G1 PO App - Definitive Handover Report & Project Plan

Date: May 17, 2025 (Reflecting all progress and chat sessions ending May 17, 2025, CDT) **Version:** 10.0 (This document supersedes all previous handover reports and incorporates all recent development and bug fixes.) **Prepared For:** Incoming Developer **Prepared By:** Mark (via collaboration with AI Assistant Gemini)

Table of Contents

1. Project Goal & Core Value Proposition
2. Current State of the Application (As of May 17, 2025)
    - Key Achievements & Functionalities (from original report)
    - Recent Developments & Enhancements (from this chat)
3. Technology Stack & Project Layout
4. Key Code Files & Their Roles (Highlighting Recent Changes)
5. Developer Setup & Environment
6. Detailed Plan for Completion & Future Tasks (Updated)
7. Known Issues & Considerations (Updated)
8. Recent Bug Fixes & Specific File Updates
9. Conclusion for Handover

---

1. Project Goal & Core Value Proposition

To develop a secure, internal web application ("G1 PO App") that automates and streamlines the purchase order (PO) and shipment process for drop-shipped items. The system ingests orders from BigCommerce, allows authorized users to manage these orders, generate necessary documentation (POs, Packing Slips, Shipping Labels), communicate with suppliers, update BigCommerce, and integrate with QuickBooks Desktop for accounting. A new "G1 Onsite Fulfillment" option has also been introduced for items fulfilled directly from stock. The primary aim is to reduce manual effort, improve accuracy, and provide a centralized platform for this critical business workflow.

2. Current State of the Application (As of May 17, 2025)

The application is substantially functional with many core features implemented and refined. Significant progress has been made on handling complex order scenarios, robust authentication/authorization, integrations with external services, UI enhancements, and the introduction of new fulfillment options.

2.1. Key Achievements & Functionalities (Consolidated from original report v9.0)

- **Order Ingestion & Management:**
    - Successfully ingests relevant orders from BigCommerce (filtered by status, e.g., "Awaiting Fulfillment").
    - Logic includes preservation of manually set statuses and stripping of sensitive patterns from customer notes.
    - Dashboard for viewing/filtering orders (by status) and importing new orders.
    - Detailed order view (OrderDetail.jsx) with original order information, customer details, and line items.
    - Dynamic spare part lookup for HPE option SKUs.
    - Adaptive UI for Single vs. Multi-Supplier PO Fulfillment.
- **Dashboard UI (Dashboard.jsx):**
    - Unified component handling "Orders" view and "Daily Sales Report" view.
    - "Daily Sales Report" integrated, accessible via navigation, fetches data from /api/reports/daily-revenue (UTC day aggregation), displays revenue for the last 14 UTC days with specific styling and timezone handling (hides current UTC day if sales are $0.00).
    - "Products" tab/UI removed from Dashboard.jsx; backend API routes for product mappings still exist.
- **Order Detail UI (OrderDetail.jsx - Original Enhancements):**
    - Prominent "ORDER PROCESSED SUCCESSFULLY!" message upon successful order processing.
    - Formats PO information as "PO # [Number] sent to [Supplier Name]".
- **Backend Processing (app.py - Python/Flask):**
    - Handles order processing requests from the frontend.

- Multi-Supplier PO Logic: Creates distinct Purchase Orders, PO Line Items, and Shipment records.
- Generates unique, sequential PO numbers for supplier POs.
- Database interaction with PostgreSQL on Google Cloud SQL.
- /api/reports/daily-revenue endpoint provides data for the Daily Sales Report.

- **Document Generation (document_generator.py - ReportLab):**
  - Purchase Orders (PDFs): Includes supplier info, items, costs, notes, and a "Partial fulfillment" flag. Company address in PO header (COMPANY_ADDRESS_PO_HEADER) is set to an empty string.
  - Packing Slips (PDFs): Differentiates items in the current shipment vs. items shipping separately. Customer's phone number is no longer included. Item descriptions use "pure" value from hpe_description_mappings.po_description or original BigCommerce name.
- **Shipping Label Generation (shipping_service.py - UPS API):**
  - Generates UPS shipping labels (PDF via GIF conversion) using OAuth 2.0.
  - Handles mapping of shipping method names to UPS service codes.
- **Email Communication (email_service.py - Postmark API):**
  - Emails PO PDF, Packing Slip PDF, and UPS Label PDF (if generated) to suppliers.
  - Sends daily IIF batch emails. send_quickbooks_data_email function (if still present) is no longer called by app.py.
- **Cloud Storage (app.py - Google Cloud Storage):**
  - Uploads generated POs, Packing Slips, and Labels to GCS. Signed URLs are used for accessing these documents.
- **BigCommerce Integration (shipping_service.py, app.py):**
  - Retrieves order data.
  - Refined logic for shipment creation and final order status updates.
- **QuickBooks Integration (iif_generator.py, app.py):**
  - Generates and emails a daily IIF file for Purchase Orders.

