Player Database Management System

A leading professional Sporting organization has decided to develop a database management system to manage its entire players detail. It has hired a technology partner to put in place, the necessary infrastructure required including creating the necessary software. As part of requirements, the below mentioned specification have been given to the partner to implement.

Requirements Specification:

1. Create the players database with fields such as ID, Name, Age, Email and Address
2. Read/Select a set of records based on ID
3. Update a player's age or email by accessing the player's detail with his /her ID
4. Delete a particular player's record

Note:

- Make your own assumptions while developing the players database.
- The developed application should be tested with any unit test framework.
- Use tools such as Selenium to perform functional testing
- All the above requirements have to be done in DevOps way .

The partner organization has selected

- GIT as its SCM tool
- Maven as a build process tool to build and test the application using Junit framework in an incremental fashion
- Selenium to test the application developed
- Artifactory to store binary to be deployed.
- Ansible to configure the server environment
- Docker to containerize the application
- Jenkins for orchestration

Note: During integrated learning project execution, you may use any alternative tools.

The partner organization has selected

- GIT as its SCM tool
- Maven as a build process tool to build and test the application using Junit framework in an incremental fashion
- Selenium to test the application developed
- Jfrog Artifactory to store binary to be deployed.
- Code Quality report from SonarQube
- Jenkins for orchestration

Note: During integrated learning project execution, you may use any alternative tools.

Project Inputs:

- Application Source code
- Web configuration files & UI pages
- Junit framework based unit test code s
- Functional Testing scripts using selenium

- Required dependency details if any. (pom.xml)
- Read Me (file name: readme.txt => Refer to know services details)

Note: Use of give sample code is optional. You may develop your own code for above requirements and tool integration. Use above code for learning/reference only.

Project Tasks

As a DevOps Professional in the project, you are expected to do the following tasks:

1. Create a Git Repository Project in GitLab – Use below mentioned naming convention
    a. <Your ADID>/DevOpsProfessional/ ILP_CI-CD/<projectTitle>
2. Create a Project structure as per the Build tool and do an initial commit /Push to Gitlab
3. Experiment to validate / verify the execution of solution as per the requirements.
    - Example: Run any/all Maven goals
4. If step 3 is successful then configure and create Jenkins job to perform build process automation
    - Ensure automated build trigger and notification
5. Take necessary/suitable snapshots as listed below.
    a. Gitlab Commit history
    b. Jenkins Global Tool Configuration
    c. Jenkins Configure System
    d. Jenkins Project Configuration
    e. Jenkins Build History
    f. Application execution copy

General Instructions

Gitlab repository should have the following artefacts

- Structured code base & Build process configuration file (pom.xml)
- Snapshots listed under project tasks (Step 5 as mentioned under Project tasks)
- Jenkins triggered email upon successful completion of below listed phases to avinash.patel@wipro.com and raghavendran.sethumadhavan1@wipro.com (An email should be triggered to yourself, upon failure at any phase)

NOTE: While sending email, please follow below convention

✓ Subject: <Your ADID>/DevOpsProfessional/ ILP_CI-CD/<projectTitle>
    - Please ensure email triggered from your email id only
    - For all failures emails should be send back to you only
    - After deliverables are pushed to Gitlab, do post your gitlab URL on suggested MS Teams link

Integrated Learning Project Objective & Deliverables

1. Continuous Integration (CI)

Objective:

- Construct project structure as per recommendation of build processing tool
- Create application source codes (Business Logic, Unit Test code, other scripts /automation scripts –if required)
- Integrate necessary tools with CI server to perform Continuous integration
    - SCM : Gitlab
    - Build Tool : Maven
    - Unit Testing : Junit
    - Artifactory : Jfrog
    - CI Server : Jenkins
- Automate CI process to produce artifact (war/jar) and notify with suitable response

Deliverables:

a. Project Structure with sources
    i. Application Source Code

ii. Application Unit Testing Source Code
iii. UI & web configurations sources
iv. Any automation scripts /other scripts (if applicable)
v. Read Me (readme.txt) having configuration and usage details

b. Snapshots
   i. Gitlab
      1. Project structure
      2. History (commit details, tags, branches etc.,)
   ii. Jenkins
      1. Project Configuration (SCM, Build Trigger, Steps, Build and Post build actions)
      2. Build History
      3. Other Configuration (Artifactory, SonarQube and other suitable setup/Configuration)
   iii. Artifacts details from Jfrog Artifactory
   iv. Auto triggered mail (as mentioned above)

2. Continuous Delivery (CI-CD)

Objective:

- Construct project structure as per recommendation of build processing tool
- Create application source codes (Business Logic, Unit Test code, other scripts /automation scripts –if required)
- Integrate necessary tools with CI server to perform Continuous integration and Continuous Delivery
  - SCM : Gitlab
  - Build Tool : Maven
  - Unit Testing : Junit
  - Artifactory : Jfrog
  - Code Review : SonarQube
  - Testing : Functional test using Selenium
  - Test Environment : server-Apache Tomcat, browser - Chrome/Firefox/chromium
  - CI Server : Jenkins
- Automate CI process to produce artifact (war/jar) and notify with suitable response

a. Snapshots
   i. Jenkins Configuration – Artifactory, SonarQube and WebServer
   ii. Jenkins Project Configuration & Build History
   iii. Artifacts details from Jfrog Artifactory
   iv. Code quality report from SonarQube
   v. WebServer UI Snapshot
   vi. Application UI responses