This detailed handover report covers the "G1 PO App" project, including its current state, architecture, progress made during our recent interactions, and a plan for completion. It incorporates information from the prior "Handover Report 20250520b.pdf" and details from our chat sessions.

---

1. Project Goal & Core Value Proposition

The primary goal is to develop the "G1 PO App," a secure, internal web application to automate and streamline the purchase order (PO) and shipment processes for both drop-shipped and G1-stocked items. This application aims to minimize manual effort, enhance accuracy, serve as a central platform for critical business workflows, and improve overall operational efficiency. Key functionalities include ingesting orders from BigCommerce, managing orders through a web interface, generating POs, packing slips, and shipping labels (UPS & FedEx), facilitating supplier communication, updating BigCommerce with shipment details, and integrating with QuickBooks Desktop via IIF files.

---

2. Current State of the Application (As of May 22, 2025)

The application has a significant number of core features implemented. A major backend refactoring introduced Flask Blueprints for better organization and scalability. The refactored backend has been successfully deployed to Google Cloud Run after resolving various deployment and environment issues.

Key Achievements & Functionalities:

- **Order Management**: Robust BigCommerce order ingestion, including parsing for third-party shipping accounts and storing billing/shipping cost data.
- **Fulfillment Modes**: Supports Single and Multi-Supplier POs, G1 Onsite Fulfillment, and customer-billed shipments.
- **Document Generation**: PDF generation for Purchase Orders and Packing Slips. The packing slip can now be generated in a "blind drop ship" version, omitting company branding and using the customer's billing address as the ship-from on the label.
- **Shipping Integration**:
    - **UPS**: Functional "Bill Sender" and "Bill Third Party" label generation.
    - **FedEx**: OAuth is working. "Bill SENDER" labels have been tested in production (test-marked). "Bill RECIPIENT" functionality is implemented in

the code but currently encounters an "SHIPMENT.ACCOUNTNUMBER.UNAUTHORIZED" error from the FedEx API when attempting to bill a recipient's account. This was a known blocker in sandbox testing and requires FedEx support/configuration to resolve for production accounts. FedEx production API credentials are set up and selected based on the environment.

- **Communication**: Email communications for POs to suppliers via Postmark.
- **BigCommerce Updates**: Integration for order data retrieval and updating order status/shipment details.
- **QuickBooks Integration**: Extensive QuickBooks Desktop IIF file generation for Purchase Orders, Sales Orders (as Invoices), and corresponding Payments. Includes US Central Time conversion and specific formatting requirements. The IIF generator was updated to handle SKUs with underscores for product mapping lookups.
- **Authentication**: Firebase Authentication (Google Sign-In with an isApproved custom claim) is implemented for user access control.
- **Frontend**: Features a Dashboard, a detailed OrderDetail.jsx page for order processing, and a Utilities section. Dark mode CSS is implemented. OrderDetail.jsx has been updated to include FedEx shipping options, UI for "Bill to Customer's FedEx Account," and a "Blind Drop Ship?" option. Layout adjustments for the "Shipment Information" section and "Process PO" button centering were made.
- **Backend Refactoring**: API route logic was moved from the main app.py into Flask Blueprints for improved modularity. app.py now handles core setup, shared resources, and blueprint registration.

Recent Critical Issue Resolved (Partially):

- An Uncaught ReferenceError: currentSelectedShippingOption is not defined in OrderDetail.jsx was causing the page to go blank upon supplier selection. This was addressed by making the getSelectedShippingOption function more robust (always returning an object) and ensuring currentSelectedShippingOption is correctly derived and available in the render scope.

