MILESTONE 4

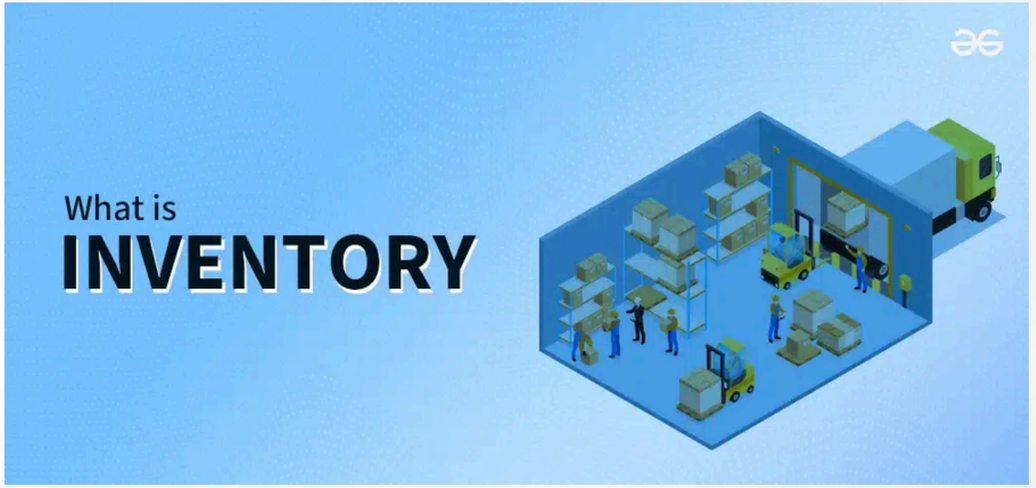
INVENTRA-INTELLIGENT INVENTORY MANAGEMENT SYSTEM

Presented by : TEAM 1

Rishwana Sirajutheen
Naga Syamala Chintapalli
Dritisai Sivarathri
Tanoor Kiran
Hemanth Ravva

INTRODUCTION

- Inventra is a web-based Inventory Management System
- It helps to track items, stock levels and transactions
- Solves problems like manual records, missing stock info
- Provides easy monitoring + role-based access

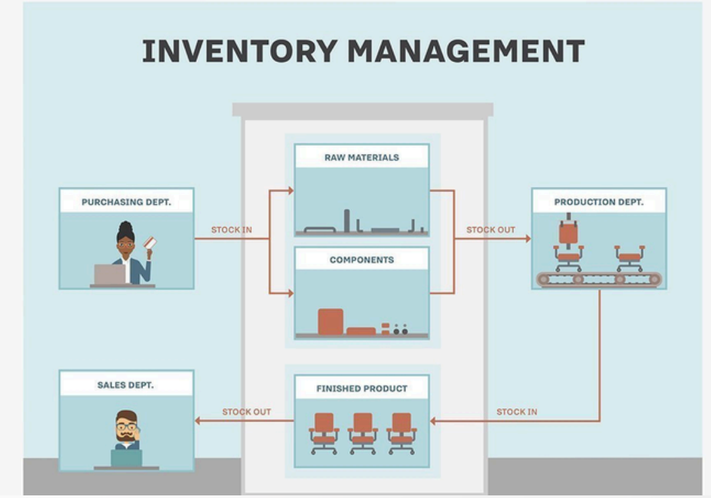


OBJECTIVE

- To understand how an inventory system works digitally
- To automate stock handling and transaction tracking
- To learn secure login + role-based control
- To improve inventory accuracy and reduce time

