COMPLETE WORKING OF BOOK STORE PROJECT

FILE: frontend/app.jsx
ABOUT : setting up routes to different location

WORKING

```
<BrowserRouter>
<Routes>
<Route path='/books/details/:id' element={<ShowBook />} />
```

NOTE: There are 5 components in frontend to take care off

- 4, CRUD operations
- 1, displaying all the data in home page

FILE: frontend/components/**createPage**.jsx
ABOUT : creates a new book by taking user input

WORKING
It solves 3 purpose

- Taking user input
- Save Button
- Sending it to server (axios)

TASK 1 (taking user input)

only this func can be used to change title ⬆️ sets default value of title ⬆️

```
const [title, setTitle] = useState('');
```

What are react hooks?

- isolate the reusable part from a functional component
- can manage state or side-effects

```
<input type="text" onChange={(e) => setTitle(e.target.value)} />
```

⬇️

event object: stores property of the text field

TASK 2 (sending data to backend)
axios: sends data to server asynchronously

```
const data = {title,author,publishYear,};
axios.post('http://localhost:5555/books', data)
```

C:post **R:**get **U:**put **D:**delete

FILE: backend/model/bookModel.js
ABOUT : to create a database schema in MongoDB

WORKING
mongoose helps us in doing that

```
const bookSchema = mongoose.Schema(
  {
    title: {
      type: String,
      required: true,
    },
    publishYear: {
      type: Number,
      required: true,
    },
  },
  {
    timestamps: true,
  }
);
```

NOTE : **collection, document in mongodb == table, record in mysql**

creating a collection named Books using the schema and will be using it in code with name book

```
export const Book = mongoose.model('Books', bookSchema);
```

FILE: backend/index.js
ABOUT : main page for backend
WORKING
It solves 3 purpose

- importing and using all important dependencies
- setting up surface level routes
- connecting to db

TASK 1(understanding the json input)

```
app.use(express.json());
```

TASK 2(setting up basic routes)

```
app.get('/', (request, response) => {
  return response.send('Welcome To MERN Stack Tutorial');
});

app.use('/books', booksRoute);
```

TASK 3 (connecting to mongoDB)

```
mongoose
  .connect(mongoDBURL)
  .then(() => {
    console.log('App connected to database');
    app.listen(PORT, () => {
      console.log(`App is listening to port: ${PORT}`);
    });
  })
  .catch((error) => {
    console.log(error);
  });
```

FILE: backend/routes/createPageRoute.js (assume we have created different components)

ABOUT : interacts with backend with the frontend data

WORKING

It solves 3 purpose

- checking for null, empty, 0
- retrieving data from user
- creating a new entry in DB

```
router.post('/', async (request, response) => {
  try {
    if (
      !request.body.title ||
      !request.body.author ||
      !request.body.publishYear
    )
    {
      return response.status(400).send({
        message: 'Send all required fields: title, author, publishYear',
      });
    }
```

```
    const newBook = {
      title: request.body.title,
      author: request.body.author,
      publishYear: request.body.publishYear,
    };
```

```
    const book = await Book.create(newBook);
    return response.status(201).send(book);
  }
  //will catch error from  Book.create(newBook);
  catch (error) {
    console.log(error.message);
    response.status(500).send({ message: error.message });
  }
});
```

At this point the book is created, now it's time for the frontend to show the books and its operations

---

FILE: frontend/components/Home.js
ABOUT : displays list of all the books along with its operations
WORKING
It solves 2 purpose

- retrieves data from the backend
- show the data along with its operations(create, delete, etc)

TASK 1 (retrive the data from backend)

```
const [books, setBooks] = useState([]);
useEffect(() => {
  axios
    .get('http://localhost:5555/books')
    .then((response) => {
      setBooks(response.data.data);
    })
    .catch((error) => {
      console.log(error);
    });
}, []);
```

What is **useEffect** hook?

The `useEffect` Hook allows you to perform side effects in your components.