---

3. Technology Stack & Project Layout

(Largely as per "Handover Report 20250520b.pdf", with updates reflected)

- **Backend (order-processing-app directory)**:
  - Language/Framework: Python 3.9+, Flask
  - WSGI Server: Gunicorn
  - Key Libraries: SQLAlchemy, google-cloud-sql-connector[pg8000], requests, python-dotenv, ReportLab, Pillow, google-cloud-storage, firebase-admin, postmarker, pytz.
  - **Updated Directory Structure**:
    - app.py: Main Flask app, shared resources, blueprint registration.
    - blueprints/: Contains individual Python files for each blueprint (e.g., orders.py, suppliers.py, quickbooks.py, hpe_mappings.py, reports.py, utils_routes.py).
    - Service Modules: document_generator.py, email_service.py, iif_generator.py, shipping_service.py.
    - Configuration: .env, requirements.txt, Dockerfile, entrypoint.sh.
- **Frontend (order-processing-app-frontend directory)**:
  - Framework/Library: React (Vite build tool)
  - Routing: react-router-dom
  - Authentication: Firebase Client SDK
  - Styling: Standard CSS, component-specific CSS (e.g., OrderDetail.css, App.css).
- **Database**: PostgreSQL on Google Cloud SQL.
  - A new column customer_shipping_country_iso2 VARCHAR(2) was added to the orders table to resolve an ingestion error.
- **Cloud Platform (GCP)**: Cloud Run, Cloud SQL, Artifact Registry, Secret Manager, Cloud Storage, Cloud Scheduler, Cloud Logging, IAM.
- **Authentication**: Firebase Authentication (Google Sign-In, Custom Claims: isApproved: true).
- **External APIs**: BigCommerce API, UPS API (OAuth 2.0), FedEx API (OAuth 2.0), Postmark API.

---

4. Key Code Files & Their Roles

- **Backend (order-processing-app)**:
    - **app.py**: Core application setup, initializes Flask, configurations, database, GCS, Firebase Admin SDK. Defines shared helpers (e.g., convert_row_to_dict, verify_firebase_token). Registers all blueprints. Contains root routes (/, /test_db).
    - **blueprints/orders.py**: Handles order listing, details, status updates, BigCommerce order ingestion, and the main order processing logic (process_order_route) which includes PO creation, document generation calls, shipping label generation calls, and BigCommerce updates. This file was updated to include FedEx label generation logic, handle a "Blind Drop Ship" flag (changing ship-from address and packing slip type), and enforce that supplier emails are only sent if all three documents (PO, Packing Slip, Label - if label attempted) are generated.
    - **blueprints/suppliers.py**: CRUD operations for suppliers.
    - **blueprints/hpe_mappings.py**: Manages HPE Part Number to PO Description mappings and SKU lookups.
    - **blueprints/quickbooks.py**: Endpoints for triggering QuickBooks IIF file generation (daily, today's, on-demand for all pending POs and Sales).
    - **blueprints/reports.py**: For generating reports like the daily revenue report.
    - **blueprints/utils_routes.py**: Utility endpoints, e.g., standalone UPS label generator.
    - **shipping_service.py**: Contains logic for interacting with UPS and FedEx APIs to get shipping rates (future) and generate shipping labels. Handles OAuth for both. FedEx logic selects production credentials based on FEDEX_API_ENVIRONMENT.
    - **document_generator.py**: Generates PDF documents (Purchase Orders, Packing Slips). Updated to accept is_blind_slip and custom_ship_from_address parameters to generate generic packing slips and conditionally omit the company logo/footer for blind drop shipments. The packing slip font was changed to Eloquia.

    - **email_service.py**: Handles sending emails (e.g., POs to suppliers) via Postmark.
    - **iif_generator.py**: Contains all logic for generating IIF files for QuickBooks Desktop (POs, Sales Orders/Invoices, Payments). Updated to handle SKUs with underscores by looking up the part after the underscore if a direct match fails.
    - Dockerfile & entrypoint.sh: Configure the Docker container and how Gunicorn runs the Flask app in Cloud Run.
- **Frontend (order-processing-app-frontend)**:
    - **OrderDetail.jsx**: The primary component for viewing and processing individual orders. This file has seen the most changes during our recent sessions:
        - Addition of FedEx shipping methods and UI to specify "Bill to Customer's FedEx Account" and their account number.
        - Addition of a global "Blind Drop Ship?" checkbox.
        - Layout adjustments for the "Shipment Information" section to improve clarity and organization of shipping method, weight, and blind ship option.
        - Layout adjustments to center the "Process PO" button.
        - Fixes for the "page goes blank" error by making getSelectedShippingOption more robust.
    - **OrderDetail.css**: Styles for OrderDetail.jsx, updated to support dark mode and the new layout adjustments for shipment information. Addressed "empty ruleset" linter warnings by removing unused styles.
    - **App.css**: Global styles, including dark mode variables and base styling.

