IMS PROJECT PRESENTATION

Harry Fresco

CONTENTS

INTRODUCTION

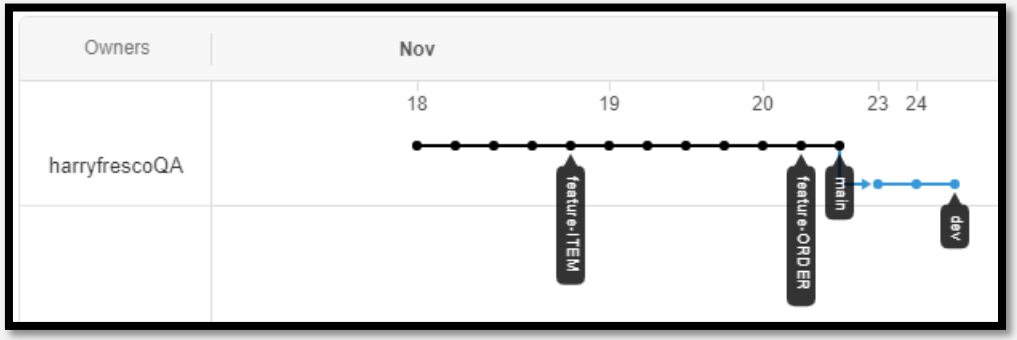
- Overview of how I approached the specification:
  - User stories and created Jira board
  - Designed and created database on GCP instance
  - Created object classes and their DAOs and Controllers
  - Found from Uni that it's better to build it layer-by-layer than all the DAOs then all Controllers (MVP)
  - Used User Stories to test along with JUnit

CONSULTANT JOURNEY

- Covered Java, Junit, SQL at Uni
- Maven
- Jira
- GCP
- Mockito
- Using DAOs and Controllers

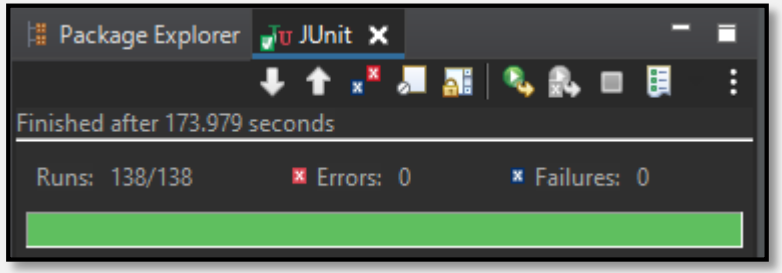
CI

- Used Main/Dev/Feature branches
- Main
  - Contained that compiled
- Dev
- Feature
  - Used for when a specific feature is being developed and code may not compile
  - Such as feature-ITEM
  - Merged to dev when complete



TESTING

- Consisted of Junit testing, Mockito, and functional testing
- Junit
  - Domains, DAO's
- Mockito
  - Controllers
- Used the user stories to perform functionality testing on the whole system.
  - Example: As a User, I want to **view all customers** in the **system** so that I can find a customer's information**.**



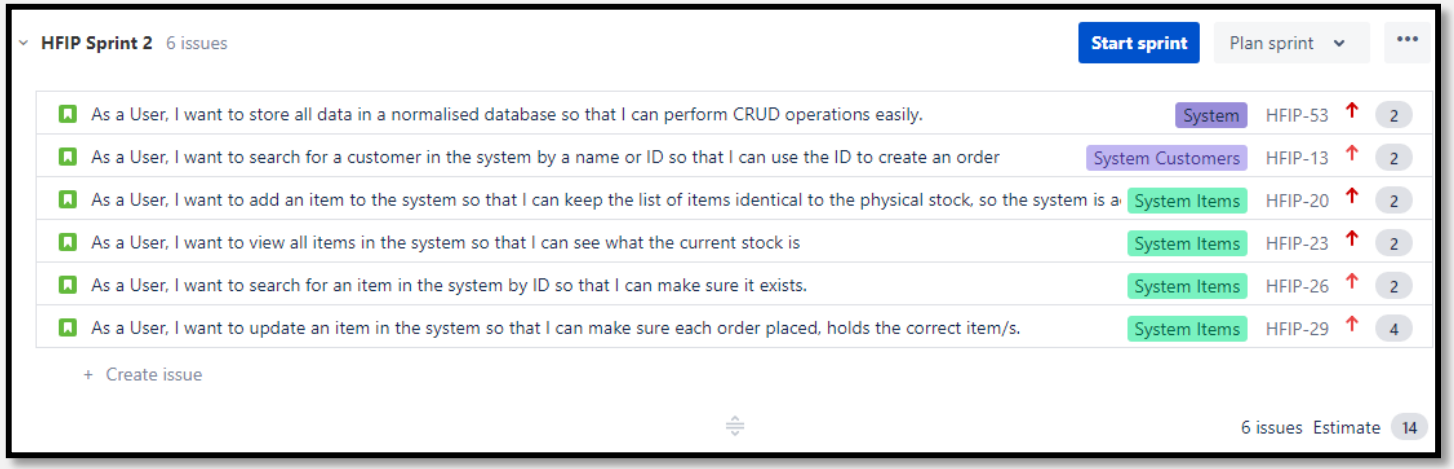
Element	Coverage	Covered Instructions	Missed Instructions	Total Instructions
IMS-Starter-master	88.3 %	5,704	759	6,463
src/main/java	74.7 %	2,238	759	2,997
src/test/java	100.0 %	3,466	0	3,466

