Fundamental Project Specification

Date of Issue: January 2020

CONTENTS

[Introduction 3](#_Toc29753593)

[Objective 3](#_Toc29753594)

[Domain 4](#_Toc29753595)

[Scope 5](#_Toc29753596)

[Constraints 6](#_Toc29753597)

[Deliverable 6](#_Toc29753598)

[Marking Scheme 7](#_Toc29753599)

[Marks 7](#_Toc29753600)

[Programming/software development 8](#_Toc29753601)

[Software Design 10](#_Toc29753602)

[Testing 11](#_Toc29753603)

[Systems integration and build 12](#_Toc29753604)

# 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 Fundamentals
* Continuous Integration
* Automated Testing

The individual project must encapsulate all aspects of the aforementioned modules. You will be provided with a domain to work with however creativity is encouraged as long as you meet the minimum requirement.

# Objective

The overall objective of the project is the following:

* To create an application with utilisation of supporting tools, methodologies and technologies that encapsulate all fundamental modules covered during training.

Specifically, you are required to create an application using the language from your Programming Fundamentals Module which interacts with a Managed Database. You must plan the approach you will take to complete this project using the design techniques learnt, and also create a CI Pipeline that can automate the building and deployment of your artifact.

# 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
* Add an **item** and quantity to an **order**.
* Calculate a cost for an **order**.
* Update the quantity of an **item** in an **order**
* Delete an **item** in the **order**
* Add an **order** to the system.
* View all **orders** in the system.

When considering the entities in this domain:

* A **customer** needs only have a name.
* An **item** needs to have a name and a value.
* An **order** needs to keep the cost of the order, needs to have a **customer** and needs to have at least one **item**.

When calculating the cost for the Order, the business has the following rules:

* If the total cost of the order exceeds 10000, bulk discount of 10% is applied.

**Extension**:

* 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:

* A Kanban 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, this database needs to have at least 3 tables in it, to demonstrate your understanding, you are also required to model a relationship.
* A functional application, following best practices and design principles, in a language that you have covered during training that meets the requirements set on your Kanban Board.
* Unit tests and integration tests for validation of the application. You must strive to provide high test coverage.
* Code fully integrated into a Version Control System which will subsequently be built through a CI server and deployed to an artifact repository manager.

You should consider the concept of MVP (Minimum Viable Product) as you plan your project, complete all the requirements above before you add extra functionality that is not specified above.

# Constraints

The time constraint of this application will be discussed when the specification is given out, as this can fluctuate based on several factors.

The other constraint for this is certain technology that needs to be used. The application needs to utilise the technology discussed during the training modules. The tech stack required would be the following:

* Kanban Board: Trello or GitHub project
* Live Database: GCP MySQL server or other Cloud Hosted managed Database.
* Test Database: Locally hosted Database or in memory database for testing purposes.
* Programming language: Language covered in Programming Fundamentals (Java)
* Testing framework: JUnit
* Version Control: Git
* Hosted Version control system: GitHub
* CI Server: Jenkins
* Cloud server: GCP virtual machine or other equivalent Cloud hosting Option.
* Artifact Repository Manager: Nexus

# 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.

A presentation of work will be required towards the end of the deadline. However, you will be required to track designs and work throughout the duration of the project and be able to show how they changed overtime.

You will be required to push you code to the Master branch once a week, this is so that the Trainer can keep track of your progress.

# Marking Scheme

Below are the skills that we will be evaluating for this inventory management system assessment. These skills are as described in the SFIA 7 framework; please see below if you wish to have more information:

<https://www.sfia-online.org/en/framework>

The mark scheme is centred around four of the SFIA skills:

* Programming/software development (PROG)
* Software design (SWDN)
* Testing (TEST)
* Systems integration and build (SINT)

Each of the skills have been broken down into different sections; each section is marked out of 5. A section has a list of competencies and for each competency completed you will receive one point. You need to get a specified number of points to achieve the next rating. The assessment will be marked out of 45, with a passmark of 27.

## Marks

Name: ………………………………………………..

Programming/software development: / 15

Software design: / 10

Testing: / 10

Systems integration and build: / 10

Total: / 45

## Programming/software development

The planning, designing, creation, amending, verification, testing and documentation of new and amended software components in order to deliver agreed value to stakeholders. The identification, creation and application of agreed software development and security standards and processes. Adopting and adapting software development lifecycle models based on the context of the work and selecting appropriately from predictive (plan-driven) approaches or adaptive (iterative/agile) approaches.

**Learners must:**

Design, code, verify, test, document, amend and refactor simple programs/scripts. Applies agreed standards and tools, to achieve a well-engineered result. Reviews own work.

|  |  |  |
| --- | --- | --- |
| Sub-skill | Rating | Details |
| Able to **design** and **test** code to fulfil an agreed specification. A point will be awarded for each of the following done correctly:   * 30% Unit Test coverage * 50% Unit Test coverage * 70% Unit Test coverage * 90% Unit Test coverage | **1** | **0/4 points** |
| **2** | **1/4 points** |
| **3** | **2/4 points** |
| **4** | **3/4 points** |
| **5** | **4/4 points** |
| Able to code, verify, amend and refactor simple programs/scripts. A point will be awarded for each of the following done correctly:   * Create functionality for user * Create functionality for item * Create functionality for order * Read functionality for user * Read functionality for item * Read functionality for order * Update functionality for user * Update functionality for item * Update functionality for order * Delete functionality for user * Delete functionality for item * Delete functionality for order * Resolves issues specified by SonarQube | **1** | **0/13 points** |
| **2** | **4/13 points** |
| **3** | **8/13 points** |
| **4** | **10/13 points** |
| **5** | **13/13 points** |
| Able to use best-practice **standards** and **tools**, to achieve a well-engineered result. Ability to **review own work** and make changes. A point will be awarded for each of the following done correctly:   * Uses PascalCase for classes and camelCase for methods and variables * Trello board utilised * SonarQube passes code for **reliability** (bugs), **security** (vulnerabilities) and **maintainability** (code smells) * Feature branch model used * Pull requests used to resolve issues | **1** | **0/5 points** |
| **2** | **2/5 points** |
| **3** | **3/5 points** |
| **4** | **4/5 points** |
| **5** | **5/5 points** |

