

Agile Endsem Project

(CSE706)

Bachelor of Technology (CSE)

By

Dhruhi Shah (22000381)

Paryapti Macwan (22000401)

4th Year, Semester 7



Department of Computer Science and Engineering

School Engineering and Technology

Navrachana University, Vadodara

Autumn Semester

(July- Nov 2025)

1. PROJECT DESCRIPTION

1.1 Initiating the Agile Project

The project aims to develop a simple *Inventory Management System* using Agile development practices. The focus is on rapid iteration, incremental delivery, continuous testing, and collaboration.

1.2 Product Vision

“To build a lightweight, reliable, and easy-to-use Inventory Management System that allows users to create, update, view, and delete products efficiently.”

1.3 Project Goal

- Provide basic inventory CRUD features
- Offer a maintainable codebase following SOLID principles
- Demonstrate Agile practices (Scrum, TDD, CI, backlog management)

Abstract

This project demonstrates the implementation of an **Inventory Management System** using **Agile Software Development Practices**, including Scrum ceremonies, TDD, CI/CD, GitHub version control, and metrics tracking.

The objective of this project is to manage inventory operations such as adding, updating, viewing, and deleting products, along with a low-stock alert feature.

The project follows a **two-sprint Agile cycle** with Scrum events such as Sprint Planning, Daily Scrum, Sprint Review, and Sprint Retrospective. Testing was performed through **JUnit Unit Tests** and **Functional Tests**, adhering to the Agile Testing Quadrants.

The system is implemented using Java, Maven, and follows SOLID principles for clean code and maintainability.

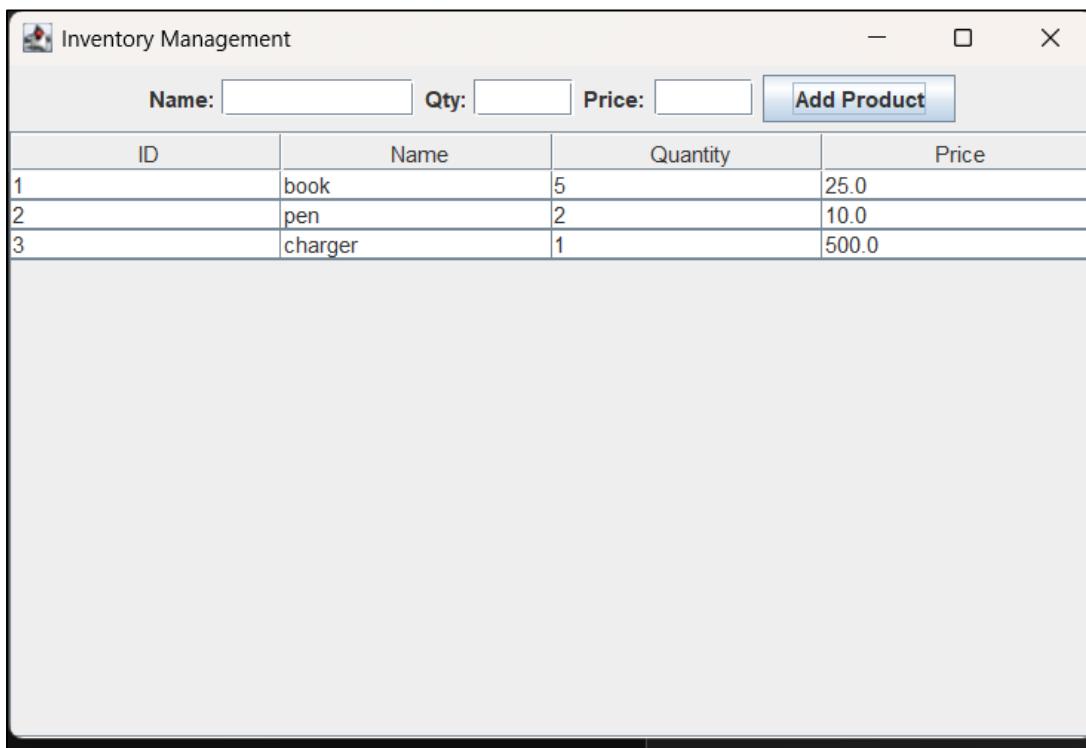
2. Introduction

An Inventory Management System helps track product quantities, pricing, and stock levels.

This project focuses on building a mini console-based version using industry-standard Agile practices.

The main features include:

- Add new products
- Update product information
- Delete product entries
- Fetch all products
- Low-stock product alert
- Test-driven modules
- GitHub version control
- CI pipeline with GitHub Actions



The screenshot shows a Windows-style application window titled "Inventory Management". At the top, there are three input fields labeled "Name:", "Qty:", and "Price:" followed by a blue "Add Product" button. Below this is a table with four columns: "ID", "Name", "Quantity", and "Price". The table contains three rows of data:

ID	Name	Quantity	Price
1	book	5	25.0
2	pen	2	10.0
3	charger	1	500.0

3. Agile methodology

We followed the Scrum framework, which includes:

