**DfE Cloud Specialism - Final Project**

Introduction

This project serves as the culmination of all of the topics you have covered as part of your training. It will involve the following concepts/technologies:

Project Management

Python Fundamentals

Python Testing

Git

Linux

Python Web Development

Databases

Continuous Integration and Deployment (CI/CD)

Cloud Fundamentals

Containerisation

This project is an individual project designed to demonstrate your knowledge.

Overview

Your objective with this project is to achieve the following:

To create a web application that integrates with a database and demonstrates CRUD functionality.

To utilise containers to host and deploy your application.

To create a continuous integration (CI)/continuous deployment (CD) pipeline that will automatically test, build and deploy your application.

Project Management and Version Control

Technologies:

Jira (or equivalent)

Git

GitHub

You must use Jira or similar project tracking software to track your project using Agile Scrum methods. You must make use of:

MoSCoW prioritisation.

Estimations of effort (e.g. story points, T-shirt sizes).

User stories.

You will be using Git as your version control system and hosting your code repository using GitHub. You must make use of a feature branch model to complete your work.

You must also produce a risk assessment to identify and analyse any potential risks to your application and infrastructure. This must be evidenced in your documentation.

Application

Technologies:

Python

Pytest

Flask

Docker/Docker Compose

MySQL

You are tasked to create a basic web application using on the Flask web framework.

It must demonstrate CRUD functionality.

The domain/topic of the application is entirely down to you. It's recommended you choose a hobby of yours to base it on. The app must keep track of two entities that have some kind of relationship with each other.

For example:

A fantasy football application that allows you to create football teams and add players to them.

A DnD character creator that allows you to create characters based on fantasy races.

A recipe book app that keeps track of recipes and ingredients.

This relationship between entities is to be modelled in the database. You must illustrate this relationships using an entity relationship diagram (ERD) in your documentation.

Application

The application is a monolithic Flask application that serves both the frontend and backend of the application.

The frontend aspect of the app will use HTML templates to serve the web pages that allow the user to perform CRUD functionality with information from the database.

The backend aspect of the application will use SQLAlchemy to model and integrate with the database.

Diagram for the application service architecture

This application will be hosted in a container to allow it to be deployed to a Docker Swarm.

Database

The application must interface with a separate database service. You are encouraged to use a MySQL container (documentation) for the sake of simplicity, but you may use a database service such as MySQL Database for Azure if you wish to challenge yourself.

The database must contain two tables with a relationship.

The relationship must at least be one-to-many, but a many-to-many relationship may garner extra marks.

For example, a fantasy football app may have a teams table and a players table. The relationship between these two tables is one-to-many, as a player has one team, but a team has many players.

Testing

Unit tests must be written for the application with the aim of achieving high coverage.

You will use pytest to write and test the application.

CI/CD Pipeline

Technologies:

Jenkins

You are also tasked with creating a CI/CD pipeline that will automate the integration and deployment of new code.

The automation server you must use is Jenkins.

You have freedom to organise the pipeline however you see fit, but the pipeline must achieve the following:

Run unit tests.

Build the Docker images.

Push the Docker images to a registry.

Deploy to a Swarm.

Every time you push new code to your GitHub repository, the pipeline should be triggered. This can be achieved using a GitHub Webhook.

Deployment

Technologies:

Docker Swarm

The application should be deployed to a Docker Swarm hosted in the cloud.

It should consist of at least one manager node and one worker node. Neither of these nodes should be the Jenkins build server.

Deliverable

The final deliverable of this project is a GitHub repository containing all of the code you have written for this project, including the software source code, Jenkins configuration, Docker configuration and any related scripts.

Your GitHub repository should also contain the write-up for your project in the form of a README.md file.

You must also provide video evidence of your application and CI/CD pipeline working.

You are required to deliver this project to complete the course. The deadline will be the final day of your bootcamp. All of your work must be merged into your main branch, ready for submission.

You must provide your trainer with the link to your repository.

Documentation

You are required to create a write-up for this project in the form of a README file on your repository.

You need to include:

An explanation of your app and how it fulfils the brief.

A technical description of how the application works.

A technical description of how the pipeline works.

A report on the success and code coverage of your unit tests.

Any future improvements you would make.

You must use diagrams to illustrate your work as much as possible and opt for a succinct writing style. Examples of diagrams to include are:

Entity Relationship Diagram (ERD).

A full CI/CD pipeline diagram.

An infrastructure diagram, illustrating the cloud resources and how they network together.

A component-level diagram, illustrating how the application interfaces with the database.

You are welcome to combine diagrams if you so wish, such as the CI/CD pipeline with the infrastructure pipeline.

Video

You should also record a brief video (between 2-5mins) demonstrating:

The CRUD functionality of your app.

Your CI/CD pipeline in action, show how after a new push to your GitHub repository your changes should be automatically reflected on the deployed application without manual configuration.

Consult the following links for guidance on recording your screen:

Recording your screen on Windows

Recording your screen on MacOS

If you are struggling to record your screen, please ask your trainer for support.

A voiceover is not required for this video evidence, though it may aid your demonstration if you can record it.

This video should be hosted on a file sharing site (Google Drive, Dropbox, etc.) and linked in your README file. Make sure that the link is public so whoever is marking it can view it.

