PROJECT SPECIFICATION

Ensuring Quality Releases

**Environment Creation & Deployment**

**CRITERIA**

**MEETS SPECIFICATIONS**

Utilize Terraform to apply Infrastructure as Code (IaC) concepts to create different environment tiers that enable effective testing of code as it reaches the different phases of the release management process.

Students will submit Terraform files for the test environment they create, as well as a screenshot of the log output of Terraform when executed by the CI/CD pipeline (ensure the timestamp is visible by toggle timestamps for the specific job). This will show an understanding of the relationships between the configuration of Terraform files, the creation of a test environment and the execution of Terraform by the CI/CD pipeline.

Add automated testing tasks during deployment of an application to a CI/CD pipeline in Azure DevOps to improve feature quality by decreasing % of failed tests.

Students will submit the azure-pipelines.yaml file which will show each task, as well as a screenshot of the successful execution of the pipeline build results page (this path will be in the URL of the correct page: /\_build/results?buildId={id}&view=results). This will not show errors and the time stamp for this ought to correspond closely to the timestamps in the screenshots that are submitted.

**Automated Testing**

**CRITERIA**

**MEETS SPECIFICATIONS**

Design and run a load test suite post-deployment using JMeter to evaluate both a web application and web service’s performance (how well application performs under stress) in order to provide resource requirements, benchmarking metrics, and meet SLA (service level agreement) objectives.

Students will submit the load test suite, and test results for each of the types of performance tests should be provided. The test results will include an HTML-generated report of each test run and not show errors for more than 10% of the test cases. A screenshot of the log output of JMeter when executed by the CI/CD pipeline (ensure the timestamp is visible by toggle timestamps for the specific job) should contain the lines that start with “summary” and “Starting standalone test @”.

Design and run functional test suites post-application-deployment using Selenium to evaluate the quality of a web application in order to identify, reduce, and prevent defects before a production release.

The Python test case files will show the proper interaction of the browser with the demo site. This will include logging in, adding 6 different items to a cart, and removing those items from the cart. The results of the test will show which user logged in, which items were added to the cart, and which items were removed from the cart, and will consist of a screenshot of the execution of the test suite by the CI/CD pipeline.

Design and run API-integration tests post-application-deployment using Postman to evaluate the quality of a web service in order to reduce defects before a production release.

The Data Validation and Regression Postman collections will include test cases that exercise each REST API endpoint of the demo site and validate a successful response (regression) along with expected values in the response (data validation). Three screenshots of the Test Run Results from Postman shown in Azure DevOps. One should be the Run Summary page (which contains 4 graphs), one should be of the Test Results page (which contains the test case titles from each test) and one should be of the output of the Publish Test Results step.

**Monitoring & Observability**

**CRITERIA**

**MEETS SPECIFICATIONS**

Configure Azure Monitor to identify and mitigate operational issues by using thoughtfully constructed alerts from a mixture of virtual machines and application services.

The student will include screenshots of the email received when the alert is triggered, the graphs of the resource that the alert was triggered for (be sure to include timestamps for the email and the graphs), and the alert rule, which will show the resource, condition, action group, alert name, and severity. Screenshots for the resource’s metrics will correspond to the approximate time that the alert was triggered.

Configure Azure Log Analytics to consume and aggregate custom application events in order to determine and address root causes of operational issues.

Students will include screenshots of log analytics queries and result sets which will show specific output of the Azure resource. The result set will include the output of the execution of the Selenium Test Suite (be sure to include timestamps).

**Suggestions to Make Your Project Stand Out!**

Cause errors or other scenarios for the AppService/VM and demonstrate those behaviors in the test suites as well as in Azure Monitor and Log Analytics.

Create a VM Scale Set in Terraform and complete each of the steps with the VM Scale set. Bonus: Include evidence of Autoscaling.