Fundamental Project Specification:

Inventory Management System (IMS)

Last revision: May 2020

CONTENTS

[Introduction 4](#_Toc44500286)

[Objective 4](#_Toc44500287)

[Domain 5](#_Toc44500288)

[Scope 6](#_Toc44500289)

[Constraints 7](#_Toc44500290)

[Deliverable 7](#_Toc44500291)

[Deliverables Checklist (MVP) 8](#_Toc44500292)

[Mark Scheme 9](#_Toc44500298)

# 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, that 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 CLI (Command Line Interface). The application needs to be 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**

**Extension (unmarked)**:

* Add a **user** to the system
* List all **users**
* Changes to **customers**, **items** and **orders** need to be tied to a **user**
* A **user** should have a **username** and **password**
* You must be able to log in as a **user** within the system to make any changes

# 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 Agile methodology and the Feature-Branch Model.
* A project management board with full expansion on user stories and tasks needed to complete the project. It should also provide a record of any issues or risks that you faced creating your project.
* A relational database used to store data persistently for the project, containing the **customers**, **products**, and **orders** 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.**

**You should complete all the requirements above before you add 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
* **Project Management**: Jira
* **Database Management System**: GCP instance of MySQL Server 5.7
* **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 **master** branch regularly – this is so that the trainers can keep track of your progress. It is recommended to use the **feature-<name>** 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 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 GCP 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 (**master**/**dev**/**features**)
* The master branch must compile
* A fat .jar which can be deployed from the command-line

## Repository & Documentation

* A completed project management board, including user stories and estimations with story points and/or MoSCoW
* A working .gitignore for ignoring build-generated files and folders
* A completed README.md explaining how to run your program
* A completed risk assessment (utilising a matrix)
* An ERD and/or a UML diagram for your Minimum Viable Product
* A copy of, or link to, your presentation

## Presentation Guideline (15 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 set up your CI pipeline and Git?
* **Testing**: Coverage, static analysis, red-green-refactor
* **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
* **Questions:** Leave 5 minutes for questions at the end of the presentation
* **Diagrams and/or screenshots used where appropriate**

# Mark Scheme

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