



# TO DO LIST WEB APPLICATION(TD L)

## QA PROJECT

BY JAMES FERNANDES

# INTRODUCTION

## APPROACHING THE PROJECT SPECIFICATION

- Reading through the specification and familiarising with concepts learned during QA academy to be applied during the project
- Aim of the Project is to create a working TDL Web Application with CRUD functionality using Java (Spring API developmen) creating two related entities and integrating the code with a MySQL database to store them. Testing of Code with industry standard coverage
- Creating a JIRA board for planning the project with epics, issues and user-stories
- Creating a Risk Assessment and an ERD to model the database structure
- Producing Base Start Code and start implementing and testing features
- Uploading to GitHub and utilising the Feature-Branch approach

# CONSULTANT JOURNEY

## TECHNOLOGIES LEARNED FOR THIS PROJECT

- JIRA : An online project planning software that utilise AGILE/SCRUM approach using Kanban Board
- GITHUB : An online software for storing, implementing and development of software in repositories using the version control as Git. Implementing the Feature-Branch approach for better workflow on the software.
- MySQL Database : Either an online platform using gcp or a local database that stores data and utilised query's for implementing changes that manipulate the database
- Java Programming Language : Learned the basics of Java programming with intermediate concepts to solve problems within the code and implement features for the project. Using the Spring IDE.
- JUnit Testing : Using Maven build and coverage testing code for unit testing

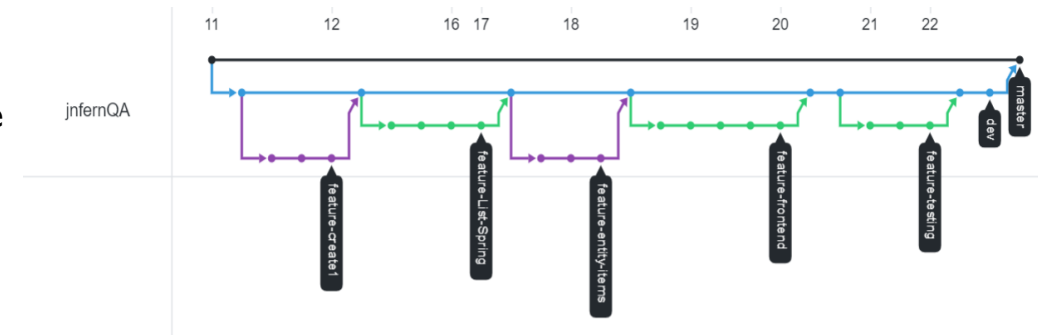
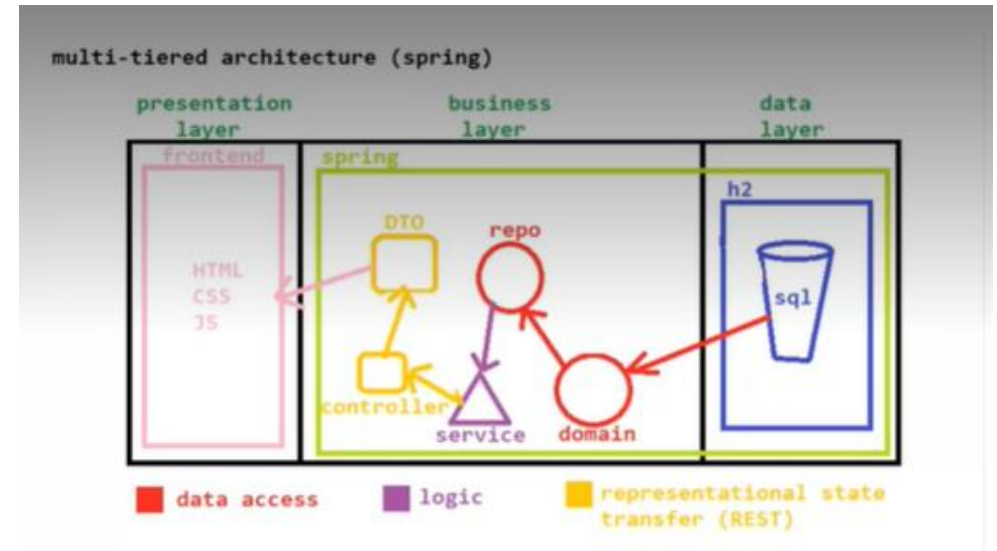
# CONSULTANT JOURNEY