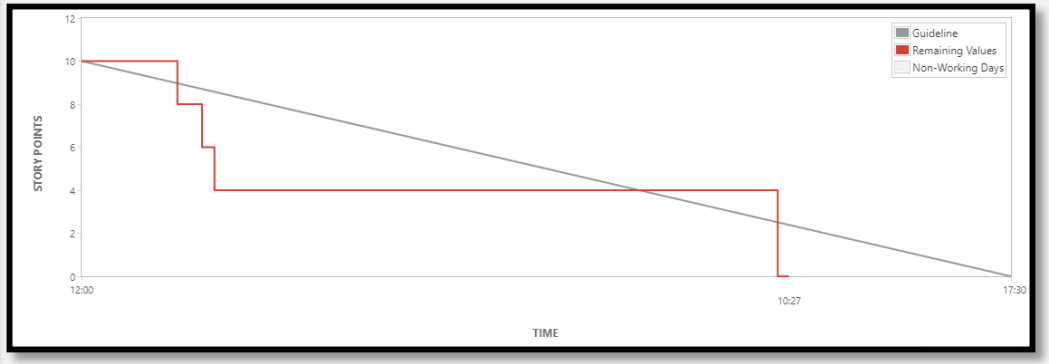
```
@Test
public void testReadAll() {
    List<Item> expected = new ArrayList<>();
    // ID Title Quantity Price
    expected.add(new Item(11, "Fender Stratocaster - White", 100, 700.00d));

    assertEquals(expected, DAO.readAll());
}
```

DEMO

- User stories to demonstrate:
- As a User, I want to **view all customers** in the **system** so that I can find a customer's information**.**
- As a User, I want to **add an item** to the **system** so that I can keep the list of items identical to the physical stock, so the system is accurate **x2**
- As a User, I want to **view all items** in the **system** so that I can see what the current stock is
- As a User, I want to **create an order with a customer and item/s** so that I can keep track of all orders easily
- As a User, I want to **view all orders** in the **system** so that I can easily search for a specific order and get the details.
- As a User, I want to **delete an item from an order** so that I can update the order so that the customer is billed correctly.
- As a User, I want to **delete an order** in the **system** so that I can keep the systems data relevant.

