Web Application Development

Create a web application using a technology stack that meets Cloud-Native Web Application Requirements. Start implementing APIs for the web application. Features of the web application will be split among various applications. For this assignment, we will focus on the backend API (no UI) service. Additional features of the web application will be implemented in future assignments. We will also build the infrastructure on the cloud to host the application. This assignment will focus on the user management aspect of the application. You will implement RESTful APIs based on user stories you will find below.

RESTful API Endpoints To Be Implemented

About the field data types in swagger docs:

1. If a field has readOnly set to true, the value in this field is populated by your application. Example fields are timestamp and id. readOnly properties are included in responses but not in requests.
2. If a field has writeOnly set to true, the value in this field is provided by the API caller in POST or PUT request but these fields are not part of the GET request. An example is the password field. writeOnly properties may be sent in requests but not in responses.
3. The keyword multipleOf is used to specify that a number must be the multiple of another number.
4. The minimum and maximum keywords are used to specify the range of possible values.

API Documentation

<https://app.swaggerhub.com/apis-docs/spring2022-csye6225/app/a01 )>

User Stories

1. All API request/response payloads should be in JSON.
2. No UI should be implemented for the application.
3. As a user, I expect all API calls to return with a proper [HTTP status code](https://en.wikipedia.org/wiki/List_of_HTTP_status_codes).
4. As a user, I expect the code quality of the application is maintained to the highest standards using the unit and/or integration tests.

Create & Setup GitHub Repository

1. [Create](https://help.github.com/articles/create-a-repo/) a new private GitHub repository for web applications in the GitHub organization you created.
2. GitHub repository name must be webservice.
3. Grant TAs access to your GitHub repository.
4. Update README.md in your repository. Your readme file must contain the following:
   1. Prerequisites for building and deploying your application locally.
   2. Build and Deploy instructions for the web application.
5. Fork the GitHub repository in your namespace. You will do all development work on your fork.
6. All web application code should now be in this repository.
7. Add appropriate .gitignore to your repository. A collection of useful .gitignore templates can be found

Implement Continuous Integration  with GitHub Actions

1. Add a GitHub actions workflow to run the application unit tests for each pull request raised. A pull request can only be merged if the workflow executes successfully.
2. Implement [GitHub branch protection](https://docs.github.com/en/repositories/configuring-branches-and-merges-in-your-repository/defining-the-mergeability-of-pull-requests/about-protected-branches)to prevent users from merging PR when GitHub Actions workflow run fails.

Documentation

Web App

https://en.wikipedia.org/wiki/List\_of\_HTTP\_status\_codes

Submission

The assignment will be considered late if commits are made to the **main** branch after the due date.

1. All work for this assignment must be done on the feature branch in your fork and merged to main when you are dev complete.
2. The feature and main branches must be in sync.
3. Submit your code from all repositories in this assignment. **Read the instructions carefully to create your zip file.**
   1. Create a folder with the naming convention **firstname\_lastname\_neuid**
   2. In the **firstname\_lastname\_neuid**clone all of your GitHub (organization) repositories with the **git clone**command. It is important that you clone the repos so that your commit history and branches are preserved.
   3. Once you have cloned all of your repositories, you will create a zip of the **firstname\_lastname\_neuid\_a\_##** directory. The zip file should be **firstname\_lastname\_neuid\_a\_##.zip** where **##** is the assignment number.
   4. Now unzip the zip file in some other directory and confirm the content of the zip files.
   5. Upload the Zip to this assignment.
4. You are allowed to resubmit. If you think there may be an issue with the ZIP file, feel free to submit it again. Only the latest will be used grading.

Grading Guidelines

The following guidelines are for information only. They are subject to change at the discretion of the instructor and TA.

Web Application Crash (10% Penalty)

* No 500 internal server errors.
* No restart of application server between API calls.

Web Application (30%)

* Students to demo the web application from their laptops.
* APIs can be demoed using any Postman or Restlet or some other REST client but not via browser.
* Check for UI. The application cannot have UI.

Git (30%)

* All students must use the Github forking workflow and their repositories (main branch which must include code for this assignment) must be in-sync. Check this by asking students to create pull requests between their main branch and their assignment branch. There should be no changes. Verify that all assignment changes are in main branch.
* Verify that students have added TAs and instructors as collaborators to the GitHub repository.
* Verify that students have README.md file in their git repository and it contains instructions on how to build, test and deploy their application including any pre-requisites for a programming language, frameworks, and third-party libraries.
* Verify that the dev environment is not set up in Downloads folder.
* Git repositories should be cloned locally using git/ssh protocol and not https.
  + Verify this by running git remote -v the command in the cloned repository in the VM.
* Validate that students have created a fork of the organization repository are working on it.
* Verify that the student has made no direct commits to their organization repository.

Git Repository Content Check (10%)

* Check the repository for any IDE-specific files. IDE configuration files must not be in the repository.
  + Verify their .gitignore configuration.
* Check the repository for build artifacts such as .class, .jar, .war files and build, node\_modules directory. None of these should be checked into the repository.
* Check for dependencies. Dependencies from the Maven repository or npm should not be committed to the git repository.

Continuous Integration (30%)

* GitHub Action workflow is triggered when PR is opened.
* GitHub Repository has branch protection configured to prevent PRs from merging when a workflow fails.