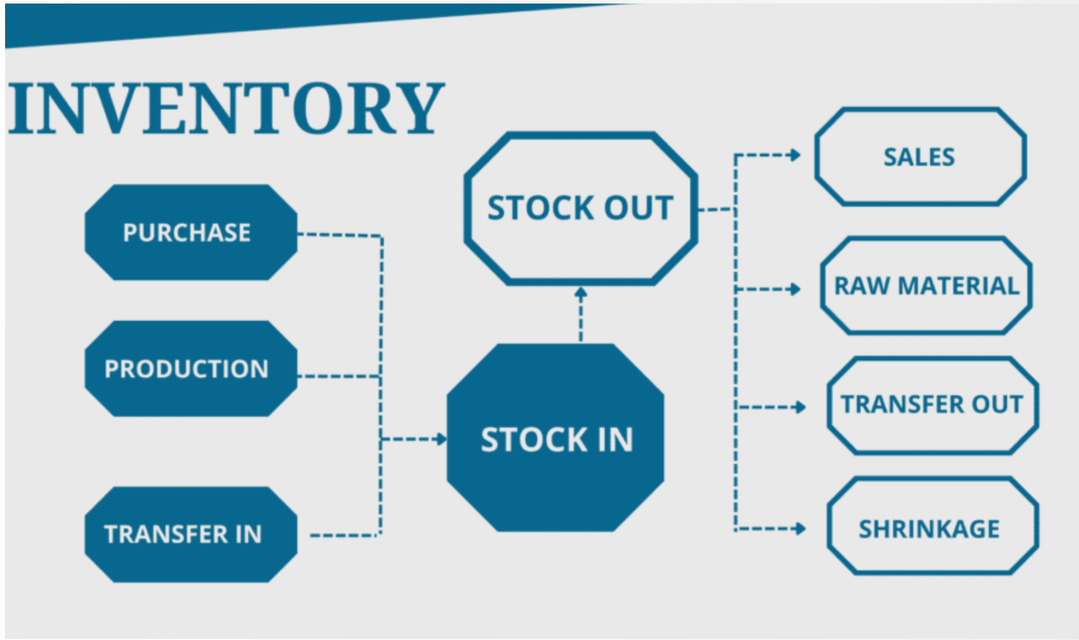
PROBLEM STATEMENT

- Manual stock handling causes errors and confusion
- Difficult to know real-time available stock
- No clear tracking of who added/issued items
- Reports and history are not maintained properly



SYSTEM OVERVIEW

- Users login using Authentication module
- Role-based access controls system operations
- Inventory modules manage:
- Add / Update / Delete stock
- Stock availability check
- Transactions track in/out movement
- Dashboard gives overall view of inventory



