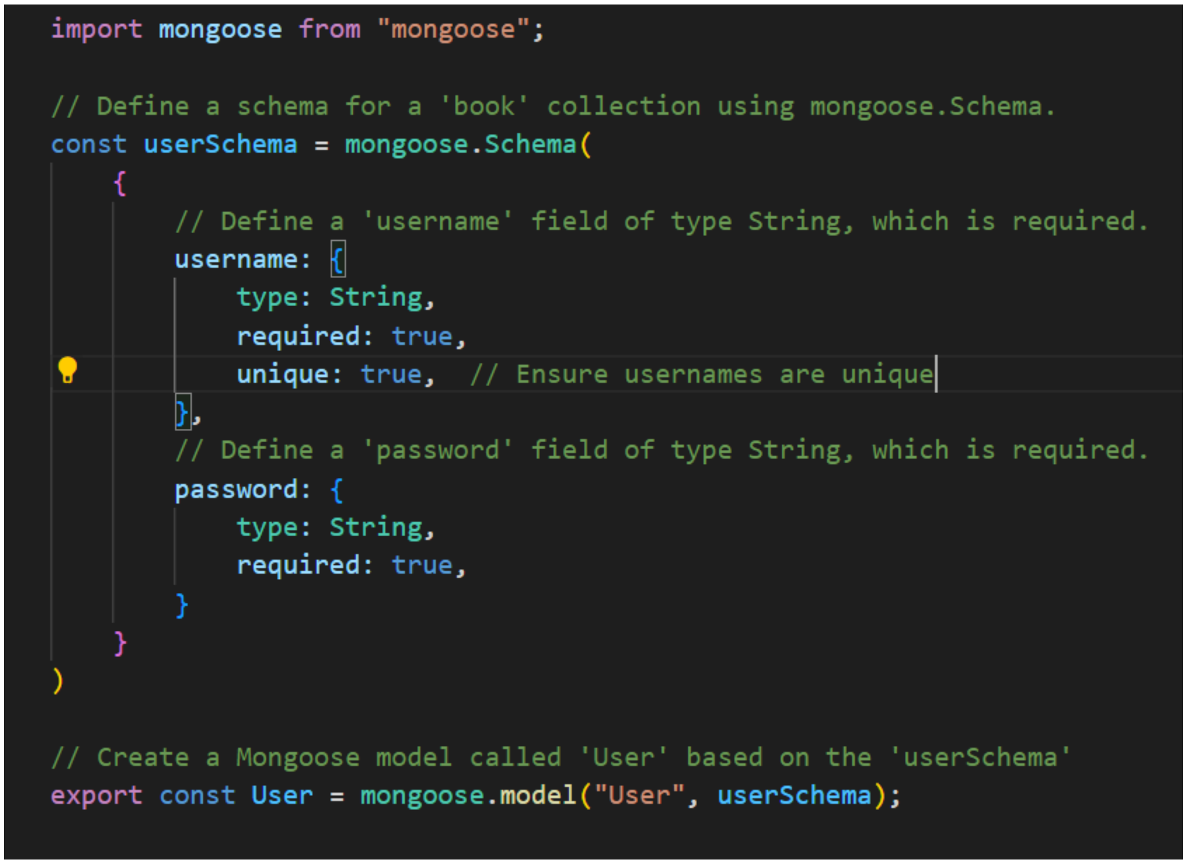
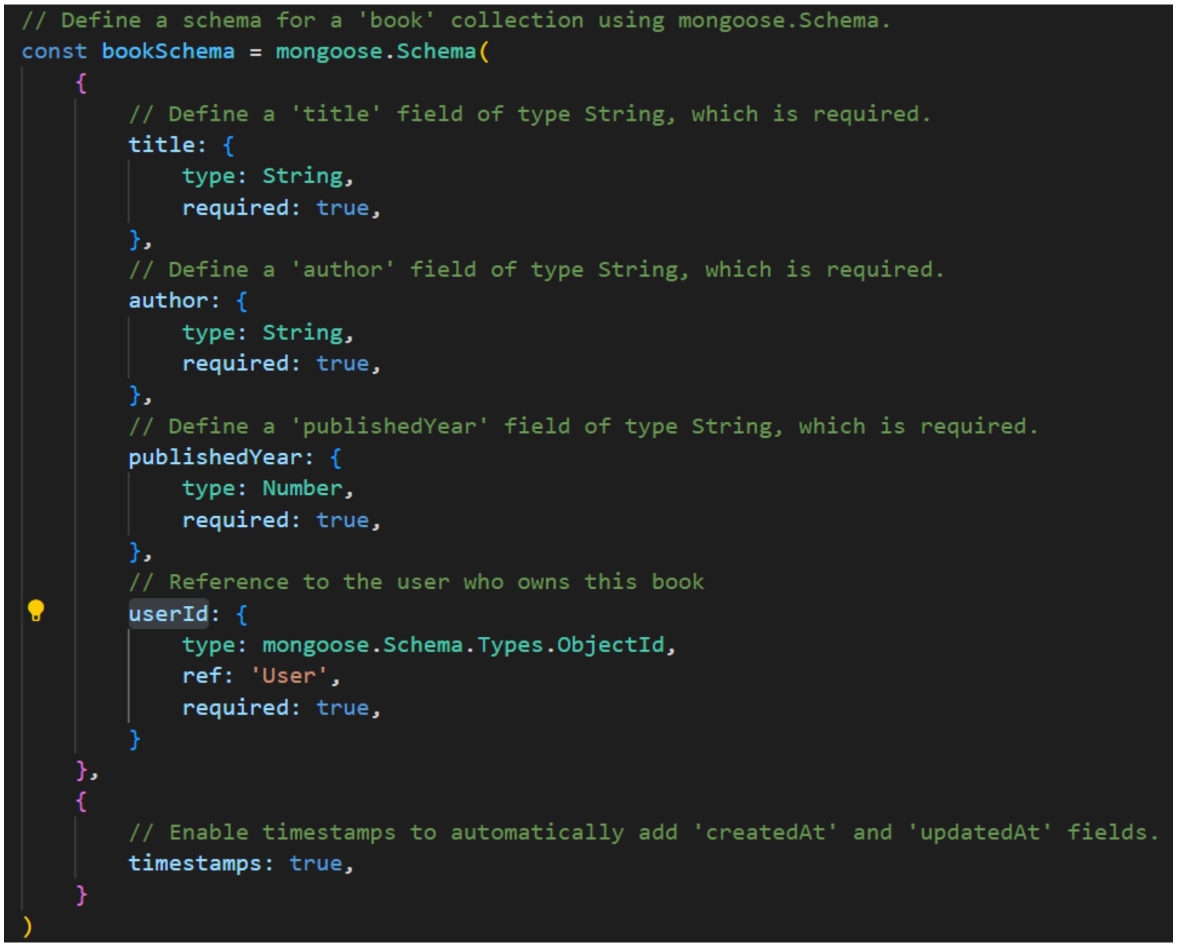
Web Development Book Store Project



