Industrial Software Development Project

E-commerce website

2025 / 2026

Team members:
Johannes **KRAUTHEIMER**, François **BESNARD**, Mathis **LECOQ**, Louis **THOMAS-GIRARDEY**
Supervisor:
Angelo SOTGIU
Date:
10/03/2026

Team Organization

What we built and how we organized the work

Team & Responsibilities

Member	Role	Features
François	[Dev]	Feature 1
Mathis	[Dev]	Feature 2
Johannes	[Dev]	Feature 3
Louis	[Lead, Dev]	Feature 4, Testing, CI / Docker

Tools used:





Agile Development Process

1

Backlog & Board

Issues created on GitHub, linked to Project board. Tasks split into sub-tasks and assigned.

2

Branching

One branch per feature/issue on the forked repo. Feature branches merged into main after collective review.

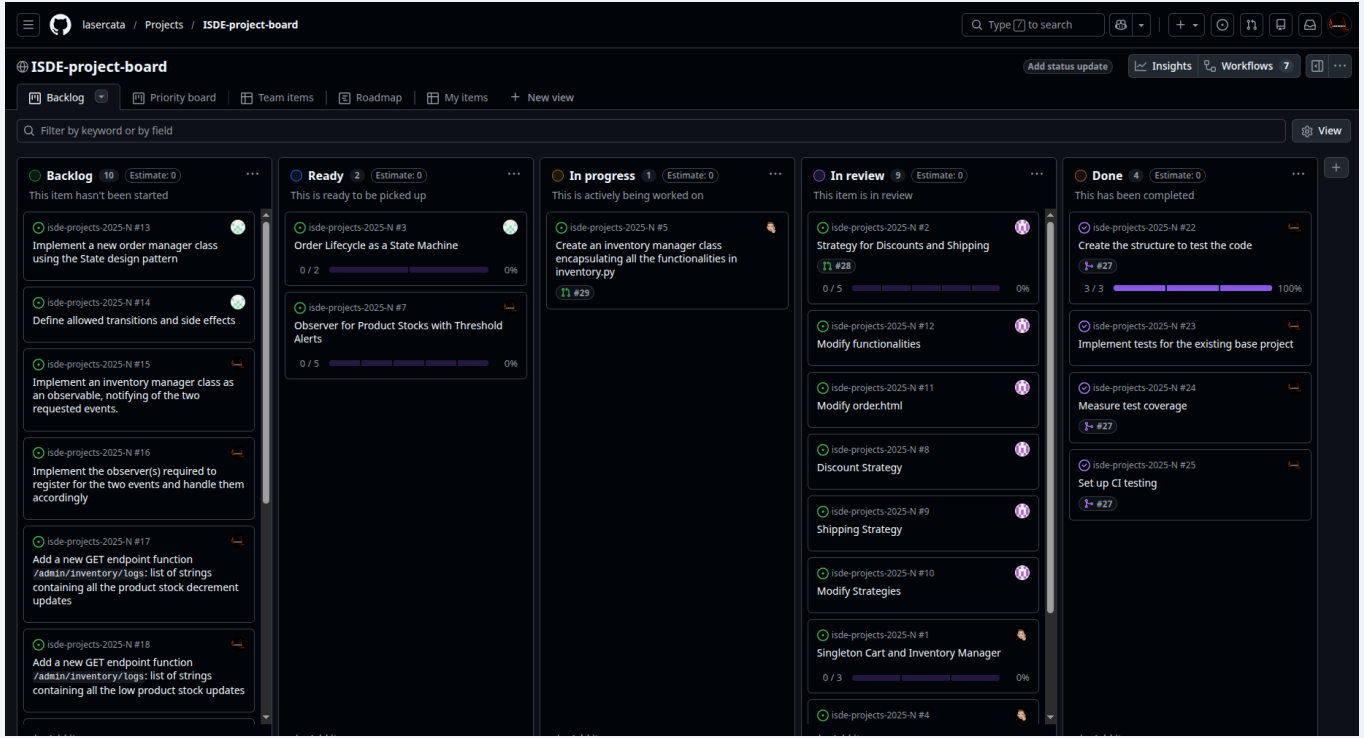
3

Pull Requests

PR with changelog & documentation sent to team's original repository after all features merged & tested.