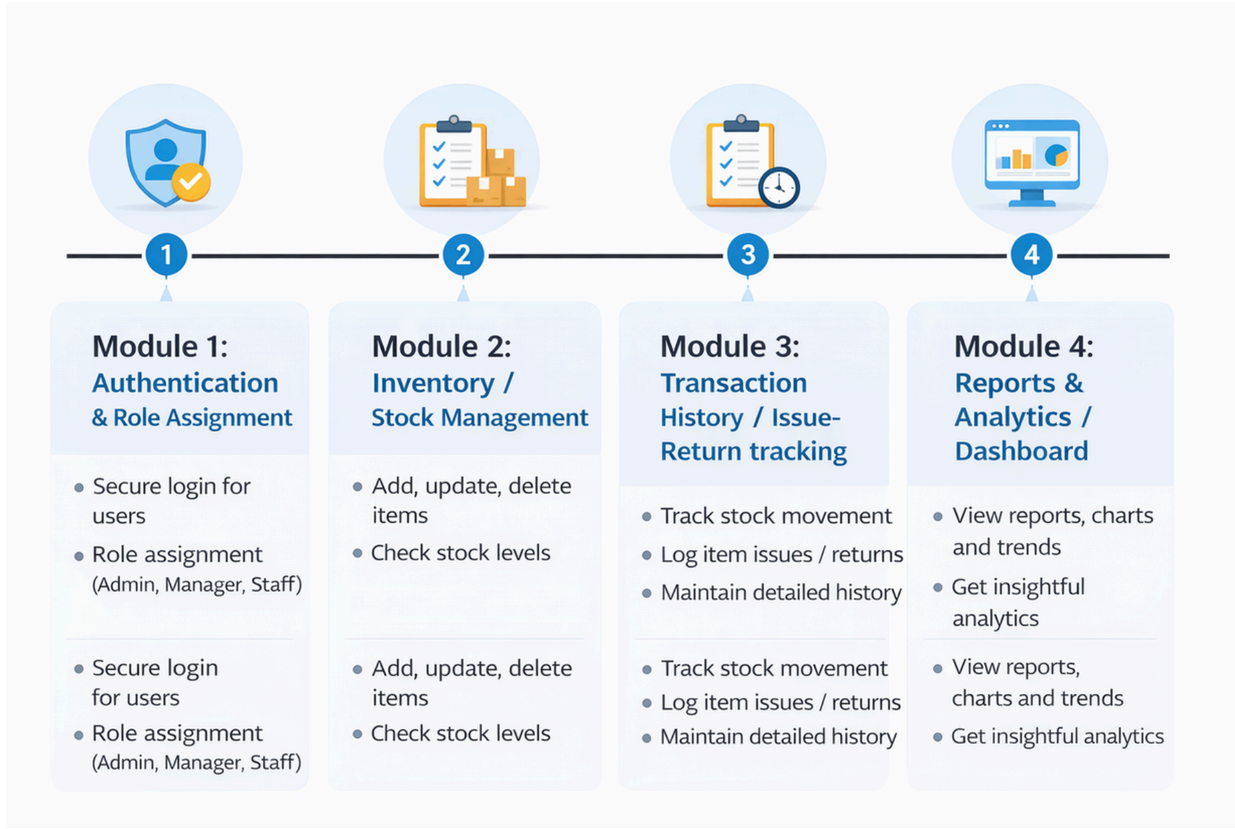
KEY MODULES IN INVENTRA

Module 1: Authentication & Role Assignment — 1

Module 2: Inventory / Stock Management — 2

Module 3: Transaction History / Issue-Return tracking — 3

Module 4: Reports & Analytics — 4



MODULE 1

AUTHENTICATION

- Entry point of the Inventory Management System
- Verifies user identity using username and password
- Allows only authorized users to access the system
- Provides role-based access (Admin, Manager, Staff)
- Restricts users to permitted features only

Username and password are received
Credentials are validated
If valid → token generated
If invalid → error message

MODULE 1 – KEY CLASSES

- **AuthController** - Handles HTTP requests
- **AuthService** - Business logic for authentication
- **User Entity** - User data model
- **UserRepository** - Database operations
- **JWTUtility** - Token generation and validation
- **EmailService** - Password reset emails

