SYNTHCORP

A Smart and Automated Manufacturing System

Presented To:

Sourish Dasgupta

Chosen Programming Language

JAVA
PostgreSQL

Presented By:

Jenil Soni	- 202412108	Leader
Anuj Shah	- 202412090	Member
Archie Shah	- 202412091	Member
Jainam Vora	- 202412122	Member

01 SCENARIO OVERVIEW

- Year 2045: SynthCorp leads in smart, AI-driven automated manufacturing.
- Features: Robotic assembly lines, real-time inventory tracking, self-optimizing machines.

02 KEY ISSUES IN THE EXISTING SYSTEM

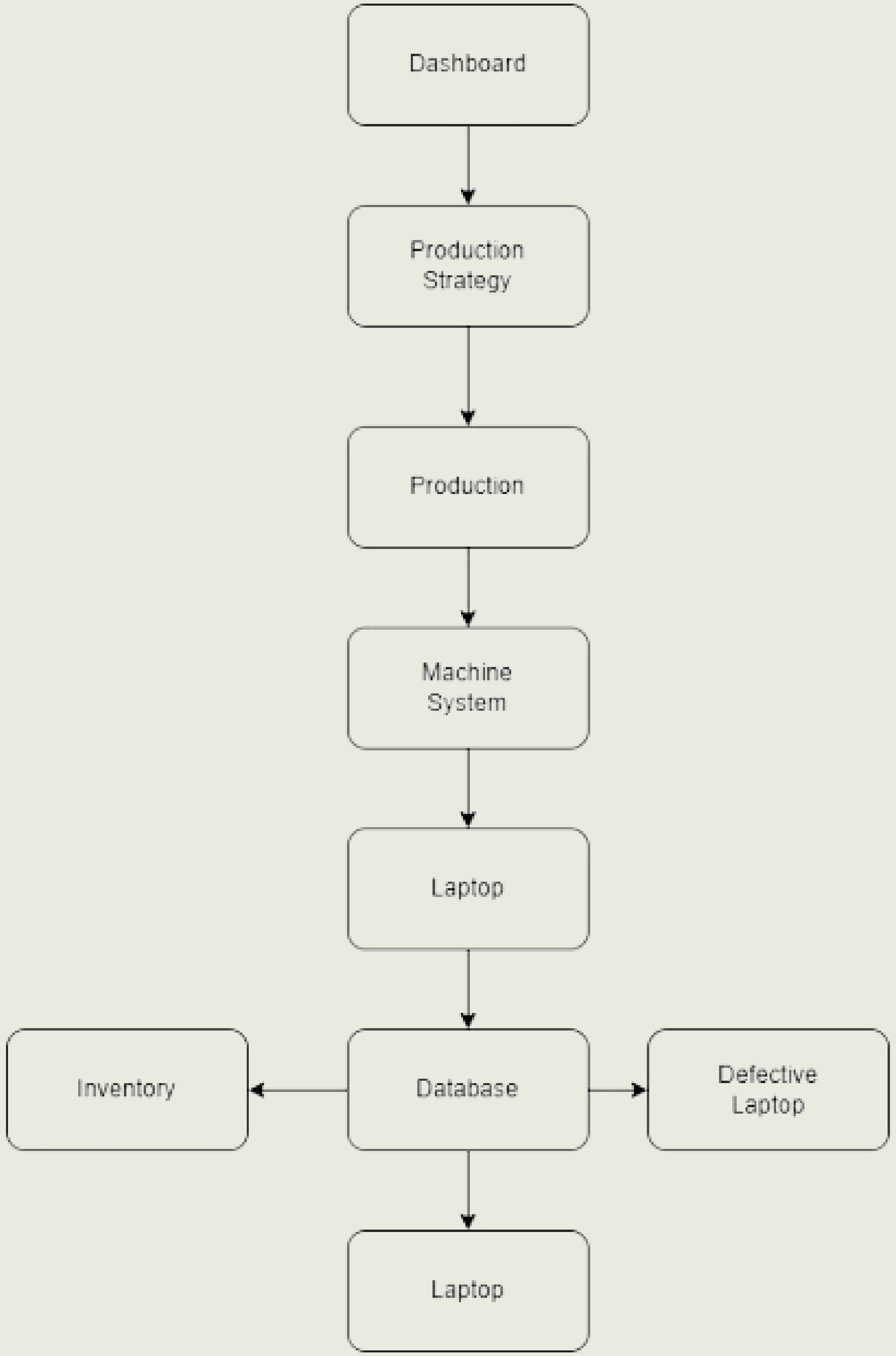
- Robotic arms malfunctioned — incorrect or stopped assembly.
- Random material orders — overstocking or shortages.
- Safety protocols ignored — hazardous work conditions.
- Unstable schedules — delayed shipments, rising costs.

