Product Requirements Document: LIGI Reporting System

Single-Page HTML/CSS/JavaScript Implementation

1. Project Overview

Project Name: LIGI Reporting - Multi-Branch Retail Management System

Purpose: A comprehensive web-based dashboard system for managing multi-branch retail operations, providing role-based access for Business Owners, Stores Managers, and Branch Managers.

Tech Stack: Pure HTML5, CSS3, Vanilla JavaScript (ES6+) - Single File Application

Architecture: Single index.html file containing all pages, styles, and functionality

2. File Structure

project/

├── index.html (ONLY FILE - contains everything)

├── assets/ (optional for images/icons)

└── README.md

3. HTML Structure Organization

3.1 Document Layout

<!DOCTYPE html>

<html lang="en">

<head>

<!-- Meta tags, title, embedded CSS -->

</head>

<body>

<!-- Navigation -->

<nav id="main-nav" class="hidden">...</nav>

<!-- Login Page -->

<div id="login-page" class="page active">...</div>

<!-- Business Owner Dashboard -->

<div id="business-owner-dashboard" class="page hidden">...</div>

<!-- Stores Manager Dashboard -->

<div id="stores-manager-dashboard" class="page hidden">...</div>

<!-- Branch Manager Dashboard -->

<div id="branch-manager-dashboard" class="page hidden">...</div>

<!-- Modals and Forms -->

<div id="eod-report-modal" class="modal hidden">...</div>

<div id="stock-requisition-modal" class="modal hidden">...</div>

<!-- Embedded JavaScript -->

<script>

// All JavaScript functionality

</script>

</body>

</html>

4. CSS Architecture (Embedded)

4.1 CSS Organization within <style> tags

/\* Reset and Base Styles \*/

/\* Typography \*/

/\* Layout System (CSS Grid + Flexbox) \*/

/\* Component Styles \*/

/\* Page-specific Styles \*/

/\* Responsive Media Queries \*/

/\* Utility Classes \*/

/\* Animation Keyframes \*/

4.2 Design System Variables

:root {

--primary-blue: #3B82F6;

--primary-dark: #1E40AF;

--success-green: #10B981;

--warning-yellow: #F59E0B;

--error-red: #EF4444;

--bg-light: #F9FAFB;

--bg-white: #FFFFFF;

--text-primary: #111827;

--text-secondary: #6B7280;

--border-color: #E5E7EB;

--shadow: 0 1px 3px 0 rgba(0, 0, 0, 0.1);

--radius: 8px;

}

5. JavaScript Architecture (Embedded)

5.1 Application State Management

// Global application state

const AppState = {

currentUser: null,

currentPage: 'login',

isAuthenticated: false,

data: {

branches: [...],

inventory: [...],

requisitions: [...],

alerts: [...]

}

};

5.2 Core Functions Structure

// Authentication System

function login(email, password) { ... }

function logout() { ... }

function checkAuth() { ... }

// Page Navigation

function showPage(pageId) { ... }

function hidePage(pageId) { ... }

function navigateTo(page, userRole) { ... }

// Data Management

function loadMockData() { ... }

function updateDashboardData() { ... }

function saveToLocalStorage(key, data) { ... }

// Form Handling

function handleEODSubmit(formData) { ... }

function handleStockRequisition(formData) { ... }

function validateForm(formElement) { ... }

// UI Updates

function updateStatsCards(data) { ... }

function renderBranchPerformance(branches) { ... }

function showAlert(message, type) { ... }

6. User Roles & Demo Data

6.1 Mock Users (JavaScript Object)

const DEMO\_USERS = [

{

id: 'user\_001',

email: 'owner@demo.com',

password: 'demo',

name: 'Tadiwa Muganiyi',

role: 'business\_owner',

branch: 'Head Office'

},

{

id: 'user\_002',

email: 'stores@demo.com',

password: 'demo',

name: 'Sarah Stores',

role: 'stores\_manager',

branch: 'Central Warehouse'