SIGN-UP (REGISTER USER) – LOGIC

Purpose

- Create new user securely
- Prevent duplicate users

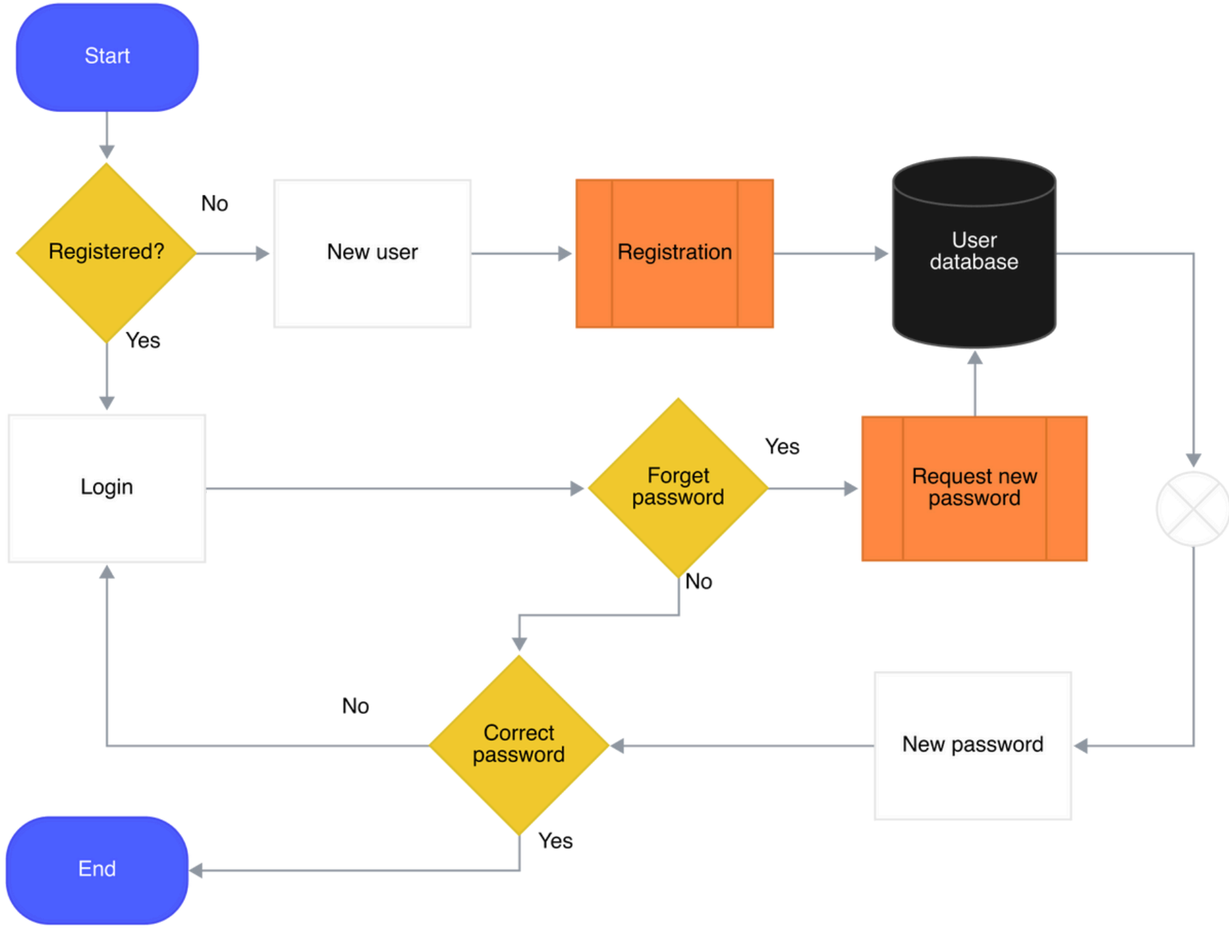
Key Explanation

- Username uniqueness enforced
- Password encryption before storage

Pseudocode

```
IF username already exists
    reject request
ELSE
    encrypt password
    save user
```