    - Automated triggering via Google Cloud Scheduler.
- **Authentication & Authorization (Firebase):**
    - Robust user authentication via Firebase (Google Sign-In, project g1-po-app-77790).
    - Authorization using Firebase Custom Claims (isApproved: true).
    - Backend API routes protected using a @verify_firebase_token decorator.
    - Correct Firebase configuration and API key usage in frontend verified.
- **GCS Signed URL IAM Permission**: "Service Account Token Creator" role for the Cloud Run service account for signed URL generation identified and instructed.
- **Routing (App.jsx):**
    - Main navigation bar links "Daily Sales" to /dashboard/sales.
    - /dashboard (and /) renders Dashboard component with initialView="orders".
    - /dashboard/sales route renders Dashboard component with initialView="dailySales".
    - Product-related routes removed.
- **Error Handling and Logging**: Implemented at various levels.

2.2. Recent Developments & Enhancements (From This Chat Session - Post May 16, 2025)

- **OrderDetail.jsx Hyperlink Behavior Update:**
    - Part number links (for original_sku, spare_sku, hpe_option_pn) in the "Order Information" section now first copy the main Order Number (e.g., order.bigcommerce_order_id) to the clipboard.
    - After copying, they open the Brokerbin URL in a new window.
    - The existing logic to update the order status to "RFQ Sent" (if the order status is 'new') when these links are clicked has been preserved within the new handlePartNumberLinkClick event handler.
- **OrderDetail.jsx Conditional "Use Multiple Suppliers" Option:**

    - The "** Use Multiple Suppliers **" option in the supplier selection dropdown now only appears if the order contains more than one line item (originalLineItems.length > 1).
    - A useEffect hook has been added to reset the multi-supplier mode if an order no longer qualifies (e.g., if line items were theoretically reduced to one or zero after the mode was selected).
- **OrderDetail.jsx "G1 Onsite Fulfillment" Feature (Frontend Implementation):**
    - A new option, "G1 Onsite Fulfillment," has been added to the supplier selection dropdown in OrderDetail.jsx. This option always appears at the bottom of the functional choices (after mapped suppliers and the conditional multi-supplier option), provided there are items in the order.
    - A new state variable isG1OnsiteFulfillmentMode controls the UI.
    - When "G1 Onsite Fulfillment" is selected:
        - The UI simplifies to only show the "Shipment Information" card (shipping method and weight) for data collection.
        - Forms/sections related to supplier POs (PO Notes, editable "Items to Purchase" for PO) are hidden.
    - handleMainSupplierTriggerChange was updated to manage these state transitions.
    - handleProcessOrder was updated to prepare a distinct payload for the backend when this mode is active, using a special supplier_id (G1_ONSITE_FULFILLMENT_IDENTIFIER). This payload includes shipment details but an empty po_line_items array.
- **app.py "G1 Onsite Fulfillment" Feature (Backend Implementation):**
    - The /api/orders/<int:order_id>/process route was modified to detect the G1_ONSITE_FULFILLMENT_IDENTIFIER.
    - If G1 Onsite Fulfillment is detected:
        - Purchase Order creation (DB records, PO PDF) is skipped. PO numbering sequence is not affected.
        - A Packing Slip PDF is generated using all original order line items.
        - A UPS Label PDF is generated if shipment information is provided.

