

INVENTRA - INTELLIGENT INVENTORY MANAGEMENT SYSTEM

MILESTONE - 2

**BY
RAJESWARI R**

WHAT IS INVENTRA?

Inventra is an intelligent inventory management system

Designed to automate inventory tracking and control

Reduces manual errors and stock inconsistencies

Follows industry-standard backend architecture

PROBLEM STATEMENT

1. Manual stock updates
2. Duplicate product
3. entries Stock mismatch
4. and losses No proper
5. access control Lack of audit and tracking

SYSTEM ARCHITECTURE

Layered Architecture

- Presentation Layer (UI)
- Controller Layer
- Service Layer (Business Logic)
- Repository Layer
- Database Layer

Flow:

UI → Controller → Service → Repository → Database

MODULE OVERVIEW

Major Modules

- Authentication & Authorization
- Product & Inventory Management
- Inventory Alerts
- Transaction History
- Reports & Analytics

MODULE 1: AUTHENTICATION

Authentication & Authorization

- User registration and login
- Encrypted password storage
- JWT-based token authentication
- Role-based access control
- Secure password recovery

Objective

- Secure access to the system
 - Validate user identity
 - Control access using roles

CLASSES USED IN MODULE 1

Architecture Flow

UI → AuthController → AuthService → UserRepository → Database

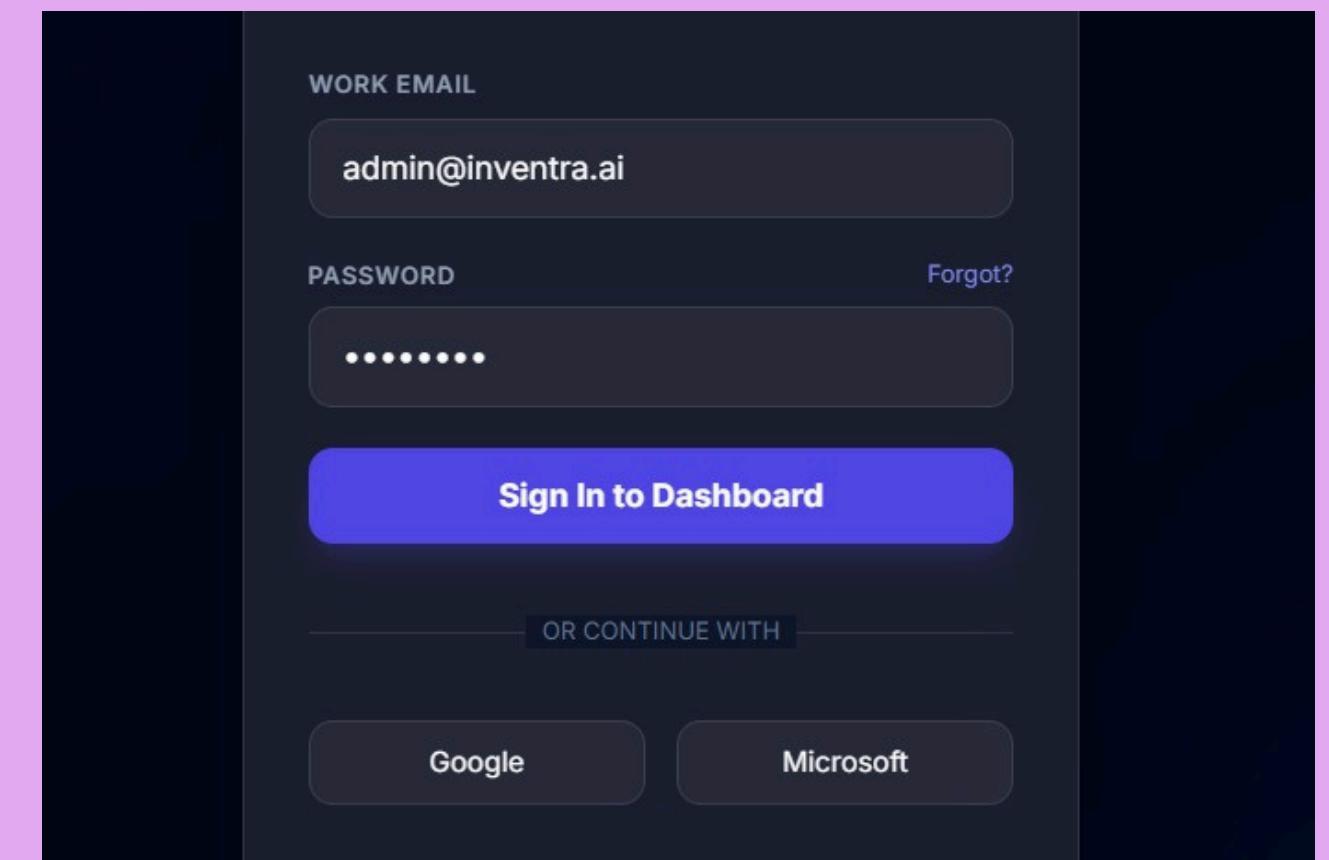
Explain

Controller handles requests

- Service applies business rules
- Repository interacts with database

Core Classes

- AuthController
- AuthService
- User
- UserRepository
- JWTUtility
- EmailService



