

**Agile Endsem Project**  
**(CSE706)**  
**Bachelor of Technology (CSE)**

By

Dhruhi Shah (22000381)

Paryapti Macwan (22000401)

4<sup>th</sup> 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.

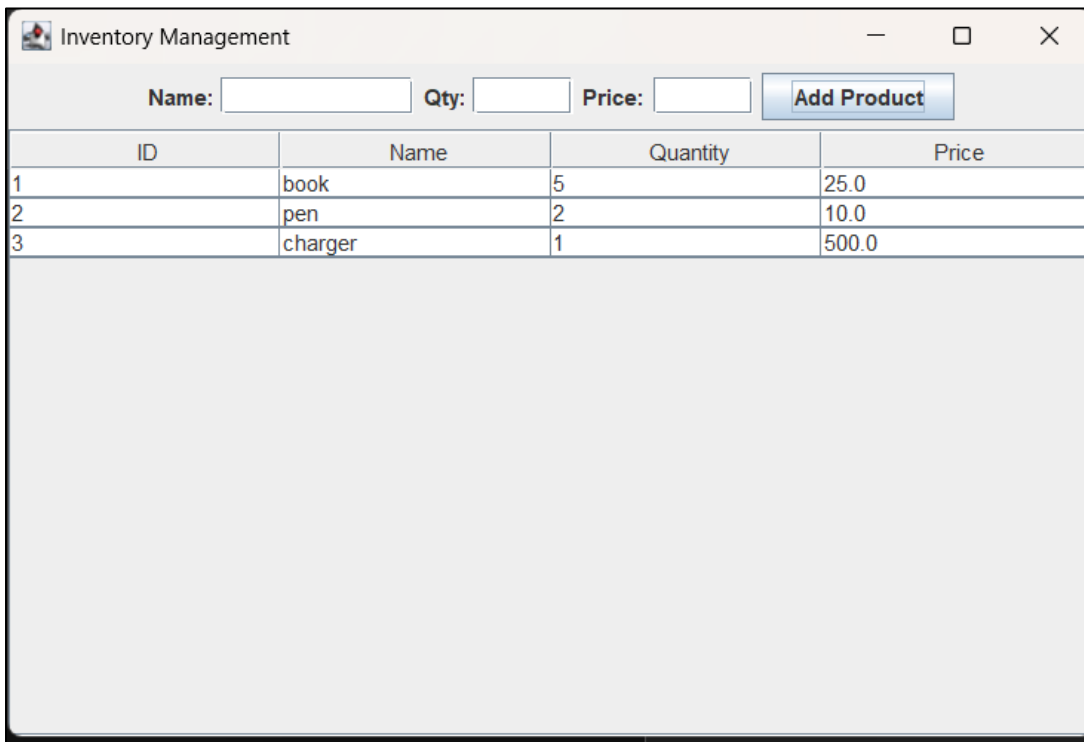
## 2. Introduction

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



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:

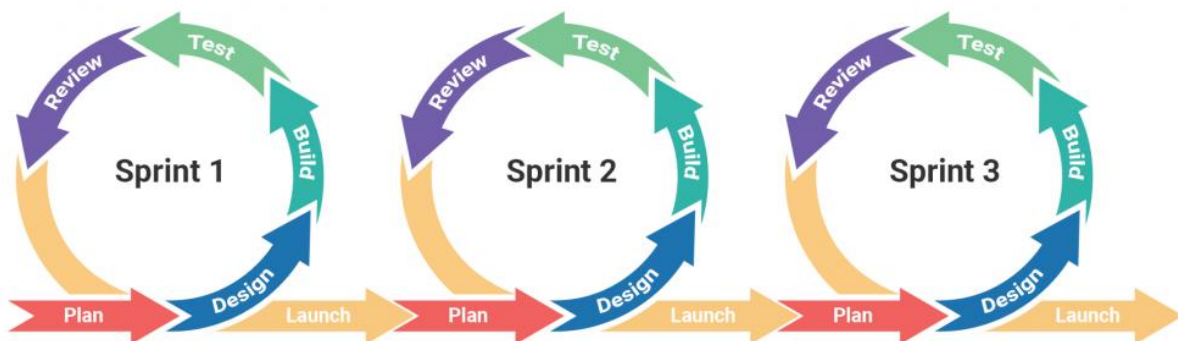
#### Scrum Roles

- 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

←

→

↺

↻

nuv-team-enexdpn7.atlassian.net/jira/software/projects/IMSI/boards/2/backlog?selectedIssue=IMSI-4

☆

🔍

Incognito

⋮

Jira

Search

+ Create

See plans

🔔

🕒

⚙️

PM

For you

Spaces

Recent

Inventory Management...

software team

More spaces

Recommended

Create a roadmap

TRY

More

Spaces

Inventory Management System (IMS)

Summary

Timeline

Backlog

Board

Calendar

List

Forms

More

6

+

Search backlog

👤

Filter

Basic inventory CRUD features

Add dates

(3 work items)

13

0

0

Start sprint

...

IMSI-4

As a user, I want to add a new product so that ...

+ Epic

TO DO

5

...

IMSI-5

As a user, I want to update product details so that s...

TO DO

5

...

IMSI-7

As a user, I want to view all products so that I can t...

TO DO

3

...

+ Create

3 of 3 work items visible

Estimate: 13 of 13

Backlog

(3 work items)

16

0

0

Create sprint

IMSI-6

As a user, I want to delete a product so that I can re...

TO DO

3

...

IMSI-8

As a user, I want low-stock alerts so that I can reor...

TO DO

8

...

IMSI-9

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

TO DO

5

...

+ Create

Jira work item

✕

Add epic

/

🔒

👁️ 1

🔗

...

IMSI-4

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

To Do

⚡

🔗

🎯

+

Description

Product is saved and visible in the list.

Details

⚙️

▼

Assignee

Unassigned

Assign to me

Labels

Add labels

Parent

Add parent

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.

---

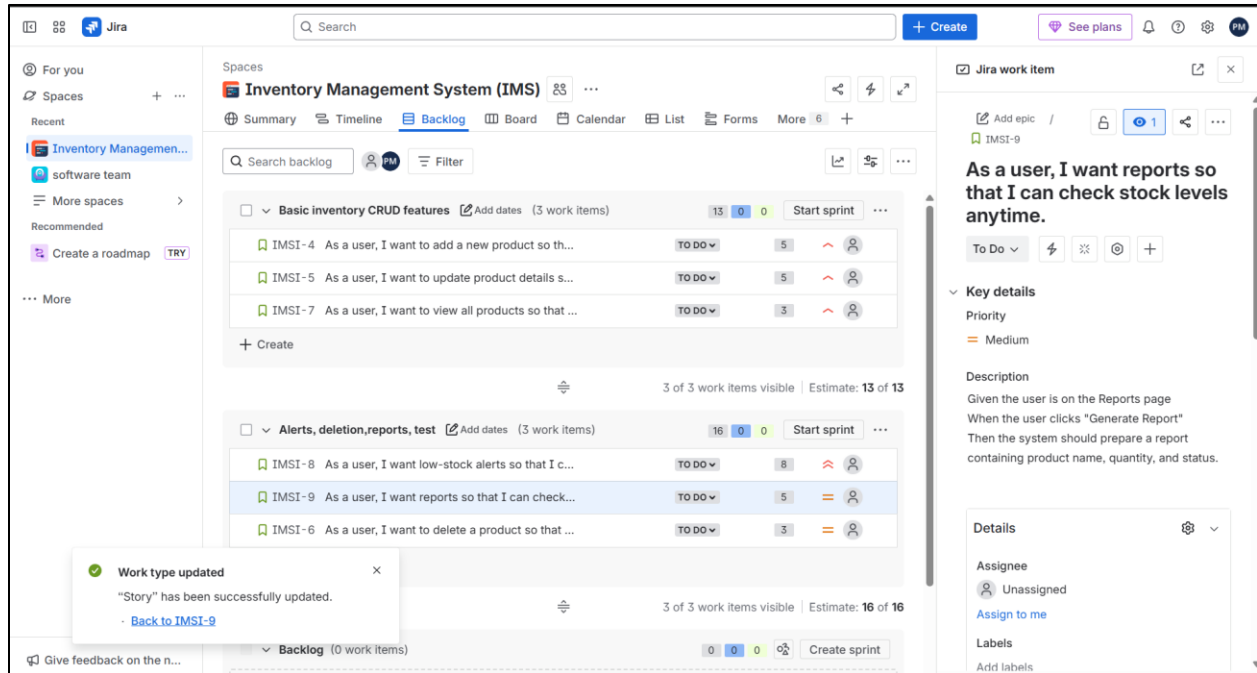
## User Story 2: Update Product
**As a** store manager
**I want to** update product details
**So that I can** keep inventory accurate.

```

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

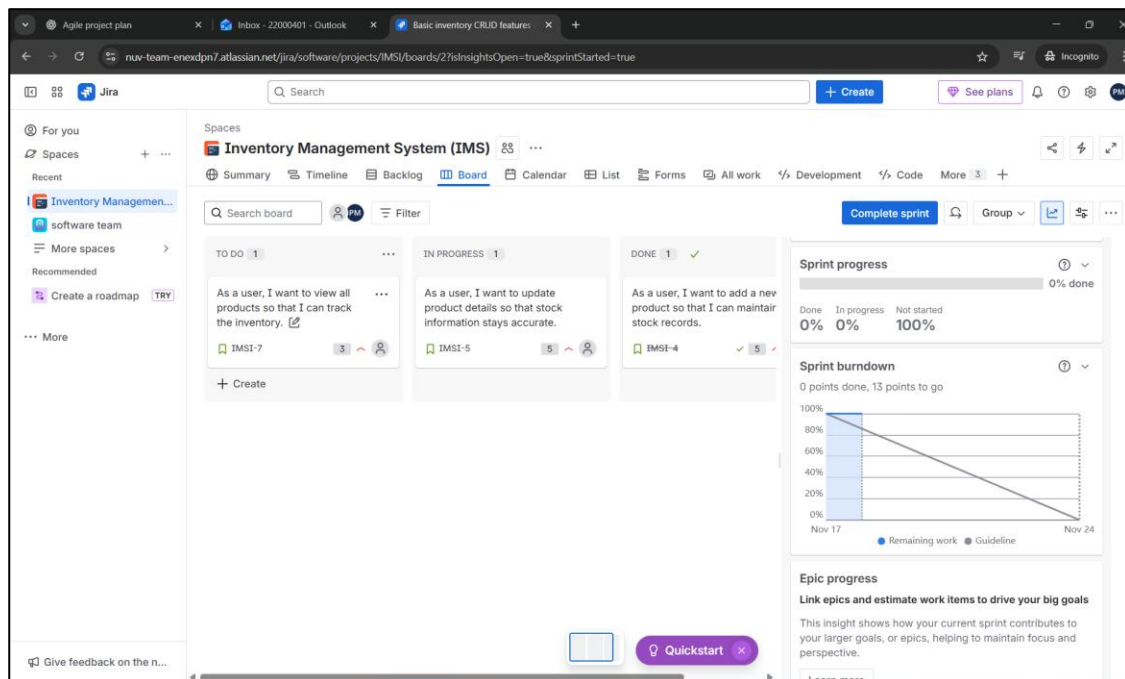
Complete Add, Update, Delete, and Fetch functionalities.

#### Sprint Backlog

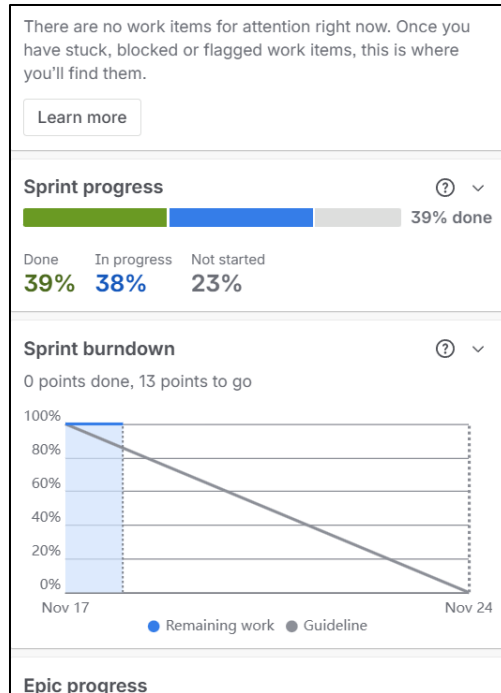
- Add Product module
- Update Product module
- Delete Product module
- List All Products
- Unit tests (TDD for Add & Update)
- Documentation setup

## Daily Scrum Notes (Sample)

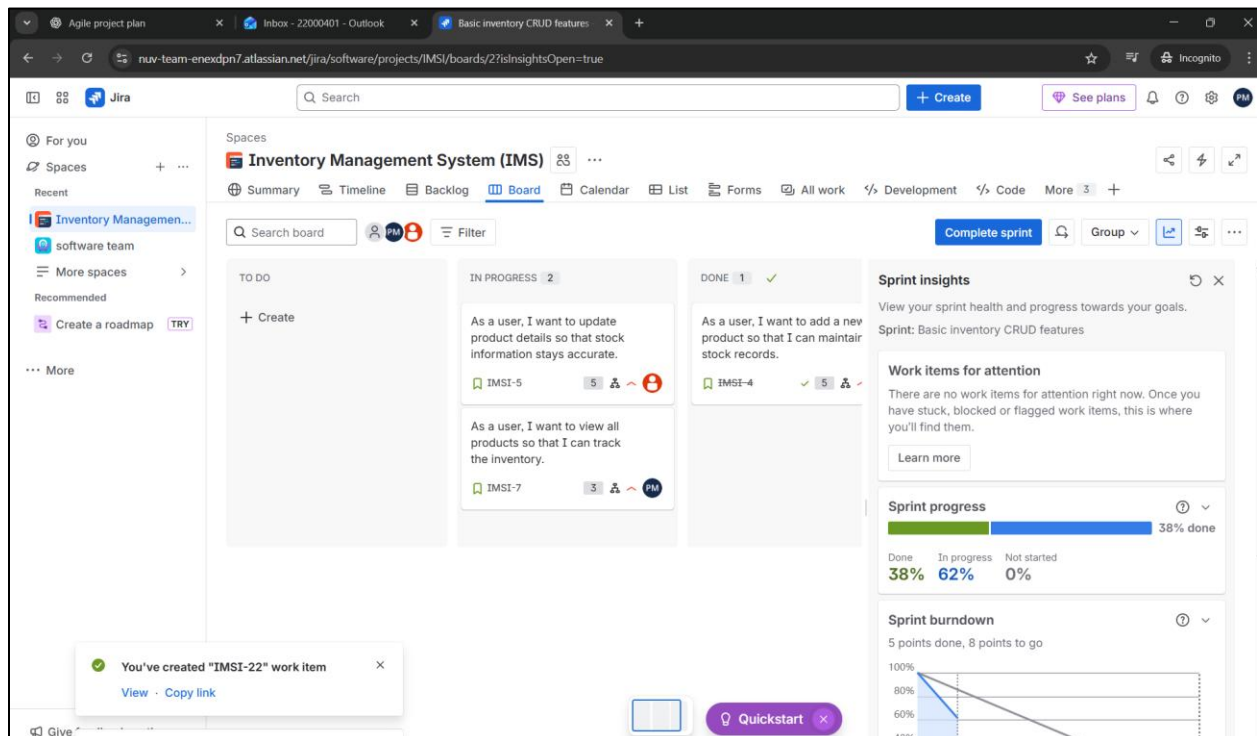
- **Day 1:** Setup project structure, created repo interface
- **Day 2:** Implemented Add Product
- **Day 3:** Implemented Update Product
- **Day 4:** Implemented Delete Product
- **Day 5:** Wrote JUnit tests
- **Day 6:** Manual testing and debugging
- **Day 7:** Sprint Review + Retrospective



Sprint board 1



Burndown chart



Sprint board 2