---

5. Database Schema Notes & Key Tables

(Refer to "Handover Report 20250520b.pdf," Section 5, pages 13-15 for the original detailed schema)

- **orders table**:

    - Added customer_shipping_country_iso2 VARCHAR(2): Stores the 2-letter ISO code for the shipping country.
    - The customer_shipping_country column should store the full country name for display purposes.
- QuickBooks sync status columns (qb_po_sync_status, qb_sales_order_sync_status, etc.) are crucial for IIF generation logic.

---

6. Developer Setup & Environment

(Refer to "Handover Report 20250520b.pdf," Section 6, pages 16-18 for base setup)

- **Backend**:
    - Python 3.9+ virtual environment is essential. Resolved "Fatal error in launcher" on Windows by recreating venv.
    - PowerShell Execution Policy: May need Set-ExecutionPolicy -ExecutionPolicy RemoteSigned -Scope CurrentUser.
    - Gunicorn on Windows: Use Flask's dev server (flask run) or Waitress for local WSGI testing due to fcntl error.
    - Line Endings: Ensure .sh files use LF (Unix) line endings.
- **Frontend**: Standard npm install and npm run dev/npm run build.
- **Deployment**:
    - Backend to Cloud Run via Docker: Use gcloud auth configure-docker, docker build, docker tag, docker push, and gcloud run deploy with appropriate environment variables and secrets.
    - Frontend to Firebase Hosting: npm run build, then firebase deploy --only hosting.
- **Crucial Environment Variables**: Extensive list in previous report and user's deploy scripts. For frontend, VITE_API_BASE_URL and Firebase SDK config are key. Ensure all SHIP_FROM_* variables, database credentials, API keys (BigCommerce, UPS, FedEx, Postmark), and GCS bucket names are correctly set in .env for local development and as secrets/env vars in Cloud Run.

---

7. Summary of Recent Interactions (This Chat Session - May 21-22, 2025)

- **FedEx Integration Advancement**:
  - Reviewed existing FedEx "Bill Recipient" logic in shipping_service.py.
  - Updated OrderDetail.jsx to include FedEx shipping methods, UI for selecting "Bill to Customer's FedEx Account," and input for the account number.
  - Modified blueprints/orders.py (process_order_route) to:
    - Receive carrier information (UPS/FedEx) and FedEx billing details from the frontend.
    - Conditionally call shipping_service.generate_fedex_label or shipping_service.generate_ups_label.
    - Pass necessary flags (is_bill_to_customer_fedex_account, customer_fedex_account_number) to the FedEx label function.
  - Diagnosed FedEx API error SHIPMENT.ACCOUNTNUMBER.UNAUTHORIZED for "Bill Recipient," confirming it requires FedEx account configuration/authorization.
- **QuickBooks IIF Generator Update**:
  - Modified iif_generator.py in the get_qb_item_name_for_option_pn function to handle SKUs containing underscores (e.g., EG001200JWJNQ_872479-B21) by attempting a lookup on the part of the SKU after the last underscore if the direct lookup fails.
- **Order Processing Enhancements**:
  - Modified blueprints/orders.py (process_order_route) to prevent sending supplier emails and updating BigCommerce order status if not all three documents (PO PDF, Packing Slip PDF, Shipping Label PDF) are generated for a supplier PO where a label was attempted. An error is raised instead.
- **Blind Drop Ship Functionality**:
  - OrderDetail.jsx: Added a global "Blind Drop Ship?" checkbox.
  - blueprints/orders.py: Updated process_order_route to:
    - Receive the is_blind_drop_ship flag.