},

{

id: 'user\_003',

email: 'branch@demo.com',

password: 'demo',

name: 'John Manager',

role: 'branch\_manager',

branch: 'Downtown Branch'

}

];

7. Page Specifications

7.1 Login Page Structure

<div id="login-page" class="page active">

<div class="login-container">

<div class="login-header">

<h1>LIGI Reporting</h1>

<p>Multi-Branch Retail Management System</p>

</div>

<div class="login-card">

<form id="login-form">

<input type="email" id="email" placeholder="Enter your email" required>

<input type="password" id="password" placeholder="Enter your password" required>

<button type="submit">Sign In</button>

</form>

</div>

<div class="demo-accounts">

<h3>Demo Accounts</h3>

<div class="demo-buttons">

<!-- Auto-fill demo account buttons -->

</div>

</div>

</div>

</div>

7.2 Business Owner Dashboard Structure

<div id="business-owner-dashboard" class="page hidden">

<div class="dashboard-header">

<h2>Business Overview</h2>

<p>Welcome back, <span id="user-name"></span></p>

<div class="last-updated">Last updated: <span id="timestamp"></span></div>

</div>

<div class="stats-grid">

<div class="stat-card revenue">

<h3>Total Revenue</h3>

<div class="stat-value">$12,450</div>

<div class="stat-change positive">+18%</div>

</div>

<!-- 3 more stat cards -->

</div>

<div class="main-content">

<div class="left-column">

<div class="branch-performance-card">

<!-- Branch performance list -->

</div>

</div>

<div class="right-column">

<div class="alerts-card">

<!-- Recent alerts -->

</div>

</div>

</div>

<div class="analytics-dashboard">

<div class="tabs">

<button class="tab-btn active" data-tab="revenue">Revenue</button>

<button class="tab-btn" data-tab="departments">Departments</button>

<button class="tab-btn" data-tab="inventory">Inventory</button>

<button class="tab-btn" data-tab="staff">Staff</button>

</div>

<div class="tab-content">

<!-- Dynamic content based on active tab -->

</div>

</div>

</div>

8. Form Implementations

8.1 EOD Report Modal

<div id="eod-report-modal" class="modal hidden">

<div class="modal-content">

<div class="modal-header">

<h3>End of Day Report</h3>

<button class="close-btn">&times;</button>

</div>

<form id="eod-form">

<div class="form-grid">

<div class="form-group">

<label for="butchery-sales">Butchery Sales (₦)</label>

<input type="number" id="butchery-sales" name="butcherySales" step="0.01">

</div>

<!-- More form fields -->

</div>

<div class="form-actions">

<button type="button" class="cancel-btn">Cancel</button>

<button type="submit" class="submit-btn">Submit Report</button>

</div>

</form>

</div>

</div>

9. Development Tasks Breakdown for Cursor AI

Phase 1: Foundation Setup

Task 1: Create Base HTML Structure

- Set up HTML5 document with semantic structure

- Create all page containers with proper IDs

- Add navigation structure

- Implement basic page switching divs

- Add all necessary form modals

Task 2: Implement CSS Design System

- Create CSS variables for colors, spacing, typography

- Implement responsive grid systems

- Style navigation and layout containers

- Create card components and form styles

- Add animation keyframes and transitions

Task 3: Basic JavaScript Framework

- Set up application state management

- Implement page navigation functions

- Create authentication system with demo users

- Add form validation utilities

- Set up local storage functions

Phase 2: Authentication & Navigation

Task 4: Login System

- Style login page with demo account buttons

- Implement login form validation

- Add demo account auto-fill functionality

- Create session management

- Add role-based page redirection

Task 5: Navigation System

- Create responsive navigation bar

- Add user info display (name, role, branch)

