Faculty of Science

Course: CSCI 3240 Adv Web Application Development

Lab: 8 CI/CD

Submission Deadline: Check the course portal or Canvas for the exact deadline.

Lab Manual: Implementing CI/CD for an Online Quiz Platform

Overview:

In this lab, you will add Continuous Integration and Continuous Deployment (CI/CD) capabilities to the Online Quiz Platform you built in the previous lab. This platform already has three main features:

1. Creating a Quiz: Users specify a title and list of questions, each with its correct answer.
2. Retrieving Quiz Details: Users retrieve quiz details by ID to view the title and questions.
3. Submitting Answers: Users submit answers for evaluation and receive a score.

The focus of this lab is on setting up GitHub Actions for CI/CD. You will configure automated workflows to handle testing, code quality checks, dependency management, and security scanning.

Please note that any helpful steps along with code here are for your guidance and not be considered fixed approach and only approach. The steps mention here should work in best cases. In case you find the need to implement additional steps to complete the lab , please do so.

Learning Objectives:

By the end of this lab, you will be able to:

- Use GitHub Actions to automate testing and code quality checks for your Flask API.
- Set up automated dependency management with Dependabot.
- Configure security scanning using CodeQL to detect vulnerabilities

Prerequisites

1. Existing Quiz Platform: Use the quiz platform you completed in the previous lab. If necessary, ensure it contains the files `quiz_controller.py`, `quiz_model.py`, `quiz_service.py`, and `test_quiz.py`.
2. GitHub Repository: The project should already be under version control. If it isn't, follow these steps:
   - Initialize a Git repository by running `git init` in your project folder.
   - Commit the code and push it to a new GitHub repository:

```bash
git add .
git commit -m "Initial commit with completed code"
git remote add origin <repository_url>
git push -u origin main
```

Part 1: Setting Up GitHub Actions for CI/CD

Step 1: Add Testing Workflow

1. **Create a Workflow File for Testing:**

- In your project, create a directory `.github/workflows` if it doesn't already exist.
- Inside `.github/workflows`, create a file named `test.yml`.
- Add the following configuration to `test.yml`:

```yaml
name: CI/CD Pipeline - Testing
on: [push, pull_request]
jobs:
  test:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout code
        uses: actions/checkout@v4
      - name: Set up Python
        uses: actions/setup-python@v5
```

```
          with:
            python-version: '3.8'
        - name: Install dependencies
          run: pip install -r requirements.txt
        - name: Run tests
          run: pytest
```

Proof to Submit:

- Screenshot of the `test.yml` file content.
- Screenshot of the Actions tab showing a successful test workflow run.

Part 2: Add Linting Workflow with Flake8

Step 1: Install and Configure Flake8

1. **Install Flake8 in the Virtual Environment:**

```bash
pip install flake8
```

2. **Add Linting Workflow:**

- In `.github/workflows`, create a file named `lint.yml`.
- Add the following configuration:

```yaml
name: Python Linting
on: [push, pull_request]
jobs:
  lint:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout code
        uses: actions/checkout@v4
```

```
      - name: Set up Python
        uses: actions/setup-python@v5
        with:
          python-version: '3.8'
      - name: Install Flake8
        run: pip install flake8
      - name: Run Flake8
        run: flake8 quiz_controller.py quiz_model.py quiz_service.py
test_quiz.py
```

Proof to Submit:

- Screenshot of the `lint.yml` file content.
- Screenshot of Actions tab showing a successful linting workflow run.

Part 3: Configure Dependabot for Dependency Management

1. **Set Up Dependabot:**

- In `.github`, create a file named `dependabot.yml`.
- Add the following configuration to set up automatic weekly checks:

```yaml
version: 2
updates:
  - package-ecosystem: "pip"
    directory: "/"
    schedule:
      interval: "weekly"
```

2. **Commit and push the file to GitHub:**

```bash
git add .github/dependabot.yml
git commit -m "Add Dependabot configuration for dependency management"
```

```
git push
```
```

**Proof to Submit:**

- Screenshot of `dependabot.yml` file content.
- Screenshot of the Dependabot section in GitHub's Insights tab

### Part 4: Add Code Coverage Reporting with Codecov

Step 1: Install `pytest-cov`

1. **Install `pytest-cov`:**

```bash
pip install pytest-cov
pip freeze > requirements.txt
```

2. **Run Tests with Coverage Locally:**

```bash
pytest --cov=app test_quiz.py
```

Proof to Submit:

- Screenshot of terminal showing test coverage results.

Step 2: Set Up Codecov Integration

1. Create a Codecov Account: Sign in to [Codecov](https://codecov.io/) and authorize access to your GitHub repository.
2. Get Codecov Token: Copy the Codecov token for your repository.
3. Add Codecov Token as a GitHub Secret:

- Go to Settings > Secrets and variables > Actions and add a new secret with the name `CODECOV_TOKEN`.

4. Modify `test.yml` to Upload Coverage Results:

- Update `test.yml` to include the Codecov step:

```yaml
```
```

```
    - name: Upload coverage to Codecov
      uses: codecov/codecov-action@v2
      with:
        token: ${{ secrets.CODECOV_TOKEN }}
```

Proof to Submit:

- Screenshot of the updated `test.yml`.
- Screenshot of Actions tab showing successful coverage report upload.
- Screenshot of Codecov dashboard showing coverage results.

Part 5: Set Up CodeQL for Security Scanning

Step 1: Create CodeQL Workflow

1. In `.github/workflows`, create a file named `codeql-analysis.yml`.
2. Add the following configuration:

```yaml
name: CodeQL Analysis
on:
  push:
    branches: [main]
  pull_request:
    branches: [main]
jobs:
  analyze:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout code
        uses: actions/checkout@v4
      - name: Initialize CodeQL
        uses: github/codeql-action/init@v1
        with:
```

```
          languages: python

      - name: Perform CodeQL Analysis

        uses: github/codeql-action/analyze@v1
```

Proof to Submit:

- Screenshot of `codeql-analysis.yml` file content.
- Screenshot of Actions tab showing successful CodeQL analysis.

Deliverables

1. Project Folder:

- Submit the complete project folder containing the `src` and `tests` directories, along with the `.github` directory for workflows.
- Ensure the folder structure matches the original provided structure.
- Delete any `__pycache__` folders or other unnecessary files to keep the submission clean.

2. Workflow and Test Screenshots:

- Provide screenshots as proof of each workflow completion. These screenshots should include:
- Testing Workflow:
- Screenshot of the Actions tab in GitHub showing a successful test workflow run.
- Screenshot of the `test.yml` file content.

- Linting Workflow:

- Screenshot of the Actions tab showing a successful linting workflow run.
- Screenshot of the `lint.yml` file content.

- Dependabot Configuration:

- Screenshot of the Dependabot setup under the Insights tab in GitHub.
- Screenshot of the `dependabot.yml` file content.

- Codecov Integration:

- Screenshot of the Actions tab showing successful Codecov coverage report upload.
- Screenshot of the Codecov dashboard displaying coverage results.
- Screenshot of the `test.yml` file content showing the Codecov integration.

- CodeQL Analysis:

- Screenshot of the Actions tab showing a successful CodeQL analysis run.
- Screenshot of the `codeql-analysis.yml` file content.

3. Submission Format:

- Compress the complete project folder (including `.github/workflows` directory) into a zip file named `your_name_quiz_platform_CI_CD_lab.zip`.
- Include all required screenshots in the zip file.

4. Submission:

- Upload the zip file to the course portal (e.g., Canvas) by the specified deadline.