Inventory Management System

Fundamental Project – By Stephanie Ashdown

Objectives

Overall Objectives

- Create a functional application
- Use technologies and tools learned during training
- Application to be written in Java and to interact with a managed database

Project Specific Objectives

- Build an inventory management system (IMS)
- System should allow user interaction via the Command-Line Interface
- Full CRUD Functionality to be implemented for Items, Customers, Orders and Order Items

Technologies Used

- **Jira** – Project management via Scrum Board

- **Git** – Version control through main/dev/feature branching

- **GitHub** – Source code management

- **MySQL Server** – Database management

- **Java** – Back-end programming language

- **Maven** – Build tool

- **JUint & Mockito** – Unit testing

Planning & Management

Jira

Risk Assessment

Jira

- Plotted out user stories and acceptance criteria
- Assigned user stories to epics
- Linked tasks to stories
- Estimated work using story points
- Assigned priority with MoSCoW
- Managed workload with sprints
- Updated progress using smart commits

Projects / IMS-22JulyEnable3

IMS Sprint 3
Finish project

SA

Epic ∨ Label ∨ Type ∨

TO DO 2 ISSUES	IN PROGRESS 3 ISSUES	DONE 10 ISSUES ✓
Create build	As a developer I want to know how to run the project on a development environment so that I can work on it.	Test Item Class
☑ IMS-78		TESTING
Add .gitignore file	DOCUMENTATION ShouldHave	☑ IMS-66
☑ IMS-79	IMS-18 5	Test Order Class
	Complete all sections of ReadMe	TESTING
	DOCUMENTATION	☑ IMS-67
	IMS-76	Test OrderLineItem Class
	Create presentation	TESTING
		☑ IMS-68

As a store worker I want to add an item to an order so that I can see which items have been purchased.

📎 Attach   🔧 Add a child issue   🔗 Link issue   ∨   ...

Description

Given the worker is in the IMS when they select to Add and Item to an order then they are able to select and add an item to an order.

Linked issues   +

relates to

☑ IMS-50	Add OrderLineItem class definition to the domain	= 👤	DONE ∨
☑ IMS-51	Add OrderLineItem to MySQL	= 👤	DONE ∨
☑ IMS-52	Add line item attribute to order class	= 👤	DONE ∨
☑ IMS-53	Add add line item method to order class	= 👤	DONE ∨
☑ IMS-54	Add OrderLineItemDAO class	= 👤	DONE ∨
☑ IMS-57	Add addLineItem method to OrderController	= 👤	DONE ∨

Risk Assessment

- Identified possible risks

- Assigned risk level using matrix

- Planned responses and control measures

- Monitored an updated as appropriate throughout

IMS Project Risk Assessment

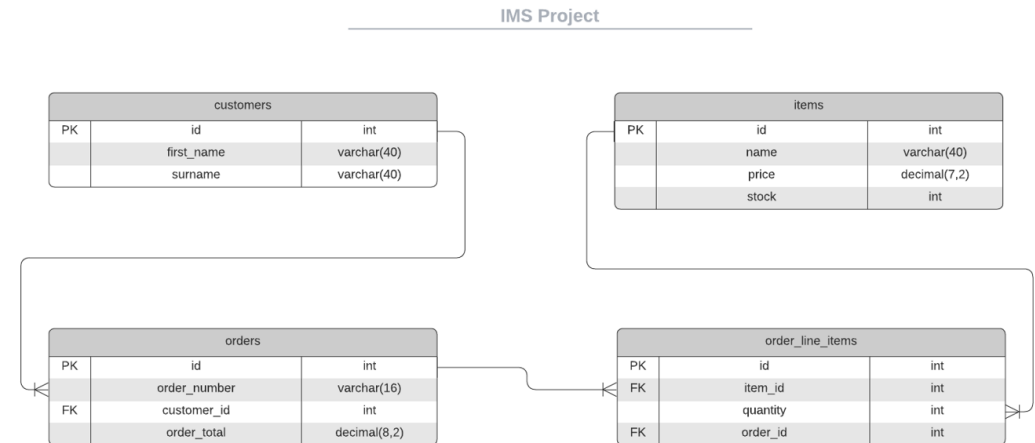| Description	Evaluation	Liklihood	Impact Level	Responsibility	Response	Control Measures	Risk Level
Sustained period of internet loss.	Not possible to update the github repository and Jira board due to lack of internet connection.	Unlikely	Moderate	Internet service provider	Continue working on the project locally committing but not pushing to github. Use mobile data to inform QA trainer. If issures persist relocate to another location with a different service provider.	Look into other internet sources either at friends or public. Make sure friends are aware that this is a possibility.	
Local machine stops functioning.	Cannot continue working on the project as laptop stops working.	Unlikely	Catastrophic	Developer	Inform the QA trainer. Work on an alternative machine cloning the repository from github.	Push commits to github regularly. Scope out possible alternative machines either from friends or shops with fast delivery.	
Problem with local machine leading to loss of data.	The local machine cashes leading to the loss of recent work on the project.	Possible	Moderate	Developer	Pull from the github repository if it is further ahead than the local copy. Repeat the lost work.	Save regularly and make regular commits and pushes to github. Track work on Jira so it is easy to see which task were being worked on.	
Github server goes down.	Issues pushing to github as the server is down.	Possible	Moderate	Github	Continue working on the project locally and push once github resolves the issue. Keep an eye on the github status updates. Inform the QA trainer if the problem persists.	Add commit and push to github regularly.	
Jira server goes down.	Issues tracking project work due to Jira server problem.	Possible	Minor	Jira	Continue working on the project and update the Jira board once Jira is back up and running.	Consider saving a local copy of the jira board so it is possible to track locally until Jira is back up and running.	
Illness.	Unable to continue working due to illness.	Unlikely	Catastrophic	Developer	Inform the QA trainer. Work as much as possible if the illness is mild enough to allow it. Ensure that the QA trainer has access to the work that has been achieved.	Eat and sleep well in the weeks leading up to the project.	
Failure to meet project deadline.	Project not completed in time for deadline as a result of bad planning on unexpected issues.	Unlikely	Major	Developer	Complete as much as possible in the timeframe focussing on the MVP and ensuring that the project is functional.	Plan the project properly in advance and monitor progess frequently adjusting priorities as required.	
23/08/2022 Come across an unknown error.	Work on the project gets delayed due to an error not seen before or unferstood	Likely	Moderate	Developer	Try to find solution via google but do not waste too much time, contact QA trainer for assistance.	Review errors from previous work in preparation. Ensure that understand how to contact QA trainer.	