- Implement logout functionality

- Add page switching logic

- Style active/inactive states

Phase 3: Business Owner Dashboard

Task 6: Business Overview Stats

- Create stats cards grid layout

- Implement dynamic data binding

- Add trend indicators (arrows, colors)

- Create responsive card layouts

- Add hover effects and animations

Task 7: Branch Performance & Alerts

- Build branch performance list component

- Create color-coded performance indicators

- Implement alerts system with timestamps

- Add interactive elements

- Style alert boxes with appropriate colors

Task 8: Analytics Dashboard

- Create tabbed interface

- Implement tab switching functionality

- Add placeholder content for each tab

- Style active/inactive tab states

- Create responsive tab layout

Phase 4: Operational Features

Task 9: EOD Report System

- Create EOD report modal

- Implement form validation

- Add auto-calculation for totals

- Create success/error handling

- Style form elements consistently

Task 10: Stock Requisition System

- Build requisition form modal

- Implement approval/decline functionality

- Create status tracking system

- Add priority level handling

- Style requisition cards and lists

Task 11: Stores Manager Dashboard

- Create inventory overview layout

- Build supplier delivery tracking

- Implement requisition approval interface

- Add stock level monitoring

- Create interactive approval buttons

Phase 5: Final Polish

Task 12: Responsive Design

- Add mobile breakpoints for all components

- Test and fix layout issues on small screens

- Implement mobile-friendly navigation

- Optimize touch interactions

- Test cross-browser compatibility

10. Implementation Guidelines for Cursor AI

10.1 Code Organization Principles

1. Keep all HTML semantic and accessible

2. Use CSS Grid and Flexbox for layouts

3. Implement vanilla JavaScript with ES6+ features

4. Maintain separation of concerns within single file

5. Use data attributes for JavaScript selectors

6. Implement proper error handling

7. Add loading states and user feedback

8. Follow mobile-first responsive design

10.2 JavaScript Patterns to Follow

// Use module pattern for organization

const App = {

state: { ... },

auth: { ... },

ui: { ... },

data: { ... },

init() { ... }

};

// Event delegation for dynamic content

document.addEventListener('click', function(e) {

if (e.target.matches('.demo-btn')) {

// Handle demo button clicks

}

});

// Use async/await for simulated API calls

async function simulateAPICall(data) {

return new Promise(resolve => {

setTimeout(() => resolve(data), 1000);

});

}

10.3 CSS Best Practices

/\* Use BEM-like naming for clarity \*/

.dashboard\_\_header { }

.dashboard\_\_stats-grid { }

.stat-card--revenue { }

.stat-card\_\_value { }

/\* Implement consistent spacing scale \*/

.spacing-xs { margin: 0.25rem; }

.spacing-sm { margin: 0.5rem; }

.spacing-md { margin: 1rem; }

.spacing-lg { margin: 2rem; }

/\* Use CSS custom properties for theming \*/

.card {

background: var(--bg-white);

border-radius: var(--radius);

box-shadow: var(--shadow);

}

11. Success Criteria

1. Single File Implementation: All functionality contained in one index.html file
2. Role-Based Access: Three distinct user experiences based on demo login
3. Responsive Design: Works seamlessly on desktop, tablet, and mobile
4. Form Functionality: EOD reports and stock requisitions work end-to-end
5. Data Persistence: User sessions and form data saved to localStorage
6. Professional UI: Clean, modern interface matching design specifications
7. No Dependencies: Pure HTML/CSS/JS with no external libraries
8. Cross-Browser: Compatible with modern browsers (Chrome, Firefox, Safari, Edge)

12. File Size Optimization

* Minify CSS and JavaScript in production
* Optimize images and use data URIs for small icons
* Implement lazy loading for non-critical sections
* Use efficient CSS selectors and minimal DOM manipulation
* Target final file size under 500KB for optimal loading

This comprehensive PRD provides all the specifications needed to build the LIGI Reporting system as a single HTML file. Reference specific sections when working with Cursor AI to maintain consistency and context throughout development.