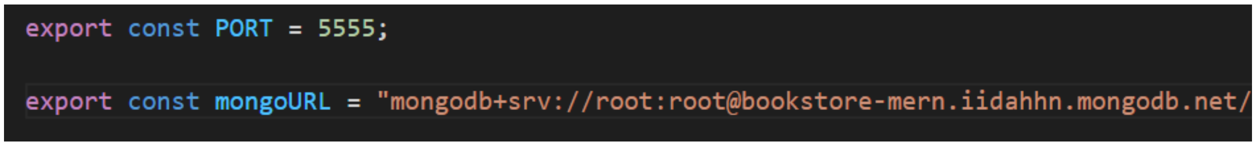
```
import mongoose from "mongoose";

// Define a schema for a 'book' collection using mongoose.Schema.
const userSchema = mongoose.Schema(
    {
        // Define a 'username' field of type String, which is required.
        username: {
            type: String,
            required: true,
            unique: true,  // Ensure usernames are unique
        },
        // Define a 'password' field of type String, which is required.
        password: {
            type: String,
            required: true,
        }
    }
)

// Create a Mongoose model called 'User' based on the 'userSchema'
export const User = mongoose.model("User", userSchema);
```

This code defines a Mongoose schema and model for a "User" collection in MongoDB. The schema specifies that each user document must have a unique `username` and a `password`, both of which are required strings. The model is then exported for use in other parts of the application, enabling user authentication and management functionality.