SIGN-UP FLOW



UI → Controller → Service → Repository → DB

SIGN-IN (LOGIN) & AUTHENTICATION

Purpose

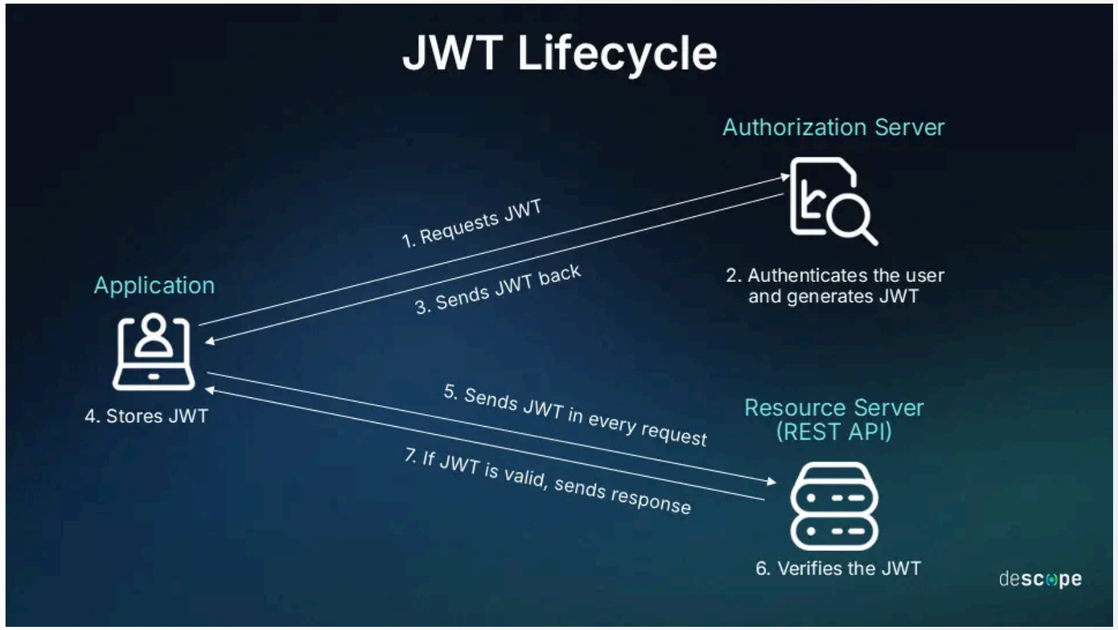
- Validate credentials
- Generate access token

Pseudocode

```
find user by username
IF password matches
    generate JWT token
ELSE
    deny access

```

JWT TOKEN – CORE SECURITY CONCEPT



Why JWT?

- Stateless authentication
- Scalable
- Secure API access

FORGOT & RESET PASSWORD (SECURITY)

Purpose

- Recover access safely
- Avoid password exposure

Pseudocode

```
generate reset token
send email
validate token
update encrypted password

```

MODULE 1 – SUMMARY & STRENGTHS

Key Strengths

- Secure authentication
- Encrypted password storage
- Token-based authorization
- Industry-standard architecture

FLOW

User Interface (UI)
↓
AuthController
↓
AuthService
↓
UserRepository
↓
Database
↑
JWT Token Generated
↑
Response sent to UI

MODULE 2: PRODUCT / INVENTORY MANAGEMENT

Module 2 is the core operational module of the Inventra – Intelligent Inventory Management System.

It is responsible for managing products and maintaining accurate stock levels in real time.

Purpose of Module 2

- The main goal of this module is to:
- Store product details in a structured manner
- Track stock availability continuously
- Handle real-world inventory operations like stock-in and stock-out
- Prevent stock mismatch and negative inventory

MODULE 2 – KEY CLASSES

Product Entity - Product data model with SKU, quantity, price

ProductController - REST endpoints for product operations

ProductService - Business logic for inventory management

ProductRepository - Database queries

TransactionService - Logs stock movements

PSEUDOCODE

Product Entity

```
CLASS Product
    productId
    sku
    name
    category
    supplier
    unitPrice
    stockQuantity
    minStockLevel
END CLASS
```

Add Product

```
FUNCTION addProduct(productData)
    IF product SKU does not exist THEN
        save product
        RETURN "Product added"
    ELSE
        RETURN "Duplicate product"
    END IF
END FUNCTION
```

Stock In

```
FUNCTION stockIn(productId, quantity)
    product = find product by productId
    product.stockQuantity += quantity
    save product
    log STOCK_IN transaction
    check low stock
END FUNCTION
```

Stock Out

```
FUNCTION stockOut(productId, quantity)
    product = find product by productId

    IF product.stockQuantity >= quantity THEN
        product.stockQuantity -= quantity
        save product
        log STOCK_OUT transaction
        check low stock
    ELSE
        RETURN "Insufficient stock"
    END IF
END FUNCTION
```

MODULE 3: INVENTORY ALERTS MODULE

Module 3 is the Inventory Alerts Module, which adds intelligence and automation to the Inventra – Intelligent Inventory Management System.

