

# IMS Project Presentation



By Jonathan Rabaja

# Introduction

## How I Approached The Project:

- Broke the project into components
- Asked for help when needed
- Researched topics I was unfamiliar in
- Used all the tools that were available to me

# Consultant Journey

## Setback

Due to issues with equipment, missed a lot of training.

## Technologies:

- Java
- Maven
- SQL
- Git
- GitHub
- JUnit Testing

# Version Control

## Git

Using git commands to add files, commit them and create new feature branches

```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Version 10.0.19042.1288]
(c) Microsoft Corporation. All rights reserved.

C:\Users\jraba>cd documents/github

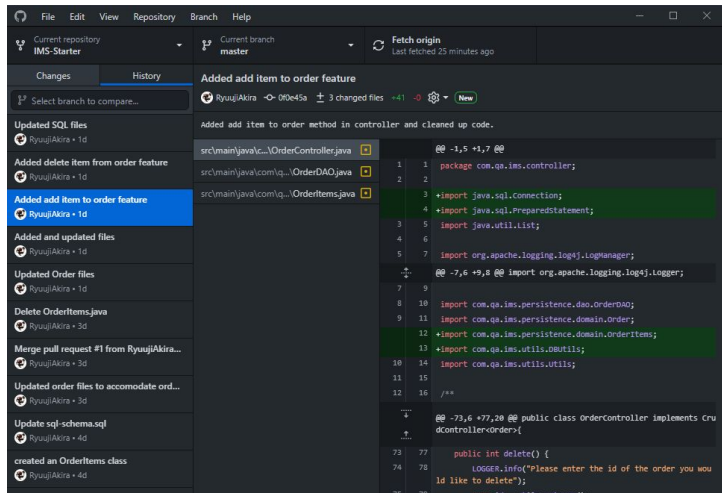
C:\Users\jraba\Documents\Github>cd IMS-Starter

C:\Users\jraba\Documents\Github\IMS-Starter>git branch
  dev
* master
  order-features
  order-features-2
  order-features-fix
  test-items
  test-order

C:\Users\jraba\Documents\Github\IMS-Starter>
```

## GitHub

Using GitHub Desktop to track changes and merge branches when code is completed



# Testing

Package Explorer JUnit X

Finished after 3.885 seconds

Runs: 29/29 Errors: 0 Failures: 0

- com.qa.ims.controllers.CustomerControllerTest [Runner: JUnit 4] (1.352 s)
- com.qa.ims.persistence.domain.CustomerTest [Runner: JUnit 4] (0.341 s)
- com.qa.ims.persistence.domain.ItemTest [Runner: JUnit 4] (0.026 s)
- com.qa.ims.persistence.domain.OrderTest [Runner: JUnit 4] (0.015 s)
- com.qa.ims.controllers.ItemControllerTest [Runner: JUnit 4] (0.050 s)
- com.qa.ims.persistence.dao.OrderDAOTest [Runner: JUnit 4] (0.720 s)
- com.qa.ims.persistence.domain.OrderItemsTest [Runner: JUnit 4] (0.027 s)
- com.qa.ims.controllers.OrderControllerTest [Runner: JUnit 4] (0.046 s)
- com.qa.ims.persistence.dao.CustomerDAOTest [Runner: JUnit 4] (0.445 s)
- com.qa.ims.persistence.dao.ItemDAOTest [Runner: JUnit 4] (0.368 s)

IMS-Starter (1 Nov 2021 01:33:10)

Element	Coverage	Covered Instructions	Missed Instructions	Total Instructions
> IMS-Starter	68.2 %	2,478	1,155	3,633

```
86 // @Test
87 // public void testAddItem() {
88 //     final long productID = 1L;
89 //     final long quantity = 2L;
90 //     final long orderID = 1L;
91 //     final OrderItems addedItem = new OrderItems(productID, quantity, orderID);
92 //
93 //     Mockito.when(utils.getLong()).thenReturn(productID, quantity, orderID);
94 //     Mockito.when(this.dao.addItemToOrder(addedItem)).thenReturn(addedItem);
95 //
96 //     assertEquals(addedItem, controller.addItemOrder());
97 //
98 //     Mockito.verify(utils, Mockito.times(3)).getLong();
99 //     Mockito.verify(dao, Mockito.times(1)).addItemToOrder(addedItem);
100 // }
101 //
102 // @Test
103 // public void testDeleteItem() {
104 //     final long ID = 1L;
105 //
106 //     Mockito.when(utils.getLong()).thenReturn(ID);
107 //     Mockito.when(dao.deleteItemFromOrder(ID)).thenReturn(1);
108 //
109 //     assertEquals(1L, this.controller.deleteItemOrder());
110 //
111 //     Mockito.verify(utils, Mockito.times(1)).getLong();
112 //     Mockito.verify(dao, Mockito.times(1)).delete(ID);
113 // }
114 //
115 // @Test
116 // public void testCalculateOrder() {
117 //     final long ID = 1L;
118 //
119 //     Mockito.when(utils.getLong()).thenReturn(ID);
120 //     Mockito.when(dao.delete(ID)).thenReturn(1);
121 //
122 //     assertEquals(1L, this.controller.delete());
123 //
124 //     Mockito.verify(utils, Mockito.times(1)).getLong();
125 //     Mockito.verify(dao, Mockito.times(1)).delete(ID);
126 // }
127 }
```

Coverage 68.1% – Could of covered more if some tests didn't end with failure

# Demonstration

I will now go through my code and run through  
some user stories on JIRA

# Sprint Review

Completed all features from specification and met the MVP

Did not meet 80% coverage in tests

Did not fix some test code

# Sprint Retrospective

## What went well:

Completed all features on time

System can be successfully launched

Backend works with front end of the system

## What could be improved:

Better understanding of tests

More detail to features

Prior learning to some technologies



# Conclusion

- Overall the project went well, completed all features
- For future steps more functionality needed, more testing coverage
- If setback did not occur possibility for better quality project

# Questions?

yes