Inventory Management System

October 29th 2021

Fraser Smith

Task Overview

To write a functional program using the fundamental modules learned during training thus far.

We will use:

-Java 11

-mySQL

-Maven

- Jira/Github

Project objective:

To construct an inventory management system that can handle customers items and orders.

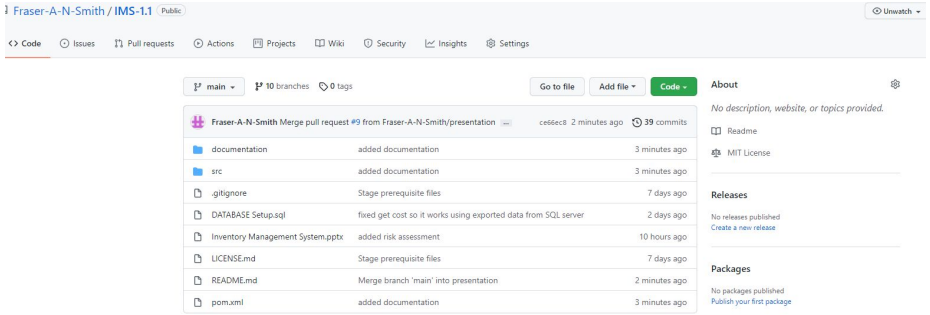
Understanding the problem: Jira Epics

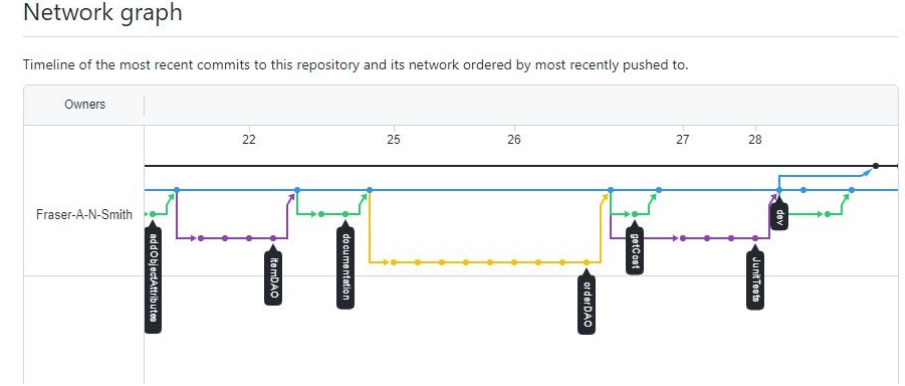
Design	Code	Documentation	Testing
• UML	• Domain	• README	• Junit Testing
• ERD	• DAO	• Risk Assessment	• Mockito
• FBM	• Controllers		

Design:

Version Control:

