Lab 4: API, Express and MongoDB

Task 1: Basic CRUD Operations in React with Express and MongoDB

Objective:

- Learn how to connect React with an Express backend
- Perform Create, Read, Update, and Delete (CRUD) operations on MongoDB using axios
- Implement the same pattern for both Users and Books collections

---

Part A: Users Management System

1. Backend (Express & MongoDB Setup)

User Schema Definition: Create a Mongoose schema with the following fields:

- `name` (String, required)
- `email` (String, required, unique)
- `age` (Number, required)
- `phone` (String, required)
- Include timestamps (createdAt, updatedAt)

API Endpoints Implementation:

1. **POST /users** → Add a user
    - Extract name, email, age, phone from request body
    - Create new User instance
    - Save to database
    - Return saved user with status 201
    - Handle validation errors with status 400
2. **GET /users** → Fetch all users
    - Find all users from database
    - Return users array
    - Handle errors with status 500
3. **PUT /users/:id** → Update a user
    - Extract id from URL parameters
    - Extract updated fields from request body
    - Use findByIdAndUpdate with options: {new: true, runValidators: true}
    - Return updated user
    - Handle "not found" case with status 404
    - Handle validation errors with status 400
4. **DELETE /users/:id** → Delete a user
    - Extract id from URL parameters
    - Use findByIdAndDelete
    - Return success message

- Handle "not found" case with status 404
- Handle errors with status 500

Part B: Books Library Management System

1. Backend (Express & MongoDB Setup)

MongoDB Connection:

- Database name: `librarydb`
- Connection string: `mongodb://localhost:27017/librarydb`

Book Schema Definition: Create schema with these fields:

- `title` (String, required)
- `author` (String, required)
- `isbn` (String, required, unique)
- `publishedYear` (Number, required)
- `genre` (String, required)
- `availableCopies` (Number, required, default: 1)
- Include timestamps

API Endpoints: Follow the exact same pattern as Users but for books:

1. **POST /books** → Add a book
2. **GET /books** → Fetch all books
3. **PUT /books/:id** → Update a book
4. **DELETE /books/:id** → Delete a book

Use the same error handling patterns and status codes.