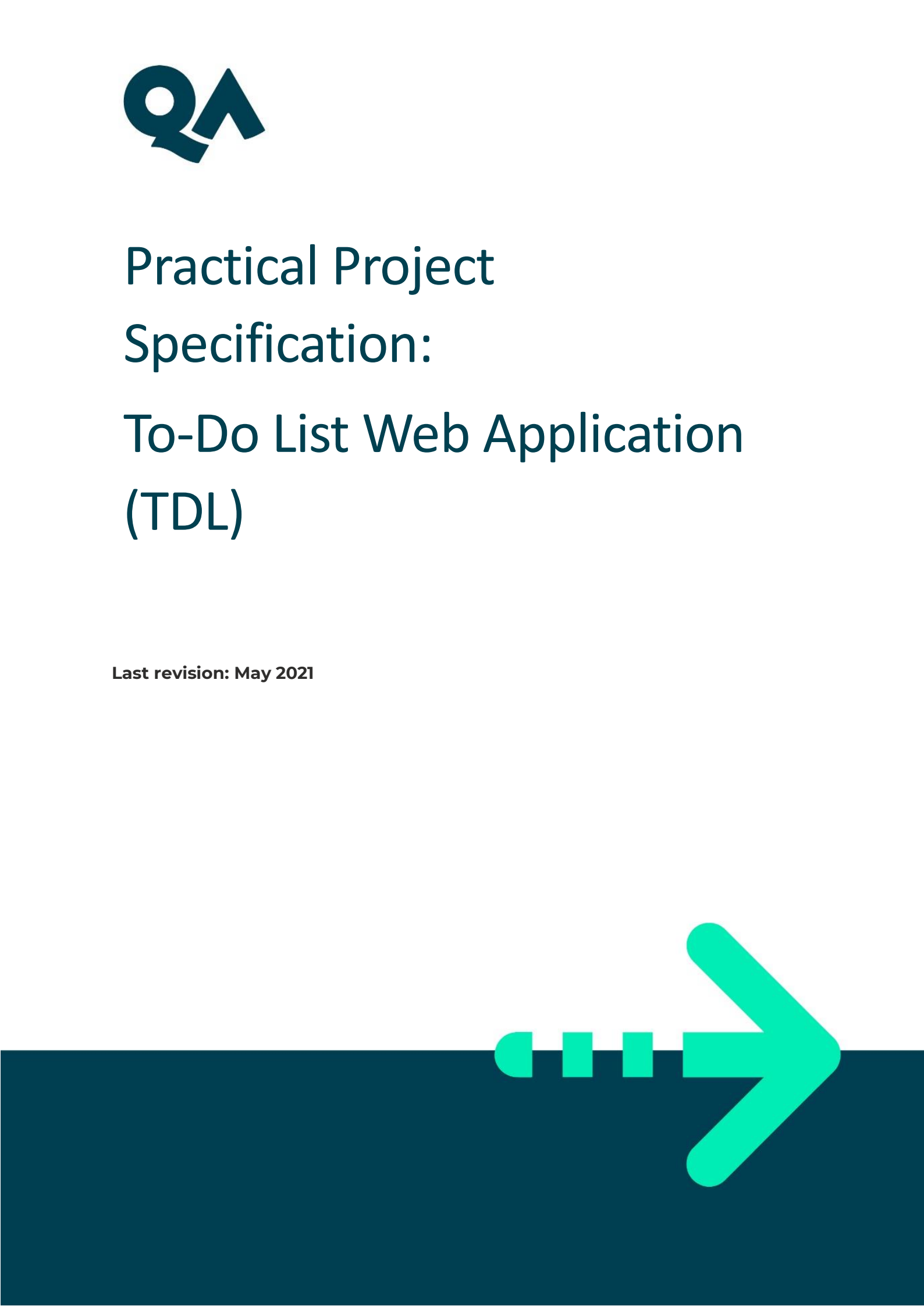

Practical Project Specification:

To-Do List Web Application (TDL)

Last revision: May 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
- Databases
- Programming Fundamentals
- Front-End Web Technologies
- API Development
- Automated Testing

Your front-end and back-end for this project 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 CRUD-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 with a multi-tier architecture, using:

- an application back-end developed using the language from your API development training - Spring Boot/Java.
- a managed database hosted locally – MySQL Server.
- a front-end developed using the languages from your Front-End Web Technologies modules – HTML, CSS & JS.

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 and document any designs you make via screenshots or saved diagrams. These should be included in your documentation folder.

Your project is expected to have been tested using the technologies covered (JUnit, Mockito).

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 (git) using the feature-branch model: **main/dev/multiple features**.
- A project management board (Jira) with full expansion on user stories, and tasks needed to complete the project. These should include story point estimations and priorities. A sprint should be used.
- 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**. If you include more than one entity, any relationship between them should be modelled with an Entity Relationship Diagram(ERD).
- A functional application 'back-end' API written using the Java framework, Spring Boot.
- A build of your application, including any dependencies it might need, produced using an integrated build tool (Maven).
- A functional 'front-end' website which connects to your back-end API and can be used to perform CRUD operations on your database.
- Unit and integration tests for your application. 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 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
- **Database Management System**: MySQL Server 8.0+
- **Back-End Programming Language**: Java
- **API Development Platform**: Spring
- **Front-End Web Technologies**: HTML, CSS, JavaScript
- **Build Tool**: Maven
- **Unit & Integration Testing**: JUnit, Mockito

Deliverable

The final deliverable for this project is the completed application with 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. It is recommended to use the **feature-<concept>** naming strategy for your feature branches. You must ensure your dev branch is merged into your main branch at the end of the project. **Only** the work present on the **main** branch will be marked.

You will be required to include your work and all supporting documentation for your project within your remote repository at close-of-business (17:30) 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 Multi-Tier Architecture.
- Sensible package structure (back-end) and folder structure (front-end).
- Adherence to best practices.

Testing

- Test suites for back-end **unit and integration testing**
    - Test coverage of the **src/main/java** folder, aiming for ~**70-80%**

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 compiled archive which can be deployed from the command-line

Repository & Documentation

- A completed **project management board(Jira)**, including user stories, story points, and MoSCoW prioritisation.
- 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
    - An image of your API endpoints.
    - 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**: Process, coverage, refactoring etc
- **Demonstration**: A brief demo of a couple of your applications features
- **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**

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.