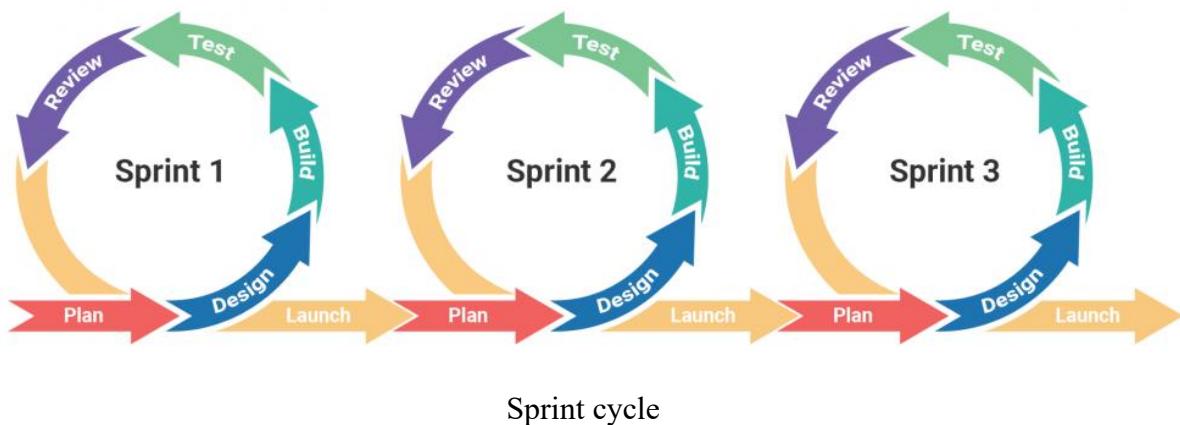
Scrum Roles

- Product Owner: Defines product vision, writes user stories
- Scrum Master: Facilitates meetings, removes blockers
- Development Team: Implements features and tests

Here Paryapti Macwan was the product owner and the developer, and Dhruhi Shah was the scrum master and developer with devops and testing.

Scrum Ceremonies

- Product Backlog Creation
- Sprint Planning
- Daily Scrum (15-minute updates)
- Sprint Development Work
- Sprint Review
- Sprint Retrospective



Sprint cycle

4. Product vision and goal

Product Vision

“To build a simple, fast, and reliable Inventory Management System that enables users to track stock levels efficiently and make inventory decisions quickly.”

Project Goals

- Implement a clean Console-based CRUD Inventory System
- Follow Agile development practices end-to-end
- Demonstrate testing discipline using TDD & functional testing
- Maintain code quality through SOLID principles
- Track progress using metrics

5. User Stories with Acceptance Criteria

User story: A user story is a concise, plain-language description of a software feature from an end-user perspective, outlining what they want to do and why.

Acceptance Criteria: In agile, acceptance criteria are a set of predefined requirements that a user's story or feature must meet to be considered complete and accepted by stakeholders.

User Story	Description	Acceptance Criteria
US-01	As a user, I want to add a product	Product is stored with name, quantity, price; validated input
US-02	As a user, I want to update a product	User can modify name, quantity, and price
US-03	As a user, I want to delete a product	Product removed; cannot fetch it afterwards
US-04	As a user, I want to view all products	List returns all stored items
US-05	As a user, I want low-stock alerts	Items below threshold appear in list

[Add epic](#) / [IMSI-4](#)

As a user, I want to add a new product so that I can maintain stock records.

[+](#) [○](#)

Key details

Priority	High
----------	------

Description

Given the user is on the Add Product page
When the user enters valid product details
And clicks the "Add" button
Then the system should save the product
And the product should appear in the product list.

User story for add a new product

The screenshot shows the Jira Backlog view for the 'Inventory Management System (IMSI)' project. The backlog is organized into sections: 'Basic inventory CRUD features' and 'Backlog'. The 'Basic inventory CRUD features' section contains three user stories:

- IMSI-4: As a user, I want to add a new product so that I can maintain stock records. Status: TO DO, Priority: 5, Assignee: Unassigned.
- IMSI-5: As a user, I want to update product details so that s... Status: TO DO, Priority: 5, Assignee: Unassigned.
- IMSI-7: As a user, I want to view all products so that I can t... Status: TO DO, Priority: 3, Assignee: Unassigned.

The 'Backlog' section contains three more user stories:

- IMSI-6: As a user, I want to delete a product so that I can re... Status: TO DO, Priority: 3, Assignee: Unassigned.
- IMSI-8: As a user, I want low-stock alerts so that I can reor... Status: TO DO, Priority: 8, Assignee: Unassigned.
- IMSI-9: As a user, I want reports so that I can check stock l... Status: TO DO, Priority: 5, Assignee: Unassigned.

On the right side of the screen, there is a detailed view of the first user story (IMSI-4), which includes fields for 'Assignee' (Unassigned), 'Labels' (Add labels), 'Parent' (Add parent), and 'Due date' (Add due date).

Jira Userstories

```

# Acceptance Criteria for User Stories

## User Story 1: Add Product
**As a** store manager
**I want to** add new products
**So that I can** maintain inventory records.

### Acceptance Criteria
- **Given** valid product details (name, quantity, price)
- **When** the user submits the add request
- **Then** the system should create a new product with a unique ID.

- Product must not be created if:
  - Name is empty
  - Quantity is negative
  - Price is negative

- System should return the created product object.

---

## User Story 2: Update Product
**As a** store manager
**I want to** update product details
**So that I can** keep inventory accurate.

