LAB REPORT : CONTINUOUS INTEGRATION

COURSE CODE: CSE 404
COURSE TITLE: SOFTWARE ENGINEERING AND ISD LABRATORY

Submitted by

Suraiya Mahmuda (364)

Submitted to

Dr. Md. Mushfique Anwar, Professor

Dr. Md. Humayun Kabir, Professor

Computer Science and Engineering

Jahangirnagar University

Dhaka, Bangladesh

January 07, 2025

Contents

1. Introduction

Continuous Integration (CI) is a core agile practice that emphasizes the automated and continuous testing of software. It encourages frequent code integration, resulting in regular builds and prompt feedback on any errors. In this lab report, I will outline my experiences and contributions to integrating CI into our project. Using CircleCI, a leading CI tool, our team focused on improving software quality through automated build pipelines and regression testing. This approach enabled quicker issue identification and resolution, ensuring more dependable and thoroughly tested software releases.

2. Why did we select CicrleCI

CircleCI is a versatile CI/CD platform that provides powerful automation features, streamlining integration and deployment workflows. Supporting various programming languages and environments, it is well-suited for a wide range of projects. With its cloud-based infrastructure, CircleCI removes the need for maintaining on-premise servers, saving both time and costs. Its seamless integration with popular version control systems like GitHub and Bitbucket simplifies code tracking and testing. Furthermore, CircleCI offers comprehensive insights and logs,

enabling efficient debugging and enhancing both software quality and developer productivity.

3. Repository Setup for Integration Test

Integration Testing Process

To set up the integration testing environment, we followed a structured approach:

1. Created a dedicated branch named **ci-test-branch** to serve as the primary branch for testing integration.
2. Merged all the relevant feature branches into **ci-test-branch** using the command:

```
git merge <branch-name>
```

3. After merging, **ci-test-branch** was prepared for integration testing.

The following feature branches were merged into **ci-test-branch**:

- **Registration** - Tamjid
- **TrackStudentsAndTeachersAttendanceDuringExam** - Tamjid
- **ApplyForMarksheet** - Tamjid
- **ApplyForCertificate** - Suraiya
- **DisableandSick** - Suraiya
- **ManageAnswerScripts** - Onu
- **ManageExamMaterials** - Nahid
- **PublishResult** - Onu
- **PublishExamCalendar** - Mamun
- **PublishExamSchedule** - Mamun
- **RegisterExam** - Nahid

- **RemunerateTeachers** - Mahfuz
- **ViewResult** - Mahfuz

This structured process ensured that all feature branches were integrated cohesively, facilitating a seamless and comprehensive integration testing phase.

4. Circle CI Setup

CircleCI Setup Process

On the CircleCI website (`https://app.circleci.com`), we began by creating an organization named **JU EXAM OFFICE**. Under this organization, we created a project called **JU Exam Office**.

Next, we connected the project to the respective GitHub repository by selecting the repository and providing the required authorization. Once the authorization was successful, CircleCI gained access to the code in the repository, enabling integration with our CI/CD workflows.

CircleCI then provided a default `config.yml` template. We updated this configuration file according to our project requirements. Afterward, CircleCI automatically committed the updated configuration file to the repository.

In the `config.yml` file, we specified the branch name **ci-test-branch** to ensure that integration testing would be executed from this branch. We also copied the updated configuration file into the **ci-test-branch**, completing the setup process.

CircleCI Configuration (`config.yml`)

```
version: 2.1

jobs:
  setup:
    docker:
      - image: circleci/python:3.8
    steps:
```

```
8       - checkout
9       - run:
10          name: Install dependencies
11          command: |
12            pip install -r 'Sprint/Sprint Implementation/
                Exam_Office_System/myproject/requirements.txt'
13      - persist_to_workspace:
14          root: .
15          paths:
16            - .
17
18  test:
19    docker:
20      - image: circleci/python:3.8
21    steps:
22      - attach_workspace:
23          at: .
24      - run:
25          name: Run pytest for all apps
26          command: |
27            cd 'Sprint/Sprint Implementation/Exam_Office_System/
                myproject'
28            for app in $(ls -d */); do
29              pytest $app/tests
30            done
31
32  docs:
33    docker:
34      - image: circleci/python:3.8
35    steps:
36      - attach_workspace:
37          at: .
38      - run:
39          name: Build Sphinx documentation
40          command: |
41            cd 'Sprint/Sprint Implementation/Exam_Office_System/
                myproject/source'
42            make html
43      - persist_to_workspace:
44          root: .
45          paths:
```

```
            - 'Sprint/Sprint Implementation/Exam_Office_System/
               myproject/source/_build'

workflows:
  version: 2
  build_and_test:
    jobs:
      - setup:
          filters:
            branches:
              only:
                - main
                - ci-test-branch
      - test:
          requires:
            - setup
          filters:
            branches:
              only:
                - main
                - ci-test-branch
      - docs:
          requires:
            - setup
          filters:
            branches:
              only:
                - main
                - ci-test-branch
```

Listing 4.1: CircleCI Configuration File (`config.yml`)

5. Result

The integration test failed due to unresolved unit test failures in Sprint 1. Since the first sprint's features, including the failed tests, were merged, the integration test outcome reflected these

issues. If we had addressed and corrected the unit tests for all features beforehand, we might have achieved a successful integration test result.