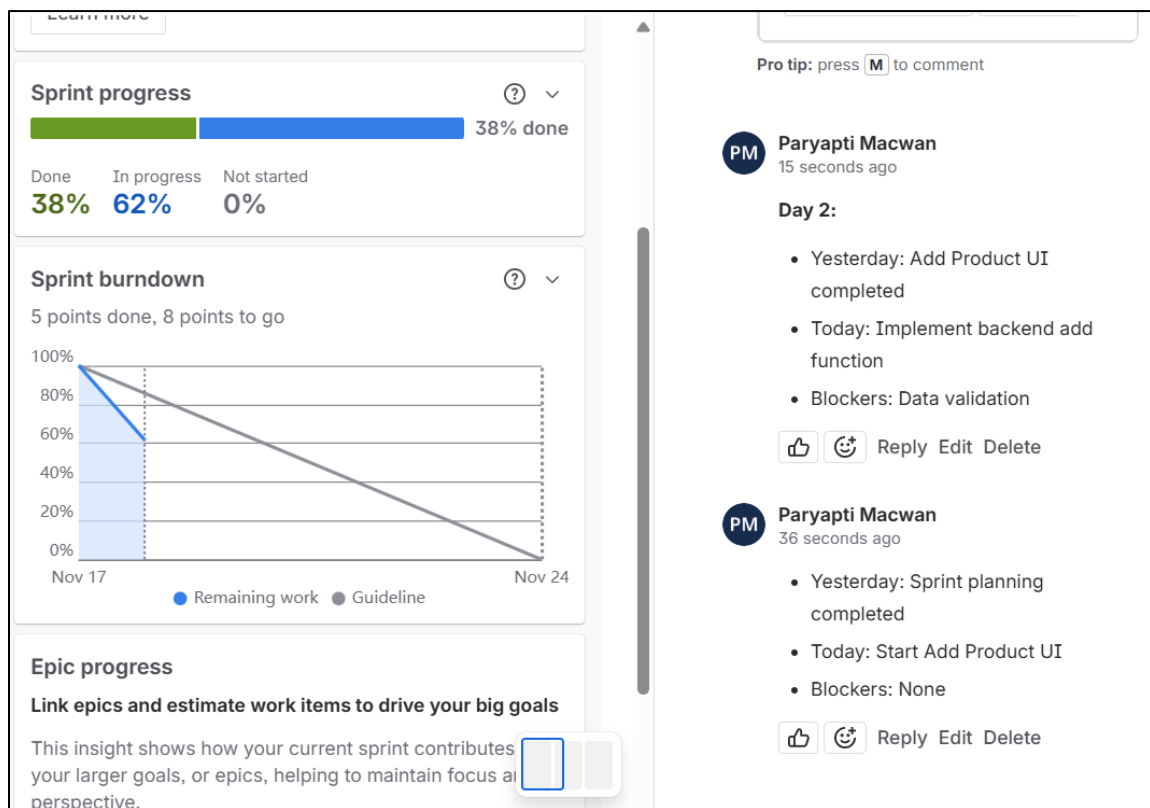
## Sprint 1 Review

### Completed:

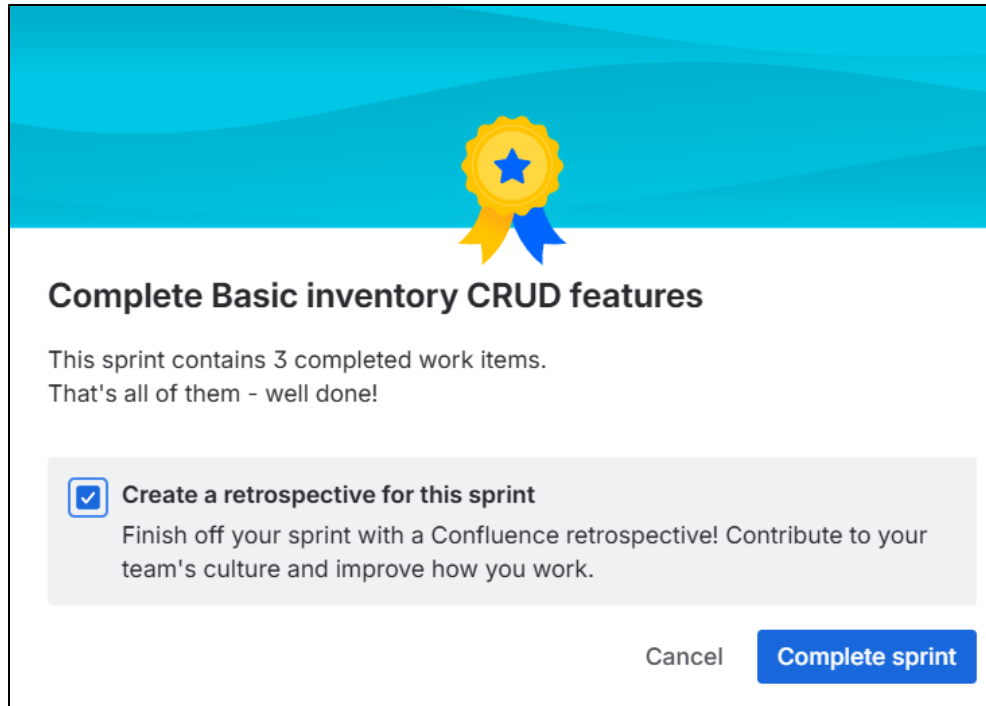
- All CRUD functionalities
- Unit tests
- Basic documentation

### Demo Outcome:

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



Comments & notes



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

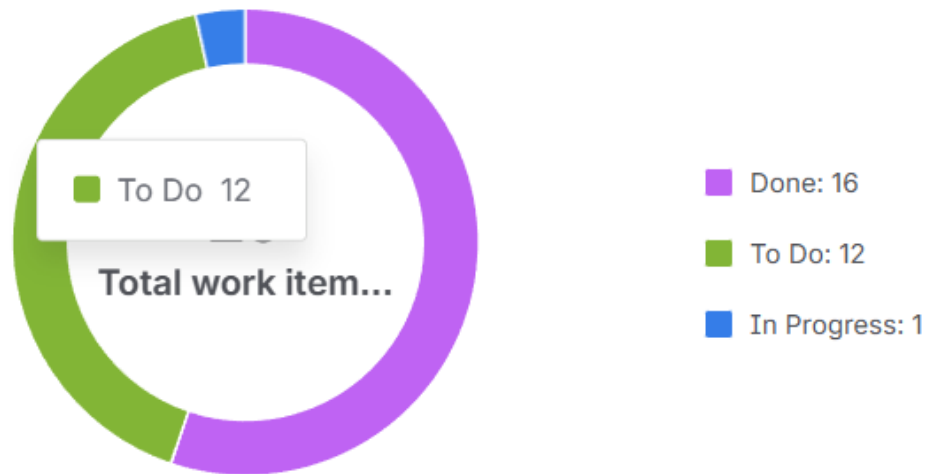
- Build the core CRUD functionality (Create, Read, Update, Delete) for the Inventory Management System and ensure basic operations work correctly.

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

### Sprint 1 retrospective notes

### Status overview

Get a snapshot of the status of your work items. [View all work items](#)



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

PM

Filter

Alerts, deletion,reports, test

18 Nov – 25 Nov (3 work items)

3

5

8

Complete sprint

...

Alerts, deletion,reports, test

IMSI-8	As a user, I want low-stock alerts so that I can reorder items on time.		DONE	8		
