Week 10 Portfolio

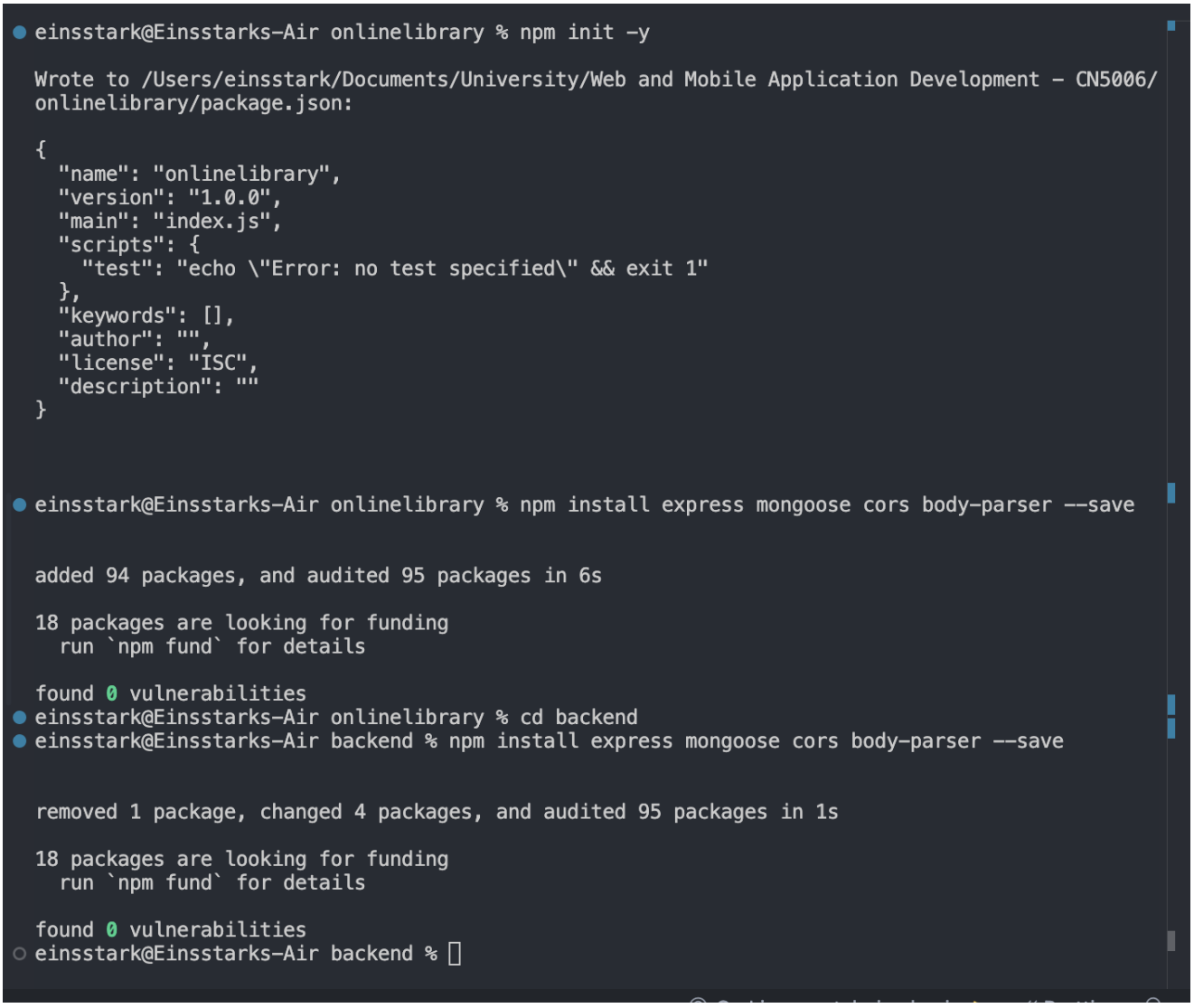
Name: Deepak Kumar
Student ID: 2284279

Backend for Online library

1. Screenshots

a. Project Set Up

I created a folder for the project, navigated to the backend folder, and ran `npm init -y` to set up the backend. Then I installed `express`, `mongoose`, `cors`, and `body-parser` using the `npm install` command to prepare for the backend setup.



b. Backend Files

1. BooksSchema.js

I created a file `BooksSchema.js`, defined fields like `booktitle`, `author`, and exported the schema using `mongoose.model()`.