        - The Packing Slip and UPS Label are uploaded to GCS, and signed URLs are generated.
        - These documents are emailed to sales@globalonetechnology.com (requires email_service.py to have a suitable function like send_sales_notification_email or a generic one that can be adapted).
        - The local order status in the orders table is updated to "Completed Offline".
        - The BigCommerce order status is updated to the standard "Shipped" status (using the existing bc_shipped_status_id environment variable).
        - A shipment record, including the tracking number, is created in BigCommerce for all original order items.
        - A shipment record is inserted into the local shipments table. **Crucially, this requires the purchase_order_id column in the shipments table to be nullable, and the order_id column should be populated.**
    - The overall order status logic at the end of process_order was adjusted to correctly handle cases that were exclusively G1 Onsite Fulfillment.
- **document_generator.py Updates:**
    - The generate_packing_slip_pdf function signature was updated to include an optional is_g1_onsite_fulfillment=False parameter.
    - The title of the packing slip is now "PACKING SLIP" by default and does not include "(G1 Onsite Fulfillment)" unless explicitly re-added using this new flag.
- **email_service.py Updates (Required):**
    - A new function, send_sales_notification_email(recipient_email, subject, html_body, text_body, attachments), needs to be implemented (or an existing generic function verified/adapted) to send G1 Onsite Fulfillment notifications (packing slip, label) to sales@globalonetechnology.com. app.py now calls this function.

    - The send_quickbooks_data_email function (if still present and previously intended for removal) should be confirmed as unused by app.py's process_order route.
- **shipping_service.py**:
    - No changes were required in this file; existing functions are used by app.py.
- **Bug Fixes Implemented During This Chat:**
    - Added import base64 to app.py.
    - Corrected document_generator.py to accept the is_g1_onsite_fulfillment argument in generate_packing_slip_pdf.
    - Corrected app.py to call the appropriate email function in email_service.py (e.g., send_sales_notification_email instead of a non-existent send_generic_email) for G1 Onsite notifications.
    - Ensured the "Completed Offline" local status for G1 Onsite Fulfillment is set correctly, while the BigCommerce status uses the standard "Shipped" ID.

3. Technology Stack & Project Layout

(This section is largely the same as original report v9.0, with minor clarifications if applicable from recent work)

- **Backend (order-processing-app directory):**
    - Python 3.9+, Flask, SQLAlchemy, pg8000, Gunicorn.
    - Key files: app.py, shipping_service.py, email_service.py, document_generator.py, iif_generator.py.
    - Dependencies: requirements.txt.
    - Configuration: .env file for local, Google Cloud Secret Manager for deployed.
- **Frontend (order-processing-app-frontend directory):**
    - React (Vite), react-router-dom, Firebase Client SDK, Workspace API.
    - Key components: App.jsx, Dashboard.jsx, OrderDetail.jsx, Login.jsx, AuthContext.jsx, ProtectedRoute.jsx, SupplierList.jsx, etc.
    - Configuration: .env.development, .env.production for VITE_API_BASE_URL and Firebase SDK config.

- **Database:** PostgreSQL on Google Cloud SQL.
    - **Schema Note for shipments table**: The purchase_order_id column **must be made nullable** to support G1 Onsite Fulfillment shipments which do not have an associated purchase order. The order_id column should be populated for all shipments.
- **Cloud Platform (GCP):**
    - Cloud Run (backend).
    - Cloud SQL (database).
    - Artifact Registry (Docker images).
    - Secret Manager (secrets).
    - Cloud Storage (GCS for documents).
    - Cloud Scheduler (IIF task).
- **Firebase:**
    - Firebase Hosting (frontend).
    - Firebase Authentication (user auth for project g1-po-app-77790).
- **External APIs:**
    - UPS API (shipping labels).
    - BigCommerce API (orders).
    - Postmark API (emails).

4. Key Code Files & Their Roles (Highlighting Recent Changes)

- **app.py (Backend Flask application):**
    - Provides all API endpoints.
    - /api/orders/<order_id>/process: Major updates to handle the new "G1 Onsite Fulfillment" scenario alongside existing single and multi-supplier PO generation. This includes conditional logic to:
        - Skip PO creation for G1 Onsite.
        - Generate appropriate documents (Packing Slip for G1 Onsite, PO & Packing Slip for suppliers).

        - Route emails correctly (to sales@globalonetechnology.com for G1 Onsite, to supplier for POs).
        - Update local and BigCommerce order statuses distinctly ("Completed Offline" locally for G1 Onsite, "Shipped" on BigCommerce for G1 Onsite; "Processed" locally for supplier POs, "Shipped" on BigCommerce if all items covered).
        - Handle GCS uploads for all generated documents.
    - /api/reports/daily-revenue: Aggregates sales data by UTC day.
    - Contains logic for order ingestion, database interactions, and calls to service modules.
- **OrderDetail.jsx (Frontend):**
    - Displays details for a single order and handles the PO/fulfillment processing form.
    - **Recent Changes:**
        - Modified part number hyperlinks to first copy Order# then open Brokerbin.
        - Conditional rendering of the "** Use Multiple Suppliers **" dropdown option based on line item count.
        - Addition of "G1 Onsite Fulfillment" option in the supplier/mode dropdown.
        - Conditional UI to show only shipment info fields when "G1 Onsite Fulfillment" is selected.
        - Updated handleProcessOrder to send a distinct payload for G1 Onsite Fulfillment.
        - Displays "ORDER PROCESSED SUCCESSFULLY!" upon success.
        - Formats PO information correctly (for supplier POs).
- **Dashboard.jsx (Frontend):**
    - Displays the main order list and the "Daily Sales Report".
    - Conditionally renders views based on the initialView prop.

  - Daily Sales Report: Displays UTC-dated revenue, hides today's UTC date if sales are zero, styles revenue numbers.
- **App.jsx (Frontend):**
  - Manages top-level application routing. Navigation bar links "Daily Sales" to /dashboard/sales. Product-related routes removed.
- **document_generator.py (Backend):**
  - Generates PO and Packing Slip PDFs using ReportLab.
  - generate_packing_slip_pdf updated to accept an optional is_g1_onsite_fulfillment flag to allow for minor template adjustments (e.g., title). By default, the title is now just "PACKING SLIP".
  - COMPANY_ADDRESS_PO_HEADER global variable remains "".
  - Customer phone number removed from the "Ship To" address on packing slips.
  - Relies on app.py to pass "pure" item descriptions for packing slip items.
- **email_service.py (Backend):**
  - Handles sending emails via Postmark.
  - **Needs a new function** (e.g., send_sales_notification_email) or adaptation of a generic function to send G1 Onsite Fulfillment notifications (Packing Slip, Label) to sales@globalonetechnology.com. app.py has been updated to call this.
  - The send_quickbooks_data_email function (if still present) is not called by app.py's process_order.
- **shipping_service.py (Backend):**
  - No changes were required for G1 Onsite Fulfillment. Existing functions generate_ups_label, create_bigcommerce_shipment, and set_bigcommerce_order_status are used by app.py.
- **Frontend Contexts/Components:** AuthContext.jsx, ProtectedRoute.jsx, etc.

5. Developer Setup & Environment

(This section is largely the same as original report v9.0)

- **Backend (order-processing-app directory):**
  - Python 3.9+.
  - Virtual environment (e.g., python -m venv venv).
  - Install dependencies: pip install -r requirements.txt.
  - Local .env file for secrets (DB credentials, API keys, GCS bucket name, Firebase Project ID, etc.).
  - GOOGLE_APPLICATION_CREDENTIALS environment variable pointing to a Firebase Admin SDK service account key JSON file for local development. This service account key *must* belong to the Firebase project (g1-po-app-77790) and have necessary permissions.
  - Google Cloud SQL Auth Proxy for connecting to the Cloud SQL PostgreSQL database locally.
- **Frontend (order-processing-app-frontend directory):**
  - Node.js (LTS version recommended).
  - Install dependencies: npm install (or yarn install).
  - Firebase project configuration snippet (API key, auth domain, etc.) must be correctly set up (e.g., via Vite environment variables in .env.development / .env.production).
  - .env.development and/or .env.local file(s) in the frontend project root for Vite:
    - VITE_API_BASE_URL (e.g., http://127.0.0.1:8080/api for local backend).
    - VITE_FIREBASE_API_KEY, VITE_FIREBASE_AUTH_DOMAIN, VITE_FIREBASE_PROJECT_ID, etc.
- **Running Locally:**
  - Start backend: flask run --port=8080 (or python app.py).
  - Start frontend: npm run dev.
- **Deployment:**
  - Backend: Dockerized, pushed to Google Artifact Registry, deployed to Google Cloud Run. The Cloud Run service account needs "Service Account Token

Creator" role on itself for GCS Signed URLs, and appropriate permissions for Cloud SQL, GCS (read/write to bucket), Secret Manager, etc.

  - Frontend: Built with npm run build, deployed to Firebase Hosting.

6. Detailed Plan for Completion & Future Tasks (Updated)

This plan integrates tasks from the original report (v9.0) and new considerations arising from recent developments.

Immediate Next Steps (Post-Handover - Critical):

1. **Task: Database Schema Update for shipments Table.**
   - **Action**: Modify the shipments table in your PostgreSQL database.
     - Make the purchase_order_id column **nullable**.
     - Ensure the order_id column is present and correctly serves as a foreign key to the orders table.
   - **Reason**: This is crucial for the "G1 Onsite Fulfillment" feature to correctly log shipments that do not have an associated supplier Purchase Order. The current app.py logic for inserting G1 Onsite shipments assumes this schema.
   - **File(s) Involved**: Database schema (SQL modification).
2. **Task: Implement/Verify send_sales_notification_email in email_service.py.**
   - **Action**: Ensure a function (e.g., send_sales_notification_email or a suitably adapted generic function) exists in email_service.py that can send an email with attachments to sales@globalonetechnology.com. app.py is now configured to call this for G1 Onsite Fulfillment. The function needs to handle attachments correctly (e.g., base64 encoding for Postmark if using the library directly).
   - **File(s) Involved**: email_service.py, app.py (to ensure the call matches).
3. **Task: Comprehensive End-to-End Testing of All Fulfillment Flows.**
   - **Action**: After the above changes, thoroughly test:
     - **Single Supplier PO Workflow**: UI, backend, PO PDF, Packing Slip PDF, Label generation, supplier email, GCS uploads, BigCommerce shipment/status updates, local DB updates.

        - **Multi-Supplier PO Workflow**: All aspects as above, ensuring distinct POs and documents are handled correctly.
        - **G1 Onsite Fulfillment Workflow**: UI (simplified form), backend (no PO created), Packing Slip PDF, Label generation, email to sales@globalonetechnology.com, GCS uploads, BigCommerce shipment/status updates ("Shipped"), local DB status update ("Completed Offline").
    - Test edge cases (e.g., order with no items for G1 Onsite Fulfillment, if applicable).
    - Verify all document content and signed URL generation/access.
    - **File(s) Involved**: Full application stack.

4. **Task: Verify Backend Signed URL Generation and Frontend Consumption (from original report).**
    - **Action**: Confirm "Service Account Token Creator" IAM role is active for Cloud Run. Deploy latest app.py and OrderDetail.jsx. Process an order and verify PO/document links in the UI.
    - **File(s) Involved**: app.py, OrderDetail.jsx, GCP IAM.
5. **Task: Verify Customer-Facing SKU/Description on Packing Slip (from original report).**
    - **Action**: Test app.py logic for prepared_items_in_this_shipment and prepared_items_shipping_separately to ensure packing slip descriptions are always the desired "pure" customer-facing ones.
    - **File(s) Involved**: app.py, document_generator.py.

Phase 2: MVP Task Completion (from original report, re-prioritize as needed)

6. **Task: PO Export Feature (Excel - Backend & Frontend).**
    - Backend (app.py): Implement GET /api/exports/pos using openpyxl.
    - Frontend (Dashboard.jsx): Add UI to trigger export.
7. **Task: Finalize Authentication & Authorization Robustness.**
    - Frontend error handling for 401/403.

  - Documentation for setAdminClaim.cjs (or prioritize Admin UI for claim management).
  - Re-verify Cloud Run IAM permissions.

Phase 3: UI/UX Enhancements & Remaining Fixes (from original report, re-prioritize as needed)

8. **Task: Supplier and Product/Mapping Management UI.**
   - Implement React components for CRUD operations on suppliers.
   - Re-evaluate need/priority for UI for hpe_part_mappings, hpe_description_mappings, qb_product_mapping, products. If needed, design an entry point (e.g., Admin section).
9. **Task: Frontend - Eloquia Font Issue.**
   - Resolve font rendering on the deployed site.
10. **Task: Loading Indicators and UX Refinements.**
    - Audit API-calling components for consistent loading indicators.
    - Review UX of the new G1 Onsite Fulfillment flow in OrderDetail.jsx.