STEP-BY-STEP TECHNICAL EXPLANATION

1□ **User Interface (UI)**

User enters username & password

Sends request to backend

2□ **AuthController**

Receives request

Does no validation

Forwards data to service layer

3□ **AuthService (CORE LOGIC)**

Verifies user existence

- Matches encrypted password
- Generates JWT Token

4□ **UserRepository**

Communicates with database

Fetches user record

Saves or updates user data

5□ **Database**

Stores encrypted passwords

Stores user roles

Never stores plain credentials

6□ **JWT Token**

Generated after successful login

Sent to UI

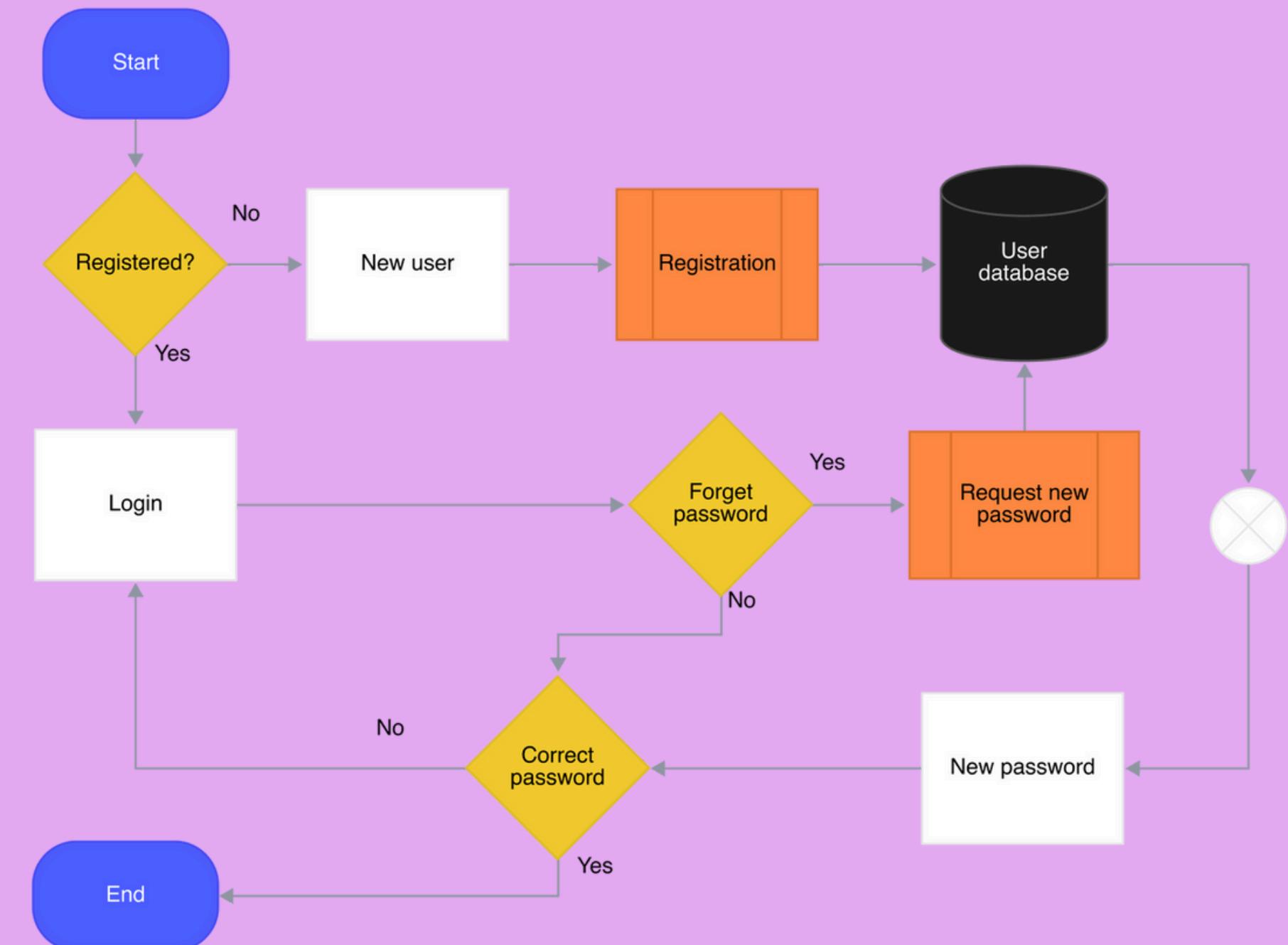
Used for all future requests

SIGN-UP (REGISTER USER) – LOGIC

Purpose

- Create new user securely
- Prevent duplicate users

SIGN-UP FLOW



Key Explanation

- Username uniqueness enforced
- Password encryption before storage

Pseudocode

```
1 IF username already exists  
2   reject request  
3 ELSE  
4   encrypt password  
5   save user  
6
```