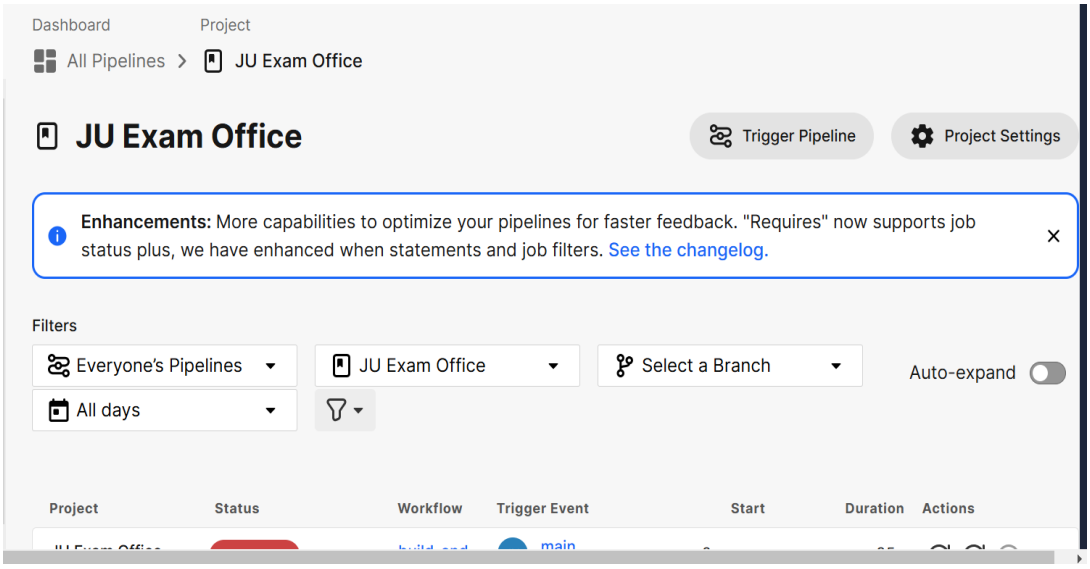

Figure 5.1: Our Project in Circle CI

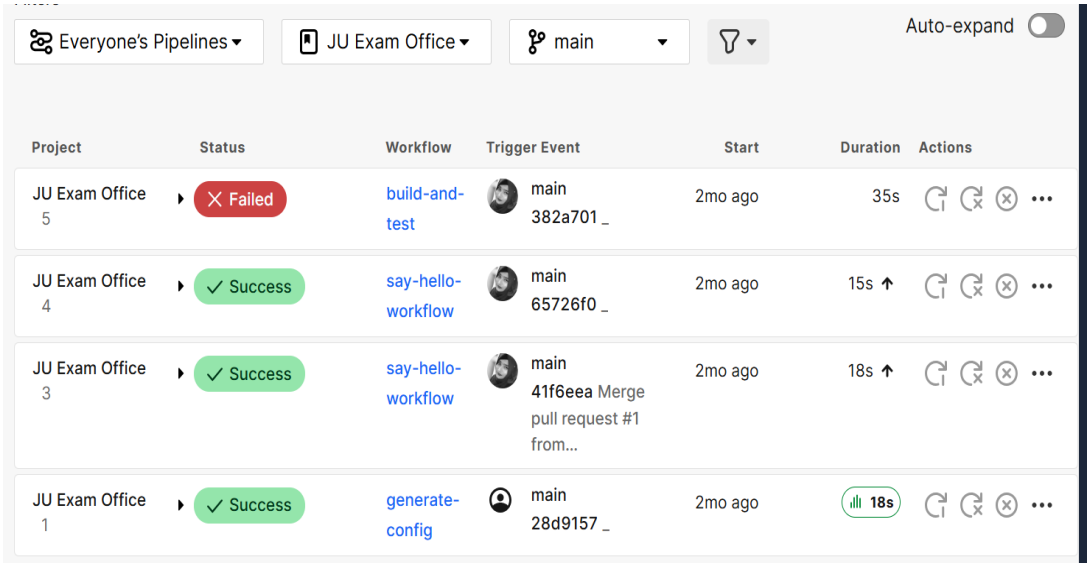

Figure 5.2: Integration Test Result