Our Project Board

Overview of the Project Board with tasks distributed across different stages



Quality Assurance & CI/CD

Tests, coverage and continuous integration

Unit Tests

- pytest framework
- Each new feature has dedicated test module
- Edge cases & invalid inputs covered
- Informative error messages validated

Test Coverage

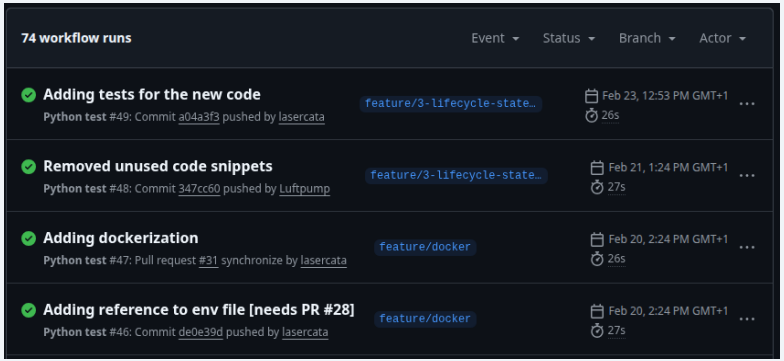
- Test coverage measured with pytest
- Coverage report generated in CI and added to the README
- Target: ≥ 80% coverage
- 99% achieved

CI with GitHub Actions

- Workflow triggers on every push & PR
- Runs linting (flake8 / pylint)
- Runs full test suite automatically
- Coverage badge in README

```
============================= tests coverage ==============================
_____________________ coverage: platform linux, python 3.14.3-final-0 _____

Name                          Stmts   Miss  Cover   Missing
-----------------------------------------------------------
app/__init__.py                   0      0   100%
app/cart.py                      20      0   100%
app/config.py                    23      4    83%   46, 48, 53, 55
app/discount_strategy.py         26      0   100%
app/inventory.py                107      0   100%
app/main.py                      64      0   100%
app/orders.py                    66      0   100%
app/pydantic_models.py           25      0   100%
app/shipping_strategy.py         38      0   100%
app/state_machine.py             43      0   100%
-----------------------------------------------------------
TOTAL                           412      4    99%
============================== 41 passed in 0.37s ==============================
```



Feature 1 — Singleton Cart and Inventory Manager

Issue #1 · Implemented by François

WHY — Motivation

A single inventory and shopping cart instance is necessary to ensure that data is consistent and up to date. This prevents inconsistencies such as an item being sold multiple times or while it was out of stock.

WHAT — Functionality

This feature ensures that the entire application shares a single instance of the shopping cart and inventory managers, guaranteeing that all operations always reflect the same up-to-date state.

HOW — Technical Approach

- Implementation of **InventoryManager** as a Singleton and adaptation of all its methods to operate through get_instance()
- Implementation of **CartManager** as a Singleton and adaptation of all its methods to operate through get_instance()
- Unit tests for both classes covering instantiation prevention, Singleton behavior, and core method logic
- Error handling (RuntimeError, ValueError) and docstring documentation



- Review test coverage and document the approach for testing individual code components
- Generate presentation structure

Feature 2 — Strategy for Discounts and Shipping

Issue #2 · Implemented by Mathis

WHY — Motivation

Dynamic calculation of discounts and shipping costs with a Strategy Design Pattern to makes code easier to maintain, understand, and reuse.

WHAT — Functionality

The order detail view now exposes a structured price breakdown:
- Subtotal
- Discount
- Shipping
- Total

HOW — Technical Approach

- Implementation of the **Discount Strategy** :
  NoDiscount / **PercentageDiscount** / **ThresholdFixedDiscount**
  Each implements the calculate_discount() method
- Implementation of the **Shipping Strategy** :
  FlatRate / **FreeOver** / **WeightBased**
  Each implements the calculate_shipping() method
- Added tests and error handling system for each strategy
- Added environment variables to select strategy



- Create the structure of the Strategy Design Pattern
- Generate docstrings documentation

Feature 3 — Order Lifecycle as a State Machine

Issue #3 · Implemented by Johannes

WHY — Motivation

Reinforcement of a stable execution of the pipeline of a single order without any unwanted behavior as shipping without paying

WHAT — Functionality

Implementation of a State Machine, which tracks for every order their current status and ensure a correct transition to the next step

HOW — Technical Approach

- Implemented a new class "**StateMachine**", which handles all the possible states of an order and their possible transitions
- Creating a unique State Machine every time a order is created, which is mapped to that order
- Used a blueprint class "State" for all states to allow an easy implementation of new possible steps in the process of an order



- Generate docstrings documentation

Feature 4 — Observer for Product Stocks with Threshold Alerts

Issue #7 · Implemented by Louis

WHY — Motivation

Monitoring the stock level is important for the shop manager, for example to reorder items when stock is low.

