**PRACTICAL NO.08**

design **test suites** specifically for an **Inventory Management System (IMS)**. A system like this typically handles:

* Product tracking
* Stock updates
* Supplier management
* Order processing
* Notifications (e.g., low stock)

**🧪 Test Suite Design for Inventory Management System**

Let’s organize the test suites by modules:

**📦 1. Product Management Test Suite**

| **Test Case ID** | **Title** | **Input Data** | **Expected Result** | **Status** |
| --- | --- | --- | --- | --- |
| TC\_PM\_01 | Add new product | Valid product details | Product added to database | Pass |
| TC\_PM\_02 | Add product with missing field | Missing product name | Validation error displayed | Pass |
| TC\_PM\_03 | Edit product details | Updated product price | Product info updated successfully | Pass |
| TC\_PM\_04 | Delete a product | Product ID | Product removed or marked inactive | Pass |

**📥 2. Stock Management Test Suite**

| **Test Case ID** | **Title** | **Input Data** | **Expected Result** | **Status** |
| --- | --- | --- | --- | --- |
| TC\_SM\_01 | Increase stock quantity | Product ID + quantity | Quantity updated in inventory | Pass |
| TC\_SM\_02 | Decrease stock (valid case) | Product ID + valid quantity | Quantity decreased | Pass |
| TC\_SM\_03 | Decrease stock below zero | Product ID + too much quantity | Error: insufficient stock | Pass |
| TC\_SM\_04 | View current stock levels | Product ID | Correct quantity displayed | Pass |

**📑 3. Supplier Management Test Suite**

| **Test Case ID** | **Title** | **Input Data** | **Expected Result** | **Status** |
| --- | --- | --- | --- | --- |
| TC\_SUP\_01 | Add new supplier | Supplier details | Supplier added successfully | Pass |
| TC\_SUP\_02 | Edit supplier info | Updated contact info | Info updated in database | Pass |
| TC\_SUP\_03 | Delete supplier | Supplier ID | Supplier removed or marked inactive | Pass |

**📦📤 4. Order Management Test Suite**

| **Test Case ID** | **Title** | **Input Data** | **Expected Result** | **Status** |
| --- | --- | --- | --- | --- |
| TC\_ORD\_01 | Create new order | Product list + quantity | Order recorded, stock decreased | Pass |
| TC\_ORD\_02 | Place order with out-of-stock item | Product ID with 0 quantity | Error: Item not available | Pass |
| TC\_ORD\_03 | View order details | Order ID | Order summary displayed | Pass |

**🔔 5. Notifications & Reports Test Suite**

| **Test Case ID** | **Title** | **Input Data** | **Expected Result** | **Status** |
| --- | --- | --- | --- | --- |
| TC\_NOT\_01 | Low stock alert triggered | Product with quantity < min | Alert/notification shown | Pass |
| TC\_RPT\_01 | Generate stock report | Date range | Accurate report generated | Pass |
| TC\_RPT\_02 | Export inventory to Excel | Click export | Download starts with correct data | Pass |

**🧪 Bonus: Validation Test Suite**

| **Test Case ID** | **Title** | **Input Data** | **Expected Result** | **Status** |
| --- | --- | --- | --- | --- |
| TC\_VAL\_01 | Add product with special characters | @!#Product name | Validation error | Pass |
| TC\_VAL\_02 | Negative price/quantity | -100 price or -5 quantity | Error: invalid input | Pass |

**🎯 Goal of Designing Test Suites**

To ensure all key functionalities of the system are tested under:

* Normal conditions
* Edge cases
* Invalid input scenarios

**🧩 Phases in Designing a Test Suite**

**1. Requirement Analysis**

* Study the **functional** and **non-functional** requirements.
* Understand the **user roles** (e.g., Admin, Store Manager, Supplier).
* Identify **key modules** (e.g., Inventory, Orders, Notifications, Reports).

**2. Identify Test Scenarios**

Each module should have:

* **Positive test scenarios**: Valid inputs and operations.
* **Negative test scenarios**: Missing fields, invalid data, unauthorized access.
* **Edge case scenarios**: Extremely high or low values.

Example for "Add Product":

* Add product with valid fields ✅
* Add product with missing name ❌
* Add product with negative price ❌
* Add duplicate product ID ❌

**3. Design Test Cases**

Each test case should include:

| **Field** | **Description** |
| --- | --- |
| **Test Case ID** | Unique identifier (e.g., TC\_INV\_001) |
| **Title** | Short name for the test (e.g., Add product with valid data) |
| **Description** | What the test is checking |
| **Preconditions** | What needs to be set before executing the test |
| **Test Steps** | Step-by-step guide to perform the test |
| **Test Data** | Data to be used in the test (e.g., product name = "Pen", qty = 100) |
| **Expected Result** | What should happen (e.g., “Product added successfully”) |
| **Actual Result** | What happened during testing |
| **Status** | Pass / Fail |
| **Remarks** | Notes or bug IDs |

**4. Organize into Test Suites**

Group related test cases into **test suites**:

| **Test Suite Name** | **Covered Modules** |
| --- | --- |
| Product Management Suite | Add, Edit, Delete Product |
| Stock Management Suite | Stock update, Inventory level checks |
| Supplier Management Suite | Supplier CRUD operations |
| Order Processing Suite | Create orders, stock updates |
| Reporting Suite | Generate reports, exports |
| Security & Validation Suite | Input validation, access control |

**5. Prioritize Test Cases**

* **High Priority**: Core features (e.g., stock update, order creation)
* **Medium**: Reports, supplier management
* **Low**: Cosmetic/UI, minor validations

Use **Risk-Based Testing** for smarter prioritization.

**6. Choose Testing Types for Each Suite**

| **Test Type** | **Description** |
| --- | --- |
| **Smoke Testing** | Basic checks (e.g., app loads, menus work) |
| **Functional** | Actual feature behavior |
| **Regression** | After code changes |
| **Integration** | Component interaction |
| **UI Testing** | Interface functionality |
| **Negative Testing** | Handling wrong/invalid inputs |

**7. Automation Consideration**

If certain actions are repeated often (e.g., login, stock update), automate them using tools like:

* **Selenium** (UI Testing)
* **Postman** (API testing)
* **JUnit/PyTest/TestNG** (Unit testing)

**8. Maintainability**

* Use consistent naming (e.g., TC\_<MODULE>\_<NUMBER>)
* Track versioning (v1.0, v2.0...)
* Add traceability to requirements

**✅ Example Test Case Template (Add Product)**

| **Field** | **Example** |
| --- | --- |
| Test Case ID | TC\_PROD\_001 |
| Title | Add product with valid details |
| Description | Verify that a product can be added successfully |
| Preconditions | User is logged in as Admin |
| Test Steps | 1. Navigate to Products  2. Click "Add Product"  3. Enter valid details  4. Submit |
| Test Data | Name: Pen, Category: Stationery, Price: 10 |
| Expected Result | Product should be added and appear in list |
| Actual Result | (To be filled during execution) |
| Status | (Pass/Fail) |
| Remarks |  |