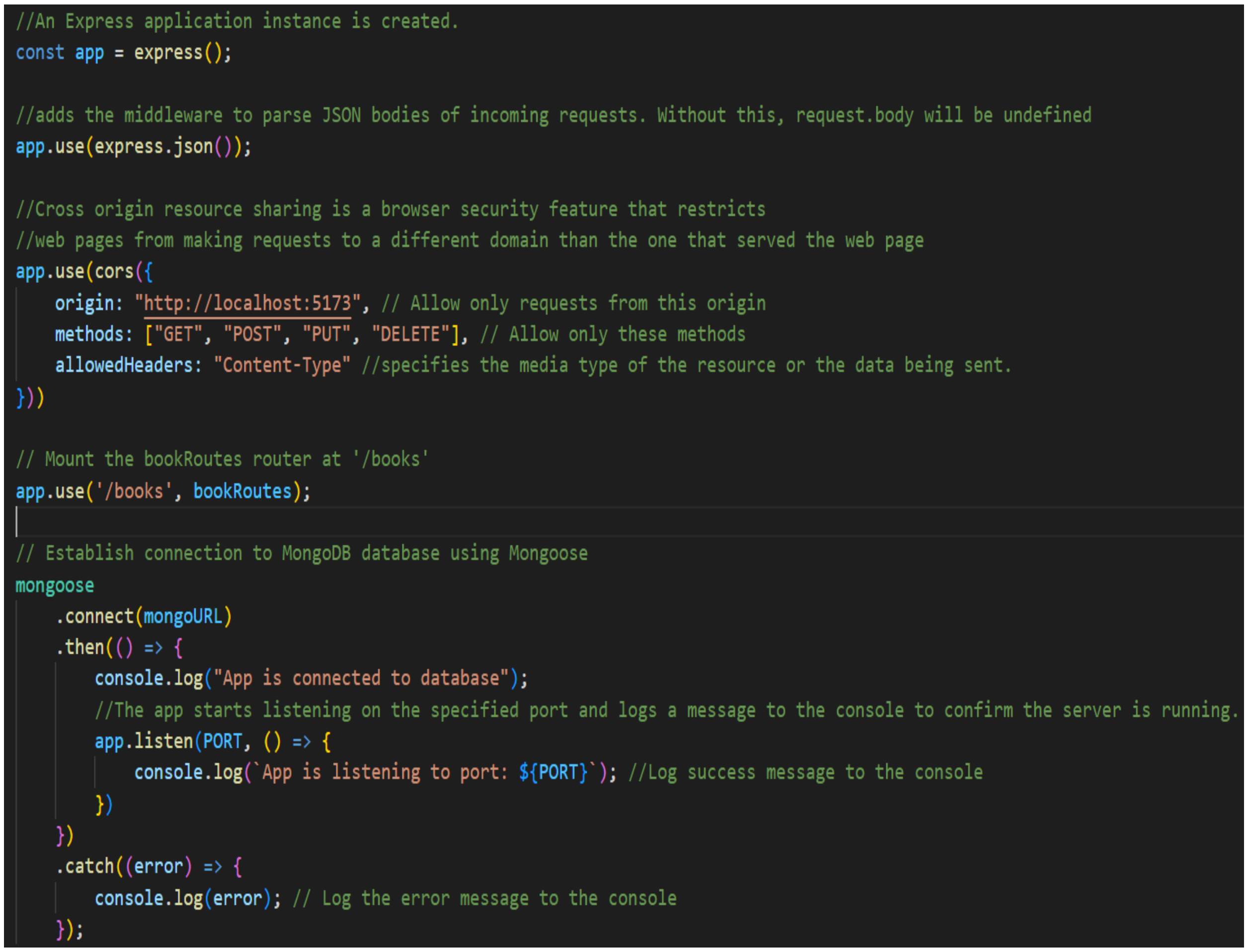
```
// Define a schema for a 'book' collection using mongoose.Schema.
const bookSchema = mongoose.Schema(
    {
        // Define a 'title' field of type String, which is required.
        title: {
            type: String,
            required: true,
        },
        // Define a 'author' field of type String, which is required.
        author: {
            type: String,
            required: true,
        },
        // Define a 'publishedYear' field of type String, which is required.
        publishedYear: {
            type: Number,
            required: true,
        },
        // Reference to the user who owns this book
        userId: {
            type: mongoose.Schema.Types.ObjectId,
            ref: 'User',
            required: true,
        }
    },
    {
        // Enable timestamps to automatically add 'createdAt' and 'updatedAt' fields.
        timestamps: true,
    }
)
```

