|  |  |
| --- | --- |
| **QA Consulting.**   |  | | --- | |  | |
| Individual Project Specification |
| QAC Training Based Project Specification |

#### Prepared by

Matthew Hunt

[matthew.hunt@qa.com](mailto:#matthew.hunt@qa.com)

Contents

[Introduction 1](#_Toc7693344)

[Objective 1](#_Toc7693345)

[Scope 2](#_Toc7693346)

[Constraints 2](#_Toc7693347)

[Deliverable 3](#_Toc7693348)

[Marking Scheme 4](#_Toc7693349)

[Programming/Software Development – marking scheme 4](#_Toc7693350)

[Systems integration and build – marking scheme 5](#_Toc7693351)

# 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
* Databases
* Java SE/EE
* Front-End Development
* Automated Testing
* Continuous Integration

The individual project can be on any subject you deem fit of encapsulating all aspects of the aforementioned modules. This could be a business case, such as a library or supermarket system, or something to do with a hobby of yours – as long as the application is CRUD functional.

This is purposefully open as we want to endorse creativity and give you an opportunity to do a project that you have full command over.

# Objective

The overall objective of the project is the following:

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

# 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 Trello board (or equivalent Kanban board tech) with full expansion on user stories, use cases 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
* A functional application created in the OOP language, following best practices and design principles, that you have covered during training that meets the requirements set on your Kanban Board with a functioning front-end website and integrated API’s.
* Fully designed test suites for the application you are creating, as well as automated tests for validation of the application. You must also meet 80% test coverage in your backend and provide consistent reports and evidence to support a TDD approach.
* Code fully integrated into a Version Control System which will subsequently be built through a CI server and deployed to a cloud-based virtual machine

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 an equivalent Kanban Board
* Database: h2
* Programming language: Java (unless stated otherwise)
* Front-end: HTML, CSS and JavaScript, possibly using a front-end library such as React
* Unit Tests: JUnit and Mockito
* Integration Tests: Selenium
* Version Control: Git
* CI Server: Jenkins
* Cloud server: AWS, GCP or Azure virtual machine

# 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 based on the domain that you have chosen.

A presentation of work may also be required towards the end of the deadline. However, you will be required to produce weekly reports of any designs and work created throughout the duration of the project.

# Marking Scheme

Below are the skills that we will be evaluating for this Databases 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 skills this assessment will discussed are the following:

* Programming/software development
* Systems integration and build

Programming/software development

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

Systems integration and build

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.

**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.

**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.

Programming/Software Development – marking scheme

Below is the list of criteria that will be assessed from your deliverable:

|  |  |  |
| --- | --- | --- |
| **SFIA Skill** | **Rating** | **Details** |
| Designs, codes, verifies, tests, documents, amends and refactors simple programs/scripts. | 1 | Software is missing functionality in major areas for creating, reading, updating and deleting. No tests implemented and designs documented were not implemented. Best practices not adhered to in the project. |
| 2 | Software functionality is working in areas but is not a fully working product. Tests were not implemented; code was commented in small areas. Parts of the project adhered to best practices but not consistently throughout the software. |
| 3 | Software is functional in all major areas but still has small bugs and/or errors. Tests were beginning to be implemented, with basic functions being tested. Best practices were adhered to for most areas of the project. |
| 4 | Software is fully functional and has been tested in all relevant areas. Best practices were consistently adhered to throughout the project. |
| 5 | Software is fully functional and has been tested in all areas, with best practices and refactoring adhered to and implemented throughout the project. Software implements concepts outside of the brief specified at a good level. |
| Applies agreed standards and tools, to achieve a well-engineered result. | 1 | Tools and workflows discussed in software documentation around design not implemented to a quality capacity. |
| 2 | Tools and workflows are referred to within the project, but the implementation is minimal. |
| 3 | Tools and workflows discussed in documentation are implemented throughout the project at a high level, but more exploration could have occurred. |
| 4 | Software produced is in line with the documentation with changes made where needed, with references to refactoring apparent in key areas. |
| 5 | Software fully implements designs with justifications as to why changes were made where needed. Refactoring of code is rife throughout all relevant areas of the software. |

Systems integration and build – marking scheme

Below is the list of criteria that will be assessed from your deliverable:

|  |  |  |
| --- | --- | --- |
| **SFIA Skill** | **Rating** | **Details** |
| Produces software builds from software source code | 1 | VCS implementation was non-existent. Build server was not installed and therefore no builds of software were created. |
| 2 | VCS implementation was attempted but structure was poor and/or content in VCS was irrelevant. Build server was installed but software did not build successfully from repository. |
| 3 | VCS was implemented and code was stored in a structured manner. Some of the content in the VCS could have been omitted but the majority was relevant. Build server installed and successfully built software manually. |
| 4 | VCS was implemented and code was stored in a structured manner with branches. All VCS content was relevant, no unnecessary files. Build server installed and successfully built software after a push. |
| 5 | VCS was implemented and code was stored in a structured manner with branches, git ignore file and scripted hooks. All files relevant to the repo, nothing unnecessary. Build server installed and successfully built software after a push with artefact produced for successful builds. |
| 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. | 1 | No tests written for back-end or front-end. No logs or reports about application produced. |
| 2 | Basic tests written for back-end, no implementation of front-end tests (or vice versa). Results produced not acknowledged in any way. |
| 3 | Basic tests written for both the back-end and front-end. Results are tabulated and logged but no further follow up. |
| 4 | Majority of tests for relevant features within the app are created in both front and back-end of application. Results of tests logged and discussed in report for test coverage. |
| 5 | Tests for all relevant CRUD features within the app are created in both the front and back-end of application. Results of tests logged and discussed in report for test coverage. Further addition of code quality checking is also implemented. |

Software Design – marking scheme

Below is the list of criteria that will be assessed from your deliverable:

|  |  |  |
| --- | --- | --- |
| **SFIA Skill** | **Rating** | **Details** |
| Creates and documents detailed designs for simple software applications or components applying agreed modelling techniques, standards, patterns and tools. | 1 | No Designs provided |
| 2 | Simple designs provided but not adhering to agreed standards and patterns. |
| 3 | Simple designs provided meeting the agreed standards and patterns. |
| 4 | Extensive Designs provided on both a system level and component level. |
| 5 | Evolution of designs evident as the project progressed in all aspects. |

Testing – marking scheme

Below is the list of criteria that will be assessed from your deliverable:

|  |  |  |
| --- | --- | --- |
| **SFIA Skill** | **Rating** | **Details** |
| Designs test cases and creates test scripts and supporting data | 1 | No tests written. |
| 2 | Basic tests written for back-end. Results produced not acknowledged in any way. |
| 3 | Adequate tests written for the back-end. |
| 4 | Majority of tests for relevant features within the app are created. |
| 5 | Tests for all relevant CRUD features within the app are created. Further addition of code quality checking is also implemented via a tool like SonarQube. |
| Analyses and reports test activities and results | 1 | No tests written. No logs or reports about application produced. |
| 2 | Basic tests written for back-end. Results produced not acknowledged in any way. |
| 3 | Results of tests are tabulated and logged but no further follow up. |
| 4 | Results of tests logged and discussed in report for test coverage. |
| 5 | Results of tests logged and discussed in report for test coverage. Further addition of code quality checking is also implemented, with the results provided. |