- TECHNOLOGIES LEARNED FOR THIS PROJECT
- Spring : It's the Java IDE for the BackEnd development for the API of the web application
- HTML, CSS and JavaScript : These technologies used for front End development creating the web application design and functionalities
- Mockito: Used for Integration testing groups of classes and test the functionalities between classes whether they are integrated.
- Selenium: Used for Web Application testing or (User Acceptance testing) tests if the CRUD functionalities work as intended in the front end.

# CONTINUOUS INTEGRATION

## HOW THE VERSION CONTROL WAS APPROACHED

- The Version Control utilised was Git using the Gitbash program to initialise the repository locally that was forked and cloned from a Starter Repo on Github.
- Implementing Changes using the Spring IDE creating a MAVEN Build, Back End API and Front-End Web page using (HTML, CSS and Javascript)
- Creating multiple branches and represented features to be added to the master branch, each feature to be added was Lists, Items, frontEnd dev and Testing. Pushing from my local branch to my online repository and merging when a feature was complete to the dev branch.
- After Completion Merging the dev branch to the master branch when the program was working



# TESTING

## WHAT WAS TESTED?


- The Coverage testing was done using JUnit and Mockito that ran a coverage unit and integration test on src/main/java:
- This was done by creating test classes in the source test folder to run the tests from and to compare expected to actual outcomes of the coverage tests.
- Static Analysis Test was done using SonarQube
- User Acceptance testing was done using selenium in the Spring IDE

## Results From the Coverage Test:

| Element                   | Coverage |
|---------------------------|----------|
| ▼ QA_TDL_Project          | 83.3 %   |
| ▼ src/main/java           | 67.5 %   |
| > com.qa.main.exceptions  | 0.0 %    |
| > com.qa.main             | 37.5 %   |
| > com.qa.main.config      | 100.0 %  |
| > com.qa.main.utils       | 94.2 %   |
| > com.qa.main.controller  | 100.0 %  |
| > com.qa.main.service     | 97.6 %   |
| > com.qa.main.persistence | 40.5 %   |
| > com.qa.main.dto         | 72.4 %   |






# TESTING

## SonarQube Static Analysis Results

 [QA\\_TDL\\_Project](#)

Passed

Last analysis: 24 minutes ago

|  |   |   |   |          |              |   |
|--|---|---|---|----------|--------------|---|
|  Bugs |  Vulnerabilities |  Hotspots Reviewed |  Code Smells | Coverage | Duplications | Lines   |
| A  | D   | E   | A   | 0.0%     | 16.7%        | 710  Java, XML |

# DEMONSTRATION



# SPRINT REVIEW

## WHAT WAS COMPLETED?

- Almost all the scope was completed with a working TDL Web program that managed a database on MySQL and development on h2 memory database with all the functionality of CRUD.
- JUnit Test was completed, Integration Test (Mockito) only create, read and delete functions were able to be tested

## WHAT GOT LEFT BEHIND?

- Integration Test with Update and providing an extent report for the selenium testing
- Front End fixes and improvements, where read list does not show items even though it shows up in console

# SPRINT RETROSPECTIVE

## WHAT WENT WELL?

- Getting the main coded program on Spring (BackEnd) working and its workflow through Github

## WHAT COULD BE IMPROVED?

- Testing could be improved as shown by the low-test coverage percent
- Front End functionality and improvements
- Ordering my front-end code more logically
- Planning for the project more efficiently and addressing problems early as possible as this caused delays in the project and incompleteness as well

# CONCLUSIONS

## REFLECTIONS ON THE PROJECT

- Learned a lot of concepts during this project and motivated its use in many fields and industry for different projects

## FUTURE STEPS

- Reinforcing Concepts learned more often through practice and addressing problems, worries, etc. much earlier to avoid delays in future projects

# QUESTIONS