}));
app.use(cors());

app.get('/hello', (req, res) => {
  return res.send('Hello Guys')
})

app.get('/myapi/mybooks', BookController.getAllBooks);
app.get('/myapi/mybooks/:bookId', BookController.findBookByBookId);

// Start Server on port 5500
app.listen(PORT, () => {
  console.log('Server is running at http://localhost:%d', PORT);
  console.log('Press CTRL-C to stop\n');
});
```

21) Start the Server

npm run server

22) Test API

http://localhost:5500/myapi/mybooks
http://localhost:5500/myapi/mybook/101
http://localhost:5500/myapi/mybook/102

23) Update BookController by adding the following methods

```
// Get Book with maximum Book Id
exports.findMaxBookId = (req, res, next) => {
   console.log("/myapi/maxId --- get()");
   Book.find({}, (error, data) => {
      if (error) {
         return next(error);
      } else {
         res.json(data);
      }
   })
      .sort({
         bookId: -1
      })
      .limit(1);
};

// Add Book
exports.addBook = (req, res) => {
   console.log("/addBook- post-");
   console.log(req.body);
   console.log("BookId : " + req.body.bookId);
   console.log("bookName : " + req.body.bookName);
   console.log("author : " + req.body.author);
   console.log("price : " + req.body.price);
   console.log("category : " + req.body.category);
   console.log("pub : " + req.body.publications);

   Book.create(req.body, (error, data) => {
      if (error) {
         return next(error);
      } else {
         res.json(data);
         console.log("Book added successfully");
      }
   });
};
```

```
// Update Book
exports.updateBook = (req, res, next) => {
   console.log("/updateBook - put()");
   console.log(req.body.bookId);
   console.log(req.body);
   Book.findOneAndUpdate({
      bookId: req.body.bookId
   },
      req.body,
      (error, data) => {
         if (error) {
            return next(error);
         } else {
            res.json(data);
            console.log("Book updated successfully");
         }
      }
   );
};

// Delete Book
exports.deleteBook = (req, res, next) => {
   console.log("/deleteBook - delete()");
   console.log(req.params.bookId);

   Book.findOneAndRemove({
      bookId: parseInt(req.params.bookId)
   },
      (error, data) => {
         if (error) {
            return next(error);
         } else {
            res.json(data);
            console.log("Book Deleted successfully");
         }
      }
   );
};
```

24) Update myserver.js

```
app.get('/myapi/maxId', BookController.findMaxBookId);
app.post('/myapi/mybooks', BookController.addBook);
app.put('/myapi/mybooks', BookController.updateBook);
app.delete('/myapi/mybooks/:bookId', BookController.deleteBook);
```

25) Start the Server

npm run server

26) Test these methods with Postman/Curl

27) Add the following method in BookController

```
// Get Books By Category
exports.findBooksByCategory = (req, res, next) => {
   console.log("/mybooks/ --- get()");
   Book.find(    {  category: req.params.category   },
   (error, data) => {
      if (error) {
         return next(error);
      } else {
         console.log(data)
         res.json(data);
      }
   }
).sort({ bookId: 1 }
);
};

// Get Books By Category and Price
exports.findBooksByCatAndPrice = (req, res, next) => {
   console.log("/mybooks/category/:/price/: --- get()");
   Book.find(
   {
      category: req.params.category,
      price:parseInt(req.params.price)
   },
   (error, data) => {
      if (error) {
         return next(error);
      } else {
         console.log(data)
         res.json(data);
      }
   }
).sort({ bookId: 1}
);
};
```

28) Update myserver.js

```
app.get('/myapi/mybooks/category/:category',
BookController.findBooksByCategory);

app.get('/myapi/mybooks/category/:category/price/:price',
BookController.findBooksByCatAndPrice);
```

29) Start the Server

```
npm run server
```

30) Test these methods with Browser

```
http://localhost:5500/myapi/mybooks/category/Java
http://localhost:5500/myapi/mybooks/category/Java/price/8000
```

BookStore App - Backend

Files Required

1. Book.js	2. BookController.js
3. .env.jlc	4. myserver.js
5. package.json	

1. Book.js

```
const mongoose = require("mongoose");
const Schema = mongoose.Schema;

let BookSchema = new Schema(
 {
 bookId: Number,
 bookName: String,
 author: String,
 category: String,
 publications: String,
 price: String,
 },
 {
 timestamps: true,
 collection: 'mybooks'
 }
);

const Book = mongoose.model('Book', BookSchema);
module.exports = Book;
```

2. BookController.js

```
let Book = require("../mymodels/Book");

// Get All Books
exports.getAllBooks = (req, res, next) => {
   console.log("/mybooks ---get()");
   Book.find((error, data) => {
      if (error) {
         return next(error);
      } else {
         res.json(data);
      }
   }).sort({
      bookId: 1
   });
};

// Get Book By Book Id
exports.findBookByBookId = (req, res, next) => {
   console.log("/mybook --- get()");
   Book.findOne({
      bookId: parseInt(req.params.bookId)
   }, (error, data) => {
      if (error) {
         return next(error);
      } else {
         console.log(data)
         res.json(data);
      }
   });
};