1. No dependency passed:

```
useEffect(() => {
  //Runs on every render
});
```

Example

2. An empty array:

```
useEffect(() => {
  //Runs only on the first render
}, []);
```

Example

3. Props or state values:

```
useEffect(() => {
  //Runs on the first render
  //And any time any dependency value changes
}, [prop, state]);
```

FILE: backend/routes/home.js

```
router.get('/', async (request, response) => {
  try {
    const books = await Book.find({});
    return response.status(200).json({
      count: books.length,
      data: books,
    });
  } catch (error) {
    console.log(error.message);
    response.status(500).send({ message: error.message });
  }
});
```

What is **Book.find({})**?
mongoose command for find the book with a particular parameter, if empty, it returns all the data found.

Sample data stored in DB

```
_id: ObjectId('651f801765dba921256eddc6')
title: "asas"
author: "swqdqwas"
publishYear: 2002
createdAt: 2023-10-06T03:33:43.034+00:00
updatedAt: 2023-10-06T03:33:43.034+00:00
__v: 0
```

TASK 2(displaying relevant data)

FILE: frontend/componets/home.js

```
return (
  <div>
    <div>
      <h1>Books List</h1>
      <a href='/books/create'><button>Create</button></a>
    </div>
    <BooksTable books={books} />
  </div>
);
```

FILE: frontend/model/homemodel.js

```
{books.map((book, index) => (
  <tr key={book._id}>
    <td>
      {index + 1}
    </td>
    <td>
      {book.title}
    </td>
    <td>
      {book.author}
    </td>
    <td>
      {book.publishYear}
    </td>
    <td>
      <div>
        <a href={`/books/details/${book._id}`}><button>Details</button></a>
        <a href={`/books/edit/${book._id}`}><button>Edit</button></a>
        <a href={`/books/delete/${book._id}`}><button>Delete</button></a>
      </div>
    </td>
  </tr>
```

Now it's time to implement other 3 (RUD) operations

SHOW BOOK OPERATION

- to send the id of book from frontend to backend
- search for the book in db with “id” and send the response to frontend
- display the sent data

FILE: frontend/components/showBook.js
ABOUT : sending id as parameter and then displaying the data returned by server
WORKING

TASK 1 - to extract id of the book from url

```
const { id } = useParams();
```

TASK 2 - sending the data to backend and receiving the data

```
const [book, setBook] = useState({});
useEffect(() => {
  axios
    .get(`http://localhost:5555/books/${id}`)
    .then((response) => {
      setBook(response.data);
    })
    .catch((error) => {
      console.log(error);
    });
}, [id]);
```

NOTE: see how id was set in dependency array
FORMAT OF URL

- www.xyz.com/books:3s5rfs6ssfsttsr (parameter format → :id)
- www.xyz.com/books?lol (query format → ?query)

FILE: backend/routes/showBook.js

```
// Route for Get One Book from database by id
router.get('/:id', async (request, response) => {
  try{
    const { id } = request.params;
    const book = await Book.findById(id);
    return response.status(200).json(book);
  }
  catch (error) {
    console.log(error.message);
    response.status(500).send({ message: error.message });
  }
});
```

EDIT BOOK

comprises of 3 parts

- getting the data of that book from backend
- taking the new input from the user
- updating the book details with user value

```
const [title, setTitle] = useState('');
const [author, setAuthor] = useState('');
const [publishYear, setPublishYear] = useState('');
const navigate = useNavigate();
const { id } = useParams();

useEffect(() => {
  axios.get(`http://localhost:5555/books/${id}`)
    .then((response) => {
    setAuthor(response.data.author);
    setPublishYear(response.data.publishYear);
    setTitle(response.data.title);
  }).catch((error) => {
    console.log(error);
  });
}, [id]);
```

FILE: backend/routes/showBook.js

```
// Route for Get One Book from database by id
router.get('/:id', async (request, response) => {
  try{
    const { id } = request.params;
    const book = await Book.findById(id);
    return response.status(200).json(book);
  }
  catch (error) {
    console.log(error.message);
    response.status(500).send({ message: error.message });
  }
});
```