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.

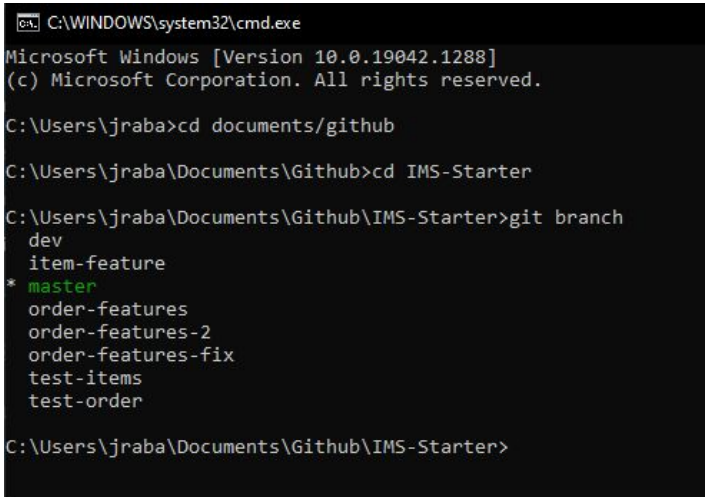
Technologies:

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

Version Control

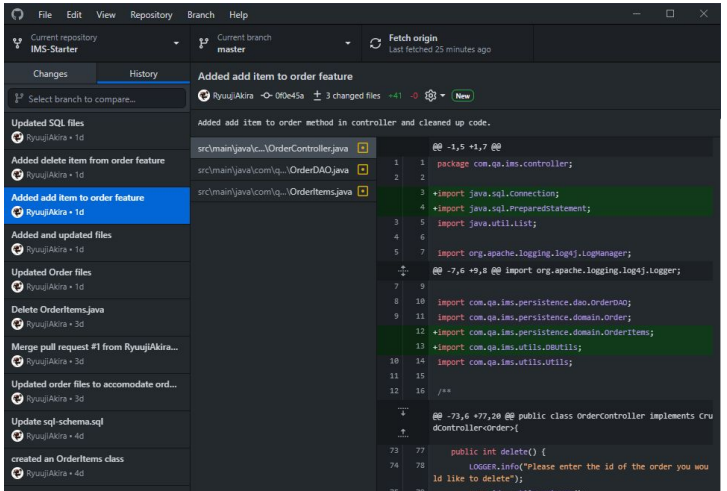
Git

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

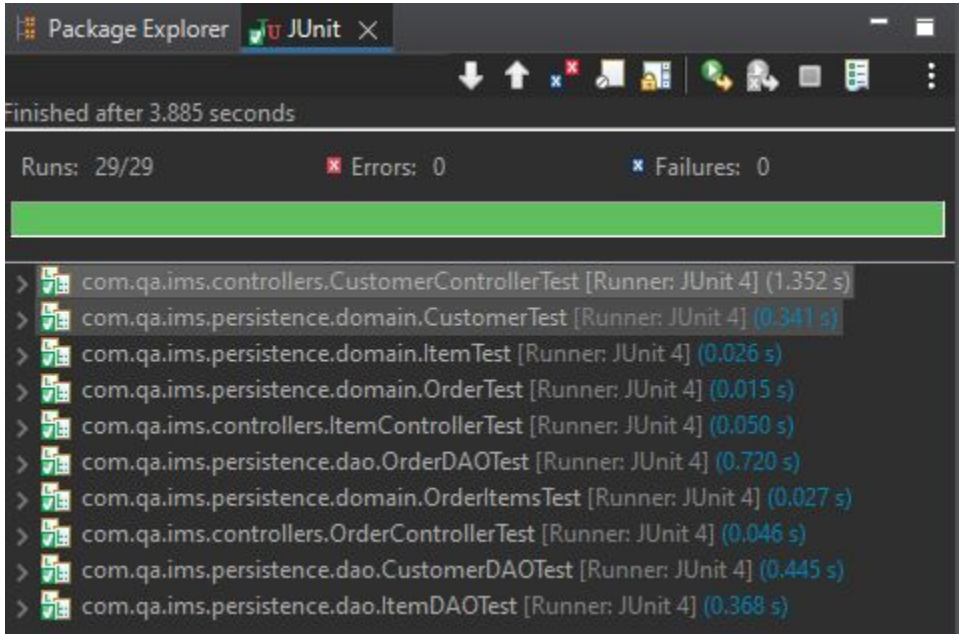


GitHub

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



Testing



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 testCaluclateOrder() {
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