// Get Book with maximum Book Id
exports.findMaxBookId = (req, res, next) => {
   console.log("/myapi/maxId --- get()");
   Book.find({}, (error, data) => {
      if (error) {
         return next(error);
      } else {
         res.json(data);
      }
   }) .sort({  bookId: -1 }).limit(1);
};
```

```
// Add Book
exports.addBook = (req, res) => {
  console.log("/addBook- post-");
  console.log(req.body);
  console.log("BookId : " + req.body.bookId);
  console.log("bookName : " + req.body.bookName);
  console.log("author : " + req.body.author);
  console.log("price : " + req.body.price);
  console.log("category : " + req.body.category);
  console.log("pub : " + req.body.publications);

  Book.create(req.body, (error, data) => {
    if (error) {
      return next(error);
    } else {
      res.json(data);
      console.log("Book added successfully");
    }
  });
};

// Update Book
exports.updateBook = (req, res, next) => {
  console.log("/updateBook - put()");
  console.log(req.body.bookId);
  console.log(req.body);
  Book.findOneAndUpdate({
    bookId: req.body.bookId
  },
    req.body,
    (error, data) => {
      if (error) {
        return next(error);
      } else {
        res.json(data);
        console.log("Book updated successfully");
      }
    }
  );
};
```

```
// Delete Book
exports.deleteBook = (req, res, next) => {
   console.log("/deleteBook - delete()");
   console.log(req.params.bookId);

   Book.findOneAndRemove({
      bookId: parseInt(req.params.bookId)
   },
      (error, data) => {
         if (error) {
            return next(error);
         } else {
            res.json(data);
            console.log("Book Deleted successfully");
         }
      }
   );
};

// Get Books By Category
exports.findBooksByCategory = (req, res, next) => {
   console.log("/mybooks/category/:/ --- get()");
    Book.find(
   {
      category: req.params.category
   },
   (error, data) => {
      if (error) {
         return next(error);
      } else {
         console.log(data)
         res.json(data);
      }
   }
).sort({
      bookId: 1
   });

};
```

```
// Get Books By Category and Price
exports.findBooksByCatAndPrice = (req, res, next) => {
   console.log("/mybooks/category/:/price/: --- get()");
   Book.find(
   {
      category: req.params.category,
      price:parseInt(req.params.price)
   },
   (error, data) => {
      if (error) {
         return next(error);
      } else {
         console.log(data)
         res.json(data);
      }
   }
).sort({
      bookId: 1
   });

};
```

3. .env.jlc

```
BASE_URL=http://localhost:5500
MONGODB_URI=mongodb://localhost:27017/mybookstoredb
```

4. myserver.js

```
const express = require("express");
const dotenv = require("dotenv");
const bodyParser = require('body-parser');
const mongoose = require('mongoose');
const path = require('path');
const cors = require("cors");

const BookController = require('./src/mycontrollers/BookController');

dotenv.config({ path: ".env.jlc" });

const app = express();
const PORT = process.env.PORT || 5500;

/**
 * Connect to MongoDB.
 */
 mongoose.connect(process.env.MONGODB_URI, {
   useUnifiedTopology: true,
```

```
  useFindAndModify: false,
  useNewUrlParser: true,
 }).then(() => {
  console.log('MongoDB connected successfully.');
 }, error => {
  console.error(error);
  console.log('MongoDB connection error. Please make sure MongoDB is running.');
  process.exit();
 });

 /**
  * Add middlewares
  */
 app.use(bodyParser.json());
 app.use(bodyParser.urlencoded({
  extended: false
 }));

 app.use(cors());

app.get("/hello", (req, res) => {
 console.log("Request for - /hello");
  return res.send("Hello Guys -- I am Ready");
});

app.get('/myapi/mybooks', BookController.getAllBooks);
app.get('/myapi/mybooks/:bookId', BookController.findBookByBookId);
app.get('/myapi/maxId', BookController.findMaxBookId);
app.post('/myapi/mybooks', BookController.addBook);
app.put('/myapi/mybooks', BookController.updateBook);
app.delete('/myapi/mybooks/:bookId', BookController.deleteBook);
app.get('/myapi/mybooks/category/:category', BookController.findBooksByCategory);
app.get('/myapi/mybooks/category/:category/price/:price',
BookController.findBooksByCatAndPrice);

/**
 * start Express server on port 5500
 */
app.listen(PORT, () => {
 console.log("Express server is running at http://localhost:%d", PORT);
 console.log("Press CTRL-C to stop\n");
});
```

5. package.json

```
{
 "name": "bookstore-app-backend",
 "version": "1.1.0",
 "description": "This is Bookstore backend",
 "main": "myserver.js",
 "scripts": {
  "test": "echo \"Error: no test specified\" && exit 1",
  "server": "nodemon --watch src myserver.js"
 },
 "keywords": [
  "MongoDB",
  "NodejS",
  "ExpressJS"
 ],
 "author": "Srinivas Dande",
 "license": "ISC",
 "dependencies": {
  "body-parser": "^1.19.0",
  "cors": "^2.8.5",
  "dotenv": "^10.0.0",
  "express": "^4.17.1",
  "mongoose": "^5.13.3",
  "nodemon": "^2.0.12"
 }
}
```