**InventoryRPA Process Definition Document (PDD)**

Project: Inventory Management RPA  
Author: Bruce Brown  
Date: 7/22/2025

**1. Business Process Overview**

Retail Innovations Inc.’s inventory team currently performs these steps each day:

1. Warehouse staff manually count on-hand inventory in each location (WH1, WH2, WH3).
2. Counts are keyed into a shared Excel workbook (inventory\_counts.xlsx).
3. The workbook is emailed to the inventory manager.
4. Manager aggregates counts, applies reorder logic in Excel formulas, and highlights low-stock SKUs.
5. Manager composes a manual email to the purchasing team with items below the reorder point.

**Pain Points:**

* Manual data entry introduces typos and duplicates (≈ 5 % error rate).
* Aggregation formulas break when columns shift.
* Notification emails are delayed by 1–2 hours.
* No audit trail of who changed what and when.

**2. Automation Objectives**

1. **Accuracy**
   * Eliminate negative and duplicated quantity errors, targeting ≥ 80 % error reduction.
2. **Speed**
   * Reduce end-to-end process time from ~ 45 minutes to < 5 minutes per run.
3. **Visibility & Audit**
   * Centralized logging of each step, with a timestamped run log.
4. **Scalability**
   * Prototype supports 500+ SKUs; architecture can scale to additional warehouses.
5. **Maintainability**
   * Modular codebase (extract, process, update, alert) with clear responsibilities.

**3. Step-by-Step Workflow**

**3.1 As-Is (Manual)**

(see diagrams/as-is.png)

1. Staff counts → Excel input.
2. Save workbook → send via email.
3. Manager opens workbook → runs formulas.
4. Manager reviews low-stock rows.
5. Manager writes email → sends to purchasing.

**3.2 To-Be (Automated)**

(see diagrams/to-be.png)

1. **Extract** (services/extract.py): load raw CSV into pandas DataFrame; log row count.
2. **Process** (services/process.py):
   * Change negative quantities to 0.
   * Drop duplicate SKU+Location pairs.
   * Compute reorder quantity: ReorderQty = max(0, ReorderPoint – OnHandQty).
3. **Update** (services/update.py): write cleaned CSV (inventory\_clean.csv); log output.
4. **Alert** (services/alert.py): filter items where ReorderQty > 0; email JSON report to inventory team.
5. **Metrics** (metrics.py): record runtime and total items; dump metrics.json.

**Orchestration:**

* main.py ties all modules using argparse for I/O paths and python-dotenv for SMTP/email config.
* Centralized logs written to logs/run.log.

**4. System Requirements**

**4.1 Software**

* Python 3.11+
* Libraries:
  + pandas
  + Faker
  + python-dotenv
  + smtplib
  + numpy
* Git & GitHub CLI (for version control)
* Lucidchart account (for diagrams)

**4.2 Hardware / Environment**

* Developer workstation (Windows 10/macOS/Linux)
* Target VM/Server for production runs (Ubuntu 20.04+ or Windows Server 2019+)
* SMTP access for sending emails (port 587 or 25)
* Write permissions for data/, logs/, and repo root

**4.3 Configuration**

* .env file with:
* SMTP\_SERVER=<host>
* SMTP\_PORT=<port>
* EMAIL\_FROM=<sender>
* EMAIL\_TO=<recipient>
* GitHub Actions (optional): ci.yml to run tests on push/PR

**5. Risks & Mitigations**

|  |  |  |
| --- | --- | --- |
| Risk | Impact | Mitigation |
| Data Quality: negative/duplicate rows | Miscomputed reorder needs | Clip negatives; dedupe on SKU+Location; log dropped rows |
| Missing or invalid env vars | Email failures or crashes | Validate env at startup; fail fast with clear error messages |
| SMTP server unavailability | No alerts delivered | Implement retry policy (3 × with backoff) and escalate to ops |
| Large dataset performance bottlenecks | Slow processing (> 5 min runtime) | Profile with cProfile/py-spy; batch processing or parallelize |
| Unhandled exceptions | Partial runs or silent failures | Wrap main steps in try/except; send ops alert on critical errors |
| Version drift in dependencies | Module breakages | Pin versions in requirements.txt; automate CI dependency scan |

**6. Metrics & Success Plan**

|  |  |  |  |
| --- | --- | --- | --- |
| Metric | Baseline | Target (3 months) | Measurement Method |
| Total runtime | 45 min/manual | < 5 min/automated | Time delta in metrics.json |
| Error rate | ~ 5 % | ≤ 1 % | Compare known injected faults |
| Data accuracy uplift | N/A | 100 % (no negatives) | Audit cleaned CSV |
| Time saved per run | N/A | ≥ 90 % reduction | Baseline vs. automated runtime |
| Alerts delivered on time | 0 % | 100 % | Compare schedule vs. email logs |

**7. References**

Faker Team. (n.d.). Faker documentation. <https://faker.readthedocs.io/>

GitHub, Inc. (n.d.). python-dotenv: Reads key-value pairs from a .env file and can set them as environment variables [Source code]. <https://github.com/theskumar/python-dotenv>

Harris, C. R., Millman, K. J., van der Walt, S. J., et al. (2020). Array programming with NumPy. Nature, 585(7825), 357–362. <https://doi.org/10.1038/s41586-020-2649-2>

The pandas development team. (n.d.). pandas: Powerful Python data analysis toolkit. <https://pandas.pydata.org/docs/>

Python Software Foundation. (n.d.). smtplib — SMTP protocol client. <https://docs.python.org/3/library/smtplib.html>

Python Software Foundation. (n.d.). logging — Logging facility for Python. <https://docs.python.org/3/library/logging.html>

Real Python. (n.d.). Python logging: An in-depth tutorial. <https://realpython.com/python-logging/>

Sweigart, A. (2015). *Automate the boring stuff with Python: Practical programming for total beginners*. No Starch Press. <https://automatetheboringstuff.com>