Blueprint

**1. Seller Creates Account**

• The seller visits the web-based software platform (the website or web-app).

• The seller clicks on a button labeled “Create Account” or “Sign Up”.

• The seller fills in a form with their personal and business details:

• Seller’s Full Name

• Business Name

• Email Address

• Mobile Number

• Password (this password is encrypted and stored securely; no one, including admins, can see it).

• After submitting the form, the seller’s information is stored securely in a database on the server.

• Not optional: The seller is required to verify their email address. By sending them a confirmation email, and asking them to click a link or enter a code. (because if email is wrong, things may go south)

• Once registration is complete, the seller logs into their account on the web platform. (submitting OTP or clicking verification link should take them right onto logged in dashboard, no separate login again)

• In their account, they have different levels of access depending on their role:

• Admin: Full control over the account and all data. (only admin can add/modify/remove marketplace/ logistic/PG etc accounts)

• Manager: Can view data, generate reports, and submit claims but cannot add/remove users or change sensitive settings.

• Viewer: Can only view reports and dashboards.

• Role management is handled by the Admin user, who can invite additional users to the same seller account and assign their roles.

**2. Seller Connects Their Marketplace Account (Amazon Seller Central Account First)**

• Once logged into their account on our web platform, the seller clicks on a button or menu option labeled “Connect Marketplace Account”.

• They select “Amazon Seller Central” from the available marketplace options.

• The system shows them clear step-by-step instructions on how to generate their API credentials. These instructions include screenshots and examples.

• The seller logs into their Amazon Seller Central account and navigates to the Developer Console, where they create a new application (our app) and receive the following credentials:

• LWA (Login with Amazon) Client ID

• LWA Client Secret

• AWS Access Key

• AWS Secret Key

• Refresh Token

• The seller copies these credentials and pastes them into the corresponding input fields on our platform.

• Our platform securely stores these credentials in the database. Before storage, each credential is encrypted using AES 256-bit encryption, a widely accepted and strong encryption standard.

• The encrypted credentials are saved in the database, which itself resides on a secure server. Even if someone accesses the physical storage of the database, they will not be able to read the credentials without decryption keys, which are managed securely in the backend.

• The connection is tested immediately. Our backend server sends a request to Amazon Seller Central’s API (SP-API) using these credentials to check if the connection is successful.

• If the connection is successful, the system confirms it to the seller.

• If there is an error (wrong credentials, connection failure, etc.), a clear error message is displayed to the seller explaining what went wrong and how to fix it.

• The system uses secure HTTPS (SSL/TLS encryption) for all communication with Amazon and with the seller’s web browser.

**3. System Fetches Order Data from Amazon**

• Our backend software (running on a cloud server) sends a request to Amazon’s SP-API endpoint, specifically the “GET /orders/v0/orders” endpoint.

• This API call retrieves data about orders that were placed by customers on Amazon.

• The data includes:

• Order ID

• Order Date

• Order Status (Shipped, Delivered, Returned, etc.)

• Product Details (SKU, Quantity, Price per unit)

• Buyer’s basic details if available (not always provided by Amazon).

• This order data is automatically saved into our platform’s database in a table called “Orders”.

• The system regularly fetches new orders automatically. This can be set to every few hours, once a day, or on-demand when the seller requests it.

• If an error occurs (Amazon API downtime, wrong credentials, etc.), the system logs the error, retries later, and sends an alert if necessary.

• Every time data is fetched or changed, an audit log records which data was accessed, who triggered it, and when it happened. This helps for future audits and debugging.

**4. System Fetches Payment Data (Settlements from Amazon)**

• Our backend software sends another request to Amazon’s SP-API, specifically the “GET /finances/v0/financialEvents” endpoint.

• This retrieves settlement reports, showing:

• The total order amount

• Amazon fees deducted (commission, shipping, referral fees, etc.)

• Taxes deducted (GST/TDS if applicable)

• The final amount paid to the seller (net payout)

• The settlement date (when the payout happened)

• This payment data is saved in a separate table in our database called “Payments”.

• Each payment record is linked to its respective order using the Order ID.

• The system keeps a record of partial settlements and funds held in reserve by Amazon for risk management reasons.

• Every data fetch and write is logged in the audit trail.

**5. System Fetches Return and Refund Data from Amazon**

• The backend sends a request to Amazon’s SP-API “GET /reports/v0/reports” endpoint for Return Reports.

• This retrieves details of customer returns:

• Order ID of returned orders

• Return status (Received by warehouse, Condition of returned product, etc.)