Marking Scheme

Below are the skills that we will be evaluating for this

assessment. These skills are as described in the SFIA 7 framework;

please see below if you wish to have more information:

https://www.sfia-online.org/en/framework

The skills this assessment will discuss are the following:

Programming/software development

Systems integration and build

Software Design

Release and Deployment

Programming & Software Development

Designs, codes, verifies, tests, documents, amends and refactors simple

programs/scripts. Applies agreed standards and tools, to achieve a

well-engineered result. Reviews own work.

Below is the list of criteria that will be assessed from your

deliverable:

SFIA Skill Rating Details

Designs, codes, verifies, tests, documents, amends and refactors simple programs/scripts. 1 Software is missing functionality in major areas for creating, reading, updating and deleting. No tests implemented and designs documented were not implemented. Best practices not adhered to in the project.

2 Software functionality is working in areas but is not a fully working product. Tests were not implemented; code was commented in small areas. Parts of the project adhered to best practices but not consistently throughout the software.

3 Software is functional in all major areas but still has small bugs and/or errors. Tests were beginning to be implemented, with basic functions being tested. Best practices were adhered to for most areas of the project.

4 Software is fully functional and has been tested in all relevant areas. Best practices were consistently adhered to throughout the project.

5 Software is fully functional and has been tested in all areas, with best practices and refactoring adhered to and implemented throughout the project. Software implements concepts outside of the brief specified at a good level.

Applies agreed standards and tools, to achieve a well-engineered result. 1 Tools and workflows discussed in software documentation around design not implemented to a quality capacity.

2 Tools and workflows are referred to within the project, but the implementation is minimal.

3 Tools and workflows discussed in the documentation are implemented throughout the project at a high level, but more exploration could have occurred.

4 Software produced is in line with the documentation with changes made where needed, with references to refactoring apparent in key areas.

5 Software fully implements designs with justifications as to why changes were made where needed. Refactoring of code is rife throughout all relevant areas of the software.

Systems Integration and Build

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.

Below is the list of criteria that will be assessed from your

deliverable:

SFIA Skill Rating Details

Produces software builds from software source code 1 VCS implementation was non-existent. Build server was not installed and therefore no builds of software were created.

2 VCS implementation was attempted but structure was poor and/or content in VCS was irrelevant. Build server was installed but software did not build successfully from repository.

3 VCS was implemented and code was stored in a structured manner. Some of the content in the VCS could have been omitted but the majority was relevant. Build server installed and successfully built software manually.

4 VCS was implemented and code was stored in a structured manner with branches. All VCS content was relevant, no unnecessary files. Use of a configuration file to make build jobs portable.

5 VCS was implemented and code was stored in a structured manner with branches, git ignore file and scripted hooks. All files relevant to the repo, nothing unnecessary. Build server installed and successfully built software after a push with artefact produced for successful builds.

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. 1 No tests written for back-end. No logs or reports about application produced. No risks recorded or monitored.

2 Basic tests written for back-end. Results produced not acknowledged in any way. Some Risks acknowledged but not formally followed up.

3 Basic tests written the back-end. Results are tabulated and logged but no further follow up. A simple Risk analysis has been performed

4 Majority of tests for relevant features within the app are created in the back-end of application. Results of tests logged and discussed in report for test coverage. A more formal Risk assessment process has been followed.

5 Tests for all relevant CRUD features within the app are created in both the back-end of the application. Results of tests logged and discussed in report for test coverage. Further addition of code quality checking is also implemented. A formal risk assessment process followed and updated, analysis performed at end of project.

Software Design

The specification and design of software to meet defined requirements by

following agreed design standards and principles. The definition of

software, components, interfaces and related characteristics.

Below is the list of criteria that will be assessed from your

deliverable:

SFIA Skill Rating Details

Creates and documents detailed designs for simple software applications or components applying agreed modelling techniques, standards, patterns and tools. 1 No Designs provided

2 Simple designs provided but not adhering to agreed standards and patterns.

3 Simple designs provided meeting the agreed standards and patterns.

4 Extensive Designs provided on both a system level and component level.

5 Evolution of designs evident as the project progressed in all aspects.

Release and Deployment

The management of the processes, systems and functions to package,

build, test and deploy changes and updates (which are bounded as

"releases") into a live environment, establishing or continuing the

specified Service, to enable controlled and effective handover to

Operations and the user community.

Below is the list of criteria that will be assessed from your

deliverable:

SFIA Skill Rating Details

Uses the tools and techniques for specific areas of release and deployment activities. 1 No release or deployment activities undertaken.

2 Release and deployment activities attempted; however results have not been a success.

3 System has been successfully deployed.

4 System can be robustly deployed with no further configuration required.

5 Advanced deployment and release techniques and tools have been utilised in order to create a very robust system.

Administers the recording of activities, logging of results and documents technical activity undertaken. 1 No deployment activities undertaken, or no logs or reports about application produced.

2 Deployment activities completed but not acknowledged in any way.

3 Deployment activities are tabulated and logged but no further follow up.

4 Deployment activities logged and discussed in a report. Certain advanced tools and techniques used have been explained thoroughly.

5