IMSI-9	As a user, I want reports so that I can check stock levels anytime.		IN PROGRESS	5		
IMSI-6	As a user, I want to delete a product so that I can remove unwanted items.		TO DO	3		

+ Create

Backlog (0 work items)

0000

Create sprint

Your backlog is empty.

+ Create

## Sprint 2

Spaces

Inventory Management System (IMS)

...

Summary

Timeline

Backlog

Board

Calendar

List

Forms

Goals

Search board

PM

Filter

TO DO 1

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

IMSI-6 3

+ Create

IN PROGRESS 1

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

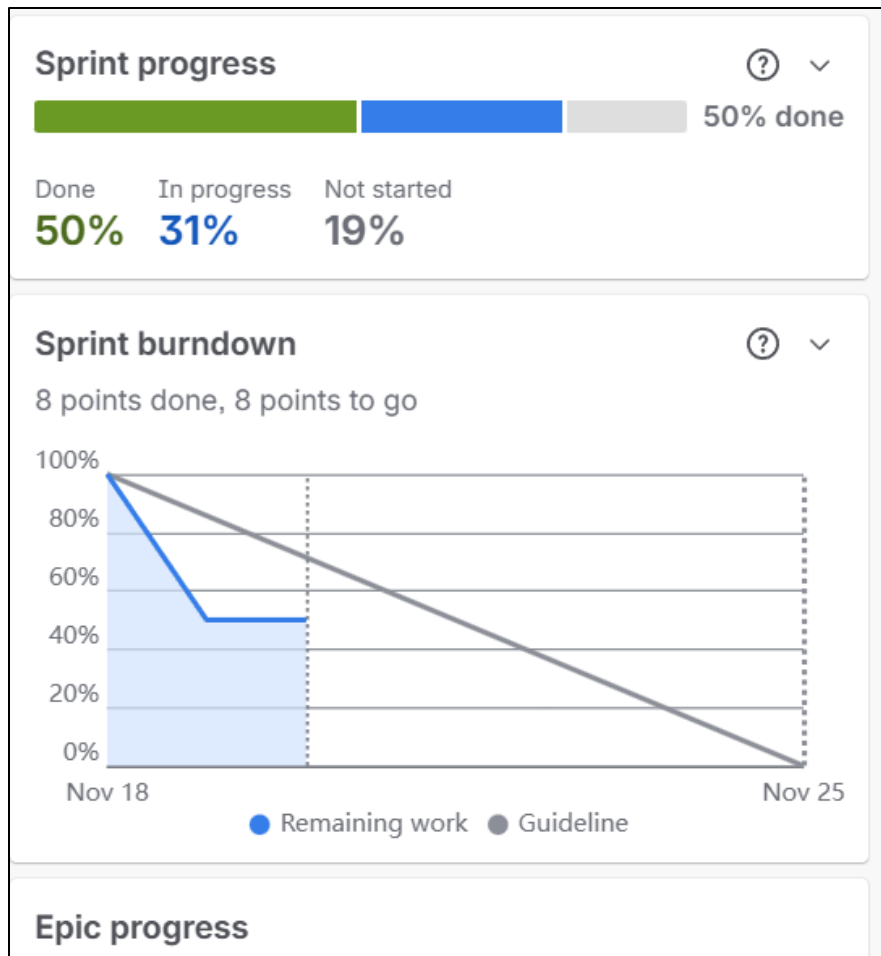
IMSI-9 5

DONE 1

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

IMSI-8 8

## 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

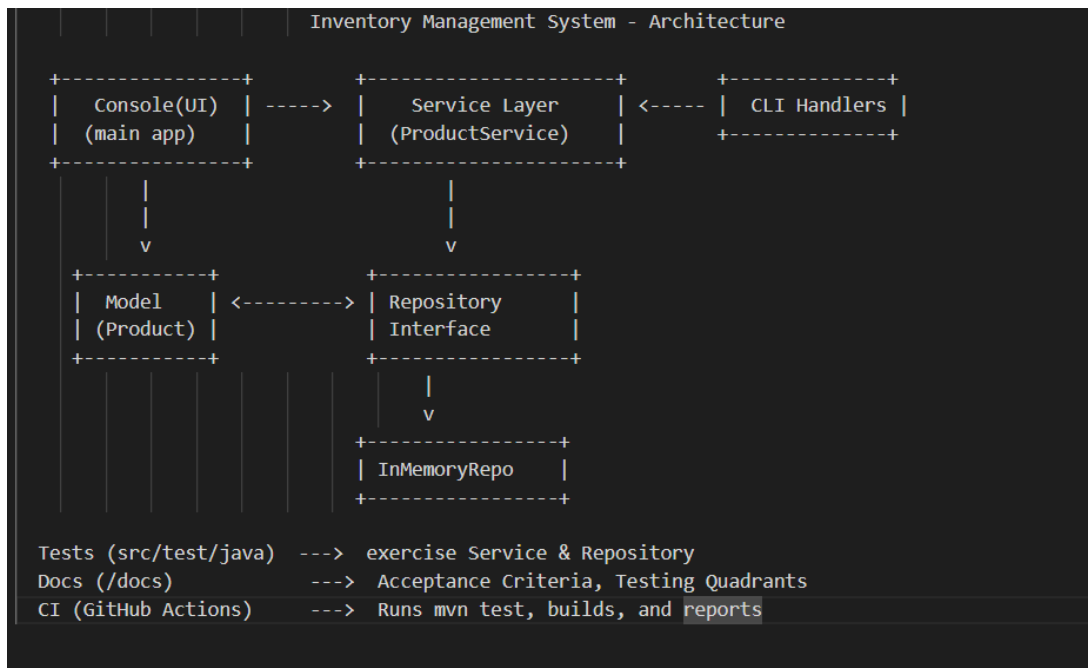
### Action Items

- Use feature branches for future enhancements

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

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$ExecJavaClassLoader.loadClass (URLClassLoaderBuilder.java:117)
    at java.lang.ClassLoader.loadClass (ClassLoader.java:525)
    at org.codehaus.mojo.exec.AbstractExecJavaBase.doExecClassLoader (AbstractExecJavaBase.java:376)
    at org.codehaus.mojo.exec.AbstractExecJavaBase.lambda$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 inventory-management-system:
[ERROR] java.lang.ClassNotFoundException: 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 article:
[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 platform dependent
[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.inventory.functional.ProductFunctionalTest
[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.inventory.service.LowStockTest
[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.inventory.service.ProductServiceTest
[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
[INFO] -----
```