Product Requirements Document: LIGI Reporting System

Single-Page HTML/CSS/JavaScript Implementation

1. Project Overview

Project Name: LIGI Reporting - Multi-Branch Retail Management System

Purpose: A comprehensive web-based dashboard system for managing multi-branch retail operations, providing role-based access for Business Owners, Stores Managers, and Branch Managers.

Tech Stack: Pure HTML5, CSS3, Vanilla JavaScript (ES6+) - Single File Application

Architecture: Single index.html file containing all pages, styles, and functionality

2. File Structure

project/

├── index.html (ONLY FILE - contains everything)

├── assets/ (optional for images/icons)

└── README.md

3. HTML Structure Organization

3.1 Document Layout

<!DOCTYPE html>

<html lang="en">

<head>

<!-- Meta tags, title, embedded CSS -->

</head>

<body>

<!-- Navigation -->

<nav id="main-nav" class="hidden">...</nav>

<!-- Login Page -->

<div id="login-page" class="page active">...</div>

<!-- Business Owner Dashboard -->

<div id="business-owner-dashboard" class="page hidden">...</div>

<!-- Stores Manager Dashboard -->

<div id="stores-manager-dashboard" class="page hidden">...</div>

<!-- Branch Manager Dashboard -->

<div id="branch-manager-dashboard" class="page hidden">...</div>

<!-- Modals and Forms -->

<div id="eod-report-modal" class="modal hidden">...</div>

<div id="stock-requisition-modal" class="modal hidden">...</div>

<!-- Embedded JavaScript -->

<script>

// All JavaScript functionality

</script>

</body>

</html>

4. CSS Architecture (Embedded)

4.1 CSS Organization within <style> tags

/\* Reset and Base Styles \*/

/\* Typography \*/

/\* Layout System (CSS Grid + Flexbox) \*/

/\* Component Styles \*/

/\* Page-specific Styles \*/

/\* Responsive Media Queries \*/

/\* Utility Classes \*/

/\* Animation Keyframes \*/

4.2 Design System Variables

:root {

--primary-blue: #3B82F6;

--primary-dark: #1E40AF;

--success-green: #10B981;

--warning-yellow: #F59E0B;

--error-red: #EF4444;

--bg-light: #F9FAFB;

--bg-white: #FFFFFF;

--text-primary: #111827;

--text-secondary: #6B7280;

--border-color: #E5E7EB;

--shadow: 0 1px 3px 0 rgba(0, 0, 0, 0.1);

--radius: 8px;

}

5. JavaScript Architecture (Embedded)

5.1 Application State Management

// Global application state

const AppState = {

currentUser: null,

currentPage: 'login',

isAuthenticated: false,

data: {

branches: [...],

inventory: [...],

requisitions: [...],

alerts: [...]

}

};

5.2 Core Functions Structure

// Authentication System

function login(email, password) { ... }

function logout() { ... }

function checkAuth() { ... }

// Page Navigation

function showPage(pageId) { ... }

function hidePage(pageId) { ... }

function navigateTo(page, userRole) { ... }

// Data Management

function loadMockData() { ... }

function updateDashboardData() { ... }

function saveToLocalStorage(key, data) { ... }

// Form Handling

function handleEODSubmit(formData) { ... }

function handleStockRequisition(formData) { ... }

function validateForm(formElement) { ... }

// UI Updates

function updateStatsCards(data) { ... }

function renderBranchPerformance(branches) { ... }

function showAlert(message, type) { ... }

6. User Roles & Demo Data

6.1 Mock Users (JavaScript Object)

const DEMO\_USERS = [

{

id: 'user\_001',

email: 'owner@demo.com',

password: 'demo',

name: 'Tadiwa Muganiyi',

role: 'business\_owner',

branch: 'Head Office'

},

{

id: 'user\_002',

email: 'stores@demo.com',

password: 'demo',

name: 'Sarah Stores',

role: 'stores\_manager',

branch: 'Central Warehouse'

