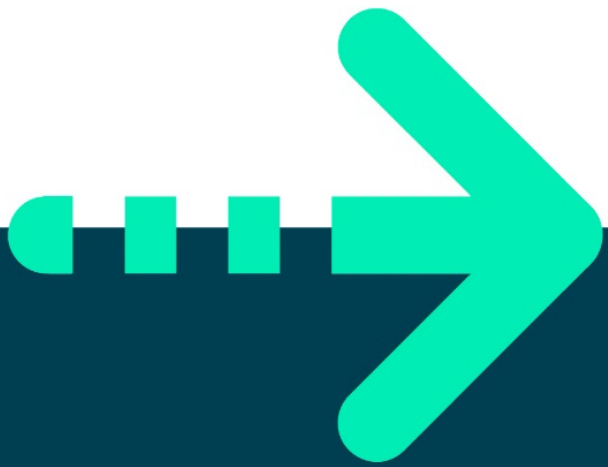
Fundamental Project Specification:

Inventory Management System (IMS)

Last revision: January 2021 by NJ



CONTENTS

Introduction

The purpose of this document is to outline the individual project specification that you will be working on during the training. This project will involve concepts from all core training modules; more specifically, this will involve:

- Agile & Project Management
- Databases & Cloud Fundamentals
- Programming & Testing Fundamentals
- Continuous Integration & Build Tools

The individual project must encapsulate all aspects of the aforementioned modules, to achieve the specification outlined in the **Domain** section below. However, creativity is encouraged, provided that you meet the minimum requirements.

Objective

The overall objective of the project is the following:

To create a functional application – using supporting tools, methodologies, and technologies – which encapsulates all fundamental modules covered during training.

Specifically, you are required to create an application using the language from your Programming Fundamentals week which interacts with a managed database.

If you wish to use any technologies which have not been covered as part of your training, you must consult your trainer first.

You must plan the approach you will take to complete this project using the design techniques which you have learned. Your project is also expected to be rigorously tested.

Domain

You are required to build an application that an end user can interact with via a Command-Line Interface (CLI). The application required is an inventory management system, that needs to be able to:

- Add a **customer** to the system
- View all **customers** in the system
- Update a **customer** in the system
- Delete a **customer** in the system
- Add an **item** to the system
- View all **items** in the system
- Update an **item** in the system
- Delete an **item** in the system
- Create an **order** in the system
- View all **orders** in the system
- Delete an **order** in the system
- Add an **item** to an **order**
- Calculate a cost for an **order**
- Delete an **item** in an **order**

When considering the entities in this domain:

- A **customer** needs to have a name
- An **item** needs to have a name and a value
- An **order** needs to have a **customer** and contains **items**
- You will need to create an intermediary **orders_items** table to handle the many-to-many relationship, as MySQL does not natively support this type of relationship.

Scope

The requirements set for the project are below. Note that these are a minimum set of requirements and can be added onto during the duration of the project.

The requirements of the project are as follows:

- Code fully integrated into a Version Control System using the feature-branch model: **main/dev/multiple features**.
- A project management board with full expansion on user stories, acceptance criteria and tasks needed to complete the project.
- A risk assessment which outlines the issues and risks faced during the project timeframe.
- A relational database used to persist data for the project, containing the **customers, products, orders, and orders_items** tables. Relationships should be modelled using an ERD.
- A functional application 'back-end', following best practices and design principles, in the language that you have covered during training, meeting the requirements set on your project management board.
- A build of your application, including any dependencies it might need, produced using an integrated build tool.
- Unit tests for validation of the application. You should aim to reach the industry standard of **80%** test coverage.

You should consider the concept of MVP (Minimum Viable Product) as you plan your project.

Ensure that you complete <u>all</u> the requirements above before adding extra functionality that is not specified above.

Constraints

The time constraints for this application will be discussed when this specification has been distributed to you.

The application must also **strictly** adhere to the following technological constraints, as encountered during the course of your training:

- **Version Control System**: Git
- **Source Code Management**: GitHub
- **Kanban Board**: Jira
- **Database Management System**: MySQL Server 5.7+ (local or GCP instance)
- **Back-End Programming Language**: Java
- **Build Tool**: Maven
- **Unit Testing**: JUnit

Deliverable

The final deliverable for this project is the completed application with full documentation around utilisation of supporting tools. This will require a fully functional application.