UI → Controller → Service → Repository → DB

SIGN-IN (LOGIN) & AUTHENTICATION

Purpose

- Validate credentials
- Generate access token

Pseudocode

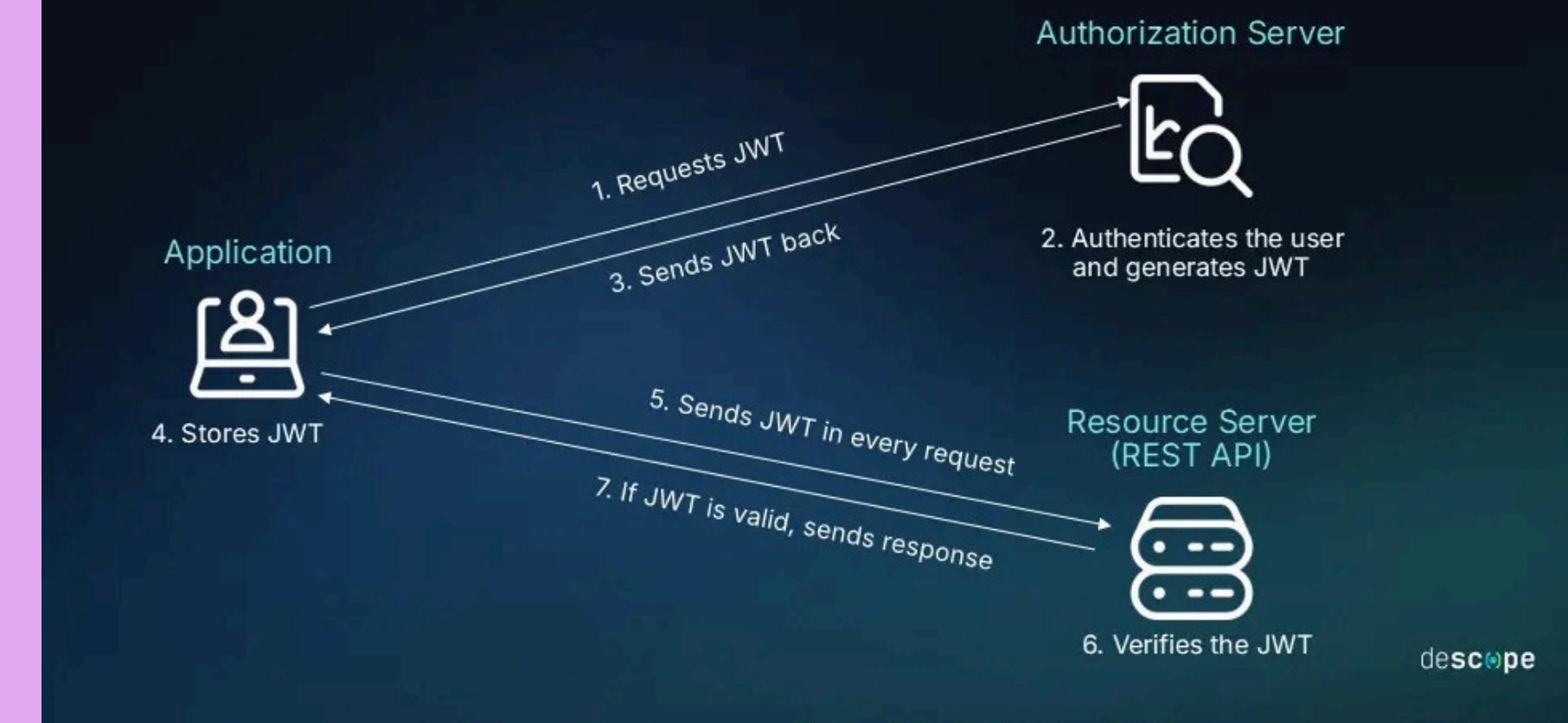
```
1 find user by username
2 IF password matches
3     generate JWT token
4 ELSE
5     deny access
```

6

7

JWT TOKEN – CORE SECURITY CONCEPT

JWT Lifecycle



Why JWT?

- Stateless authentication
- Scalable
- Secure API access

FORGOT & RESET PASSWORD (SECURITY) MODULE 1 – SUMMARY & STRENGTHS

Purpose

- Recover access safely
- Avoid password exposure

Pseudocode

```
1 generate reset token
2 send email
3 validate token
4 update encrypted password
5
6
```

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

Objective

- Manage products
- Track stock accurately
- Enforce business rules
- Prevent inventory inconsistency

Problems Without Proper Inventory Logic

- Duplicate products
- Negative stock
- Manual errors
- No tracking or accountability

Solution

- Centralized stock control
- Validation before update
- Automatic logging and alerts

ARCHITECTURE

Architecture Flow

UI → ProductController → ProductService → ProductRepository → Database

PRODUCT ENTITY DESIGN

ADD PRODUCT – BUSINESS LOGIC

Purpose

SKU ensures uniqueness
stockQuantity is dynamic
minStockLevel triggers alerts

Purpose

Add new product safely
Avoid duplicates

Product Class (Entity)

```
1 CLASS Product
2     productId
3     sku
4     name
5     category
6     unitPrice
7     stockQuantity
8     minStockLevel
9 END CLASS
10
```

Pseudocode

```
1 FUNCTION addProduct(productData)
2     IF sku already exists THEN
3         RETURN "Duplicate Product"
4     ELSE
5         save product
6     END IF
7 END FUNCTION
8
```

STOCK IN NO OPERATION



Pseudocode

```
1 FUNCTION stockIn(productId, quantity)
2     product = find product
3     product.stock += quantity
4     save product
5     log transaction
6 END FUNCTION
7 |
```