},

{

id: 'user\_003',

email: 'branch@demo.com',

password: 'demo',

name: 'John Manager',

role: 'branch\_manager',

branch: 'Downtown Branch'

}

];

7. Page Specifications

7.1 Login Page Structure

<div id="login-page" class="page active">

<div class="login-container">

<div class="login-header">

<h1>LIGI Reporting</h1>

<p>Multi-Branch Retail Management System</p>

</div>

<div class="login-card">

<form id="login-form">

<input type="email" id="email" placeholder="Enter your email" required>

<input type="password" id="password" placeholder="Enter your password" required>

<button type="submit">Sign In</button>

</form>

</div>

<div class="demo-accounts">

<h3>Demo Accounts</h3>

<div class="demo-buttons">

<!-- Auto-fill demo account buttons -->

</div>

</div>

</div>

</div>

7.2 Business Owner Dashboard Structure

<div id="business-owner-dashboard" class="page hidden">

<div class="dashboard-header">

<h2>Business Overview</h2>

<p>Welcome back, <span id="user-name"></span></p>

<div class="last-updated">Last updated: <span id="timestamp"></span></div>

</div>

<div class="stats-grid">

<div class="stat-card revenue">

<h3>Total Revenue</h3>

<div class="stat-value">$12,450</div>

<div class="stat-change positive">+18%</div>

</div>

<!-- 3 more stat cards -->

</div>

<div class="main-content">

<div class="left-column">

<div class="branch-performance-card">

<!-- Branch performance list -->

</div>

</div>

<div class="right-column">

<div class="alerts-card">

<!-- Recent alerts -->

</div>

</div>

</div>

<div class="analytics-dashboard">

<div class="tabs">

<button class="tab-btn active" data-tab="revenue">Revenue</button>

<button class="tab-btn" data-tab="departments">Departments</button>

<button class="tab-btn" data-tab="inventory">Inventory</button>

<button class="tab-btn" data-tab="staff">Staff</button>

</div>

<div class="tab-content">

<!-- Dynamic content based on active tab -->

</div>

</div>

</div>

8. Form Implementations

8.1 EOD Report Modal

<div id="eod-report-modal" class="modal hidden">

<div class="modal-content">

<div class="modal-header">

<h3>End of Day Report</h3>

<button class="close-btn">&times;</button>

</div>

<form id="eod-form">

<div class="form-grid">

<div class="form-group">

<label for="butchery-sales">Butchery Sales (₦)</label>

<input type="number" id="butchery-sales" name="butcherySales" step="0.01">

</div>

<!-- More form fields -->

</div>

<div class="form-actions">

<button type="button" class="cancel-btn">Cancel</button>

<button type="submit" class="submit-btn">Submit Report</button>

</div>

</form>

</div>

</div>

9. Development Tasks Breakdown for Cursor AI

Phase 1: Foundation Setup

Task 1: Create Base HTML Structure

- Set up HTML5 document with semantic structure

- Create all page containers with proper IDs

- Add navigation structure

- Implement basic page switching divs

- Add all necessary form modals

Task 2: Implement CSS Design System

- Create CSS variables for colors, spacing, typography

- Implement responsive grid systems

- Style navigation and layout containers

- Create card components and form styles

- Add animation keyframes and transitions

Task 3: Basic JavaScript Framework

- Set up application state management

- Implement page navigation functions

- Create authentication system with demo users

- Add form validation utilities

- Set up local storage functions

Phase 2: Authentication & Navigation

Task 4: Login System

- Style login page with demo account buttons

- Implement login form validation

- Add demo account auto-fill functionality

- Create session management

- Add role-based page redirection

Task 5: Navigation System

- Create responsive navigation bar

- Add user info display (name, role, branch)

