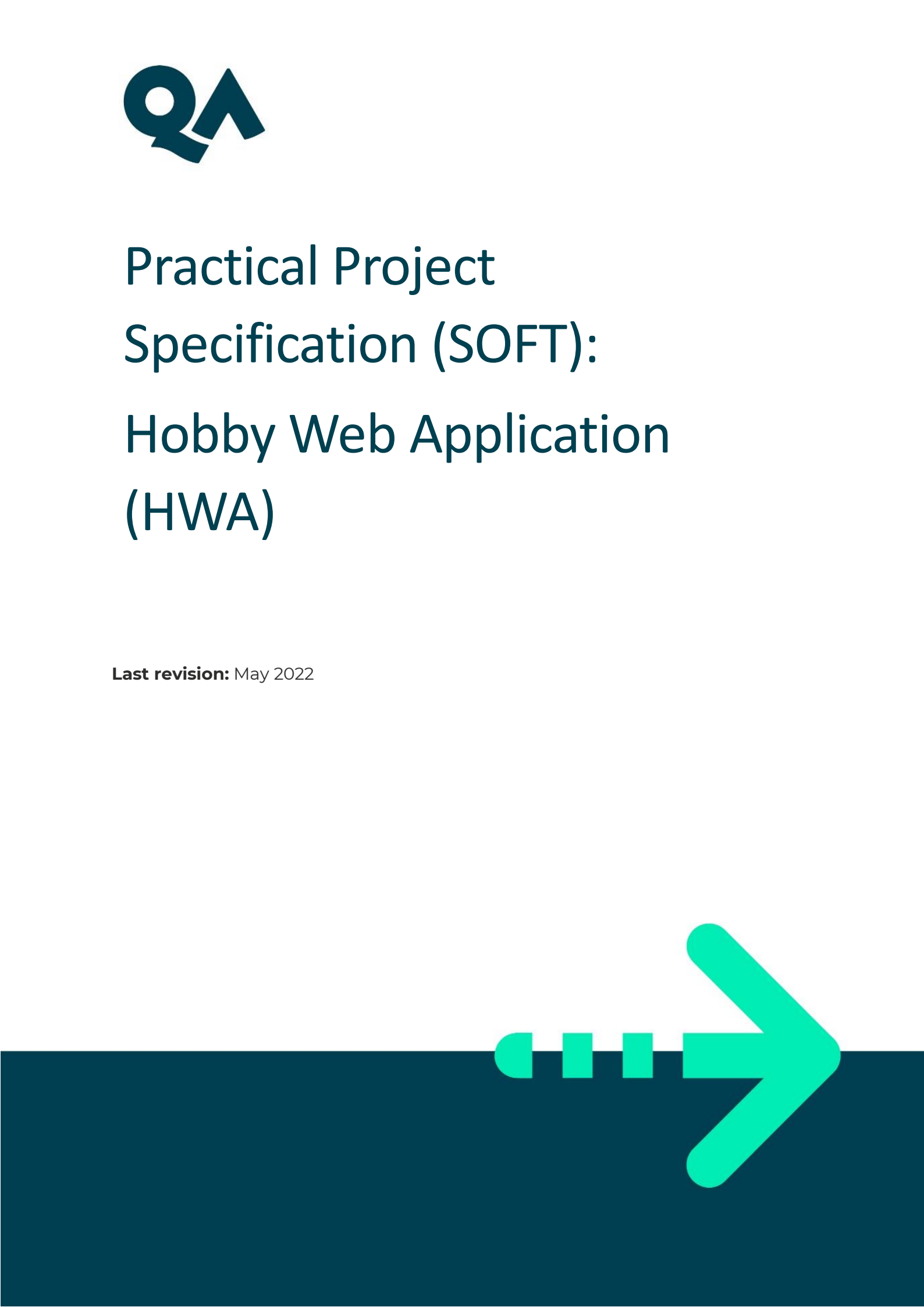

Practical Project Specification (SOFT): Hobby Web Application (HWA)

Last revision: May 2022

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
- Databases & Cloud Fundamentals
- Java/Spring Boot API Development
- Front-End Development
- Automated Testing

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.

Your front-end and back-end must be linked together and fully functional, please bear this in mind when organising your time.

.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 Java (Spring Boot Framework).
- A test suite utilising JUnit and Mockito for integration and unit tests of the back-end.
- A managed MySQL database hosted locally.
- A front-end developed using Javascript, HTML and CSS.

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.

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, hosted locally, which is used to persist data for the project. This database must contain at least **one entity**, with their structure and relationships modelled using an ERD. (Still produce ERD even if only one entity is used).
- A functional application created in Java (Spring Boot), 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.
- A build of your application, including any dependencies it might need, produced using an integrated build tool.
- A functional 'front-end' website which connects to your back-end API.
- Fully designed test suites for the application you are creating. You should aim to reach the industry-standard of **80%** test coverage through **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 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:

- **Version Control System**: Git
- **Source Code Management**: GitHub
- **Kanban Board**: Jira
- **Database Management System**: MySQL Server (Local)
- **Back-End Programming Language**: Java
- **API Development Platform**: Spring Boot
- **Front-End Web Technologies**: HTML, CSS, JavaScript
- **Build Tool**: Maven
- **Unit Testing**: JUnit, Mockito

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.

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 **main** branch at the end, for the marking of the project. It is recommended to use the **feature-<concept>** 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 (17:30) on your deadline day – please refer to the **Deliverables Checklist (MVP)** for further details.

Deliverables Checklist (MVP)

Codebase

- CRUD functionality in both the API and front-end.
- Sensible package structure (back-end) and folder structure (front-end).
- Adherence to best practices.

Testing

- Unit and integration testing for the project back-end.
- Test coverage of the **src/main/java** folder, aiming for **80%**

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

Repository & Documentation

- A completed **project management board**, including user stories, acceptance criteria, estimations via story points, and prioritisation via MoSCoW methodology.
- A working **.gitignore** for ignoring build-generated files and folders.
- A completed **README.md**, explaining how to use and test your application
- A **documentation** folder containing:
    - A completed **risk assessment**, utilising a matrix, in **.pdf** format
    - **At least one ERD** and **one UML** diagram, in **.png** format
    - A Screenshot of the **Swagger API** documentation
    - A copy of your presentation, in **.pdf** format (slides only – no notes)

Presentation Guideline

- **Introduction**: Who are you? How did you approach the specification?
- **Concept**: How did you initially approach the problem?
- **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 approach version control?
- **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
- Your presentation should last a total of **15 minutes**