2. MongoDBConnect.js

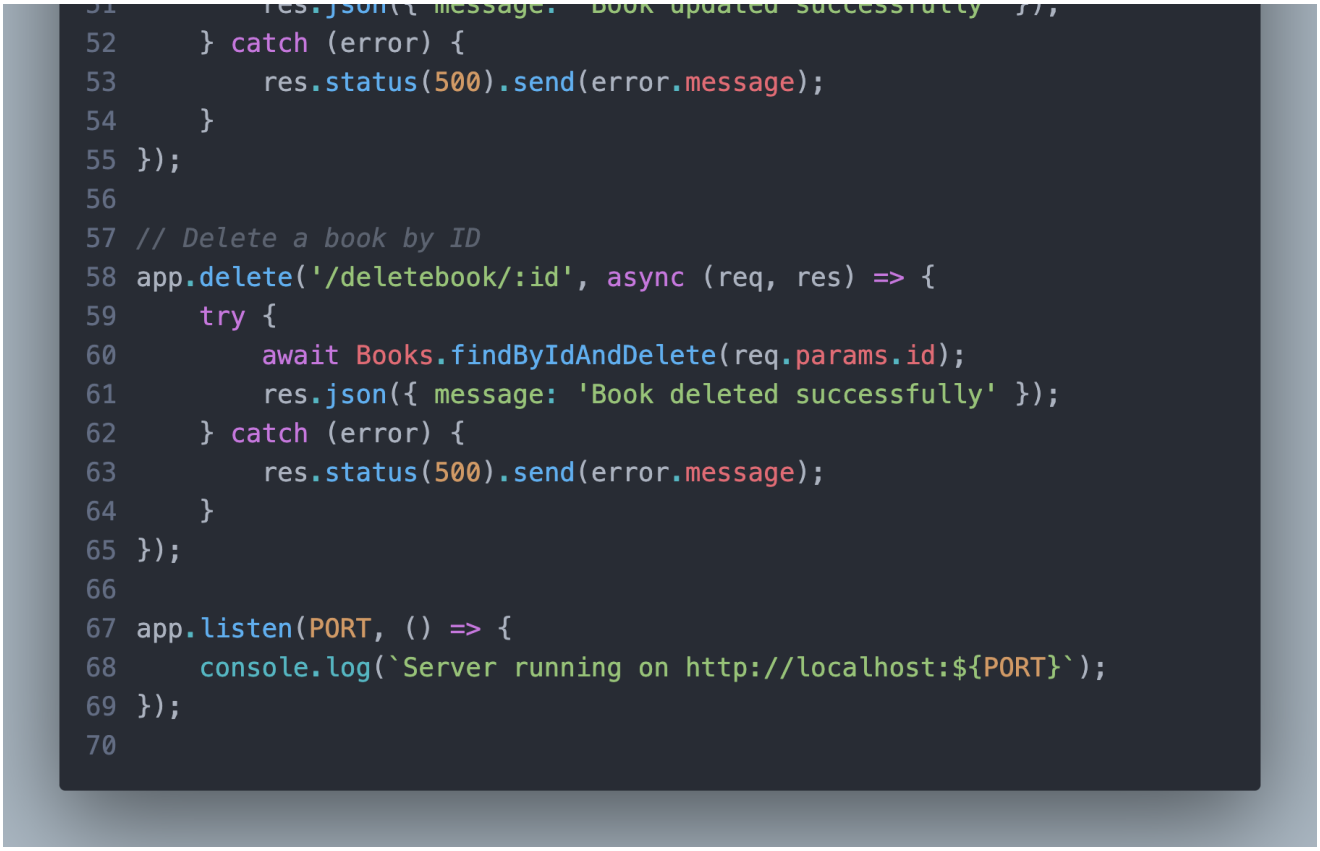
In `MongoDBConnect.js`, I connected to a MongoDB database using Mongoose and added error handling for connection issues.

```
const mongoose = require('mongoose');

const MONG_URI = 'mongodb://localhost:27017/BooksData';

mongoose.connect(MONG_URI, {
    useUnifiedTopology: true,
    useNewUrlParser: true,
});

const db = mongoose.connection;

Tabnine | Edit | Test | Explain | Document | Ask
db.on('error', (err) => {
    console.log('Error:', err);
});

Tabnine | Edit | Test | Explain | Document | Ask
db.once('connected', () => {
    console.log('Connected to MongoDB:', MONG_URI);
});

module.exports = db;

