**Development**

Java-Solo Project Brief

Brief

# Overview

The solo project represents your opportunity to implement all of the techniques and topics that you have covered during your training into a functional application whilst working in an Agile way.

You will work on it alongside your training for a period of **3 weeks**, at which point, you will give a presentation covering what you have produced, how you have implemented it and what you learned from the experience.

## Learning Outcomes

* Experience of implementing functional business requirements into a full application.
* Experience of using various technologies and techniques covered during your training together
* Practise of time management. You will be working on your project alongside the taught course content. It will be up to you to manage your time to balance your learning with completion of your project.

## Things You Will Need Before You Start

Your trainer should have gone through the Agile Project Briefing with you before handing out the project brief.

You will be provided with an A3 piece of paper and some post-it notes that you should use as a **Kanban** board to keep track of your tasks and sprints.

## Approach

Before you start, you should produce a project brief from the perspective of a User; this should form a set of **Functional Requirements.**

You should then translate these functional requirements into **Technical Requirements** and a **Project Backlog**.

You should work in **2-day sprints**, aiming to produce a concrete **deliverable** at the end of every sprint.

At the end of each Sprint, you should perform a **Sprint Retrospective** to reflect on what you have completed over the previous 2 days. What went well and what will you do differently next time?

# Project

You should select a project from those listed below. If you have already been placed, aim to produce a project within the business domain of the client you will be working with.

All projects should be Web Applications utilising the technologies and techniques that are covered over the course of the training.

## Projects Suggestions

* E-Commerce
* Auction Site
* Charity Giving
* CMS with Social Features
* Online Gaming (sports betting)
* Video on Demand
* News
* Parcel / Delivery Tracking
* Chat System
* CRM System
* Helpdesk System
* Defect Tracking Tool

## Notes on Implementation

All projects must conform to the following architecture:

1. Database (Oracle)
2. MVC Implementation using Servlets or Spring Web-MVC
3. Web Server

Technologies

* Tomcat Hosted Web Application
* All logic should be unit tested
* TDD: required for all classes

Acceptable Web Server architecture:

* MVC
* TDD: required for all “model” and “controller” classes
* TDD: not required for “view” components (jsp/css/javascript)

## Advice

You will be given the project before learning about all of the technologies you might use.

As you learn about a technology, consider using it in your architecture. Where you have a choice of technology – choose one.

Have a rough idea of how you might split logic into different classes/packages whilst conforming to the SOLID design principles.

Before beginning TDD – have a rough plan of the classes and interfaces you think you will need to develop.

Work in an Agile fashion (deliver a single working feature before moving to another).

Create a good design of classes/interfaces that is **easy to extend,** to enable you to add new features later.

The project will require a lot of work; implementing good SOLID design principles will help you with code re-use and TDD.

## Assessment

The Presentation Code Review assessment is primarily looking for **good design** and **extensive use of Test Driven Development**.

**A well designed and implemented system is more important than a large system that has been built poorly.**

You will be assessed on:

* Presentation technique
* UML (Use Case, Class Diagram, ERD) – these may be hand drawn
* Implementation of SOLID and OO design principles
* Appropriate use of Design Patterns
* Implementation of a fully normalised database
* Implementation of MVC/N-Tier
* Demonstration of functionality
* Use of a suitable logging framework
* Discussion of technology stack used
* Use of Source Control
* Implementation of Security and Authentication
* Implementation of TDD using a suitable framework, including mocking
* Suitable code coverage
* Usage of a data access framework
* Evidence of reflection regarding lessons learnt