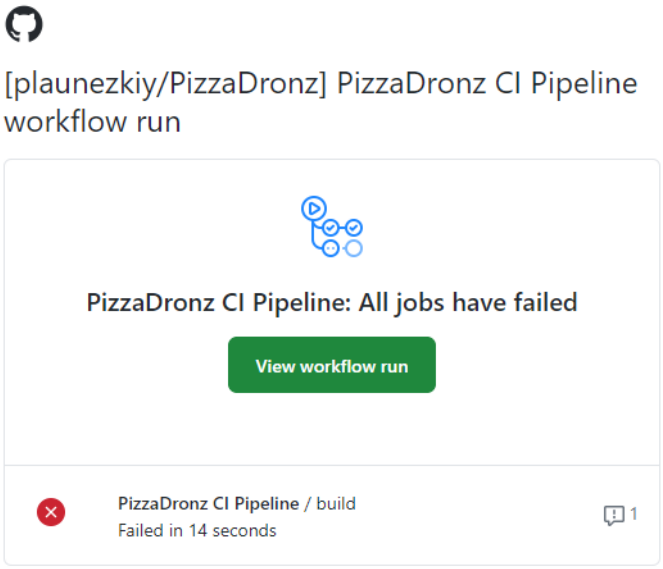
This document describes approaches, methods and techniques that were used and will be implemented in the future iterations of the project to conduct reviews and improve the overall quality of the process. Moreover, a construction of a Continuous-Integration pipeline and its consequences are described as well its simplest implementation is demonstrated later on.

The Inspection:

- Develop guidelines and employ code linters to enforce rules and ensure consistency of the code style.
- Implement a checklist of important attributes to be checked at every major release when new features are deployed. (This can be by thoroughly documenting the development and deployment process).
- Manual Review & Team Consultations with other developers working on the project when a big update is made to cross-examine the updated codebase and identify potential issues.
- CI pipeline conducts remote testing on pushes to the `main` branch and sends an email when the tests fail, this is an important security and quality assuring feature that, when the project grows in complexity and employs more people, will signal immediately about compilation and testing issues.
  Example error notification email:



CI/CD pipeline

The continuous Integration & Continuous Delivery pipelines are a modern approach that combines multiple practices from a variety of subfields surrounding software engineering that are employed to ensure maintainable quality of the codebase and allow to new features seamlessly and eliminate human error when delivering an update. They are also useful to identify issues early-on by automating the regression-testing and inform the developers about failing parts of the application.
The base version of the pipeline shall include 4 key stages:

1. Source

Also referred to as Version Control Stage. Forms the foundation of any pipeline, it involves storage and management of the codebase in a versioned manner. This stage ensures that every version of the product and every change are tracked, can be retrieved and can easily be reverted to in case something goes wrong. This is

automatically managed by Github, which is used for this project and requires the developer to simply label commits and utilise branches when working on experimental features.

2. Build

Building is a critical phase that uses the source code of the application to create a tangible product that can be served and interacted with in a controlled environment. Any dependencies of the project are ensured to be available and compliable in a remote environment. It is also useful to eliminate "it-works-on-machine" issues as the build is done in a remote isolated environment. This will be done using Github's feature called "Workflows" that provides a set of tools that can be combined programmatically to create an environment and specify the compiler version, testing operating system and list the necessary steps to be executed.

3. Test

Vital stage of the process, where all the Integration and System tests can take place to assess the overall performance of the application in a live environment before the delivery. The testing will ensure that the application behaves as expected and should anything be out of the ordinary, a pre-defined sequence of actions will be executed and the developers shall be notified about any discrepancies in the behaviour, such as warning email shown above.

4. Deploy

The final part of the CI/CD pipeline, where the application is released into the deployment environment, such as a live server that the customers can interact with. The deployment can be automated using the tools like Docker/Kubernetes and AWS cloud services that provide a toolkit that enables to link the compiled, tested codebase to the live environment. It is integral to fully automate this stage to eliminate any human error in the process, as at this very stage customers could stop receiving the service should something go wrong and make the application unavailable.

The pipeline is invoked and operated automatically by writing a markup document that specifies a set of activities to be performed at every marked action, such as Push to the Main branch.
Stage 3 (Testing) can be additionally improved in the future to develop End-to-End tests that can simulate user interaction with the graphical interface, such as a frontend or a mobile app to ensure that the user is able to reach the endpoints and performs the activities they are advertised.
For the purposes of the project, the pipeline only implements 3 stages as no live server is implemented at the time of development, so the first three stages are demonstrated below:

```yaml
name: PizzaDronz CI Pipeline

on:
  push:
    branches: [ "main" ]
  pull_request:
    branches: [ "main" ]

jobs:
  build:

    runs-on: ubuntu-latest

    steps:
      - uses: actions/checkout@v3
      - name: Set up JDK 18
        uses: actions/setup-java@v3
        with:
          java-version: '18'
          distribution: 'temurin'
          cache: maven
      - name: Build and Test the Application with Maven
        run: mvn -B package --file pom.xml
```

A successful completion can also be observed below: