Graduate Portal: CI/CD

Conducted by:

Team GO

Team Members	*Email*
Alexander Muendesi	u20426918@tuks.co.za
Jason Antalis	u19141859@tuks.co.za
Ying Hao Li	u20460687@tuks.co.za
Chris Magerat	u19029242@tuks.co.za
Priyolan Pillay	u20471582@tuks.co.za
Regan Zhao	u20556455@tuks.co.za
Josh Brink	u19185678@tuks.co.za

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA
Denkleiers • Leading Minds • Dikgopolo tša Dihlalefi

April 16, 2022

Table of Contents

1 Introduction

1.1 Purpose

Team GO was contacted by Mr A Schreuder to create a graduates portal where students and industry can collaborate. It is meant to be a platform that gives graduates and industry a chance to find potential talent with the intent of getting job opportunities.

Mr A Schreuder wanted to fast track the tedious and often repetitive process of selecting and interviewing prospective candidates after an impressive Capstone Project. Mr A Schreuder requested that we create a portal similar to LinkedIn, however specially catered towards graduates.They requested for the important academic information that an institute may require when interviewing potential graduates be displayed on the graduates page. Their main concern was that of authenticity and security and as a result tasked us with certifying the graduates degree on the block-chain.

Mr A Schreuder allowed us creative freedom and suggested that we add fun elements into the graduates portal comprising of but not limited to, a social competent and a social feed.

1.2 Scope

1.2.1 Name

Continuous integration / Continuous deployment

1.2.2 Explanation of CI/CD

Continuous Integration/Continuous Deployment is a method for allowing frequent application delivery via automation during the development of the application. The main aspects of Continuous Integration/Continuous Deployment include delivery and deployment.Team GOś role in this project will be to accomplish the above by introducing automation scripts,as well as maintaining and ensuring the health of the repository by monitoring pull requests, commits and merges. Automation scripts include those for testing code, building the code and releasing validated code to correct branch.

1.3 Definitions, Acronyms and Abbreviations

Table 1: Definitions, Acronyms and Abbreviations

Role	**Name**
CI/CD	Continuous Integration/Continuous Deployment
GitHub	GitHub is a website and cloud-based service that helps developers store and manage their code, as well as track and control changes to their code.
SRS	Software Requirements Specification

1.4 References

References

[1] *What Is GitHub? A Beginner's Introduction to GitHub.* 2021. URL: `https://kinsta.com/knowledgebase/what-is-github`.

[2] *CI/CD Pipeline: What, Why How to Build The Best One | 2022 Updated.* 2022. URL: `https://katalon.com/resources-center/blog/ci-cd-pipeline`.

[3] *What Is CI/CD and How Does It Work? | Synopsys?* URL: `https://www.synopsys.com/glossary/what-is-cicd.html`.

[4] *Continuous Delivery | Get started with CI/CD | Atlassian.* URL: `https://www.atlassian.com/continuous-delivery`.

[5] Dean Leffingwell. *Nonfunctional Requirements - Scaled Agile Framework.* URL: `https://www.scaledagileframework.com/nonfunctional-requirements/`.

[6] *Continuous Integration: A 'Typical' Process | Red Hat Developer.* URL: `https://developers.redhat.com/blog/2017/09/06/continuous-integration-a-typical-process`.

[7] *Setting Issues as High Priority.* URL: `https://help.zenhub.com/support/solutions/articles/43000495285-setting-issues-as-high-priority`.

[8] DataVisBob Laramee. *Software Engineering Principles Lecture 06: Subsystems.* 2017. URL: `https://www.youtube.com/watch?v=3A0gY8f8x2A`.

[9] David C. Kung. *The Principles of Quantum Mechanics.* McGraw-Hill Education, 2014. ISBN: 9780073376257.

1.5 Overview

1.5.1 Contents of the SRS

The SRS will provide a description of the CI/CD. It will provide an outline of the requirements. It will also provide an overview of the characteristics and constraints of the CI/CD.

1.5.2 SRS Organization

The following section, section 2, will provide an overview of the CI/CD and its perspectives, functions, characteristics, constraints, and dependencies. Section 3 will outline in more detail the specific requirements for the software.

2 Overall Description

2.1 Product Perspective

This section introduces Continuous Integration/Continuous Deployment that was selected by our team along with the functional requirements that need to be met in order for successful completion of our feature.

2.1.1 Product Function

Continuous Integration/Continuous Deployment is a method for allowing frequent application delivery via automation during the development of the application. The main aspects of Continuous Integration/Continuous Deployment include delivery and deployment.Team GOś role in this project will be to accomplish the above by introducing automation scripts,as well as maintaining and ensuring the health of the repository by monitoring pull requests, commits and merges. Automation scripts include those for testing code, building the code, linting code and releasing validated code to correct branch.

2.1.2 User Characteristics

The users of the CI/CD system include students and faculty of the University of Pretoria. The users are assumed to have a good understanding of coding and coding related applications, e.g. GitHub.

2.1.3 General Constraints

Due to the fact that the CI/CD system will only be active on the COS301/Graduate Portal GitHub, users will have to be a University of Pretoriastudent who is registered for COS301 and be granted access to the repository by the owners of the repository.

The pipeline may (at any time) break due to erroneous code being merged into the main repository.

3 Specific Requirements

3.1 Business Needs

The following is an overview of business needs which should be implemented:

- Provide a system that enables code to be continually integrated and deployed.
- Allow for users to test their code before merging with current code ready for deployment.

3.2 System Requirements

3.2.1 Functional Requirements

- **FR1.** The implemented system shall notify users if there are any issues during the integration of their changes.
- **FR2.** The system shall be automated so that there is no manual intervention required from the user in terms of building,testing and deploying.
- **FR3.** The system shall automatically integrate code/changes that have successfully passed the checks to the relevant repository.
- **FR4.** Continuous delivery shall ensure that there is always a code base ready for deployment to a production environment.
- **FR5.** Each pull request shall trigger an automatic test and build sequence.
- **FR6.** Each pull request shall trigger a meta tag test to ensure that the correct meta tags are used before merging.
- **FR7.** The system shall have badges in the README.md to display the test coverage and the status of the system.
- **FR8.** The system shall enable manual triggers for the automation scripts to allow for manual testing of the code.

3.2.2 Non-Functional Requirements

- **N-FR1.** The implemented system shall give users an adequate description of any issues experienced through integration to ease the correction process.(Usability)
- **N-FR2.** The Continuous integration/Continuous delivery system shall prevent any faulty code form being merged into the develop branch(Safety requirements)
- **N-FR3.** The Continuous integration/Continuous delivery system shall be available 99 percent of the time to handle pull requests(Availability)

3.3 System Decomposition

3.3.1 Subsystems

.