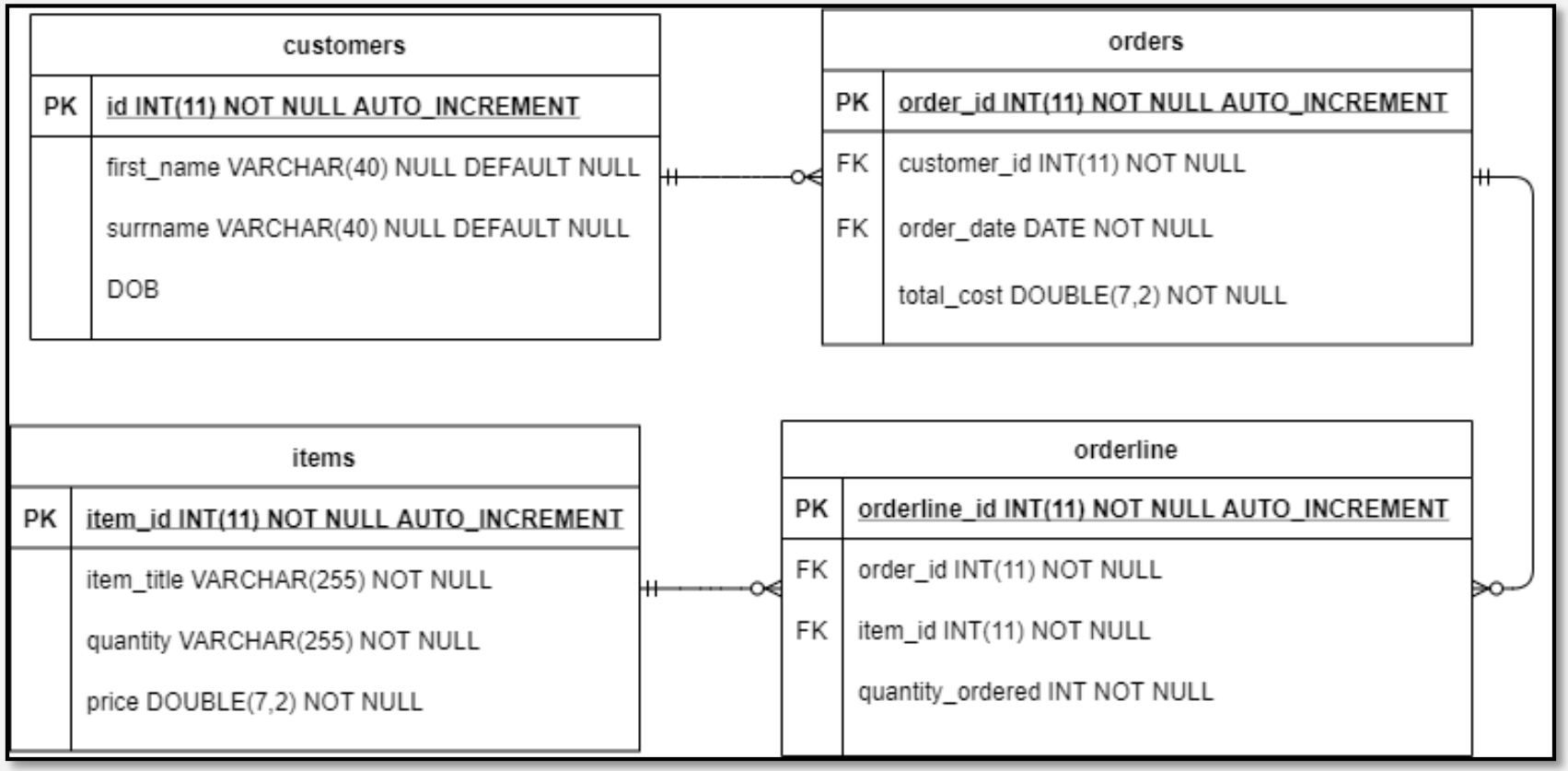
SPRINT 1





- Was mainly to do with Items and it's CRUD functions
- Went easier than predicted
- Updating Item took a while to decide how to ask the user to enter information

