# Practical Project (DFE) Specification

**Last revision:** November 2021

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

- Agile & Project Management (Git, Jira)
- Databases & Cloud Fundamentals (H2, MySQL)
- Programming Fundamentals (Java)
- API Development (Spring Boot)
- Automated Testing (JUnit)

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

Your API must be fully functional and capable of handling HTTP requests from a tool such as Postman. You must be able to show that data has been persisted to your database using a tool such as MySQL Workbench or the H2 console..

# Objective

The overall objective of the project is the following:

**To create a Spring Boot API, 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 Spring Boot API using:

- an application back-end developed using the language from your Programming Fundamentals module (e.g. Java)
- a managed database hosted locally or within the Cloud Provider examined during your Cloud Fundamentals module (e.g. H2 or MySQL (local / GCP)
- a means of making API calls (Postman) and a means of checking persistence (Workbench/H2 console)

**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.

Your project is expected to have been rigorously tested in the key areas explored above, using the technologies covered during all fundamental and practical modules (e.g. JUnit).

# 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, locally or within the Cloud, which is used to persist data for the project.
- A functional application ‘back-end’, written in a suitable framework of the language covered in training (Java/Spring Boot), which meets the requirements set on your Scrum Kanban board.

- A build (.jar) of your application, including any dependencies it might need, produced using an integrated build tool (Maven).
- A series of API calls designed with postman, used for CRUD functionality. (Create, Read, Update, Delete)
- Fully designed test suites for the application you are creating, including both **unit** and **integration** tests**.**

**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 your training:

- **Version Control System**: Git
- **Source Code Management:** GitHub
- **Kanban Board**: Jira (Scrum Board)
- **Database Management System**: H2 or MySQL Server (local or GCP)
- **Back-End Programming Language**: Java
- **API Development Platform**: Spring
- **Build Tool:** Maven
- **Unit & Integration testing:** JUnit

# Deliverable

The final deliverable for this project is a github repository containing the completed Spring Boot API & a build of the application (.jar), as well as any documentation listed in the **scope** section above. You should also include a link to your Jira board in the README.md file.

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 deliverables for your project within your remote repository at close-of-business (17:30) on the day of submission – please refer to the **Deliverables Checklist (MVP)** below for further details.

# Deliverables Checklist (MVP)

### Codebase

- Spring Boot API with full CRUD functionality.
- Sensible package structure (back-end).
- Adherence to best practices.

### Testing

- Unit and integration testing for the project back-end.
- Reasonable test coverage of the **src/main/java folder.**

### Continuous Integration

- GitHub 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 .**jar** which can be deployed from the command-line (java -jar <filename.jar>)

### Repository & Documentation

- A completed **project management board**, including user stories, acceptance criteria, estimations via story points, and prioritisation via MoSCoW methodology. **You must add Morgan Walsh and Jordan Benbelaid as collaborators on your Jira board.**
- A working **.gitignore** for ignoring build-generated files and folders
- You are also expected to make a **README.md** file and to fill this **README** with information about this project. It should contain the following headers:
    - Why are we doing this?
    - How I expected the challenge to go.
    - What went well? / What didn't go as planned?
    - Possible improvements for future revisions of the project.
    - Screenshots showing your postman requests and the output from the API.
    - Screenshots of your database to prove that data is being persisted.
    - Screenshot of your test results, including coverage report.
    - Link to Jira Board - **You must add Morgan Walsh and Jordan Benbelaid as collaborators on your Jira board.**
- A **documentation** folder containing:
    - A completed **risk assessment**, utilising a matrix, in **.pdf** format

### Stretch Goals:

- Create an ERD diagram for your specified domain and include it in the documentation folder.
- Use custom queries, such as ‘find by name’.
- Include custom exceptions in your service.
- Include the use of DTOs
- Include the use of Lombok
- Implement Mockito
- Achieve 80% test coverage of the **src/main/java** folder.

# 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.