Its primary role is to monitor stock levels continuously and notify responsible users before a stock-out situation occurs.

🔹 Purpose of Module 3

- The main objectives of this module are to:
- Detect low-stock conditions in real time
- Notify users at the right time
- Avoid repeated or unnecessary alerts
- Improve inventory decision-making

ALERT TRIGGER FLOW

Stock Update → Check Minimum Level → Alert Triggered

PSEUDOCODE

🔹 Alert Logic Using Stack

```
CLASS AlertManager
    CREATE Stack alertStack
END CLASS
```

MODULE 3 – KEY CLASSES

Transaction – stores issue/return record

IssueService – handles issuing items + stock reduction

ReturnService – handles return + stock update

TransactionHistory – view/search/filter transactions

TransactionRepository – DB insert/fetch transactions

- Check Low Stock

```
FUNCTION checkLowStock(product)
    IF product.stockQuantity < product.minStockLevel THEN
        alertMessage = "Low stock for " + product.name
        addAlert(alertMessage)
    END IF
END FUNCTION
```

- Add Alert (Avoid Duplicates)

```
FUNCTION addAlert(alertMessage)
    IF alertStack is empty OR top of stack != alertMessage THEN
        PUSH alertMessage into stack
        DISPLAY alertMessage
    END IF
END FUNCTION
```

- View Latest Alert

```
FUNCTION getLatestAlert()
    IF stack not empty THEN
        RETURN top alert
    ELSE
        RETURN "No alerts"
    END IF
END FUNCTION
```

MODULE 4: REPORTS & ANALYTICS

1) Introduction

Module 4 is the Reports & Analytics part of Inventra.

It converts inventory + transaction data into useful insights.

It shows the dashboard view with charts and summary counts.

Helps admins/managers monitor stock and decisions easily.

2) Purpose / Objective

To generate real-time reports from system data

To display:

Total items

Available stock

Low stock alerts

Recently issued/returned items

To support analytics like:

Frequently issued items

Monthly/weekly transaction summary

MODULE 4 – KEY CLASSES

Dashboard – shows KPIs (total stock, low stock, recent logs)

Report – report object (stock/transaction/low stock)

ReportGenerator – generates reports based on type

AnalyticsService – trends (most issued, monthly summary)

ChartDataBuilder – converts data to chart format

PSEUDOCODE

```
START

FUNCTION loadDashboard(userRole):
    IF userRole NOT IN ["ADMIN", "MANAGER"]:
        SHOW "Access Denied"
        RETURN

    inventoryData = fetchInventoryData()
    transactionData = fetchTransactionData()

    totalItems = count(inventoryData)
    totalStock = sum(inventoryData.quantity)

    lowStockItems = filter(inventoryData WHERE quantity <= threshold
        )

    issuedCount = count(transactionData WHERE type == "ISSUE")
    returnCount = count(transactionData WHERE type == "RETURN")

    recentTransactions = getLatest(transactionData, 5)
```

```
 DISPLAY totalItems
 DISPLAY totalStock
 DISPLAY issuedCount
 DISPLAY returnCount
 DISPLAY lowStockItems
 DISPLAY recentTransactions

) FUNCTION


NCTION generateReport(reportType, dateFrom, dateTo):

 IF reportType == "STOCK_REPORT":
     data = fetchInventoryData()
     EXPORT data as PDF/Excel

 ELSE IF reportType == "TRANSACTION_REPORT":
     data = fetchTransactionDataBetween(dateFrom, dateTo)
     EXPORT data as PDF/Excel
```

```
    ELSE IF reportType == "TRANSACTION_REPORT":
        data = fetchTransactionDataBetween(dateFrom, dateTo)
        EXPORT data as PDF/Excel

    ELSE IF reportType == "LOW_STOCK_REPORT":
        data = fetchLowStockItems()
        EXPORT data as PDF/Excel

    ELSE:
        SHOW "Invalid report type"

END FUNCTION

STOP
```