```

Acceptancecriteria.md

6. Product Backlog (Prioritized)

ID	User Story	Priority	Story Points
PB-01	Add Product	High	5
PB-02	Update Product	High	3
PB-03	Delete Product	High	3
PB-04	List All Products	Medium	2
PB-05	Low-stock Alerts	Medium	2
PB-06	Testing (TDD + Functional)	High	5
PB-07	GitHub + CI Setup	Low	2
PB-08	Documentation	Medium	2

ProductBacklog

7. Sprint Planning

We executed **2 Sprints**, each of 1 week.

Sprint 1 – CRUD Operations

Sprint Goal

Complete Add, Update, Delete, and Fetch functionalities.

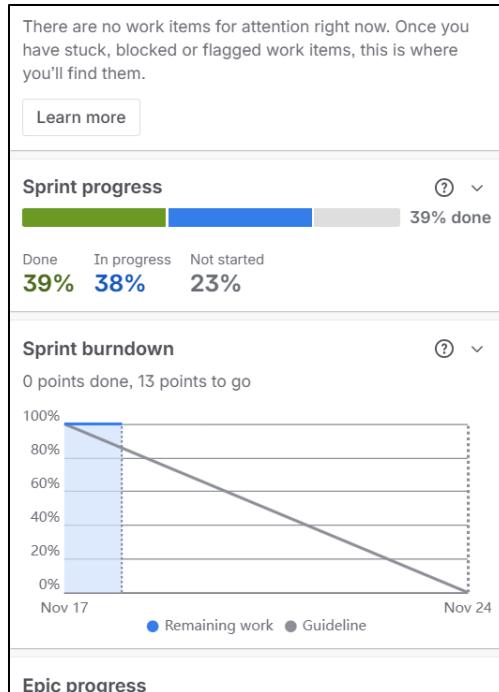
Sprint Backlog

- Add Product module
- Update Product module
- Delete Product module
- List All Products
- Unit tests (TDD for Add & Update)
- Documentation setup

Daily Scrum Notes (Sample)

- **Day 1:** Setup project structure, created repo interface
- **Day 2:** Implemented Add Product
- **Day 3:** Implemented Update Product
- **Day 4:** Implemented Delete Product
- **Day 5:** Wrote JUnit tests
- **Day 6:** Manual testing and debugging
- **Day 7:** Sprint Review + Retrospective

Sprint board 1



Burndown chart

The screenshot shows the Jira Sprint board 2 interface. On the left, a sidebar includes "For you", "Spaces" (selected), "Recent", "Inventory Management...", "software team", "More spaces", "Recommended", "Create a roadmap", and "More".

The main area displays the "Inventory Management System (IMS)" board. It has sections for "TO DO", "IN PROGRESS", and "DONE".

- TO DO:**
 - + Create
 - As a user, I want to update product details so that stock information stays accurate. (IMSI-5)
 - As a user, I want to view all products so that I can track the inventory. (IMSI-7)
- IN PROGRESS:**
 - As a user, I want to add a new product so that I can maintain stock records. (IMSI-4)
- DONE:**
 - 1 ✓

Sprint insights

View your sprint health and progress towards your goals.
Sprint: Basic inventory CRUD features

Work items for attention

There are no work items for attention right now. Once you have stuck, blocked or flagged work items, this is where you'll find them.

Sprint progress

A horizontal bar chart showing progress: 38% Done (green), 62% In progress (blue), and 0% Not started (grey). The total is 38% done.

Status	Percentage
Done	38%
In progress	62%
Not started	0%

Sprint burndown

5 points done, 8 points to go.

A burndown chart showing remaining work from Nov 17 to Nov 24. The Y-axis represents points from 40% to 100%. The X-axis shows dates Nov 17 and Nov 24. A grey line represents the guideline, and a blue line represents the remaining work, which starts at approximately 90% and ends at 0% by Nov 24.

Nov 17 Nov 24

You've created "IMSI-22" work item

View · Copy link

Quickstart

Sprint board 2

Sprint 1 Review

Completed:

- All CRUD functionalities
- Unit tests
- Basic documentation

Demo Outcome:

Application successfully performs add, update, delete, and list operations.

Sprint progress

38% done

Done In progress Not started

38% 62% 0%

Sprint burndown

5 points done, 8 points to go

Nov 17 Nov 24

Remaining work Guideline

Epic progress

Link epics and estimate work items to drive your big goals

This insight shows how your current sprint contributes to your larger goals, or epics, helping to maintain focus at a perspective.

Comments & notes

Pro tip: press M to comment

PM Paryapti Macwan 15 seconds ago

Day 2:

- Yesterday: Add Product UI completed
- Today: Implement backend add function
- Blockers: Data validation

Reply Edit Delete

PM Paryapti Macwan 36 seconds ago

- Yesterday: Sprint planning completed
- Today: Start Add Product UI
- Blockers: None

Reply Edit Delete



Complete Basic inventory CRUD features

This sprint contains 3 completed work items.
That's all of them - well done!

Create a retrospective for this sprint
Finish off your sprint with a Confluence retrospective! Contribute to your team's culture and improve how you work.

[Cancel](#) [Complete sprint](#)

Sprint 1 completed

Sprint 1 Retrospective

What Went Well

- Clear understanding of task breakdown
- Faster development due to small sprint size
- Tests helped reduce bugs

What Could Be Improved

- Write acceptance criteria earlier
- Improve naming conventions

Action Items

- Maintain consistent coding style
- Expand testing coverage

➡ Retrospective: Basic inventory CRUD features

By Paryapti Macwan 1 min See views Add a reaction

Reflect on past work and identify opportunities for improvement by following the instructions for the [Retrospective Play](#).

