Web services

Title: Assignment 1

Weight: 15%

Description

Over the last number of weeks, we had a chance in our labs to create basic web services. The main issue we face is when we want to provide a solution to our customers, the entire process is very manual. In this assignment, we will create a REST-based API and use a complete DevOps lifecycle for the delivery and testing processes.

Part 1 - REST API

Create a REST-based API using Flask-Restful. The API must have the following endpoints available:

/getProducts	A complete JSON list of products is returned to the user and is stored in a local MongoDB database.
/getTitles	Return a list of product titles only. This API endpoint must communicate with a GraphQL server to retrieve the information from the MongoDB database that contains the products table. A correct schema should be added in GraphQL to handle this.
/insertProduct	The user should be able to call this API endpoint using Postman and send across a product id, product title and product cost that will then be stored in the MongoDB database. A custom API key should be passed to the URL also to ensure the user is authenticated before they can send data. If the user has not passed an API key, an error message should be returned to alert

	them.
/	The root page should provide a list of the API URLs that are available and a brief description of how they work e.g., what data is sent to or received from the API endpoint.

A list of records for your local MongoDB database is available on the Moodle page. This should be inserted into your local MongoDB database.

Part 2: DevOps

Create a Jenkins workflow that will

1. Pull the latest code from GitHub when a commit has been made.
2. After the code has been pulled down, the GraphQL server, REST API, and MongoDB database will need to be triggered to turn on as part of the building and testing process.
3. Unit tests should be added to test that each of the application API URLs is working correctly and returning sample data as expected.
4. At the end of the process, documentation outlining what the results of the unit tests were should be copied into a folder titled **test results** and placed in a final zip file that will be generated with the current date and time.
5. Each of the servers that were started e.g., API, GraphQL, and MongoDB should all be stopped.
6. A message should be printed to Jenkins outlining “Process completed and ready for the customer”.

Deliverable

Upload your code in a PDF document with screenshots of the outputs and a description of each to Moodle.

Remember to add your name and student number to the document.