        - If true, use the order's *billing address* as the ship_from_address for label generation.
        - Pass flags to document_generator.py to produce a generic packing slip.
    - document_generator.py: Modified generate_packing_slip_pdf to:
        - Accept is_blind_slip and custom_ship_from_address parameters.
        - Omit company logo and use generic/custom ship-from details on the packing slip if is_blind_slip is true.
        - Conditionally omit the company-specific footer.
- **Font Change**: Updated document_generator.py to use "Eloquia" fonts (already registered) for the Packing Slip instead of Helvetica.
- **Frontend Layout Adjustments (OrderDetail.jsx & OrderDetail.css)**:
    - Addressed several iterations of layout requests for the "Shipment Information" section, specifically for "Method," "Weight (lbs)," and "Blind Ship?" elements for both desktop and mobile views.
    - Adjusted CSS to center the "Process PO" button.
- **Database Schema Update**: Added customer_shipping_country_iso2 VARCHAR(2) to the orders table to fix an ingestion error.
- **Critical Bug Fix (OrderDetail.jsx)**: Resolved a Uncaught ReferenceError: currentSelectedShippingOption is not defined that caused the page to go blank on supplier selection. This involved making the getSelectedShippingOption helper function more robust (always returning an object) and ensuring it was correctly utilized by currentSelectedShippingOption.

---

8. Detailed Plan for Completion & Future Tasks

This plan integrates tasks from the previous "Handover Report 20250520b.pdf" with new items and progress from our recent sessions.

Phase 0: Immediate Post-Handover Validation & Critical Fixes *(Some of these were in the previous report and remain critical)*

1. **Thorough API Testing (Critical)**:

   - **Action**: Systematically test all API endpoints (GET, POST, PUT for orders, suppliers, HPE mappings; POST for QuickBooks triggers, label generation, order processing) using the deployed frontend.
   - **Goal**: Confirm the backend refactoring into blueprints and recent changes (FedEx, Blind Drop Ship, IIF underscore fix, strict document check) work as expected without regressions. Verify data handling, document generation, and external API calls.

2. **Monitor Cloud Run Logs (Ongoing)**:
   - **Action**: While testing, continuously monitor Cloud Run logs for the deployed backend service.
   - **Goal**: Ensure application stability, identify any new runtime errors, unexpected behavior, or performance issues.
3. **Resolve Firebase auth/quota-exceeded Error (Critical if still occurring)**:
   - **Action**: Check Google Cloud Console for "Identity Platform" / "Firebase Authentication" quotas.
   - **Goal**: If quotas are the issue, upgrade the Firebase project to the "Blaze" (pay-as-you-go) plan to ensure reliable authentication.
4. **Review & Tune Gunicorn Production Settings**:
   - **Action**: Current entrypoint.sh uses debug settings. After testing confirms stability, adjust for production (e.g., increase threads/workers, set log-level to info or warning).
   - **Goal**: Optimized and stable Gunicorn configuration.
5. **Frontend Bug Squashing (Post currentSelectedShippingOption fix)**:
   - **Action**: After applying the fix for currentSelectedShippingOption, thoroughly test all supplier/mode selections in OrderDetail.jsx to ensure no more "blank page" errors occur. Check browser console for any other JS errors.
   - **Goal**: Stable OrderDetail.jsx component during dynamic state changes.

Phase 1: Frontend UI/UX Refinement (OrderDetail.jsx)

1. **Finalize Shipment Information Layout (Desktop & Mobile)**:

- **Action**: Implement the latest discussed CSS and JSX for the "Method", "Weight (lbs)", and "Blind Ship?" fields to achieve the desired layout:
  - Desktop: "Method" on its own row. "Weight (lbs)" label + input and "Blind Ship?" label + checkbox on the next row, sharing space as requested.
  - Mobile: Ensure these elements stack cleanly and logically.
- **Goal**: User-approved layout for shipment inputs.

2. **Item Purchase Grid Layout - "Unit Cost" (Desktop)**:
   - **Action**: Adjust CSS for .item-row or .qty-cost-row within the "Items to Purchase" section to ensure the "Unit Cost" input field appears directly under its "Unit Cost" label, similar to how "Qty" is structured.
   - **Goal**: Consistent and clear layout for item entry.
3. **Mobile View - Status Alignment**:
   - **Action**: Verify the order status badge aligns correctly with the order title on mobile views. Adjust CSS if necessary.
   - **Goal**: Improved mobile layout.
4. **Desktop View - Address Duplicate Status / Layout Transformation**:
   - **Action**: Clarify with the user which "table" layouts need conversion to cards and where any duplicate statuses appear. Implement a responsive multi-column card layout.