Node .Js Scenario-Based Coding Assignment

Project Title:

"Book Management REST API" – Manage a list of books in a JSON file.

User Story Format

Story 1 – Project Setup with NPM

As a backend developer,
I want to initialize my project with npm init and install required packages,
So that my Node.js application has a structured dependency management system.

Acceptance Criteria:

- npm init -y should create package.json.
- Install express and nodemon (dev dependency).
- Add a start script in package.json to run the app with nodemon.

Story 2 – Create Server with Express.js

As a backend developer,
I want to create an HTTP server using Express.js,
So that I can handle client requests and send responses.

Acceptance Criteria:

- server.js file starts an Express server on port 3000.
- Root route / returns "Welcome to Book Management API" as JSON.

Story 3 – Manage Books in a File (fs module)

As a backend developer,
I want to store book data in a JSON file using Node.js fs module,
So that the data persists between server restarts.

Acceptance Criteria:

- Create a data/books.json file with an initial empty array [].
- Implement a utility module bookService.js that reads and writes from the JSON file asynchronously.

Story 4 – CRUD Routes for Books

As a client,
I want to be able to Create, Read, Update, and Delete books via API endpoints,
So that I can manage my book collection.

Acceptance Criteria:

- GET /books → Returns all books.
- GET /books/:id → Returns a book by ID.
- POST /books → Adds a new book (with title and author).
- PUT /books/:id → Updates book details by ID.
- DELETE /books/:id → Deletes a book by ID.

Story 5 – Use of Async/Await and Events

As a backend developer,
I want to use asynchronous programming and events,
So that file operations don't block the main thread and I can log changes.

Acceptance Criteria:

- Use async/await in file read/write functions.
- Create an eventEmitter to log "Book Added", "Book Updated", "Book Deleted".

Time Allocation (120 min)

Task	Time (min)
Project setup & NPM basics	15
Create server & basic route	15
File handling with fs	20
Implement CRUD API	40
Add async/await & events	20
Final testing & submission	10

Submission Guidelines

- Folder Structure:

book-api/

├── data/books.json

├── services/bookService.js

├── server.js

├── package.json

├── package-lock.json

└── README.md

1. Dependencies:
    - "express" in dependencies.
    - "nodemon" in devDependencies.
2. Scripts:
    - "start": "nodemon server.js" in package.json.
3. Coding Standards:
    - Use ES6+ syntax (const, let, arrow functions, async/await).
    - Proper error handling with try/catch.
4. Testing:

- Test all CRUD routes using Postman or curl.

5. Documentation:
    - README should include setup instructions and API endpoints.