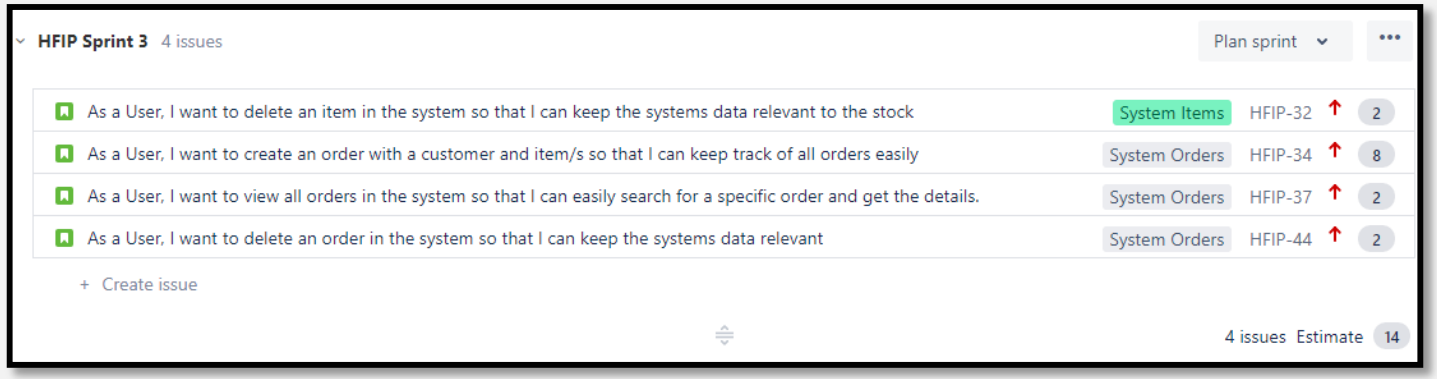
DATABASE DESIGN

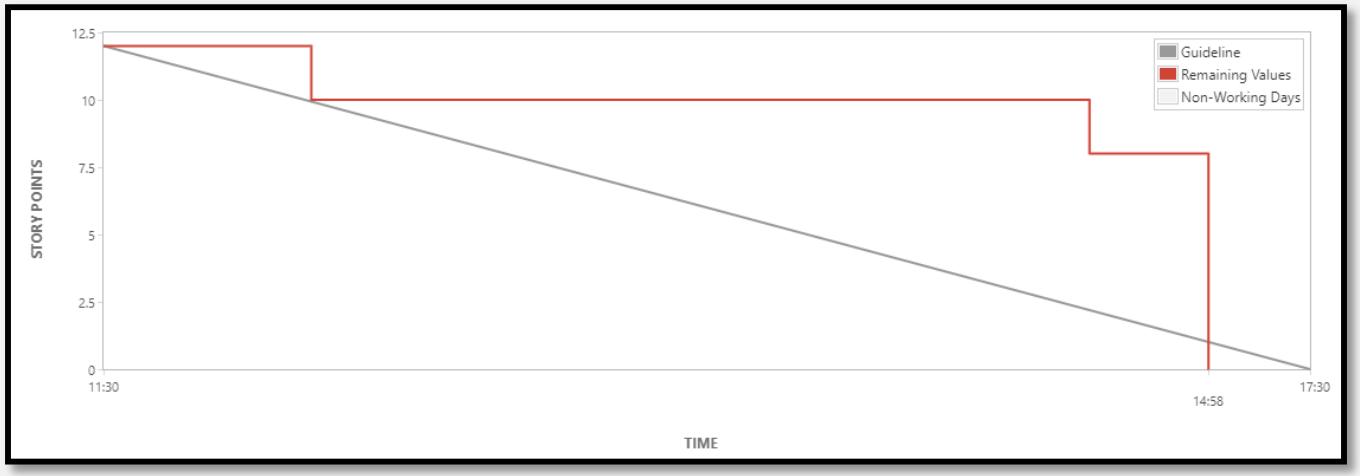


SPRINT 2

- Mainly to do with Orders and it's CRUD functions
- Difficult to decide how the orderline should connect
- Compromises:
  - Created Orderline with own id so could delete one. As the delete function in the interface only accepted one id. (Could not pass in order_id and item_id)
  - Created Orderline object and DAO so each item in an order can have a quantity
    - Tried to have an Order have list of items but found it easier to use separate object. May not have been the best way.