You will be required to present your work to at least one trainer – this may be your course leader, another trainer, or several trainers. This will take the form of a presentation of work lasting 15 minutes, plus a 5-minute Q&A session.

Given the above, you will therefore be required to track your designs and workflow (e.g. through screenshots) throughout the duration of the project, and be able to show how they changed over time.

You will be required to utilise the feature-branch model, and to push a working copy of your code to the **main** branch regularly. It is recommended to use the **feature-<concept>** naming strategy for your feature branches.

You will be required to include all supporting documentation for your project within your remote repository at close-of-business (17:30) on the day of presenting your project – please refer to the **Deliverables Checklist (MVP)** for further details.

Deliverables Checklist (MVP)

Codebase

- CRUD functionality following the Enterprise Architecture Model for the **customers**, **items**, and **orders** entities
- The project connects via JDBC to a local or GCP-based MySQL instance
- Sensible package structure
- Adherence to best practice (e.g. OOP principles, SOLID, refactoring)

Testing

- Unit test coverage of the **src/main/java** folder, aiming for **80%**

Continuous Integration

- Git repository utilising the feature-branch model
- The **main** branch must compile
- A build of the application is present in the root folder of your git repo
    - A **fat .jar** which can be deployed from the command-line

Repository & Documentation

- A completed **project management board**, including user stories, acceptance criteria, estimations via story points, and prioritisation via MoSCoW methodology
- A working **.gitignore** for ignoring build-generated files and folders
- A completed **README.md**, explaining how to use and test your application
- A **documentation** folder containing:
    - A completed **risk assessment**, utilising a matrix, in **.pdf** format
    - **At least one ERD** and **one UML** diagram, in **.png** format
    - A copy of your presentation, in **.pdf** format (slides only – no notes)

Presentation Guideline (15+5 mins)

- **Introduction**: Who are you? How did you approach the specification?
- **Consultant Journey**: What technologies have you learned for this project?
- **CI**: How did you approach version control?
- **Testing**: What was tested? Show the coverage of the **src/main/java** folder.
- **Demonstration**: Run through a couple of user stories
- **Sprint review**: What did you complete? What got left behind?
- **Sprint retrospective**: What went well? What could be improved?
- **Conclusion**: Reflections on the project, future steps, any other relevant info
- Diagrams and/or screenshots used where appropriate
- Your presentation should last a total of **15 minutes**
- **Questions**: Leave 5 minutes for questions at the end of the presentation

Mark Scheme

The skills evaluated within this project are described within the SFIA 7 framework; please see **https://sfia-online.org/en/framework** for further information.

The skills which this project will evaluate are the following:

Programming & Software Development (PROG)

- Designs, codes, verifies, tests, amends, and refactors simple programs/scripts.
- Tests, documents, amends, and refactors simple programs/scripts.
- Applies agreed standards and tools, to achieve a well-engineered result.

Software Design (SWDN)

- Creates and documents detailed designs for simple software applications or components applying agreed modelling techniques, standards, patterns, and tools.
- Creates and documents the development and/or deployment of an application, applying agreed standards and tools.

Testing (TEST)

- Designs test cases and creates test scripts and supporting data.
- Analyses and reports test activities and results.

Systems Integration & Build (SINT)

- Produces software builds from software source code.
- Conducts tests as defined in an integration test specification, records the details of any failures. Analyses and reports on integration test activities and results.
- Identifies and reports issues and risks.

Marking Procedure

You will be emailed your marks for this project via your QA email address in a timely manner by your Lead Trainer during the week following your project deadline. All marks, once given, are final.

Individual feedback will only be available upon specific request to your Lead Trainer via a direct reply to the email containing your marks.

Please note that suboptimal performance in this project may constitute grounds for immediate dismissal from the Academy, unless reasonable adjustments have already been agreed with your Lead Trainer prior to your presentation taking place.

Any further questions regarding the marking procedure should be directed to your Lead Trainer.