Explanation

- Used during purchase or restocking
- No restriction on upper limit

STOCK OUT OPERATION

Pseudocode

```
FUNCTION stockOut(productId, quantity)
    product = find product
    IF product.stock >= quantity THEN
        product.stock -= quantity
        save product
        log transaction
    ELSE
        RETURN "Insufficient Stock"
    END IF
END FUNCTION
```

ALERT TRIGGERING

Purpose

- Prevents stock-out
- Reduces manual monitoring
- Improves efficiency

Pseudocode

```
FUNCTION checkLowStock(product)
    IF product.stock < product.minStockLevel THEN
        send alert
    END IF
END FUNCTION
```

TRANSACTION LOGGING

Purpose

- Full audit trail
- Transparency
- Report generation

Pseudocode

```
FUNCTION logTransaction(type, productId, quantity)
    transaction.date = current date
    transaction.type = type
    transaction.productId = productId
    save transaction
END FUNCTION
```

SUMMARY & STRENGTHS

Key Strengths

- Centralized business logic
- Validation before updates
- Automatic alerts
- Complete audit trail

User / UI



ProductController



ProductService



ProductRepository



Database



Transaction Log + Alert Check

USER TABLE (Authentication Module)

Field	Description
user_id (PK)	Unique user ID
username	Unique login name
password	Encrypted password
email	User email
role	Admin / Staff
reset_token	Password reset token

PRODUCT TABLE (Inventory Module)

Field	Description
product_id (PK)	Unique product ID
sku	Unique product code
name	Product name
category	Product category
unit_price	Price per unit
stock_quantity	Current stock
min_stock_level	Alert threshold

TRANSACTION TABLE (Audit & History)

Field	Description
transaction_id (PK)	Unique transaction ID
product_id (FK)	Related product
transaction_type	STOCK_IN / STOCK_OUT
quantity	Quantity changed
transaction_date	Date & time
performed_by	User ID

ALERT TABLE (Optional - Alerts Module)

Field	Description
alert_id (PK)	Alert ID
product_id (FK)	Related product
alert_type	LOW_STOCK
alert_status	Active / Resolved
created_at	Timestamp

CRITICAL ALERTS

⚠️ MacBook Pro M3 is at 50% of its minimum safety stock level.

⚠️ Wireless Mouse inventory is critically low with only 8 units remaining against a target of 20.

⚠️ LED Monitor 27" has fallen below its 15-unit minimum threshold.

AI STRATEGY

Prioritize immediate replenishment of MacBook Pro units to avoid significant revenue loss on high-margin hardware.

Create a bundled 'Workstation Package' featuring the surplus Ergonomic Desk Chairs and A4 Paper to optimize warehouse space.

Establish a higher safety stock buffer for Wireless Mice to prevent frequent stockouts of essential peripherals.

MARKET OUTLOOK

Demand for professional workstation upgrades remains high, suggesting a risk of supply-side lead time volatility for premium laptops and monitors.

Dashboard Overview

Welcome back, Managing inventory for Warehouse A.

+ New Product



Total Products
5 +12% this month



Total Inventory Value
\$31,030



Low Stock Alerts
3 -2 from yesterday



Critical OOS
0

Stock Levels by Product

Comparing current vs min threshold



Inventory Catalog

Search by product or supplier...

All Categories

PRODUCT	CATEGORY	STOCK	PRICE	SUPPLIER	ACTIONS
MacBook Pro M3 ID: 1	Electronics	5 units LOW STOCK	\$1,999	Apple Inc.	Edit Delete
Ergonomic Desk Chair ID: 2	Furniture	45 units	\$299	Steelcase	Edit Delete
Wireless Mouse ID: 3	Electronics	8 units LOW STOCK	\$49	Logitech	Edit Delete
A4 Printing Paper (Box) ID: 4	Stationery	120 units	\$35	Office Depot	Edit Delete
LED Monitor 27" ID: 5	Electronics	12 units LOW STOCK	\$249	Samsung	Edit Delete

Thankyou