WHAT — Functionality

Implementation of an automatic **notification system** for product stock levels:

- Product stock **decremented**;
- Product stock reaches reorder **threshold.**

The threshold can be adjusted for each product

HOW — Technical Approach

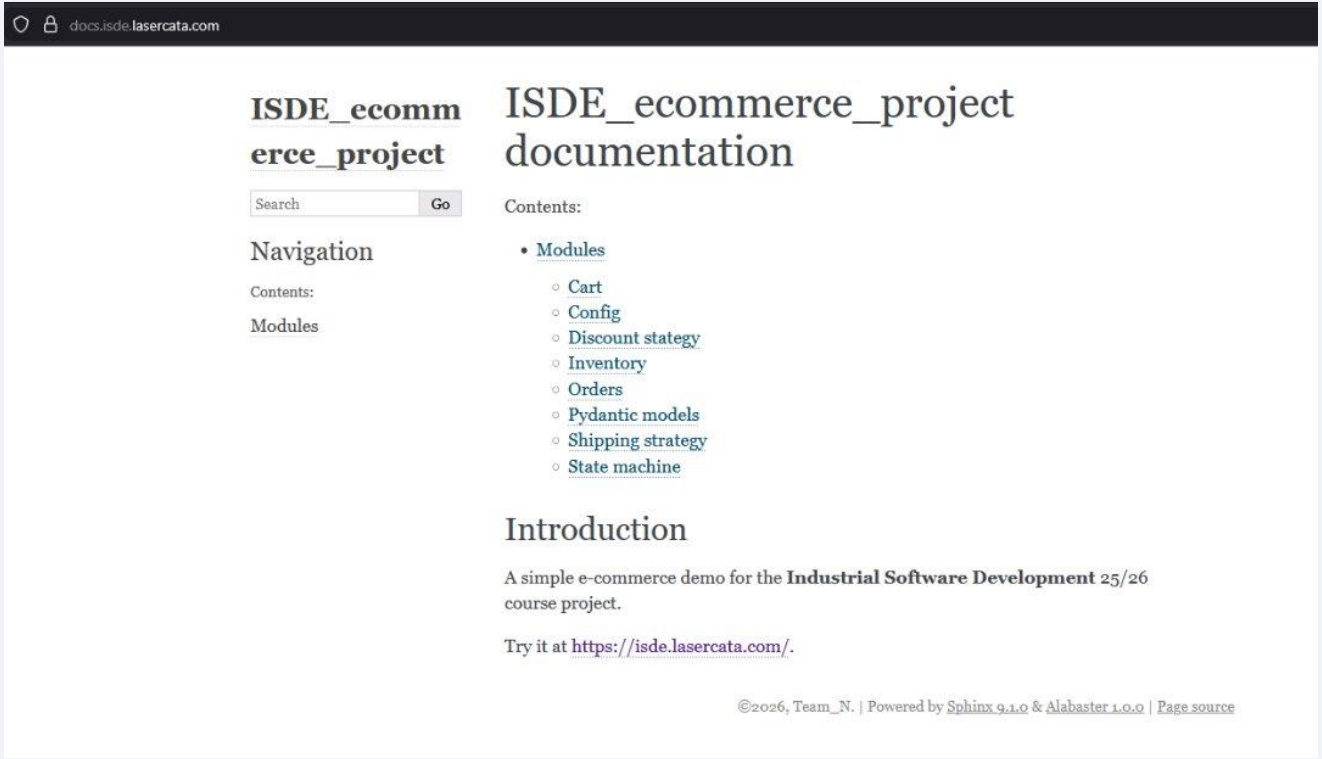
- Followed the **Observer** design pattern
- Modified InventoryManager to publish on events
- Implemented new class InventoryObserver that subscribes to InventoryManager
- Wrote tests for the new code



- Create base for the tests
- Debug
- Web research about syntax framework

Project documentation

Overview of the Sphinx project documentation



Project URLs :

Website
https://isde.lasercata.com

Documentation
https://docs.isde.lasercata.com

Video

User experience on the application