☰ 📋 Overview

Date	Nov 18, 2025
Team	Team Inventory
Participants	Dhruhi, Paryapti

💡 Sprint Retrospective – CRUD Operations Sprint

Sprint Goal

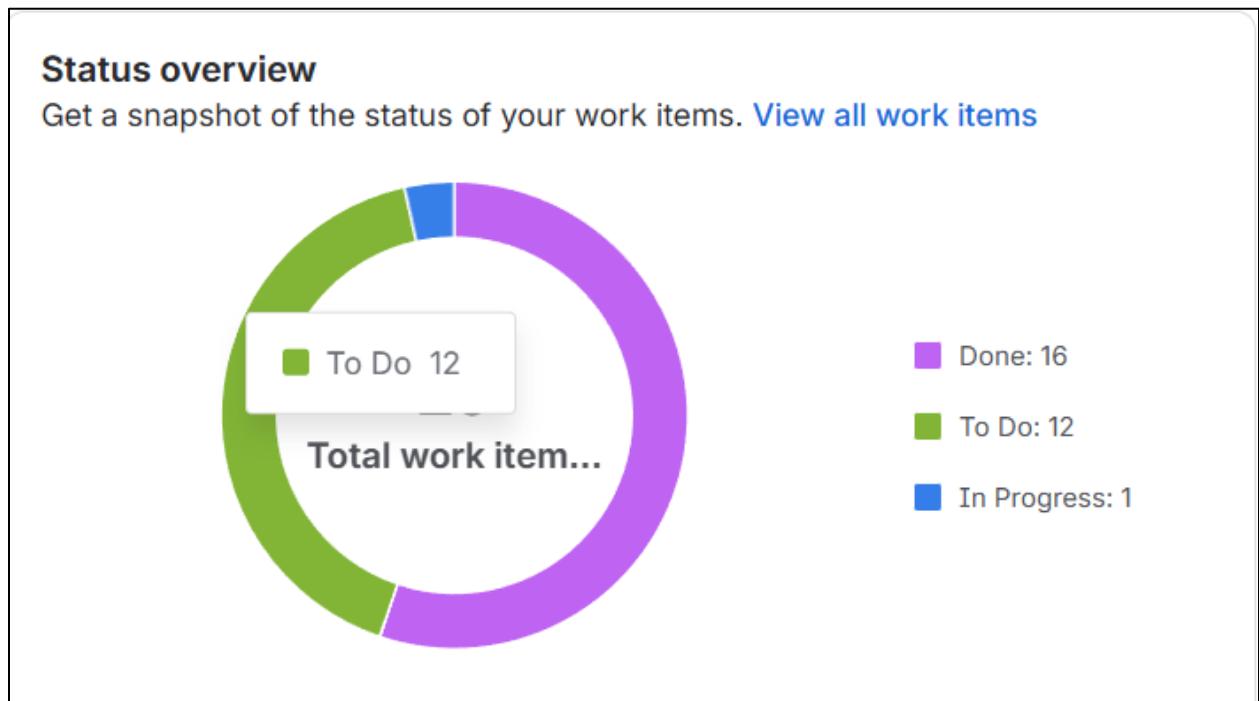
Sprint 1 retrospective

Sprint Goal

- + :: Build the core CRUD functionality (Create, Read, Update, Delete) for the Inventory Management System and ensure basic operations work correctly.

➡ What went well	🔴 What could be improved	💡 What we learned
<ul style="list-style-type: none">• Clear requirements made implementation straightforward.• Repository pattern (using <code>InMemoryProductRepository</code>) made development flexible.• Unit tests executed successfully, validating add/update/delete logic early.• TDD helped catch issues early, reducing debugging time.	<ul style="list-style-type: none">• No input validation layer yet → system accepts negative quantity/price.• Some test cases had issues due to early mistakes with return types.• Need to improve naming conventions and comments for better readability.• Functional testing was added later — could have been planned earlier.• Difficulties in Java version control.	<ul style="list-style-type: none">• Importance of writing tests before code (TDD mindset).• How Maven compiles tests differently from main code.• How to structure Java applications using services and repositories.• How crucial clean separation of responsibilities is (SOLID).• Git setup and version control are critical to track changes and improvements.

Sprint 1 retrospective notes



Work to do after sprint 1

Sprint 2 – Testing, Low-Stock, CI/CD

Sprint Goal

Complete low-stock module + functional testing + GitHub Actions CI.

Sprint Backlog

- Implement low-stock logic
- Functional tests
- GitHub Actions CI pipeline
- Testing Quadrants documentation
- Architecture diagram

Search backlog Filter

Alerts, deletion,reports, test 18 Nov – 25 Nov (3 work items)

Alerts, deletion,reports, test

IMSI-8 As a user, I want low-stock alerts so that I can reorder items on time. DONE 8 8 1 User

IMSI-9 As a user, I want reports so that I can check stock levels anytime. IN PROGRESS 5 5 1 User

IMSI-6 As a user, I want to delete a product so that I can remove unwanted items. TO DO 3 3 1 User

+ Create

Backlog (0 work items) Create sprint

Your backlog is empty.

+ Create

Sprint 2

Spaces Inventory Management System (IMS) Summary Timeline Backlog Board Calendar List Forms Goals

Search board Filter

TO DO 1

As a user, I want to delete a product so that I can remove unwanted items.

IMSI-6 3 8 1 User

+ Create

IN PROGRESS 1

As a user, I want reports so that I can check stock levels anytime.

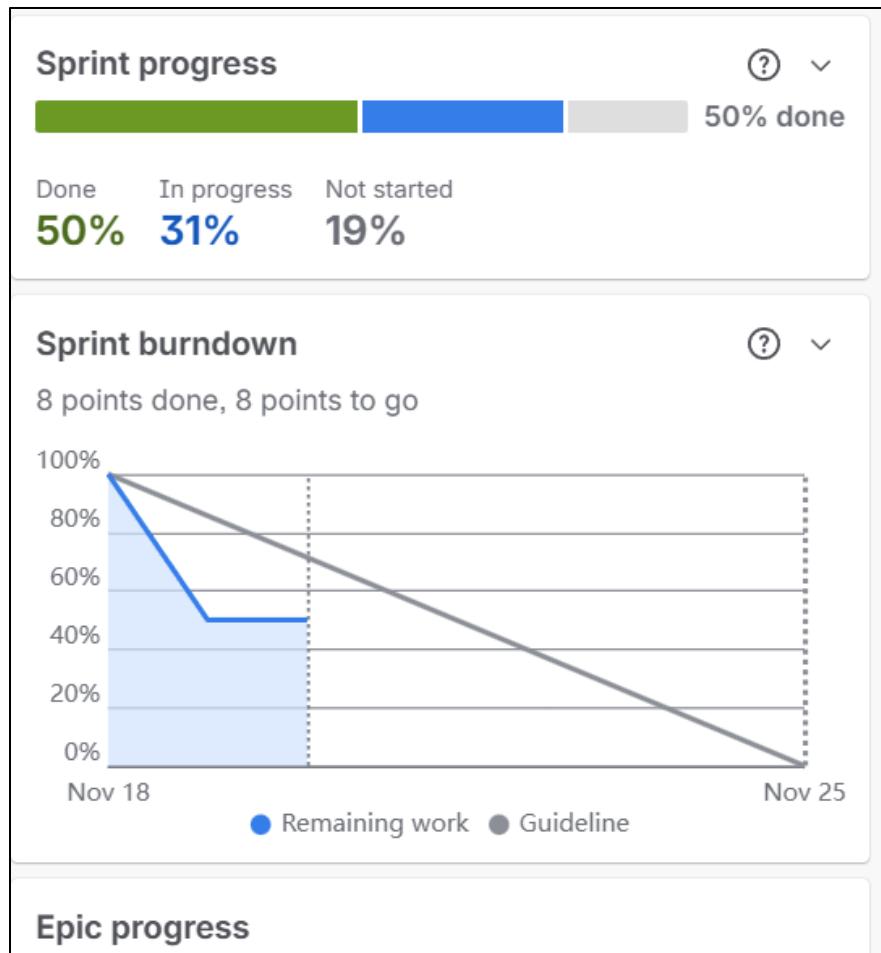
IMSI-9 5 8 1 User

DONE 1 ✓

As a user, I want low-stock alerts so that I can reorder items on time.

IMSI-8 8 8 1 User

Sprint 2 board



Burndown chart Sprint 2

Daily Scrum Notes (Sample)

- **Day 1:** Wrote low-stock logic
- **Day 2:** Implemented functional test
- **Day 3:** Pushed to GitHub
- **Day 4:** Configured CI workflow
- **Day 5:** Documentation
- **Day 6:** Bug fixing
- **Day 7:** Sprint Review + Retrospective

Sprint 2 Review

Completed:

- Low-stock feature
- Functional testing
- GitHub Actions CI pipeline
- Documentation & diagrams

Demo Result:

Build passes CI pipeline, all tests green.

Sprint 2 Retrospective

What Went Well

- CI pipeline automated testing
- Low-stock logic correct
- Functional test ensuring workflow