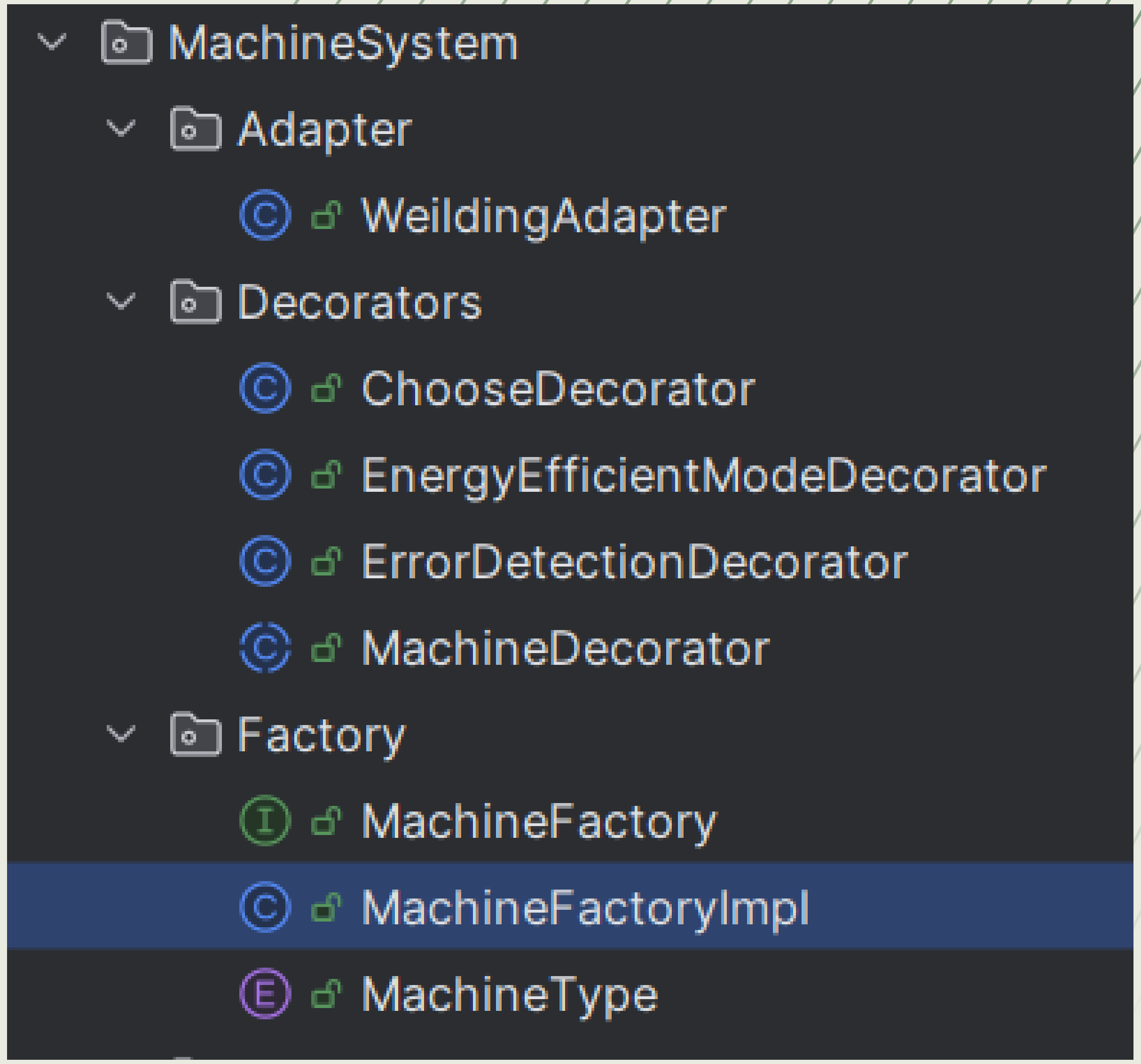
03 OBJECTIVES OF OUR SOLUTION

- Restore system stability and safe operations.
- Redesign system using Object-Oriented Programming.
- Apply design patterns for scalability and maintainability.
- Optimize inventory, scheduling, and production flow.

