**Online Bookstore Project**

**Project Overview**

This project is a dynamic and responsive website built using React for the frontend and Node.js with Express for the backend. The website connects to an Oracle database to perform CRUD operations (Create, Read, Update, Delete). It includes features like product management, order processing, and customer management. The project also ensures a professional, mobile-friendly UI with a fully functional API.

**Technologies Used**

* **Frontend:** React, Bootstrap
* **Backend:** Node.js, Express.js
* **Database:** Oracle Database
* **Tools:** Nodemon, dotenv, CORS, Body-Parser
* **Middleware:** Express Middleware for routing, body parsing, and CORS handling

**Project Features**

* **Dynamic Pages:**
* Home, Products, Contact, and more.
* Dynamic product listing connected to the database.
* **API Integration:**
* RESTful API to interact with Oracle Database.
* CRUD operations supported for products, customers, and orders.
* **Responsive Design:**
* Optimized for mobile, tablet, and desktop.
* **Database Connectivity:**
* OracleDB integration for secure and efficient data operations.

**Setup Instructions**

**Prerequisites**

* **Node.js and npm** installed
* **Oracle Database** installed and running
* **OracleDB Node.js driver** dependencies (ensure Oracle Instant Client is installed)

**1. Backend Setup**

**i. Install Dependencies**

Run the following commands:

npm init -y

npm install express oracledb body-parser cors dotenv nodemon

**ii. Environment Variables**

Create a .env file in the backend directory and add the following:

DB\_USER=your\_oracle\_username

DB\_PASSWORD=your\_oracle\_password

DB\_CONNECT\_STRING=your\_oracle\_connect\_string

PORT=5000

**iii. Run the Server**

nodemon server.js

**2. Frontend Setup**

**i. Install Dependencies**

In the frontend directory, initialize React and install required packages:

npx create-react-app frontend

cd frontend

npm install axios react-router-dom bootstrap

**ii. Run the React App**

npm start

**Backend API Documentation**

**Base URL:**

[http://localhost:5000](http://localhost:5000/)

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Endpoint** | **Description** | **Parameters/Body** |
| GET | /api/:table | Fetch all rows from a table | table (e.g., Products, Orders) |
| POST | /api/:table | Insert new data into a table | JSON body with key-value pairs |
| PUT | /api/:table/:id | Update data in a table by ID | JSON body with key-value pairs, id |
| DELETE | /api/:table/:id | Delete data from a table by ID | id (primary key value) |
| POST | /api/products | Insert a new product | JSON body: {ProductID, ProductName, ...} |
| POST | /api/customers | Insert a new customer | JSON body: {CustomerID, FirstName, ...} |
| GET | /health | Check server status | None |

**Database Schema**

**Products Table:**

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Description** |
| ProductID | Number | Primary Key |
| ProductName | Varchar2 | Name of the product |
| Category | Varchar2 | Product category |
| Price | Number | Price of product |
| QuantityInStock | Number | Stock count |

**Customers Table:**

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Description** |
| CustomerID | Number | Primary Key |
| FirstName | Varchar2 | First name of customer |
| LastName | Varchar2 | Last name of customer |
| Email | Varchar2 | Email address |
| Phone | Varchar2 | Contact number |
| Address | Varchar2 | Address of customer |

**Orders Table:**

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Description** |
| OrderID | Number | Primary Key |
| CustomerID | Number | Foreign Key |
| OrderDate | Date | Date of the order |
| TotalAmount | Number | Total order amount |

**Frontend Structure**

**File Structure**

frontend/

├── public/

├── src/

│ ├── components/

│ │ ├── Navbar.js

│ │ ├── Products.js

│ │ ├── Contact.js

│ │ └── Footer.js

│ ├── pages/

│ │ ├── Home.js

│ │ └── About.js

│ ├── App.js

│ ├── api.js

│ ├── index.js

├── package.json

**Key Frontend Files**

**1.api.js**

**2.Products.js**

**Testing and Debugging**

**Backend Testing**

* Use **Postman** to test API endpoints.
* Test all CRUD operations for products, customers, and orders.

**Frontend Testing**

* Test dynamic rendering of product lists.
* Ensure the website is responsive on mobile and desktop.

**Challenges**

* Setting up OracleDB connectivity with Node.js.
* Ensuring SQL injection prevention for dynamic table queries.

**Conclusion**

This project successfully demonstrates a full-stack web application integrating React, Node.js, and Oracle Database. It is highly scalable and ready for deployment with professional and dynamic features.