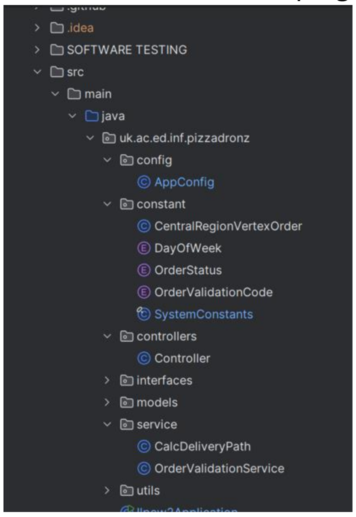
Learning Outcome 5: Conduct Reviews, Inspections, and Design and Implement Automated Testing Processes

5.1 Identify and Apply Review Criteria to Selected Parts of the Code and Identify Issues in the Code

- **Review Criteria:**
  - **Code Quality:** Ensure the code adheres to industry standards for readability, maintainability, and efficiency.
  - **Functional and Non-functional Requirements:** Validate that the code meets all specified requirements, including performance, scalability, and security.
  - **Modularity:** Confirm the code is designed for easy extension and modification, enabling future scalability.
- **Code review:**
  - Conduct peer reviews and inspections for critical components like the Controller, OrderValidationService, and CalcDeliveryPath classes.
  - Compare the outputs of the same tests with other people creating the same system as me to ensure it is correct
  - Intellijs built in error highlighter was extremely helpful in getting the system to run as it gives suggestions on solutions which saved time. This was used throughout the build of the system.
- **Code quality:**
  - Since this was the biggest project I have ever completed it was important to have quality code that was easy to navigate and edit. This was done by having structured classes with identifiable names. The code was also well commented further helping improve the code quality.



```
> .idea
> SOFTWARE TESTING
v src
  v main
    v java
      v uk.ac.ed.inf.pizzadronz
        v config
            AppConfig
        v constant
            CentralRegionVertexOrder
            DayOfWeek
            OrderStatus
            OrderValidationCode
            SystemConstants
        v controllers
            Controller
        > interfaces
        > models
        v service
            CalcDeliveryPath
            OrderValidationService
        > utils
```

        - The main issue in the project was that it was not thoroughly cleaned before the end of the project, this meant that there was a lot of code that was made redundant

5.2 Construct an Appropriate CI Pipeline for the Software

Although I have used CI pipelines before for testing I have never actually created one, so this was a new skill I had to learn

- Integration with GitHub as the source code repository and for **version control**.

  This was useful because I was able to work on certain parts of code and when I made errors I was not able to solve I could just pull the last commit.

- **Automated triggering of tests** for every commit was done using github workflows which was much easier than manually testing large parts of the code.

- In order to deploy this system for grading I had to ensure that I had a running executable using docker. This was the final step to completing the project.

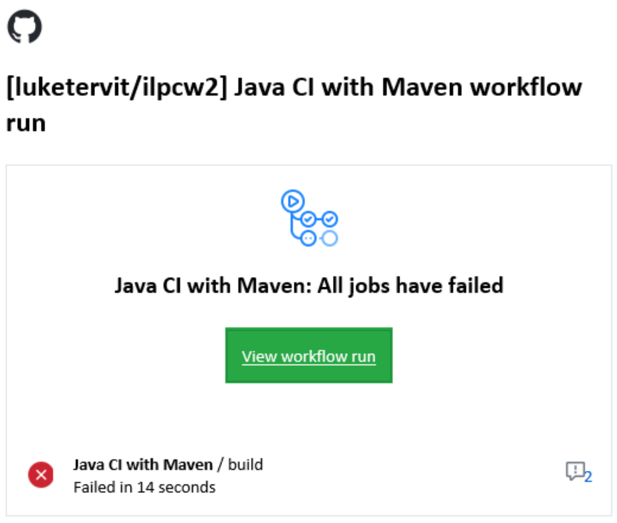
5.3 Automate Some Aspects of the Testing

- **Testing Scope:**
    - **Automated Unit Tests:** Develop comprehensive tests for critical services, such as:
        - validateOrder method to ensure all order validation rules are met.
- **Testing Tools:**
    - Used JUnit for unit testing and Mockito for mocking dependencies.
    - Integrated REST-assured for API testing to validate endpoints like /validateOrder and /calcDeliveryPath.

5.4 Demonstrate the CI Pipeline Functions as Expected

- **Verification Process:**
        - Detects and reports failing tests.

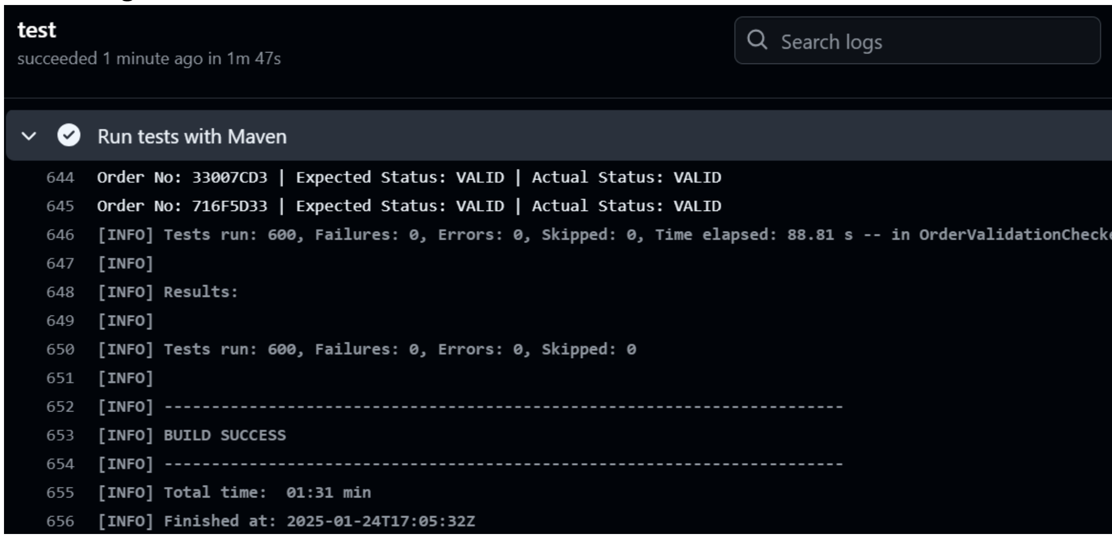
          This was verified by doing Mutation testing for ordervalidation where I purposely made tests fail to ensure that the pipeline would detect them and respond correctly. When the tests failed this was shown in github and also send as an email receipt.


[luketervit/ilpcw2] Java CI with Maven workflow run

Java CI with Maven: All jobs have failed

Java CI with Maven / build
Failed in 14 seconds

- In the results section at the end, all failed tests are counted and displayed. It was then easy to see which tests specifically were passing and failing since the first 300 were valid and the second 300 were invalid.


test
succeeded 1 minute ago in 1m 47s

Run tests with Maven

```
Order No: 33007CD3 | Expected Status: VALID | Actual Status: VALID
Order No: 716F5D33 | Expected Status: VALID | Actual Status: VALID
[INFO] Tests run: 600, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 88.81 s -- in OrderValidationChecke
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 600, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  01:31 min
[INFO] Finished at: 2025-01-24T17:05:32Z
```

All tests passing after pushing to github.

- Successfully deploys builds for passing all tests.