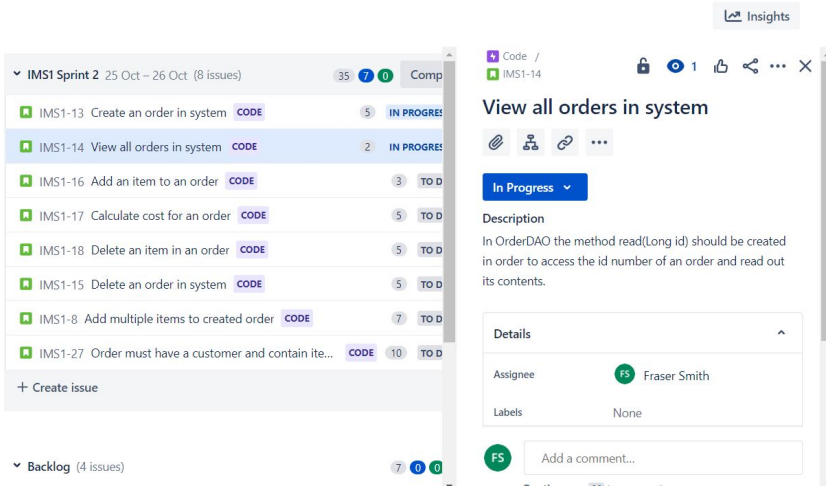
GitHub

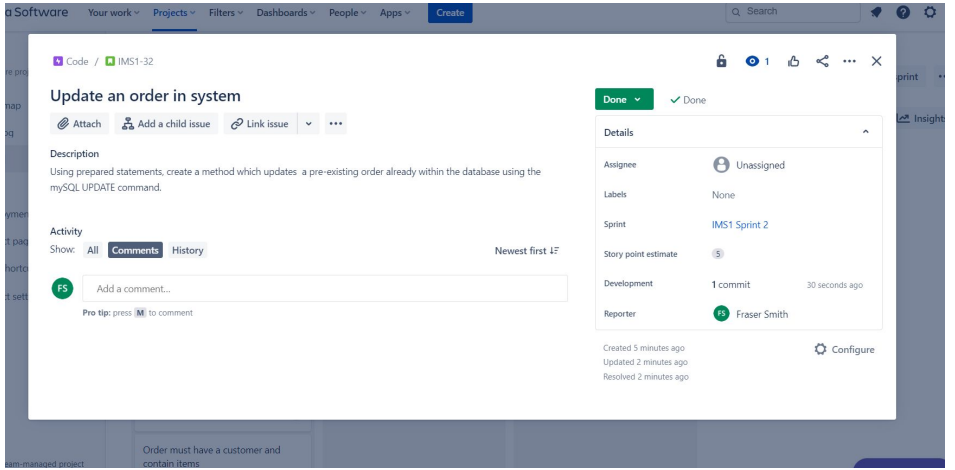




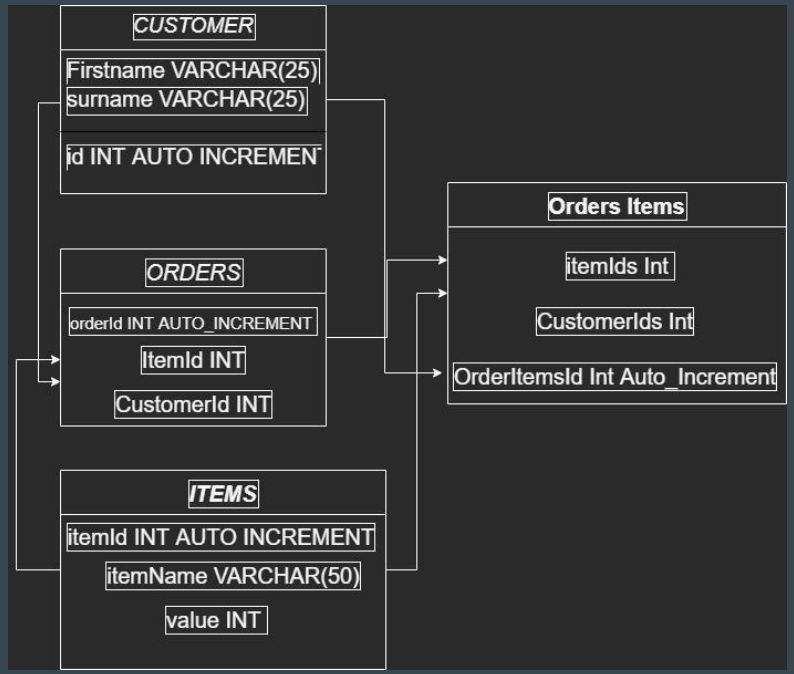
Scrum

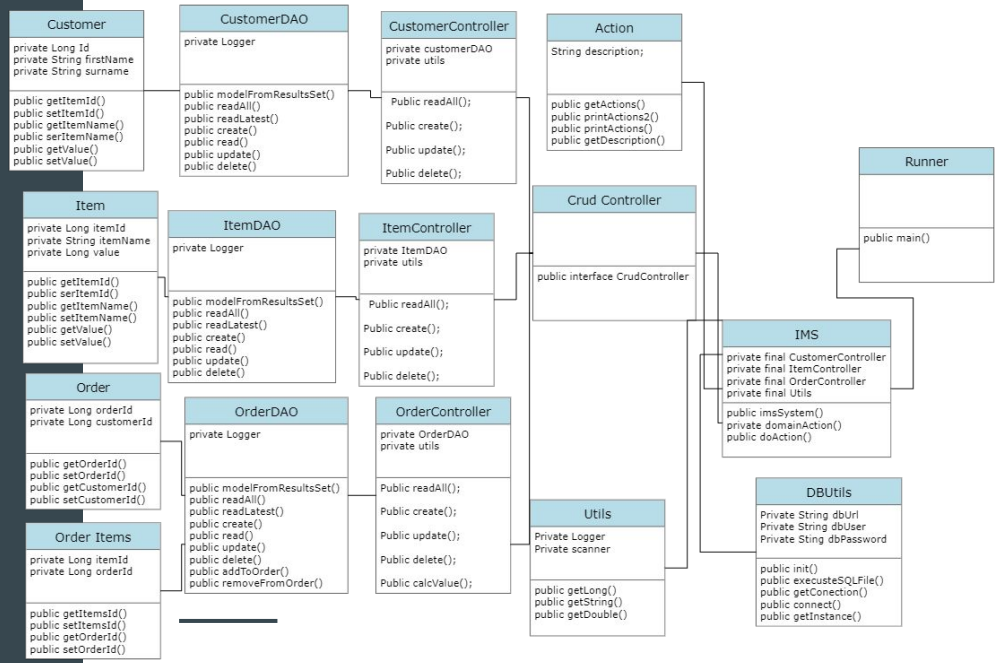
Jira


Insights
IMS1 Sprint 2 25 Oct – 26 Oct (8 issues)
IMS1-13 Create an order in system CODE
IMS1-14 View all orders in system CODE
IMS1-16 Add an item to an order CODE
IMS1-17 Calculate cost for an order CODE
IMS1-18 Delete an item in an order CODE
IMS1-15 Delete an order in system CODE
IMS1-8 Add multiple items to created order CODE
IMS1-27 Order must have a customer and contain ite... CODE
Create issue
Backlog (4 issues)
Code / IMS1-14
View all orders in system
In Progress
Description
In OrderDAO the method read(Long id) should be created in order to access the id number of an order and read out its contents.
Details
Assignee Fraser Smith
Labels None
Add a comment...


Code / IMS1-32
Update an order in system
Attach
Add a child issue
Link issue
Description
Using prepared statements, create a method which updates a pre-existing order already within the database using the mySQL UPDATE command.
Activity
Show: All Comments History
Newest first
Add a comment...
Pro tip: press M to comment
Done
Details
Assignee Unassigned
Labels None
Sprint IMS1 Sprint 2
Story point estimate 5
Development 1 commit 30 seconds ago
Reporter Fraser Smith
Created 5 minutes ago
Updated 2 minutes ago
Resolved 2 minutes ago
Configure

UML & ERD:


CUSTOMER
Firstname VARCHAR(25)
surname VARCHAR(25)
id INT AUTO INCREMEN
ORDERS
orderId INT AUTO_INCREMENT
ItemId INT
CustomerId INT
ITEMS
itemId INT AUTO INCREMENT
itemName VARCHAR(50)
value INT
Orders Items
itemIds Int
CustomerIds Int
OrderItemsId Int Auto_Increment


Customer
private Long Id
private String firstName
private String surname
public getItemId()
public setItemId()
public getItemName()
public serItemName()
public getValue()
public setValue()
CustomerDAO
private Logger
public modelFromResultsSet()
public readAll()
public readLatest()
public create()
public read()
public update()
public delete()
CustomerController
private customerDAO
private utils
Public readAll();
Public create();
Public update();
Public delete();
Action
String description;
public getActions()
public printActions2()
public printActions()
public getDescription()
Runner
public main()
Item
private Long itemId
private String itemName
private Long value
public getItemId()
public serItemId()
public getItemName()
public setItemName()
public getValue()
public setValue()
ItemDAO
private Logger
public modelFromResultsSet()
public readAll()
public readLatest()
public create()
public read()
public update()
public delete()
ItemController
private ItemDAO
private utils
Public readAll();
Public create();
Public update();
Public delete();
Crud Controller
public interface CrudController
IMS
private final CustomerController
private final ItemController
private final OrderController
private final Utils
public imsSystem()
private domainAction()
public doAction()
Order
private Long orderId
private Long customerId
public getOrderId()
public setOrderId()
public getCustomerId()
public setCustomerId()
OrderDAO
private Logger
public modelFromResultsSet()
public readAll()
public readLatest()
public create()
public read()
public update()
public delete()
public addToOrder()
public removeFromOrder()
OrderController
private OrderDAO
private utils
Public readAll();
Public create();
Public update();
Public delete();
Public calcValue();
Order Items
private Long itemId
private Long orderId
public getItemsId()
public setItemsId()
public getOrderId()
public setOrderId()
Utils
Private Logger
Private scanner
public getLong()
public getString()
public getDouble()
DBUtils
Private String dbUrl
Private String dbUser
Private Sting dbPassword
public init()
public executeSQLFile()
public getConection()
public connect()
public getInstance()

Code:

Domain:

Object Setup

- Customer
- Item
- Order
- Item Order

DAOs:

Method Setup -3 Methods

- Create
- ReadAll
- Read
- Update
- Delete

Controllers:

Customer, Item, Order

Order changed to add additional methods

Documentation:

README:

and orders. It also allows for the update of any of th three and the addition of a number of items to an order.

Coverage: 75.3%

IMS Project

Creation of an inverntory management system to process and handle people, items, and orders.

Getting Started

In the target folder, run the ims-0.0.1-jar-with-dependencies in the command line.

Prerequisites

Java compatible operating system 2GB RAM Java 11

Java 11 A Java Capable Operating System 4GB RAM

```
System was successfully run on:
ERAZER DEPUTY P10
Intel I7
RTX 2060
=======
Project was run on:

ERAZER DEPUTY P10
4GB RAM
Intel I7
RTX2060
```

Installing

Access the target folder, run command line, then run the following command:

Risk Assessment:

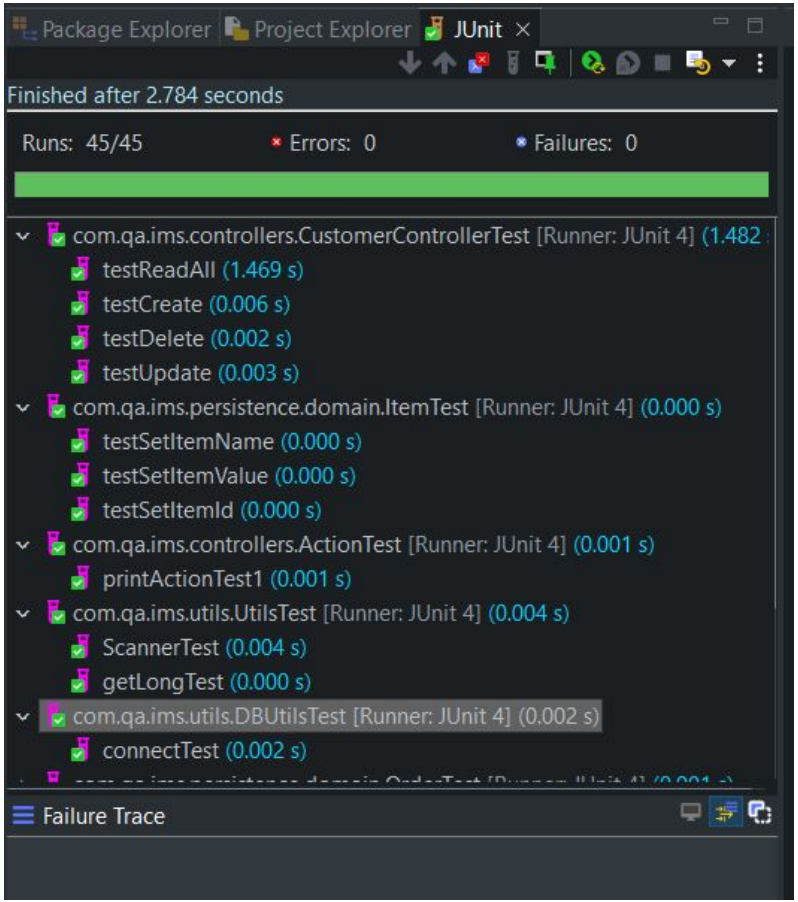
IMS Risk Assessment

Index	Risk Description	Mitigation strategies	Severity	Likelihood
1	Repetitive Strain Injury (RSI)	Take smaller more frequent breaks, exercise wrist and hand mobility		
2	Eye Strain / Visual Acuity loss	Take small frequent breaks to vary the distance at which your eyes are focused		
3	Power Outage	Operate on a laptop with battery power, and ensure the batteries are charged		
4	Loss of work, lost data	Use a version control software, such as github, so as not to loose data		
5	Destruction of hardware	Save all work on a cloud based storage system or version controll such as github		
6	Illness / Loss of time	Use a jira board / scrum in order to mange time and ensure MVP philosophy is observed		
7	Memory Loss / Head injury	Comment your code, including detailed descriptions of how complex methods work		
8	Carpal Tunnel Syndrome (CTS)	Wrist exercises to promote blood flow to the region		
9	Back Strain	Use a chair with sufficient lumbar support, or use furniture that supports better posture		
10	Fire	Ensure fire alarms are armed and are powered with regular checks		

Risk Assessment Matrix	Severity ->	Insignificant	Minor	Moderate	Major	Extreme
Probability \/						
Very Likely						
Likely						
Possible						
Unlikely						
Very Unlikely						

Testing:

JUnit Tests:

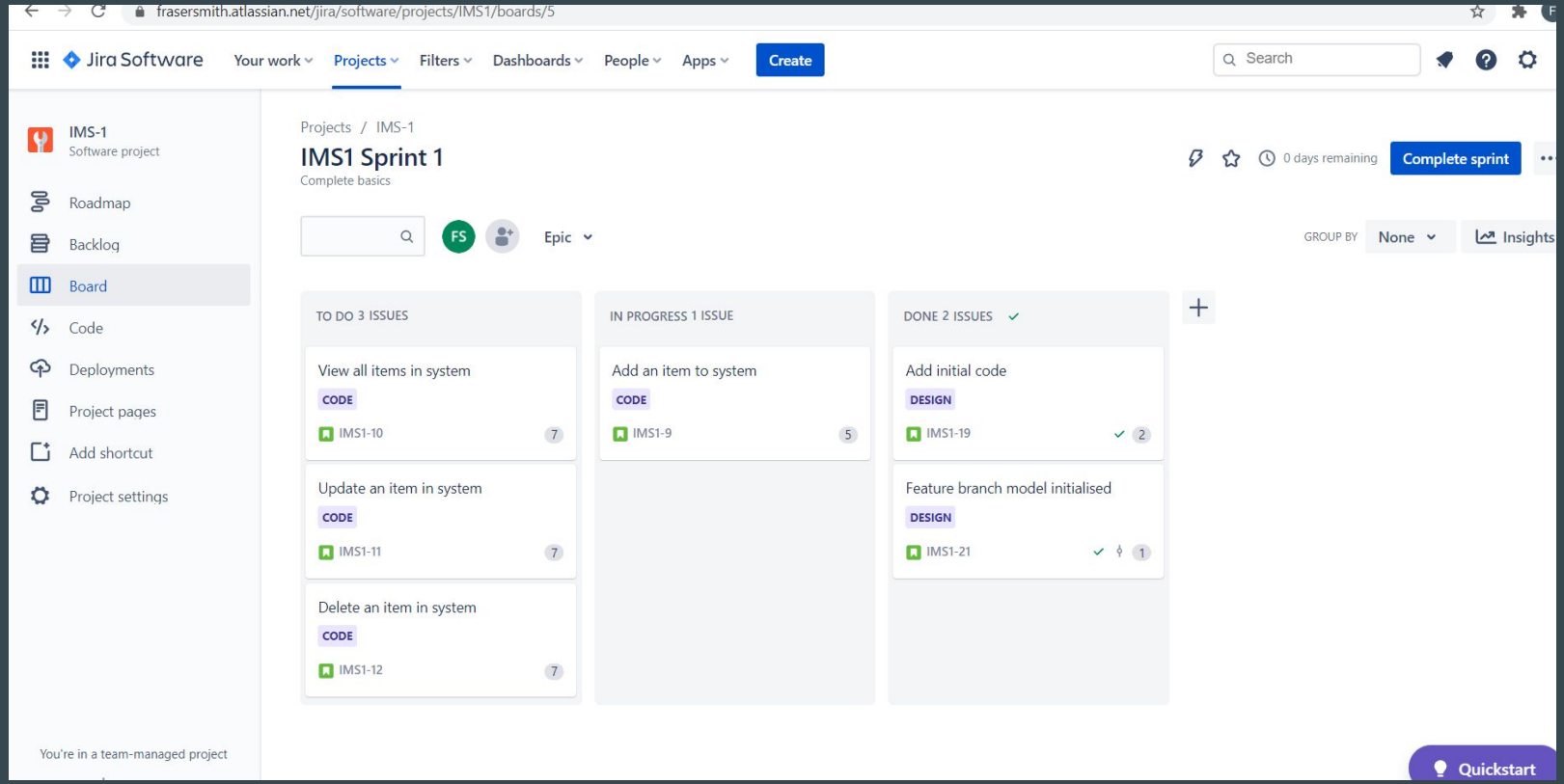

Package Explorer
Project Explorer
JUnit
Finished after 2.784 seconds
Runs: 45/45
Errors: 0
Failures: 0
com.qa.ims.controllers.CustomerControllerTest [Runner: JUnit 4] (1.482
testReadAll (1.469 s)
testCreate (0.006 s)
testDelete (0.002 s)
testUpdate (0.003 s)
com.qa.ims.persistence.domain.ItemTest [Runner: JUnit 4] (0.000 s)
testSetItemName (0.000 s)
testSetItemValue (0.000 s)
testSetItemId (0.000 s)
com.qa.ims.controllers.ActionTest [Runner: JUnit 4] (0.001 s)
printActionTest1 (0.001 s)
com.qa.ims.utils.UtilsTest [Runner: JUnit 4] (0.004 s)
ScannerTest (0.004 s)
getLongTest (0.000 s)
com.qa.ims.utils.DBUtilsTest [Runner: JUnit 4] (0.002 s)
connectTest (0.002 s)
Failure Trace

Mockito Tests:

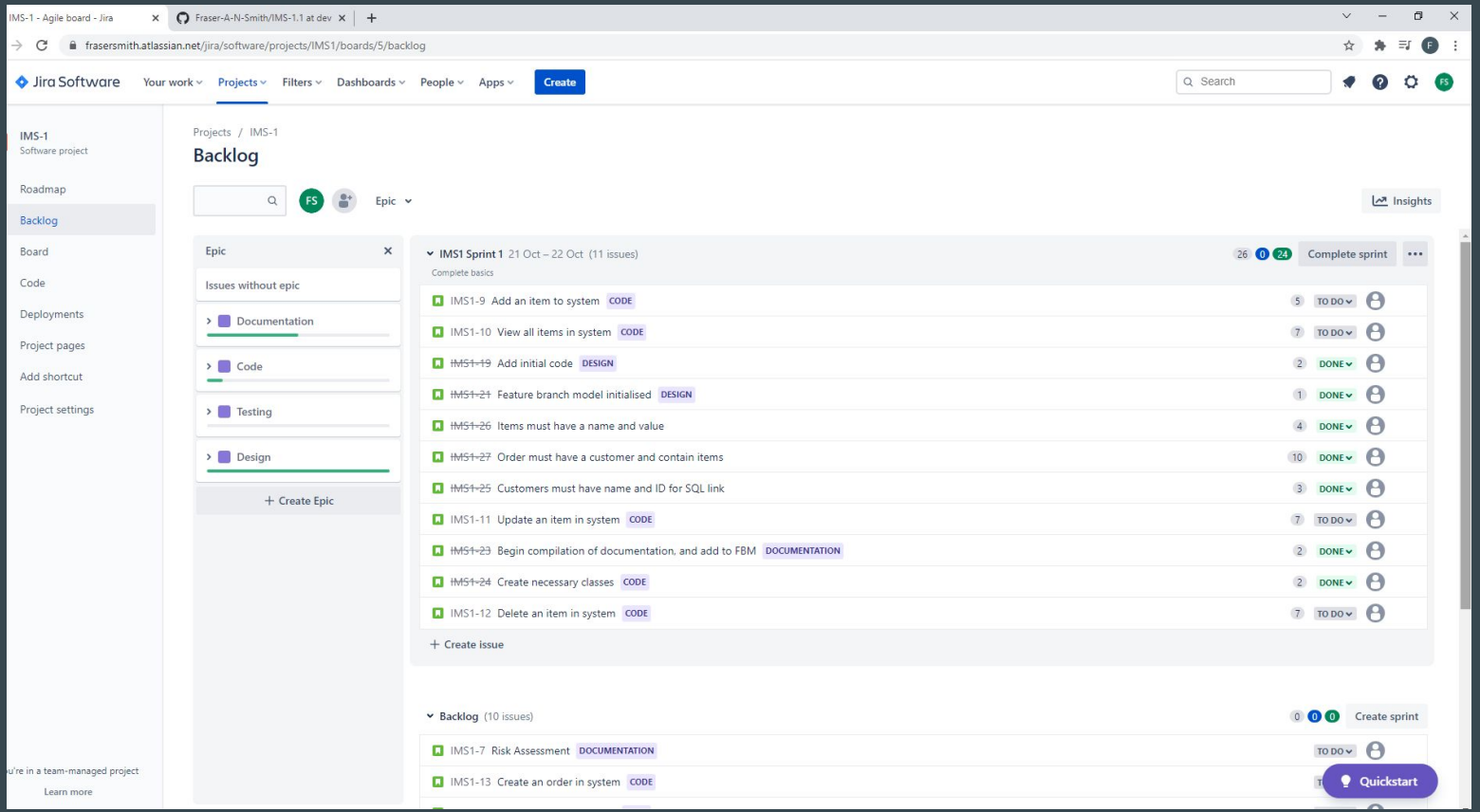
CustomerControllerTest.java

```
22
23      @Mock
24      private Utils utils;
25
26      @Mock
27      private CustomerDAO dao;
28
29      @InjectMocks
30      private CustomerController controller;
31
32      @Test
33      public void testCreate() {
34          final String F_NAME = "barry", L_NAME = "scott";
35          final Customer created = new Customer(F_NAME, L_NAME);
36
37          Mockito.when(utils.getString()).thenReturn(F_NAME, L_NAME);
38          Mockito.when(dao.create(created)).thenReturn(created);
39
40          assertEquals(created, controller.create());
41
42          Mockito.verify(utils, Mockito.times(2)).getString();
43          Mockito.verify(dao, Mockito.times(1)).create(created);
44      }
45
46
47      @Test
48      public void testReadAll() {
49          List<Customer> customers = new ArrayList<>();
50          customers.add(new Customer(1L, "jordan", "harrison"));
51
52          Mockito.when(dao.readAll()).thenReturn(customers);
53
54
```

Sprint Review



Sprint Retrospective



Conclusion

- Improvements
- Failings
- Future Steps

Further Questions?