- Implement logout functionality

- Add page switching logic

- Style active/inactive states

Phase 3: Business Owner Dashboard

Task 6: Business Overview Stats

- Create stats cards grid layout

- Implement dynamic data binding

- Add trend indicators (arrows, colors)

- Create responsive card layouts

- Add hover effects and animations

Task 7: Branch Performance & Alerts

- Build branch performance list component

- Create color-coded performance indicators

- Implement alerts system with timestamps

- Add interactive elements

- Style alert boxes with appropriate colors

Task 8: Analytics Dashboard

- Create tabbed interface

- Implement tab switching functionality

- Add placeholder content for each tab

- Style active/inactive tab states

- Create responsive tab layout

Phase 4: Operational Features

Task 9: EOD Report System

- Create EOD report modal

- Implement form validation

- Add auto-calculation for totals

- Create success/error handling

- Style form elements consistently

Task 10: Stock Requisition System

- Build requisition form modal

- Implement approval/decline functionality

- Create status tracking system

- Add priority level handling

- Style requisition cards and lists

Task 11: Stores Manager Dashboard

- Create inventory overview layout

- Build supplier delivery tracking

- Implement requisition approval interface

- Add stock level monitoring

- Create interactive approval buttons

Phase 5: Final Polish

Task 12: Responsive Design

- Add mobile breakpoints for all components

- Test and fix layout issues on small screens

- Implement mobile-friendly navigation

- Optimize touch interactions

- Test cross-browser compatibility

10. Implementation Guidelines for Cursor AI

10.1 Code Organization Principles

1. Keep all HTML semantic and accessible

2. Use CSS Grid and Flexbox for layouts

3. Implement vanilla JavaScript with ES6+ features

4. Maintain separation of concerns within single file

5. Use data attributes for JavaScript selectors

6. Implement proper error handling

7. Add loading states and user feedback

8. Follow mobile-first responsive design

10.2 JavaScript Patterns to Follow

// Use module pattern for organization

const App = {

state: { ... },

auth: { ... },

ui: { ... },

data: { ... },

init() { ... }

};

// Event delegation for dynamic content

document.addEventListener('click', function(e) {

if (e.target.matches('.demo-btn')) {

// Handle demo button clicks

}

});

// Use async/await for simulated API calls

async function simulateAPICall(data) {

return new Promise(resolve => {

setTimeout(() => resolve(data), 1000);

});

}

10.3 CSS Best Practices

/\* Use BEM-like naming for clarity \*/

.dashboard\_\_header { }

.dashboard\_\_stats-grid { }

.stat-card--revenue { }

.stat-card\_\_value { }

/\* Implement consistent spacing scale \*/

.spacing-xs { margin: 0.25rem; }

.spacing-sm { margin: 0.5rem; }

.spacing-md { margin: 1rem; }

.spacing-lg { margin: 2rem; }

/\* Use CSS custom properties for theming \*/

.card {

background: var(--bg-white);

border-radius: var(--radius);

box-shadow: var(--shadow);

}

11. Success Criteria

1. Single File Implementation: All functionality contained in one index.html file
2. Role-Based Access: Three distinct user experiences based on demo login
3. Responsive Design: Works seamlessly on desktop, tablet, and mobile
4. Form Functionality: EOD reports and stock requisitions work end-to-end
5. Data Persistence: User sessions and form data saved to localStorage
6. Professional UI: Clean, modern interface matching design specifications
7. No Dependencies: Pure HTML/CSS/JS with no external libraries
8. Cross-Browser: Compatible with modern browsers (Chrome, Firefox, Safari, Edge)

12. File Size Optimization

* Minify CSS and JavaScript in production
* Optimize images and use data URIs for small icons
* Implement lazy loading for non-critical sections
* Use efficient CSS selectors and minimal DOM manipulation
* Target final file size under 500KB for optimal loading

This comprehensive PRD provides all the specifications needed to build the LIGI Reporting system as a single HTML file. Reference specific sections when working with Cursor AI to maintain consistency and context throughout development.