Green

## 11. Testing and Validation

### 11.1 Unit Tests

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

```
[INFO] -----
[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] Results:
[INFO] Tests run: 2, Failures: 0, Errors: 0, Skipped: 0
[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

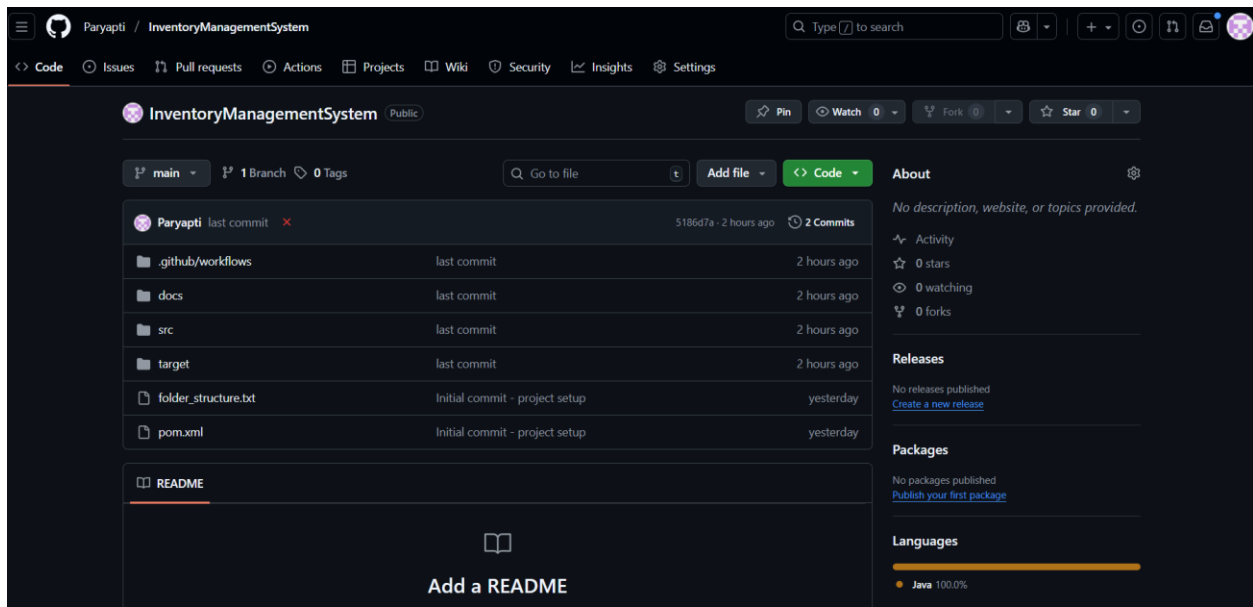
## 12. Version Control Using GitHub

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

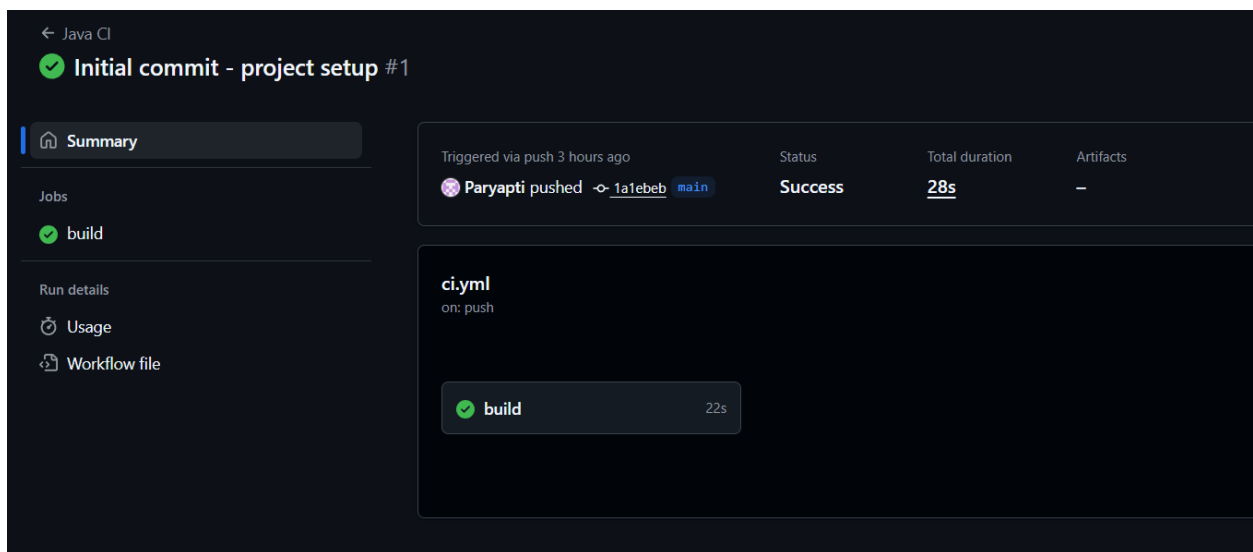
```
D:\College\Agile\Inventory Management System>git remote add origin https://github.com/Paryapti/InventoryManagementSystem.git
error: remote origin already exists.