## Software Design

The specification and design of software to meet defined requirements by following agreed design standards and principles. The definition of software, components, interfaces and related characteristics. The identification of concepts and patterns and the translation into a design which provides a basis for software construction and verification. The evaluation of alternative solutions and trade-offs. The facilitation of design decisions within the constraints of systems designs, design standards, quality, feasibility, extensibility and maintainability. The development and iteration of prototypes/simulations to enable informed decision-making. The adoption and adaptation of software design models, tools and techniques based on the context of the work and selecting appropriately from predictive (plan-driven) approaches or adaptive (iterative/agile) approaches.

**Learner must**:  
Create and document detailed designs for simple software applications or components applying agreed modelling techniques, standards, patterns and tools. Contributes to the design of components of larger software systems. Reviews own work.

|  |  |  |
| --- | --- | --- |
| Sub-Skill | Rating | Points |
| Able to **document** project for understanding of its purpose. A point will be awarded for each of the following done correctly:   * Package documentation with package-info.java * Each method and class should have documentation explaining its purpose. This will be measured as a 5% document coverage per class. * Uses JavaDoc instead of comments * Has a README for the project * Risk analysis conducting on project | **1** | **0/5 points** |
| **2** | **2/5 points** |
| **3** | **3/5 points** |
| **4** | **4/5 points** |
| **5** | **5/5 points** |
| Able to create details designs for simple software applications or components. Apply agreed modelling techniques, standards, patterns and tools. One point will be awarded for each done correctly:   * Using UML to design Java classes * Uses ERD to design database tables * README contains information on prerequisites (items which need to be installed on the machine) * README contains information on what tools will be used to build the project. | **1** | **0/4 points** |
| **2** | **1/4 points** |
| **3** | **2/4 points** |
| **4** | **3/4 points** |
| **5** | **4/4 points** |

## Testing

The planning, design, management, execution and reporting of tests, using appropriate testing tools and techniques and conforming to agreed process standards and industry specific regulations. The purpose of testing is to ensure that new and amended systems, configurations, packages, or services, together with any interfaces, perform as specified (including security requirements), and that the risks associated with deployment are adequately understood and documented. Testing includes the process of engineering, using and maintaining testware (test cases, test scripts, test reports, test plans, etc) to measure and improve the quality of the software being tested.

**Learner must:**

Define test conditions for given requirements. Designs test cases and creates test scripts and supporting data, working to the specifications provided. Interpret, execute and record test cases in accordance with project test plans. Analyses and reports test activities and results. Identifies and reports issues and risks.

|  |  |  |
| --- | --- | --- |
| **Sub-skill** | **Rating** | **Details** |
| Able to **design**, **create** and **run** unit tests. One point will be awarded for each of the following done correctly:   * 30% code coverage * 50% code coverage * 70% code coverage * 90% code coverage | **1** | **0/4 points** |
| **2** | **1/4 points** |
| **3** | **2/4 points** |
| **4** | **3/4 points** |
| **5** | **4/4 points** |
| Analyse and report test activities. Identifies issues and risks. One points will be awarded for each of the following done correctly:   * Code analysed with SonarQube * Issues added to remote repository * Maven Surefire reports include junit tests. Report added to remote repository * Test analyse included in README * Risk analysis completed for the project | **1** | **0/5 points** |
| **2** | **2/5 points** |
| **3** | **3/5 points** |
| **4** | **4/5 points** |
| **5** | **5/5 points** |

## Systems integration and build

The planning, implementation and control of activities to integrate/build components, subsystems and interfaces to create operational systems, products or services for delivery to customers, or for internal or interim purposes such as testing. The development of organisational capabilities for systems integration and build including automation and continuous integration.

**Learners must:**

Produce software builds from software source code. Conduct 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.

|  |  |  |
| --- | --- | --- |
| **Sub-skill** | **Rating** | **Details** |
| Produce software builds from software source code. One point will be awarded for each of the following done correctly:   * Have a pom.xml in project root and installs maven on remote machine * Use the maven-jar-plugin to build an executable jar. * Automate creation of a jar with jenkins * Automation of deployment to Nexus * README contains information on how to conduct software build | **1** | **0/5 points** |
| **2** | **2/5 points** |
| **3** | **3/5 points** |
| **4** | **4/5 points** |
| **5** | **5/5 points** |
| Creates and runs **integration tests** and **reports** findings, reports issues and risks. One point will be awarded for each of the following done correctly:   * Integration tests run against a dummy database while source code runs against production database * Integration tests are not just big-bang integration * Maven Surefire reports that contain integration tests. Reports added to remote repository * Issues added to GitHub * Risk assessment submitted to GitHub | **1** | **0/5 points** |
| **2** | **2/5 points** |
| **3** | **3/5 points** |
| **4** | **4/5 points** |
| **5** | **5/5 points** |