This code defines a Mongoose schema and model for a "Book" collection in MongoDB. The schema specifies that each book document must have a `title`, `author`, and `publishedYear`, all of which are required fields. Additionally, it includes a reference to a `userId`, which links each book to a specific user. Timestamps are enabled to automatically add `createdAt` and `updatedAt` fields. The model is then exported for use in other parts of the application, enabling book management functionality.



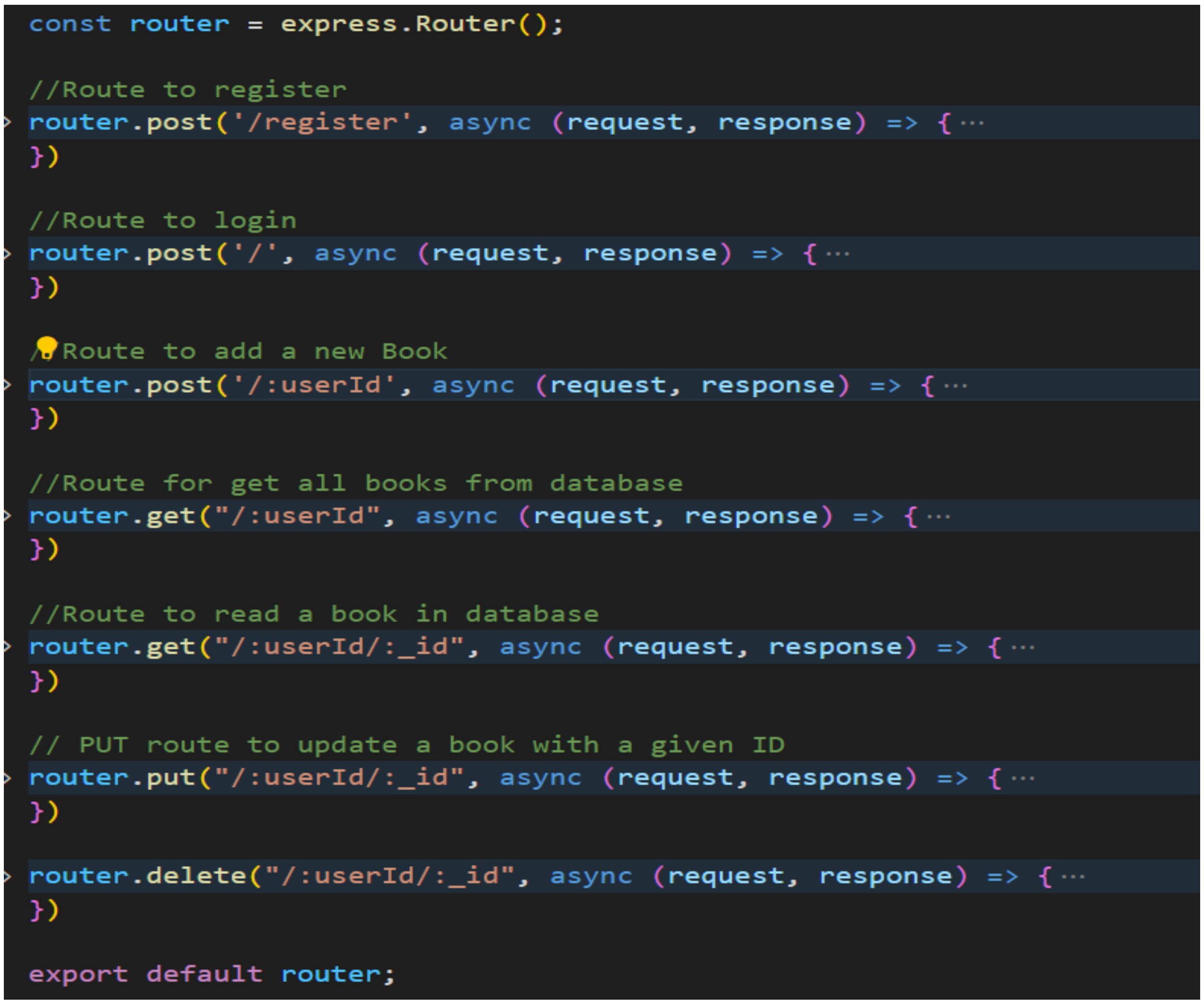
```
export const PORT = 5555;

export const mongoURL = "mongodb+srv://root:root@bookstore-mern.iidahhn.mongodb.net/
```