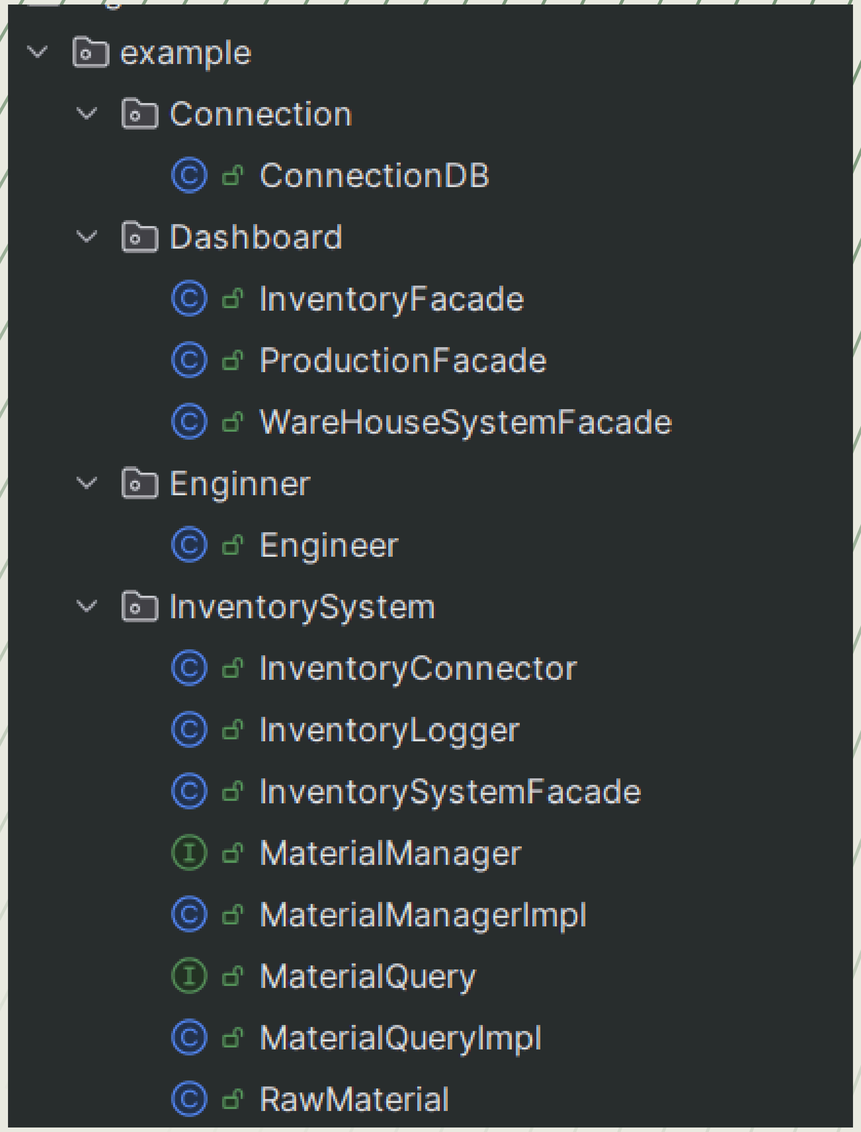
04 OUR AIM

- To create a smarter, safer, and more efficient control system for SynthCorp's next-gen manufacturing.





- MachineSystem
  - Adapter
    - WeildingAdapter
  - Decorators
    - ChooseDecorator
    - EnergyEfficientModeDecorator
    - ErrorDetectionDecorator
    - MachineDecorator
  - Factory
    - MachineFactory
    - MachineFactoryImpl
    - MachineType



- example
  - Connection
    - ConnectionDB
  - Dashboard
    - InventoryFacade
    - ProductionFacade
    - WareHouseSystemFacade
  - Enginner
    - Engineer
  - InventorySystem
    - InventoryConnector
    - InventoryLogger
    - InventorySystemFacade
    - MaterialManager
    - MaterialManagerImpl
    - MaterialQuery
    - MaterialQueryImpl
    - RawMaterial

DESIGN PATTERNS

CREATIONAL PATTERN:

FACTORY DESIGN PATTERN

In this pattern we hide the creation logic of object like in our system we have applied this for creation of machine object.

SINGLETON DESIGN PATTERN

In this pattern only single instance of an class can be created and used we have applied this thing in our database connection as well as object creation.