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'.
```



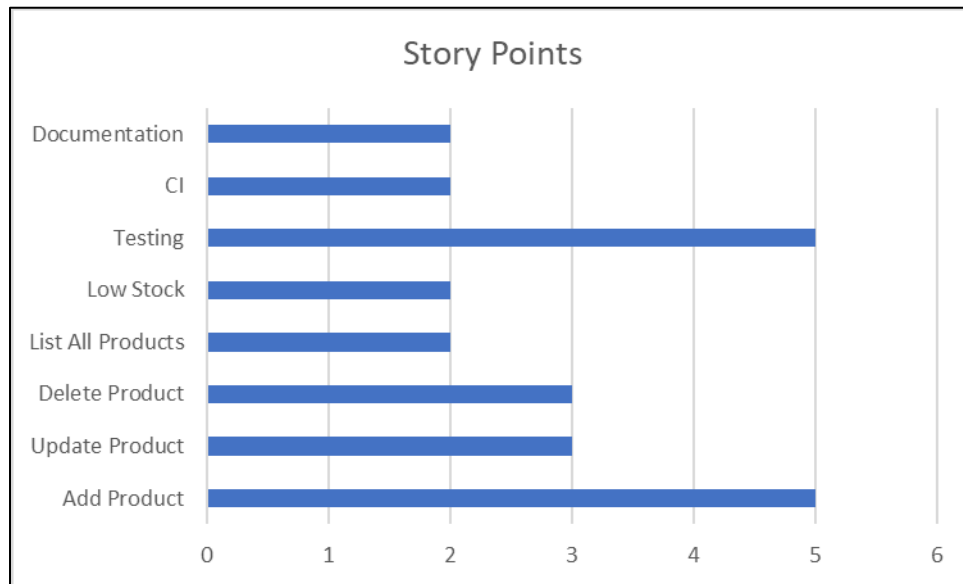
Github repository



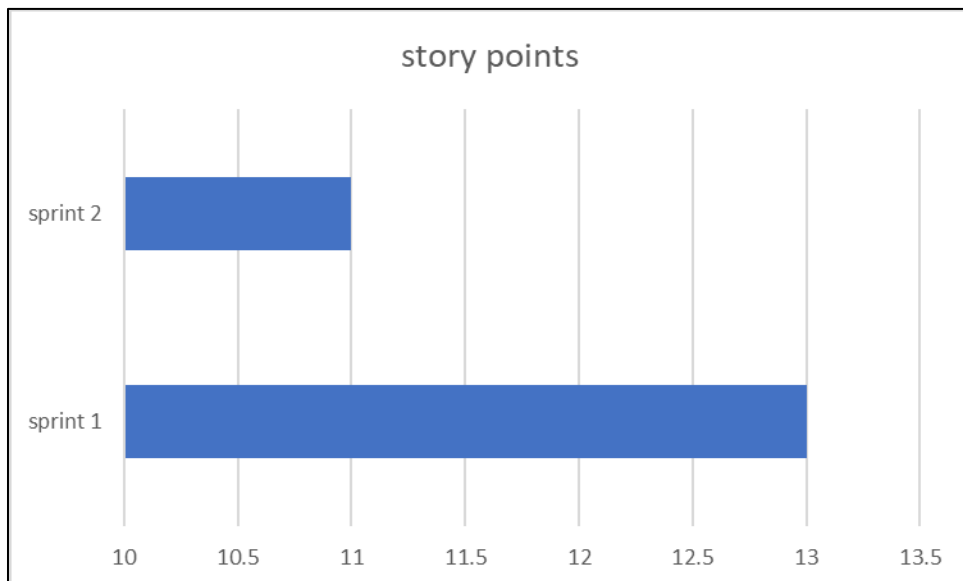
Java CI

### 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.