What Could Be Improved

- More branching discipline on GitHub

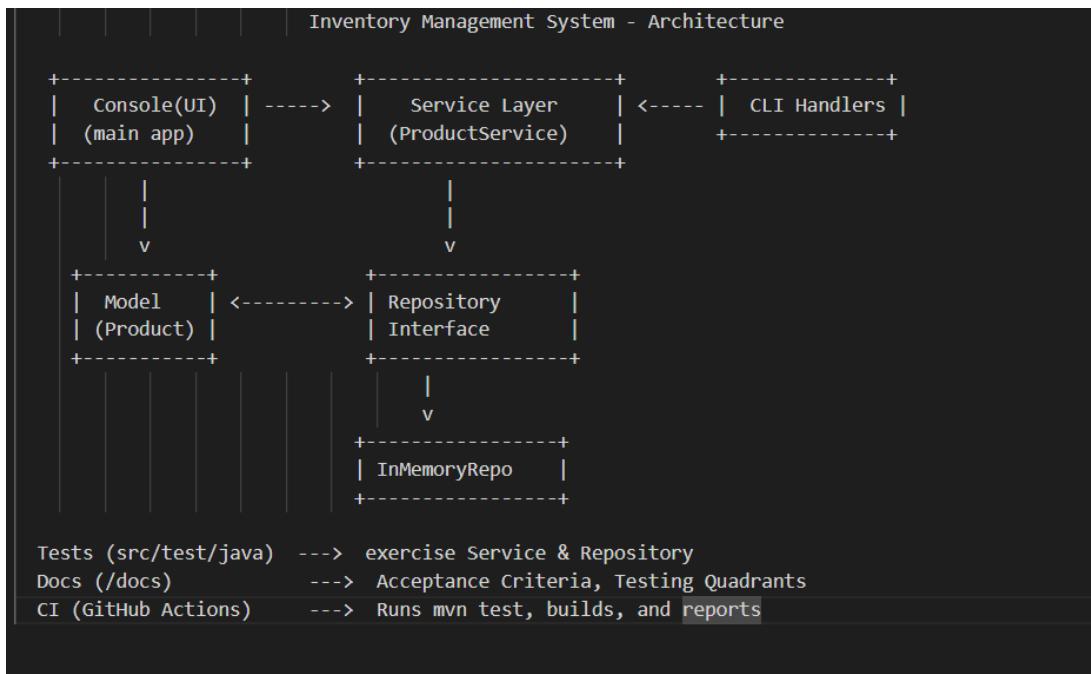
Action Items

- Use feature branches for future enhancements

10	11	+	12	13	14
17	18	+	19	20	21
	↳ Basic inventory CRUD features				
24	25	+	26	27	28
	↳ Alerts, deletion, reports, test				

Planning

8. Architecture Diagram



9. Design Principles (SOLID)

S – Single Responsibility

`ProductService` handles product logic only.

It has a single responsibility: **managing business logic for products**.

Other concerns (storage, UI, tests, controllers) are properly separated.

```

public class ProductService {

    private final ProductRepository repo;
    private int autoId = 1;

    public ProductService(ProductRepository repo) {
        this.repo = repo;
    }

    public Product addProduct(String name, int qty, double price) {
        Product p = new Product(autoId++, name, qty, price);
        repo.add(p);
        return p;
    }

    public void updateProduct(int id, String name, int qty, double price) {
        Product p = new Product(id, name, qty, price);
        repo.update(p);
    }

    public void deleteProduct(int id) {
        repo.delete(id);
    }
}

```

O – Open/Closed

Repository interface allows new storage types.

```
src > main > java > com > example > inventory > repository > ProductRepository.java > ...
1 package com.example.inventory.repository;
2 |
3
4 import com.example.inventory.model.Product;
5 import java.util.List;
6
7 public interface ProductRepository {
8     void add(Product product);
9     Product findById(int id);
10    List<Product> findAll();
11    void update(Product product);
12    void delete(int id);
13 }
14
```

L – Liskov Substitution

Any repository implementation can replace another.

I – Interface Segregation

Repository only exposes required methods.

D – Dependency Inversion

Service depends on the repository interface, not concrete classes.

```
src > main > java > com > example > inventory > repository > ProductRepository.java > ...
1 package com.example.inventory.repository;
2 |
3
4 import com.example.inventory.model.Product;
5 import java.util.List;
6
7 public interface ProductRepository {
8     void add(Product product);
9     Product findById(int id);
10    List<Product> findAll();
11    void update(Product product);
12    void delete(int id);
13 }
14
```

Product Repository

```
package com.example.inventory.service;

import com.example.inventory.model.Product;
import com.example.inventory.repository.ProductRepository;
import java.util.List;
import java.util.stream.Collectors;

public class ProductService {

    private final ProductRepository repo;
    private int autoId = 1;

    public ProductService(ProductRepository repo) {
        this.repo = repo;
    }

    public Product addProduct(String name, int qty, double price) {
        Product p = new Product(autoId++, name, qty, price);
        repo.add(p);
        return p;
    }
}
```

Product service depends on product repository interface not concrete classes

10. Test-Driven Development (TDD)

TDD cycle followed:

Red → Green → Refactor

Example Module

- Add Product
- Update Product

Both were implemented using JUnit 5.

```
[INFO] --- exec:3.6.2:java (default-cli) @ inventory-management-system ---
[WARNING]
java.lang.ClassNotFoundException: com.example.inventory.InventoryApplication
    at org.codehaus.mojo.exec.URLClassLoaderBuilder$ExecJavaClassLoader.loadClass (URLClassLoaderBu
    at java.lang.ClassLoader.loadClass (ClassLoader.java:525)
    at org.codehaus.mojo.exec.AbstractExecJavaBase.doExecClassLoader (AbstractExecJavaBase.java:376)
    at org.codehaus.mojo.exec.AbstractExecJavaBase.lambda$execute$0 (AbstractExecJavaBase.java:287)
    at java.lang.Thread.run (Thread.java:840)
[INFO] -----
[INFO] BUILD FAILURE
[INFO] -----
[INFO] Total time:  1.532 s
[INFO] Finished at: 2025-11-18T23:59:11+05:30
[INFO] -----
[ERROR] Failed to execute goal org.codehaus.mojo:exec-maven-plugin:3.6.2:java (default-cli) on project
      executing the Java class. com.example.inventory.InventoryApplication -> [Help 1]
[ERROR]
[ERROR] To see the full stack trace of the errors, re-run Maven with the -e switch.
[ERROR] Re-run Maven using the -X switch to enable full debug logging.
[ERROR]
[ERROR] For more information about the errors and possible solutions, please read the following art
[ERROR] [Help 1] http://cwiki.apache.org/confluence/display/MAVEN/MojoExecutionException
```