This code exports configuration constants for a server application. It sets the `PORT` constant to `5555`, defining the port number on which the server will listen for incoming requests. It also exports the `mongoURL` constant, which contains the MongoDB connection string used to connect to a MongoDB database hosted on MongoDB Atlas. This connection string includes authentication details (username and password) and specifies the database cluster to connect to.



```
//An Express application instance is created.
const app = express();

//adds the middleware to parse JSON bodies of incoming requests. Without this, request.body will be undefined
app.use(express.json());

//Cross origin resource sharing is a browser security feature that restricts
//web pages from making requests to a different domain than the one that served the web page
app.use(cors({
    origin: "http://localhost:5173", // Allow only requests from this origin
    methods: ["GET", "POST", "PUT", "DELETE"], // Allow only these methods
    allowedHeaders: "Content-Type" //specifies the media type of the resource or the data being sent.
}))

// Mount the bookRoutes router at '/books'
app.use('/books', bookRoutes);

// Establish connection to MongoDB database using Mongoose
mongoose
    .connect(mongoURL)
    .then(() => {
        console.log("App is connected to database");
        //The app starts listening on the specified port and logs a message to the console to confirm the server is running.
        app.listen(PORT, () => {
            console.log(`App is listening to port: ${PORT}`); //Log success message to the console
        })
    })
    .catch((error) => {
        console.log(error); // Log the error message to the console
    });
```

This code snippet sets up an Express.js application with middleware to parse JSON request bodies and enables Cross-Origin Resource Sharing (CORS) for requests from http://localhost:5173 using specific HTTP methods. It mounts bookRoutes at the /books path. The code also establishes a connection to a MongoDB database using Mongoose, logging a success message to the console upon a successful connection and an error message if the connection fails. Once the database connection is successful, the server starts listening on the specified port, logging a message to confirm it is running.



```
const router = express.Router();

//Route to register
router.post('/register', async (request, response) => {…
})

//Route to login
router.post('/', async (request, response) => {…
})

//Route to add a new Book
router.post('/:userId', async (request, response) => {…
})

//Route for get all books from database
router.get("/:userId", async (request, response) => {…
})

//Route to read a book in database
router.get("/:userId/:_id", async (request, response) => {…
})

// PUT route to update a book with a given ID
router.put("/:userId/:_id", async (request, response) => {…
})

router.delete("/:userId/:_id", async (request, response) => {…
})

export default router;
```

This code defines an Express router with routes for user registration (POST /register), login (POST /), and various operations on books. It includes routes to create a new book (POST /:userId), get all books (GET /:userId), read a specific book (GET /:userId/:id), update a book (PUT /:userId/:id), and delete a book (DELETE /:userId/:id). Each route uses asynchronous handlers to process the requests. The router is exported as the default export.

Yassir Zoaka

20222022559

Software Engineering