MODULE 1: AUTHENTICATION – DATABASE MANAGEMENT

Column Name	Data Type	Description
user_id	INT (PK)	Unique user identifier
username	VARCHAR	Unique login name
password	VARCHAR	Encrypted password
role	VARCHAR	ADMIN / STAFF
email	VARCHAR	Used for password reset
created_at	TIMESTAMP	Account creation time

```sql
CREATE TABLE users (
    user_id INT PRIMARY KEY AUTO_INCREMENT,
    username VARCHAR(50) UNIQUE NOT NULL,
    password VARCHAR(100) NOT NULL,
    role VARCHAR(20) NOT NULL,
    email VARCHAR(100),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

- How Module 1 Uses the DB
- Signup → INSERT new user
- Login → SELECT username & password
- Role validation → SELECT role
- Forgot password → SELECT by email

✔ Ensures security, authentication, and authorization

MODULE 2: PRODUCT / INVENTORY MANAGEMENT – DATABASE MANAGEMENT

Column Name	Data Type	Description
product_id	INT (PK)	Unique product ID
sku	VARCHAR	Unique product code
name	VARCHAR	Product name
category	VARCHAR	Category
stock_quantity	INT	Current stock
min_stock_level	INT	Alert threshold

```sql
CREATE TABLE products (
    product_id INT PRIMARY KEY AUTO_INCREMENT,
    sku VARCHAR(50) UNIQUE,
    name VARCHAR(100),
    category VARCHAR(50),
    stock_quantity INT,
    min_stock_level INT
);
```

- PRODUCTS Table

Column	Type	Description
transaction_id	INT (PK)	Unique transaction
product_id	INT (FK)	Related product
type	VARCHAR	STOCK_IN / STOCK_OUT
quantity	INT	Quantity changed
transaction_date	TIMESTAMP	When it happened

```sql
CREATE TABLE inventory_transactions (
    transaction_id INT PRIMARY KEY AUTO_INCREMENT,
    product_id INT,
    type VARCHAR(20),
    quantity INT,
    transaction_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (product_id) REFERENCES products(product_id)
);
```

🔹 INVENTORY_TRANSACTIONS Table

🔹 How Module 2 Uses the DB

Add product → INSERT into products

Stock in/out → UPDATE stock_quantity

Audit trail → INSERT into inventory_transactions

MODULE 3: INVENTORY ALERTS – DATABASE MANAGEMENT

Column Name	Data Type	Description
alert_id	INT (PK)	Unique alert
product_id	INT (FK)	Related product
alert_message	VARCHAR	Alert text
alert_time	TIMESTAMP	Alert timestamp
status	VARCHAR	ACTIVE / RESOLVED

```sql
CREATE TABLE alerts (
    alert_id INT PRIMARY KEY AUTO_INCREMENT,
    product_id INT,
    alert_message VARCHAR(255),
    alert_time TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    status VARCHAR(20),
    FOREIGN KEY (product_id) REFERENCES products(product_id)
);
```

- How Module 3 Uses the DB
  - Low stock detected → INSERT alert
  - Duplicate prevention → CHECK last alert
  - Alert resolution → UPDATE status

REAL-TIME / PRACTICAL EXAMPLE

Where Inventra can be used:

Colleges / Labs

equipment issue & return

Hospitals

medicine and equipment stock

Retail shops

product stock monitoring

IT companies

asset tracking (laptops, cables, devices)

ADVANTAGES

- Saves time and reduces manual work
- Provides real-time stock visibility
- Ensures accountability using transaction logs
- Prevents duplicate allocation of resources
- Improves decision making using reports

LIMITATIONS

- Needs correct data entry for accurate results
- Requires proper database + backup
- Role management is critical for security
- System performance depends on server/network

CONCLUSION

- Inventra helps in efficient inventory control
- Provides structured modules with clear workflow
- Improves tracking, security, and stock accuracy
- Useful for real-world industry inventory problems

THANK YOU