Red

```
[INFO] --- compiler:3.13.0:testCompile (default-testCompile) @ inventory-management-system ---
[INFO] Recompiling the module because of changed source code.
[WARNING] File encoding has not been set, using platform encoding windows-1252, i.e. build is plat
[INFO] Compiling 4 source files with javac [debug target 1.8] to target\test-classes
[WARNING] bootstrap class path not set in conjunction with -source 8
[INFO]
[INFO] --- surefire:3.1.0:test (default-test) @ inventory-management-system ---
[INFO] Using auto detected provider org.apache.maven.surefire.junitplatform.JUnitPlatformProvider
[INFO]
[INFO] -----
[INFO] T E S T S
[INFO] -----
[INFO] Running com.example.inventory.functional.ProductFunctionalTest
[INFO] Tests run: 2, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.106 s - in com.example.in
[INFO] Running com.example.inventory.service.LowStockTest
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.002 s - in com.example.in
[INFO] Running com.example.inventory.service.ProductServiceTest
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.003 s - in com.example.in
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 4, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time:  5.388 s
[INFO] Finished at: 2025-11-18T22:16:18+05:30
-----
```

Green

11. Testing and Validation

11.1 Unit Tests

- testAddProduct()
- testUpdateProduct()
- testLowStockProducts()

```
INFO] T E S T S
INFO] -----
INFO] Running com.example.inventory.service.LowStockTest
INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.042 s - in com.example.inventory.service.LowStockTest
INFO] Running com.example.inventory.service.ProductServiceTest
INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.002 s - in com.example.inventory.service.ProductServiceTest
INFO]
INFO] Results:
INFO]
INFO] Tests run: 2, Failures: 0, Errors: 0, Skipped: 0
INFO]
INFO] -----
INFO] BUILD SUCCESS
INFO] -----
INFO] Total time: 2.439 s
```

11.2 Functional Tests (end-to-end flow)

- Add → Update → Delete product
- Low-stock filter

```
T E S T S
-----
Running com.example.inventory.functional.ProductFunctionalTest
Tests run: 2, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.106 s
Running com.example.inventory.service.LowStockTest
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.002 s
Running com.example.inventory.service.ProductServiceTest
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.003 s

Results:

Tests run: 4, Failures: 0, Errors: 0, Skipped: 0

-----
BUILD SUCCESS
-----
Total time: 5.388 s
Finished at: 2025-11-18T22:16:18+05:30
-----
```

11.3 Agile Testing Quadrants

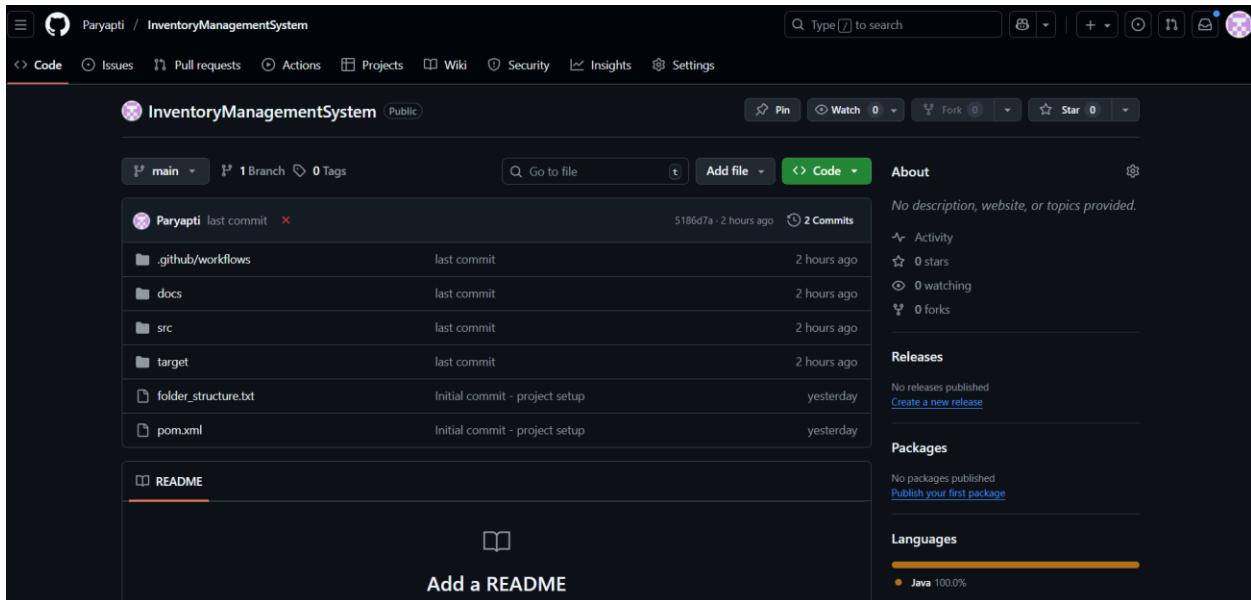
Quadrant	Type of Testing	Applied
Q1	Unit tests	✓
Q2	Functional tests	✓
Q3	User acceptance	✓
Q4	Performance/Automation	Limited