Version Control

- Project updates tracked using git

- Work carried out on different branches

- Branches merged into dev branch once completed satisfactorily

- Main branch kept clean for build commit

Database Design

- Database designed around four tables

- Customer table for customer information

- Item table for item details

- Order table for order information

- Order line item table for order items

IMS Project

customers

PK	id	int
	first_name	varchar(40)
	surname	varchar(40)

items

PK	id	int
	name	varchar(40)
	price	decimal(7,2)
	stock	int

orders

PK	id	int
	order_number	varchar(16)
FK	customer_id	int
	order_total	decimal(8,2)

order_line_items

PK	id	int
FK	item_id	int
	quantity	int
FK	order_id	int

Class Design

- Classes created for the four database tables

- User interaction with each domain entity controlled by controllers

- Database actions controlled by DAO classes
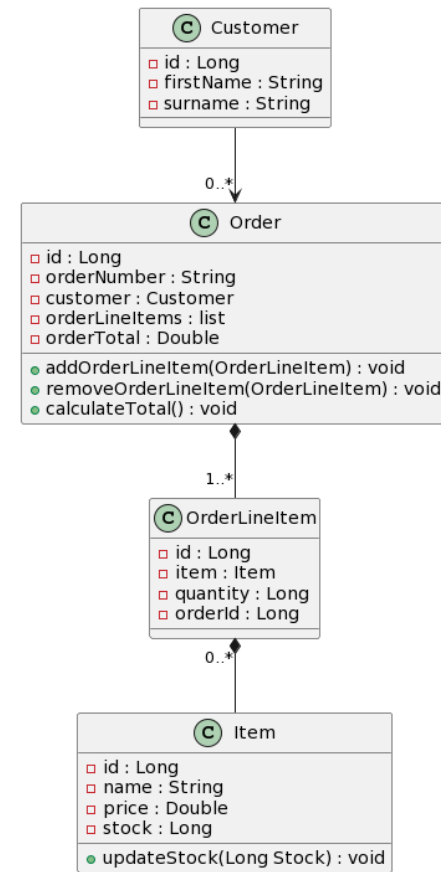
Demo

- A demonstration of some of the user stories will now be carried out in the project itself

- This will focus on the main CRUD functionality

Unit Testing

- Tests created for each of the domain entities

- Tests created for the controller classes

  - Mockito used for these tests

- Tests created for the DAO classes

- Coverage of main folder currently 78%

- Main lack of coverage in IMS and Utils

- Work continuing to improve coverage

Element		Coverage	vered Instructions	lissed Instructions	Total Instructions
⌄ 🗃 IMS-22JulyEnable3		88.6 %	5,752	743	6,495
⌄ 📁 src/main/java		78.0 %	2,616	739	3,355
> ⊞ com.qa.ims.persistence.dao		78.5 %	989	271	1,260
> ⊞ com.qa.ims		0.0 %	0	213	213
> ⊞ com.qa.ims.controller		89.0 %	723	89	812
> ⊞ com.qa.ims.utils		65.7 %	163	85	248
> ⊞ com.qa.ims.persistence.domain		90.5 %	741	78	819
> ⊞ com.qa.ims.exceptions		0.0 %	0	3	3
> 📁 src/test/java		99.9 %	3,136	4	3,140

Conclusions

What Went Well

- The MVP was met

- Initial planning of user stories

- Version control and use of branching

What Could Have Been Better

- Jira organisation and story point estimation

- More screenshots of mid stages

- Functionality implementation over complicated particularly in order controller

- More focus on SOLID principles and best practice