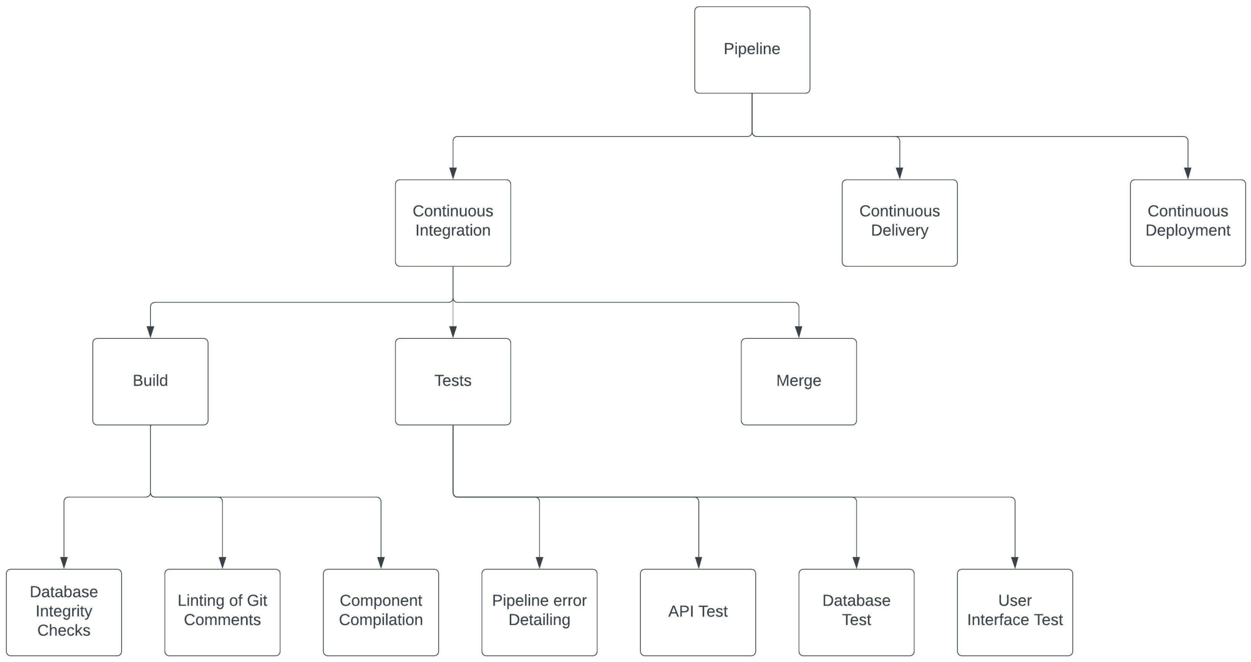

Figure 1: System Functional Block Diagram

3.4 System requirements associated with the subsystems

3.4.1 Component compilation

- The system shall successfully build individually components.

3.4.2 API Tests

- The system shall run available API tests and give detailed feedback on the conducted tests.

3.4.3 Database Tests

- The system shall run available Database tests and give detailed feedback on the conducted tests.

3.4.4 User Interface Tests

- The system shall run available UI tests and give detailed feedback on the conducted tests.

3.4.5 Merge

- The system shall accept pull requests once a successful build has occurred and all tests have been passed.

3.4.6 Continuous Delivery

- The system shall automatically release the validated code to the repository.This should occur when pull requests are accepted.

3.4.7 Continuous Deployment

- The system shall automate the release of the application to production.

3.4.8 Linting of git comments

- The system shall maintain a clean, easy to read and maintainable project history.

3.4.9 Pipeline Error Detailing

- The system shall display any and all errors that have been found or have occurred in processes in the pipeline.

4 Acceptance Criteria

CI/CD is essential for maintaining and monitoring the health of the pipeline. It enables automation to allow for smooth integration and deployment of a system. This feature thus needs to meet the following acceptance criteria to be ready for production:

- The building of the system should be automated for each pull request opened.
- Unit testing and e2e testing should be automated.
- Linting of code should be automated.
- The correct standard for meta tags should be tested.
- The automation scripts should also contain manual triggers for manual testing of code.
- Automation scripts should run in parallel.
- Test the entire code base after each push commit to main branches in the repository.

Appendices

A Tools

Name	Description	Link
GitHub	Version Control Software	`https://github.com/COS-301`
Angular	A platform for building mobile and desktop web applications.	`https://angular.io/`
NGXS	A state management pattern and a library for Angular	`https://www.ngxs.io/`
NestJS	A framework for building efficient, scalable Node.js web applications	`https://nestjs.com/`
Narwhal	Mono-repo management tool	`https://nx.dev/`
Prisma	An evidence-based minimum set of items for reporting in systematic reviews and meta-analyses	`https://www.prisma.io/`
Postgres	An application database	`https://www.google.com/search?client=safari&rls=en&q=postgres&ie=UTF-8&oe=UTF-8`
GraphQL	A graph-based API standard	`https://graphql.org/`
CQRS	Command and Query Request Segregation	`https://docs.nestjs.com/recipes/cqrs`
Docker	A set of platform as a service products that use OS-level virtualization to deliver software in packages called containers	`https://www.docker.com/`
LucidChart	A web-based proprietary platform that allows users to collaborate on drawing, revising and sharing charts and diagrams.	`https://www.lucidchart.com/`