BUILDER DESIGN PATTERN

In this pattern we can create object step by step.We have used this in creation of customize laptop

DESIGN PATTERNS

STRUCTURAL PATTERN:

FACADE DESIGN PATTERN

In this context, we can draw an analogy to a factory where the production of goods is concealed from view. Similarly, within our own factory, we have implemented a system that obscures the processes of inventory management, production, and database operations.

ADAPTER DESIGN PATTERN

The Adapter Design Pattern allows incompatible interfaces to work together by acting as a bridge between them. It converts the interface of a class into another interface that a client expects.We have applied adapter in our welding machine

DECORATOR DESIGN PATTERN

The Decorator Pattern allows you to dynamically add new behavior to objects without changing their existing code. We have added this for error detection or energy-efficient operation modes.

DESIGN PATTERNS

BEHAVIORAL PATTERN:

OBSERVER DESIGN PATTERN

A subject notifies all subscribed observers automatically when its state changes.We have added a engineer and notified him on the case of error

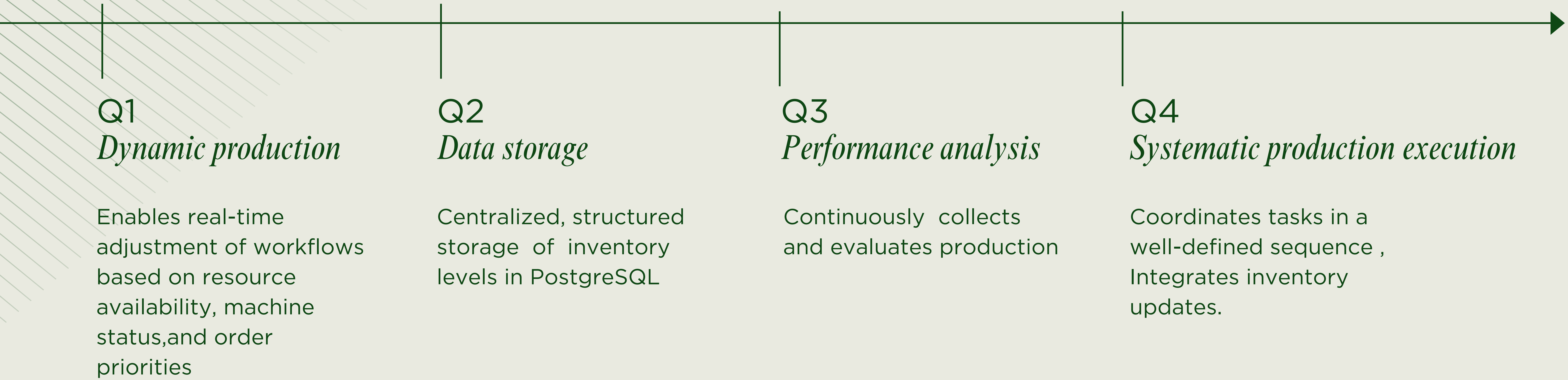
STRATEGY DESIGN PATTERN

Defines a family of algorithms, encapsulates each one, and makes them interchangeable at runtime.We have used this to do production at demand or as per available resources.

STATE DESIGN PATTERN

Allows an object to alter its behavior when its internal state changes, appearing to change its class.we have applied this in material changing state ideal , active or error

CORE FEATURES



Q1
Dynamic production

Enables real-time adjustment of workflows based on resource availability, machine status,and order priorities

Q2
Data storage

Centralized, structured storage of inventory levels in PostgreSQL

Q3
Performance analysis

Continuously collects and evaluates production

Q4
Systematic production execution

Coordinates tasks in a well-defined sequence , Integrates inventory updates.