• Refund status (Was the customer refunded? Was the seller reimbursed?)

• This return and refund data is saved in a separate “Returns” table in our database.

• This data is linked to the correct order and payment in the database.

• Each time the data is retrieved or changed, an audit log records the event.

**6. Reconciliation Engine Runs**

• This is the logic module that compares data from the Orders, Payments, and Returns tables.

• The system checks for:

• Payment mismatches (Was the seller paid the correct amount for each order?)

• Incorrect fees (Were Amazon’s charges as per their policy, or are there overcharges? E.g., excessive commission or incorrect shipping fees)

• Returns and refunds mismatches:

• If a customer returned a product and it was received by Amazon, the seller should get fees refunded (if applicable).

• If a product was returned by the customer but never received by Amazon (lost or damaged by Amazon), the seller should receive a reimbursement.

• If no reimbursement was given in such cases, the system flags a discrepancy.

• The system runs this comparison on a scheduled basis (once daily by default) or manually if triggered by the seller.

• The comparison rules and logic are fully documented and based on Amazon’s official policies and fee structures.

• Sellers can set custom rules and thresholds. For example:

• Only flag discrepancies above ₹500

• Flag if returns are more than 20% of orders for a SKU

• Every reconciliation run is recorded in the audit log for transparency and compliance.

**7. Discrepancy Report Generation**

• After the reconciliation engine runs, the system creates reports showing all detected discrepancies.

• The report includes:

• Order ID

• Type of discrepancy (e.g., Payment shortfall, Incorrect fees, Missing reimbursement for lost/damaged return)

• Amount of money involved

• Date of discrepancy detection

• Recommended action

• Examples of discrepancies flagged:

• Amazon failed to reimburse the seller when a returned item was lost or damaged.

• Amazon deducted fees incorrectly on a returned/refunded order.

• Seller was charged excess commissions or storage fees not matching the agreed rates.

• The report is displayed on the seller’s dashboard with:

• Summary cards (Total Orders Processed, Total Discrepancies Found, Total Amount at Risk, Total Amount Recovered)

• A detailed table listing each discrepancy

• The seller can filter discrepancies by date, type, status (Open, Claimed, Resolved).

• Reports can be exported as CSV, Excel, or PDF files.

• Additional insights are provided:

• Recovery Success Percentage

• High-Risk SKUs with frequent return/reimbursement issues

• Monthly/Quarterly summaries for decision-making

**8. Claim Preparation**

• The seller reviews discrepancies on the dashboard.

• For each discrepancy, the system prepares a claim template that includes:

• Order details

• Explanation of the discrepancy

• Supporting evidence (report snapshots or downloadable CSVs if needed)

• In the MVP version, the seller manually submits the claim by email using the prepared template.

• In future versions, claims can be submitted automatically via API if Amazon or other platforms allow.

• Claim submission status and history are recorded in the database and audit log.

**9. Claim Status Tracking**

• Sellers can track the progress of each claim in the dashboard.

• Status options include Submitted, In Progress, and Resolved.

• Sellers can manually update claim status after receiving feedback from Amazon.

• In future versions, automated claim tracking will be implemented using platform webhooks or feedback APIs.

• Claim history reports can be downloaded for compliance and internal audits.

**10. Notifications and Alerts**

• Sellers receive email notifications for:

• New discrepancies found

• Claim submission deadlines

• Claim resolution updates

• Notifications can be expanded to Slack or WhatsApp channels if the seller enables these options.

• Sellers can configure notification frequency (daily, weekly, real-time).

**11. Security and Compliance Measures**

• All API credentials and other sensitive data are encrypted before being stored in the database.

• Encryption uses AES 256-bit standard, preventing unauthorized access even if the database is compromised.

• SSL/TLS encryption is used for all data transfers between the seller’s browser, our platform, and the marketplace APIs.

• Role-based access control ensures that users only have access to data appropriate to their role.

• Privacy policies and terms of service are in place and comply with GDPR and Indian IT Act requirements.

• Data deletion requests from sellers are honored within stipulated timelines.

• Audit logs track every critical action in the system.

• Backup files are also encrypted.

• Long-term security roadmap includes preparation for SOC2 or ISO certifications.

**12. Multi-Marketplace Scalability Planning**

• The system is designed to support additional marketplaces (Flipkart, Shopify, WooCommerce, etc.).

• The database structure and backend code use an abstraction layer to separate each marketplace’s data models and logic.

• The seller can view and manage data from multiple marketplaces through a unified dashboard interface.