12. Version Control Using GitHub

- Main & develop branches
- Frequent commits
- Push via command line
- GitHub Actions CI pipeline automatically runs:

```
D:\College\Agile\Inventory Management System>git remote add origin https://github.com/Paryapti/InventoryManagementSystem.git
error: remote origin already exists.

D:\College\Agile\Inventory Management System>git branch -M main
D:\College\Agile\Inventory Management System>git push -u origin main
info: please complete authentication in your browser...
Enumerating objects: 69, done.
Counting objects: 100% (69/69), done.
Delta compression using up to 12 threads
Compressing objects: 100% (44/44), done.
Writing objects: 100% (69/69), 19.01 KiB | 374.00 KiB/s, done.
Total 69 (delta 8), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (8/8), done.
To https://github.com/Paryapti/InventoryManagementSystem.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
```



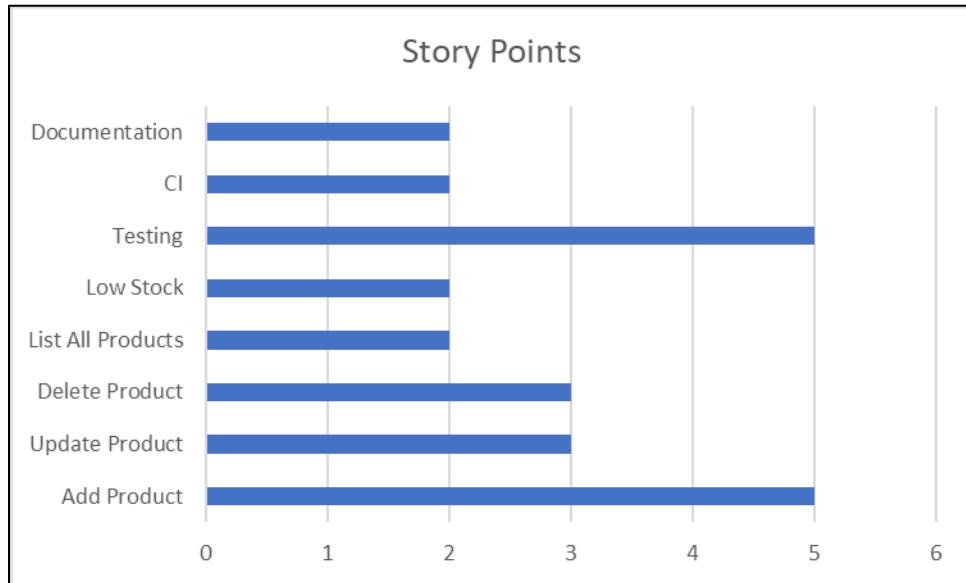
Github repository

The screenshot shows a GitHub CI pipeline for a Java project. The pipeline has one job named 'build' which triggered via push 3 hours ago. The status is Success with a duration of 28s. The workflow file is named ci.yml and contains a single build step. The build step is successful with a duration of 22s.

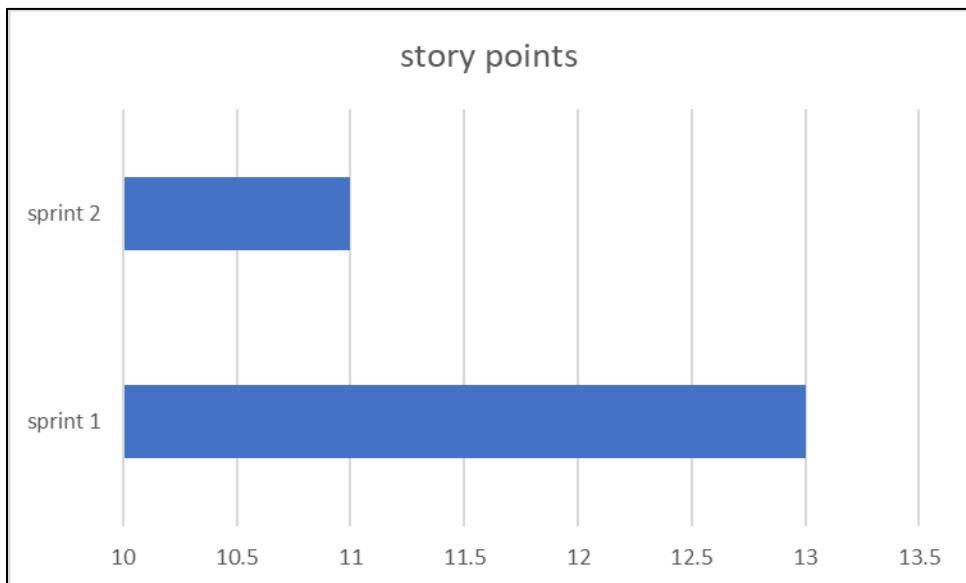
Java CI

13. Metrics: Velocity, Story Points, Burndown

Backlog bar chart



Velocity Chart



14. Conclusion

This project successfully demonstrates the complete application of Agile methodologies through Scrum practices. All backlog items were implemented over two sprints using TDD, functional testing, SOLID principles, and CI/CD integration.

The Inventory Management System works efficiently and fulfills the product vision by providing an easy-to-use, maintainable, and testable mini application.