```

3. Server.js

I made a `server.js` file, used Express to create an app, and added middleware like `body-parser` and `cors`. And added RESTful routes (`/allbooks`, `/getbook/:id`, etc.) in `server.js` to handle adding, updating, deleting, and retrieving books.

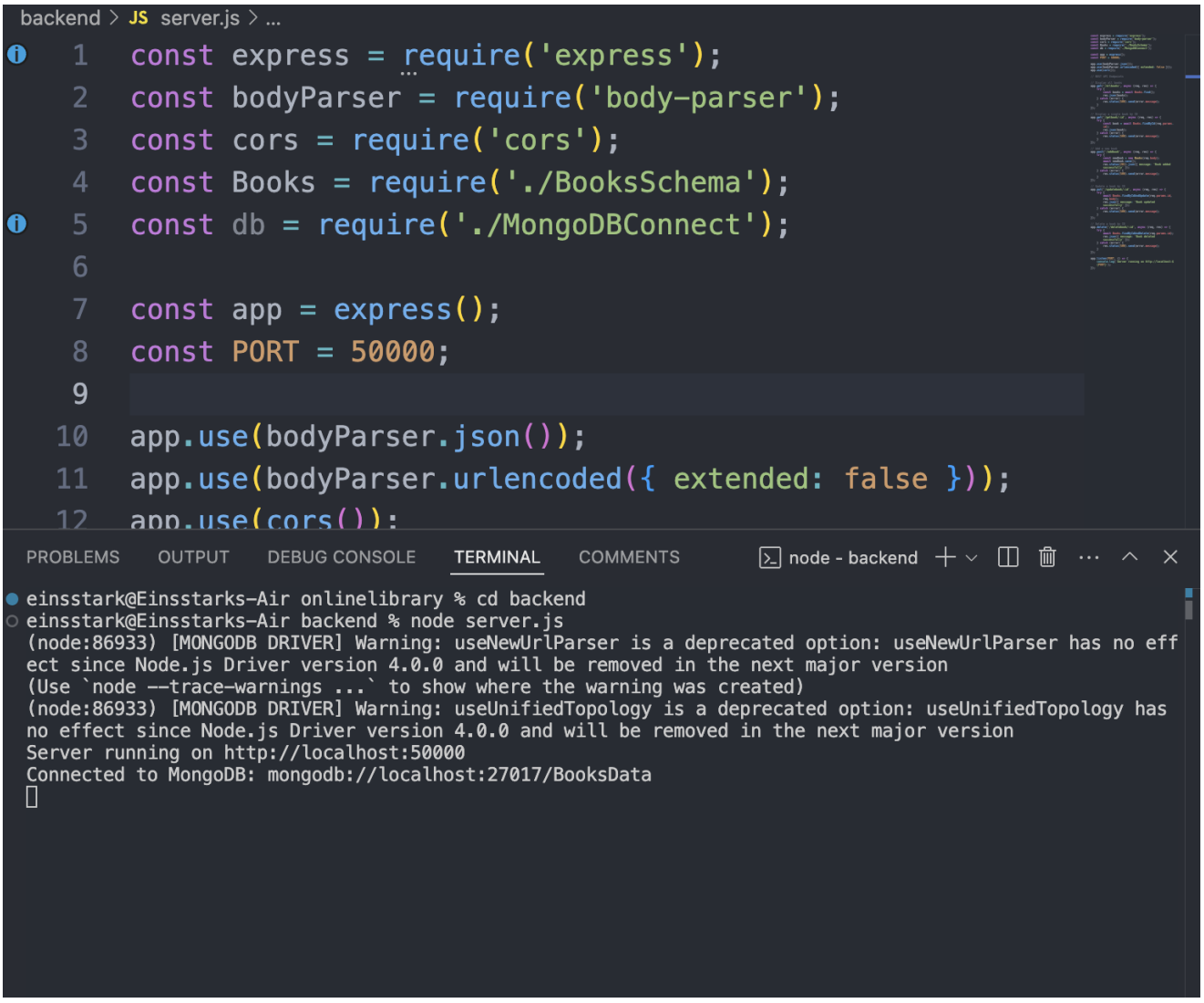
```
const express = require('express');
const bodyParser = require('body-parser');
const cors = require('cors');
const Books = require('./BooksSchema');
const db = require('./MongoDBConnect');

const app = express();
const PORT = 5000;

app.use(bodyParser.json());
app.use(bodyParser.urlencoded({ extended: false }));
app.use(cors());

// REST API Endpoints

// Display all books
app.get('/allbooks', async (req, res) => {
    try {
        const books = await Books.find();
        res.json(books);
    } catch (error) {
        res.status(500).send(error.message);
    }
});

// Display a single book by ID
app.get('/getbook/:id', async (req, res) => {
    try {
        const book = await Books.findById(req.params.id);
        res.json(book);
    } catch (error) {
        res.status(500).send(error.message);
    }
});

// Add a new book
app.post('/addbook', async (req, res) => {
    try {
        const newBook = new Books(req.body);
        await newBook.save();
        res.status(201).json({ message: 'Book added successfully' });
    } catch (error) {
        res.status(500).send(error.message);
    }
});

// Update a book by ID
app.put('/updatebook/:id', async (req, res) => {
    try {
        await Books.findByIdAndUpdate(req.params.id, req.body);
