Frontend-Backend Integration Plan

Generated: 18 Aug 2025

Replace dummy data with dynamic API calls to the Node.js/Sequelize backend

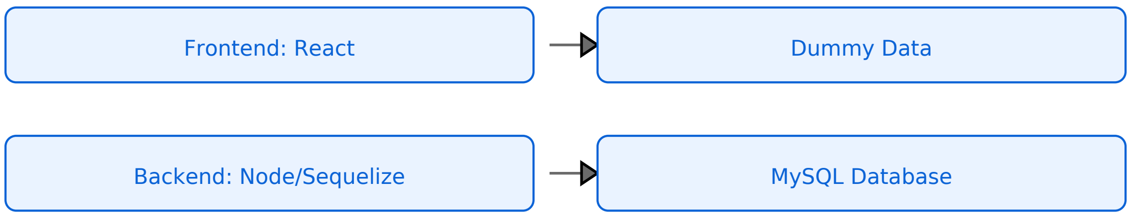
Generated: 18 Aug 2025

1. Overview

Objective: Replace static dummy data with dynamic API calls to the Node.js/Sequelize backend.
Scope: Create API services · Update components to fetch real data · Implement form submissions · Handle loading/error states.
Key benefits: Real-time synchronization, persistent storage, and full CRUD support.

2. Current Architecture

Current high-level architecture (visual):



3. Proposed Integration Approach

3.1 API Service Layer

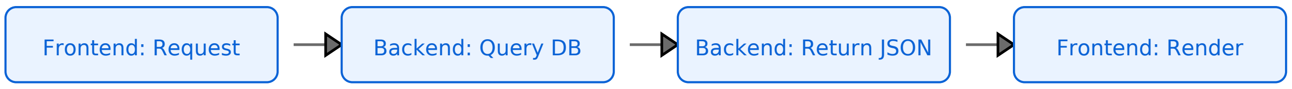
```
import axios from 'axios';

const API_URL = process.env.REACT_APP_API_URL || 'http://localhost:3000/api';

// Example service method
export const fetchProducts = async () => {
const response = await axios.get(`${API_URL}/products`);
return response.data;
};

export const createPurchase = (purchaseData) => {
return axios.post(`${API_URL}/purchases`, purchaseData);
};
```

3.2 Component Integration Strategy

For each component: remove dummy data imports → add useEffect for data fetching → implement loading/error states → replace static data with API responses.

3.3 Data Flow

Typical request/response flow (visual):



4. API Endpoint Mapping (Swagger/OpenAPI)

The table below presents the frontend-to-backend endpoint mappings and is aligned with the project's Swagger/OpenAPI specification.

Frontend Component	Backend Endpoint (guess)	Method	Notes
Brands.jsx	/api/brands	GET	-

Frontend Component	Backend Endpoint (guess)	Method	Notes
Billers.jsx	/api/billers	GET	-
Categories.jsx	/api/categories	GET	-
CustomerDueReport.jsx	/api/reports/customer-due	GET	-
CustomerReport.jsx	/api/reports/customer	GET	-
Customers.jsx	/api/customers	GET	-
Dashboard.jsx	/api/dashboard	GET	summary KPIs
Invoices.jsx	/api/invoices	GET	-
POS.jsx	/api/pos	GET	POS product & pricing data
Products.jsx / AddProduct.jsx	/api/products	GET / POST	GET listed; POST for create
PurchaseReport.jsx	/api/reports/purchase	GET	-
Purchases.jsx / AddPurchase.jsx	/api/purchases	GET / POST	GET for listing
Sales.jsx	/api/sales	GET	-
SalesReport.jsx	/api/reports/sales	GET	-
SalesReturn.jsx	/api/sales-returns	GET	-
Suppliers.jsx	/api/suppliers	GET	-
Units.jsx	/api/units	GET	-
ProductList.jsx	/api/products	GET	tables/listing
ProductForm.jsx	/api/products	POST/PUT	create/update endpoints (not GET)
CustomerReport.jsx	/api/reports/customer	GET	-

5. State Management Updates

Before:

```
const [brands] = useState(dummyBrands);
```

After:

```
const [brands, setBrands] = useState([]);
const [loading, setLoading] = useState(true);
const [error, setError] = useState(null);

useEffect(() => {
const fetchData = async () => {
try {
const data = await fetchBrands();
setBrands(data);
} catch (err) {
setError(err.message);
} finally {
setLoading(false);
}
};

fetchData();
}, []);
```

6. Form Submission Handling

```
const handleSubmit = async (e) => {
e.preventDefault();
try {
await createPurchase({
supplier: form.supplier,
date: form.date,
items: form.items,
total: calculateTotal()
});
navigate('/purchases');
} catch (error) {
setSubmissionError('Failed to create purchase');
}
};
```

7. Error Handling Strategy

• HTTP Errors: Intercept 400/500 responses · Network Errors: Show user-friendly messages · Validation Errors: Highlight form fields · Fallback UI: Maintain structure during loading

8. Testing Plan

Unit tests for API functions · Integration tests for data-fetching components · Publish Swagger/OpenAPI spec for endpoint reference · Cypress end-to-end: product creation, purchase submission, report loading.

9. Dependencies

Axios · react-query (optional) · dotenv · CORS middleware · Swagger/OpenAPI tooling (e.g., swagger-jsdoc/swagger-ui) for documentation

10. Implementation Timeline

Phase Tasks Duration

Setup Configure Axios; Create API services 1 day

Read Ops Integrate GET endpoints; Add loading states 2 days

Write Ops Implement form submissions; Handle POST/PUT 2 days

Error Handling Create error boundaries; Form validation 1 day

Testing Write test cases; Verify all flows 2 days

11. Risks & Mitigation

• API Schema Mismatch: Use Swagger/OpenAPI for a canonical spec · CORS Issues: Add proper headers · State Sync: Use refetch patterns · Performance: Add pagination

Generated on: 18 August 2025