**Phase 5: Apex Development**

1. **Triggers**

Triggers allow you to run code **before or after database operations** (Insert, Update, Delete, Undelete). They enforce **real-time business rules**.

A screenshot of a computer program

AI-generated content may be incorrect.

**Examples in your Bookstore:**

* **Update stock after order creation**
  + When a Sale\_\_c and related Sale\_Line\_Item\_\_c records are inserted, a trigger automatically updates the **Stock\_Quantity\_\_c** of each Book\_\_c.
  + This ensures stock is always accurate.

A screenshot of a computer program

AI-generated content may be incorrect.

* **Restrict order if stock unavailable**
  + A before insert trigger checks if requested quantity > available stock.
  + If stock is insufficient, the trigger throws an error and prevents the sale.

1. **Batch Apex**

Batch Apex is used for **processing large datasets asynchronously** in manageable chunks.

A screenshot of a computer program

AI-generated content may be incorrect.

**Example in your Bookstore:**

* **Daily Low-Stock Report**
  + A Batch Apex job queries all Book\_\_c records where Stock\_Quantity\_\_c < 10.
  + Compiles the results into a report.
  + Sends an email summary to the store manager every night.

1. **Future Methods**

Future methods run **asynchronously** — meaning they execute in the background without blocking user actions.

**Example in your Bookstore:**

* **Send asynchronous notification emails**
  + When stock falls below threshold, trigger calls a @future method.
  + This method composes and sends emails to managers/suppliers.
  + Async execution ensures the main transaction (book update) isn’t delayed.

1. **Exception Handling**

Errors are inevitable — missing data, unexpected input, or failed operations. Apex must **gracefully handle errors** to avoid corrupting records.

**Example in your Bookstore:**

* If stock data is missing or a Book record cannot be queried, the system:
  + Rolls back the transaction (Database.rollback) to keep data consistent.
  + Returns a clear error message to the user:  
    *“Error: Stock information for Book XYZ not available. Please contact admin.”*
* Use **try/catch blocks** to capture exceptions and log them.

1. **Test Classes**

Salesforce requires **at least 75% code coverage** for deployment to production. Test classes simulate real-world use cases to ensure Apex logic works correctly.

**Example in your Bookstore:**

* **BookControllerTest** → Tests CRUD and search methods.

A screenshot of a computer program

AI-generated content may be incorrect.

* **SaleControllerTest** → Verifies order creation, stock updates, and insufficient stock errors.

A screenshot of a computer program

AI-generated content may be incorrect.

* **Batch Tests** → Ensure daily low-stock report generates correctly.
* **Trigger Tests** → Confirm that stock reduces and alerts send properly.