        res.json({ message: 'Book updated successfully' });
```



```
        res.json({ message: 'Book updated successfully' });
    } catch (error) {
        res.status(500).send(error.message);
    }
});

// Delete a book by ID
app.delete('/deletebook/:id', async (req, res) => {
    try {
        await Books.findByIdAndDelete(req.params.id);
        res.json({ message: 'Book deleted successfully' });
    } catch (error) {
        res.status(500).send(error.message);
    }
});

app.listen(PORT, () => {
    console.log(`Server running on http://localhost:${PORT}`);
});

```

c. Testing the server

I tested the server and it is successfully running.



2. Reflection

In this lab, I learned how to create the backend for an online library system using Node.js, Express, and MongoDB. First, I defined the book schema using Mongoose, which included fields like `booktitle`, `author`, and `format`. Then, I connected the backend to a MongoDB database and built REST API endpoints for CRUD operations. Fixing the port error (after port was busy) and removing deprecated options in Mongoose was tricky but taught me how to handle real-world problems.

Compared to previous labs, this felt more like building an actual project. In earlier labs, I only worked on simple JavaScript functions and parts of backend and frontend. Those helped me understand logic and structure, but this lab showed me how to apply those skills to something bigger. Next week, I will learn learn how the backend to the frontend because it will give me a idea of how full online library will work.