SUMMARY

Team Challenges

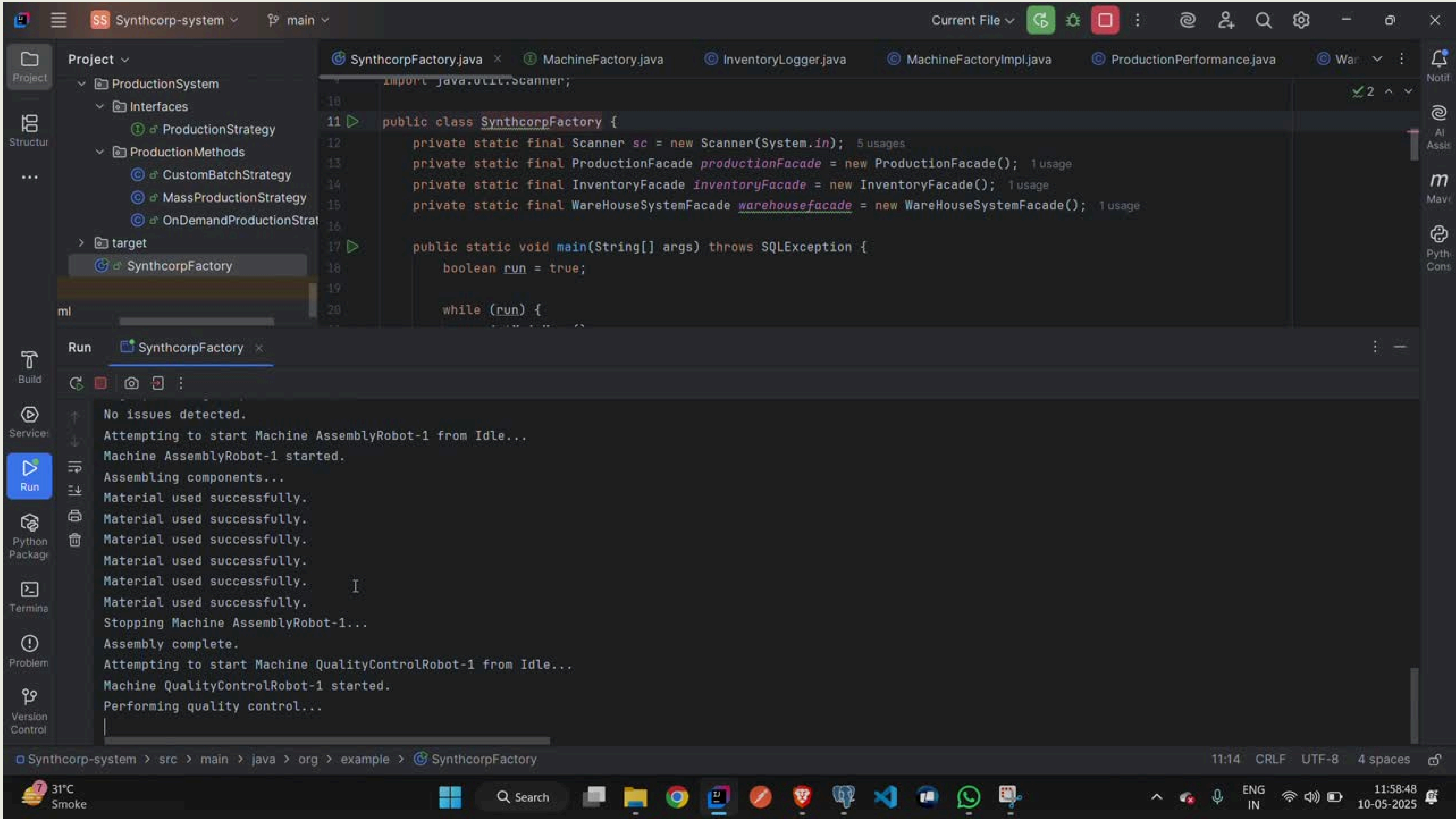
- Database connection using singleton
- Automated production
- Machine state and maintanance

Team Learnings

- Collabarative work
- Problem solving
- Shared design ideas
- Solved bugs together

Future Scope

- Multiple machines for production
- Machine Maintenance



```java
import java.util.Scanner;

public class SynthcorpFactory {
    private static final Scanner sc = new Scanner(System.in);  5 usages
    private static final ProductionFacade productionFacade = new ProductionFacade();  1 usage
    private static final InventoryFacade inventoryFacade = new InventoryFacade();  1 usage
    private static final WareHouseSystemFacade warehousefacade = new WareHouseSystemFacade();  1 usage

    public static void main(String[] args) throws SQLException {
        boolean run = true;

        while (run) {
```

```
No issues detected.
Attempting to start Machine AssemblyRobot-1 from Idle...
Machine AssemblyRobot-1 started.
Assembling components...
Material used successfully.
Material used successfully.
Material used successfully.
Material used successfully.
Material used successfully.
Material used successfully.
Stopping Machine AssemblyRobot-1...
Assembly complete.
Attempting to start Machine QualityControlRobot-1 from Idle...
Machine QualityControlRobot-1 started.
Performing quality control...
```

Thank you.

Synthcorp