B Diagrams

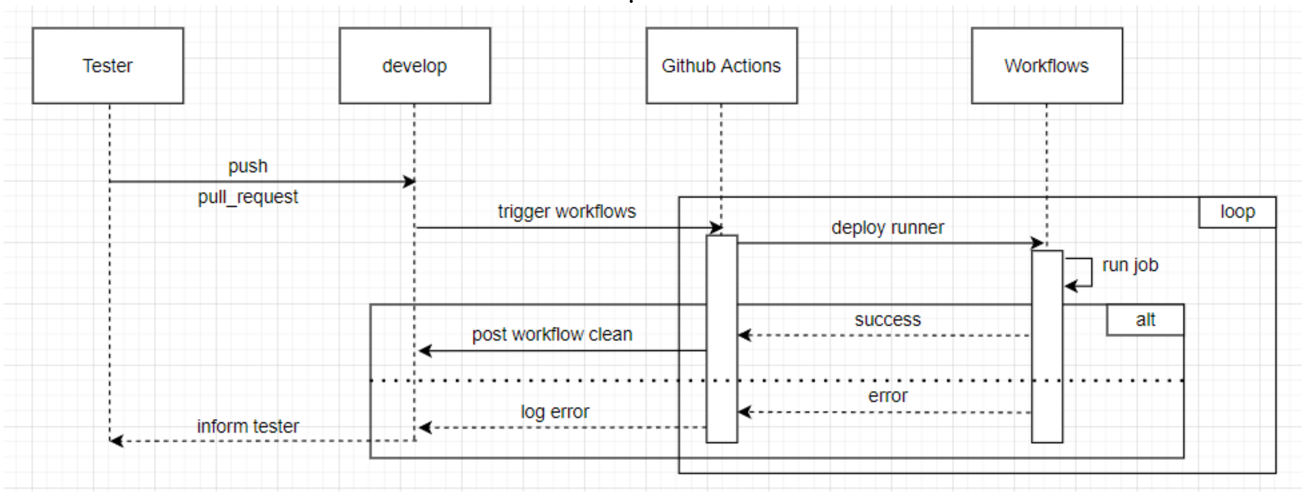

Figure 2: Github workflow Sequence Diagram

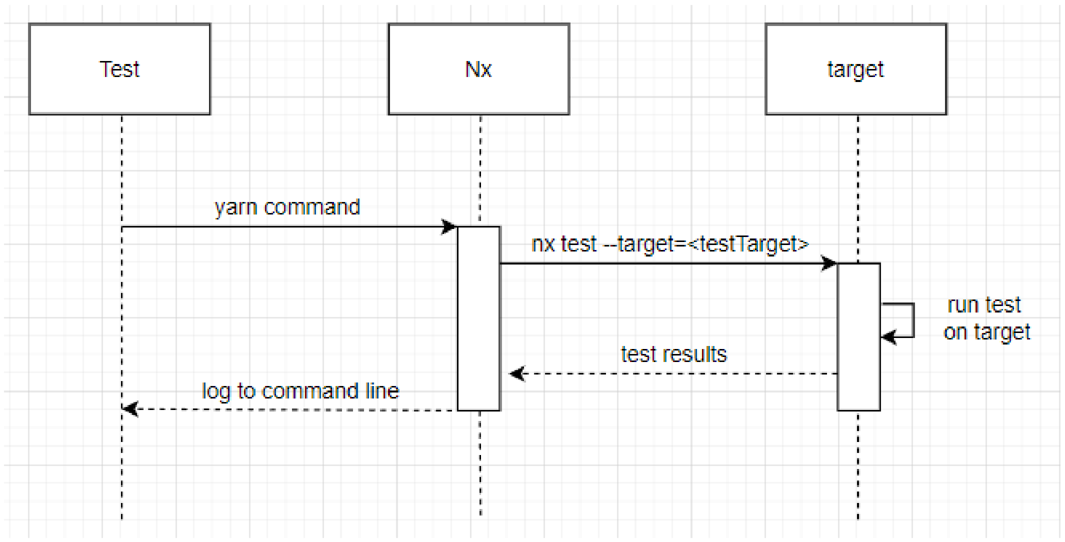

Figure 3: Nx Test Sequence Diagram