Phase 4: Infrastructure & Deployment Finalization (from original report, re-prioritize as needed)

11. **Task: Postmark - External Domain Sending Approval.**
    - Ensure Postmark is fully configured (DKIM, SPF, Custom Return-Path) for reliable email delivery.
12. **Task: Cloudflare DNS for Production Domains.**
    - Configure g1po.com (frontend) and api.g1po.com (backend).

Phase 5: Long-Term Improvements & Best Practices (from original report, re-prioritize as needed)

13. **Task: Backend Daily Revenue Aggregation Timezone.**
    - If business reporting requires alignment with a specific local timezone (e.g., CDT) for the Daily Sales Report, refactor the backend query in app.py (/api/reports/daily-revenue) to group revenue by that target timezone's day boundaries.

14. **Task: Security Hardening.**
    - Cloud Run Ingress (consider IAP).
    - Comprehensive server-side input validation for all endpoints.
    - Review and enhance structured, detailed, leveled logging.
15. **Task: Asynchronous Task Processing.**
    - For /api/orders/<id>/process, especially document generation and external API calls, consider using Google Cloud Tasks to improve API response times and robustness.
16. **Task: Admin User Approval UI.**
    - Implement an admin interface for managing isApproved Firebase custom claims, replacing the manual setAdminClaim.cjs script.
17. **Task: Data Integrity for QuickBooks Lists (TERMS and SHIPVIA).**
    - Ensure data used for IIF generation matches QuickBooks lists precisely.
18. **Task: Automated Unit/Integration Tests.**
    - Implement a testing suite using Pytest (backend) and Jest/React Testing Library (frontend).

7. Known Issues & Considerations (Updated)

- **shipments Table Schema**: As highlighted, purchase_order_id **must be made nullable** to support G1 Onsite Fulfillment. This is a **critical immediate task**.
- **IIF Import Sensitivities**: QuickBooks Desktop IIF import is sensitive. Thorough testing of generated IIF files is crucial.
- **UPS Production Costs**: Live UPS API will incur costs. Ensure proper testing in the UPS test environment before going live.
- **Postmark Deliverability**: DNS setup (DKIM, SPF, Return-Path) is critical for email deliverability.
- **Scalability of Synchronous Processing**: Current order processing in /api/orders/<id>/process is synchronous. For higher volumes, consider moving to asynchronous tasks (see Phase 5).

- **Environment Management**: Maintain strict separation and configuration for development, staging (if any), and production environments.
- **Custom Claim Management**: Admin UI for Firebase custom claims (isApproved) is a future task; currently manual.
- **Firebase & GCP Quotas**: Monitor usage of Firebase services and GCP resources.
- **Service Account Key Security**: Protect any downloaded service account key JSON files diligently. Use them only for local development and prefer workload identity federation or service account impersonation in deployed environments where possible.
- **Timezone for Reporting**: The Daily Sales Report currently uses UTC day aggregation. Consider backend changes for local business day alignment if required by the business.

8. Recent Bug Fixes & Specific File Updates (During this chat)

- **app.py:**
    - Added import base64.
    - Integrated G1 Onsite Fulfillment logic into /api/orders/<id>/process.
    - Corrected calls to email_service to use an appropriate function (e.g., send_sales_notification_email) instead of a non-existent one.
- **document_generator.py:**
    - Modified generate_packing_slip_pdf to accept is_g1_onsite_fulfillment parameter.
    - Removed "(G1 Onsite Fulfillment)" from the default packing slip title; the new flag can be used for any such conditional text.
- **OrderDetail.jsx:**
    - Implemented hyperlink modification (copy Order# then open Brokerbin).
    - Implemented conditional rendering of "Use Multiple Suppliers" option.
    - Implemented "G1 Onsite Fulfillment" UI and logic.
- **email_service.py**:

    - Requires addition/verification of a function like send_sales_notification_email to be called by app.py for G1 Onsite notifications.

9. Conclusion for Handover

The G1 PO App has evolved significantly and now includes enhanced UI features and more flexible fulfillment options. The immediate next steps, particularly the database schema update for the shipments table and verification of the email service function for G1 onsite notifications, are critical for the stability of the newly added features. This document provides a comprehensive snapshot of the project's current state, recent changes, and a clear path forward for continued development and refinement.