Practical Project Specification:

Hobby Web Application (HWA)

Last revision: May 2020

CONTENTS

[Introduction 3](#_Toc41496493)

[Objective 3](#_Toc41496494)

[Scope 4](#_Toc41496495)

[Constraints 5](#_Toc41496496)

[Deliverable 5](#_Toc41496497)

[Deliverables Checklist (MVP) 6](#_Toc41496498)

[Mark Scheme 7](#_Toc41496504)

# 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 encapsulate concepts from all core training modules – more specifically, this will involve:

* Agile & Project Management
* Databases & Cloud Fundamentals
* Programming Fundamentals
* Automated Testing
* Continuous Integration
* Front-End Web Technologies
* API Development

You may tweak this project to any subject, business case, or hobby which you deem fit – e.g. a library, supermarket system, statistical tracker, etc. – provided that it encapsulates all aspects of the aforementioned modules, and that it is fully CRUD functional.

# Objective

The overall objective of the project is the following:

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

Specifically, you are required to create a full-stack web application following the Enterprise Architecture Model, using:

* an application back-end developed using the language from your Programming Fundamentals module (e.g. Java)
* a managed database hosted within the Cloud Provider examined during your Cloud Fundamentals module (e.g. MySQL in GCP)
* a front-end developed using the language from your Front-End Web Technologies module (e.g. JavaScript)

**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 you have learned. You are also expected to utilise a CI Pipeline that can automate the building and deployment of your artefact. Finally, your project is expected to have been rigorously tested.

# 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, which will subsequently be built through a CI server and deployed to an artefact repository manager.
* 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, located within the Cloud, which is used to persist data for the project. This database must contain at least **two entities**, with their relationships modelled using an ERD.
* A functional application ‘back-end’, written in the language that you have covered during training, created following OOP principles and best practice, that meets the requirements set on your Kanban board.
* A build of your application, including any dependencies it might need, produced using an integrated build tool, and with relevant static analysis and artefact repository tools utilised.
* A functional ‘front-end’ website (and integrated APIs) which connect to your back-end.
* Fully designed test suites for the application you are creating, as well as automated tests for validation of the application. You should aim to reach the industry-standard of **80%** test coverage through both **unit** and **integration** testing.

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

**Ensure that you complete all the requirements above before adding extra functionality that is not explicitly 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**: Trello / Jira / GitHub Boards
* **Database Management System**: GCP instance of MySQL Server
* **Back-End Programming Language**: Java
* **API Development Platform**: Spring
* **Front-End Web Technologies:** HTML, CSS, JavaScript
* **Build Tool:** Maven
* **Unit Testing:** JUnit, Mockito
* **CI Server:** Jenkins
* **Static Analysis:** SonarQube
* **Test Reporting**: Surefire
* **Artefact Repository:** 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.

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. 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
* Sensible package structure in the back-end
* Sensible folder structure in the front-end
* Adherence to best practice (e.g. OOP principles, SOLID, refactoring)

## Testing

* Unit testing for the project back-end
* Integration testing for the project front-end
* Test coverage of the **src/main/java** aiming for **80%**

## Continuous Integration

* GitHub repository utilising the feature-branch model (**master**/**dev**/**features**)
* A **master** Jenkins pipeline job (for final builds)
* A **dev** Jenkins pipeline job (for development purposes)
* Code pushed to your personal SonarQube
* Code pushed to your trainer’s SonarQube
* A fat .jar deployed to your trainer’s Nexus

## Repository & Documentation

* A completed Kanban board, including user stories, acceptance criteria, story points, and MoSCoW methodology
* A working .gitignore for ignoring build-generated files and folders
* A reasonably-sized README.md explaining how to use your hobby system
* A completed risk assessment (utilising a matrix)
* An ERD for your Minimum Viable Product
* A folder for your testing reports
* Your presentation

## Presentation Guideline

* **Introduction**: Welcome, specification, scope
* **Concept:** What hobby did you use and why?
* **Sprint plan:** What needed to be included? What did you hope to achieve?
* **Consultant Journey:** What technologies have you learned for the project?
* **CI:** How did you set up your CI pipeline?
* **Testing**: Coverage, static analysis, red-green-refactor
* **Demonstration:** Run through a couple of user stories
* **Sprint review:** What did you complete during the sprint?
* **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

# 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.