SPRINT 2

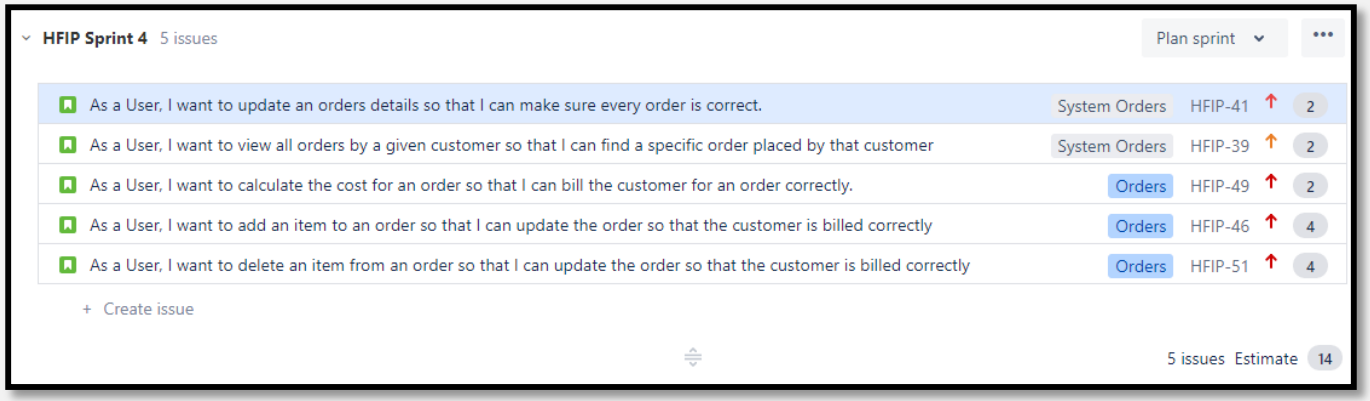

HFIP Sprint 3 4 issues
Plan sprint
As a User, I want to delete an item in the system so that I can keep the systems data relevant to the stock
System Items
HFIP-32
2
As a User, I want to create an order with a customer and item/s so that I can keep track of all orders easily
System Orders
HFIP-34
8
As a User, I want to view all orders in the system so that I can easily search for a specific order and get the details.
System Orders
HFIP-37
2
As a User, I want to delete an order in the system so that I can keep the systems data relevant
System Orders
HFIP-44
2
+ Create issue
4 issues Estimate 14

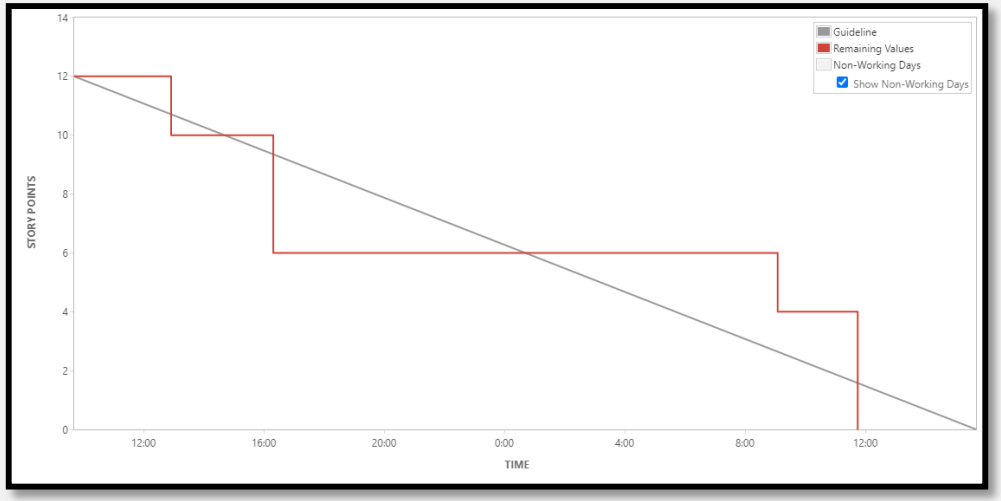

12.5
10
7.5
5
2.5
0
STORY POINTS
11:30
14:58
17:30
TIME
Guideline
Remaining Values
Non-Working Days

SPRINT 3

- To do with integrating everything together. Customers have orders and Orders contain items.
- The most difficult part of the whole project. Mainly adding and deleting items to/from an order. This was because the Order object has a list of orderlines. A list of items instead may have solved this but I could not get it to work well with the database.
- Compromises:
  - The Create_order function calls to an AddToOrder function to add items to the order. The deletefromOrder function is also called when delete from order is called
    - Not the best way
    - Made it harder to test each function separately.

SPRINT 3


HFIP Sprint 4 5 issues
Plan sprint
As a User, I want to update an orders details so that I can make sure every order is correct. System Orders HFIP-41 2
As a User, I want to view all orders by a given customer so that I can find a specific order placed by that customer System Orders HFIP-39 2
As a User, I want to calculate the cost for an order so that I can bill the customer for an order correctly. Orders HFIP-49 2
As a User, I want to add an item to an order so that I can update the order so that the customer is billed correctly Orders HFIP-46 4
As a User, I want to delete an item from an order so that I can update the order so that the customer is billed correctly Orders HFIP-51 4
+ Create issue
5 issues Estimate 14


Guideline
Remaining Values
Non-Working Days
Show Non-Working Days
STORY POINTS
14
12
10
8
6
4
2
0
12:00
16:00
20:00
0:00
4:00
8:00
12:00
TIME

CONCLUSION

- **What went well**
  - Using Jira helped to have idea of what was needed to be done
  - Database design and implementation
  - Integrating into the current design pattern
  - Adding and deleting items from orders
- **What I enjoyed**
  - Allowing Orders to have many Items
  - Creating a easy-to-read layout for orders
- **What could be improved**
  - Order should have list of Items instead of orderlines.
- **What have I taken away / is useful for future**
  - Using Jira is very useful to get a project started
  - Create objects first (To avoid problems like: Order having list of Orderlines)
  - Creating Junit tests help to understand code / fix it a lot better