INVENTORY MANAGEMENT SYSTEM PROJECT-QA

A PRESENTATION BY LLOYD LOW

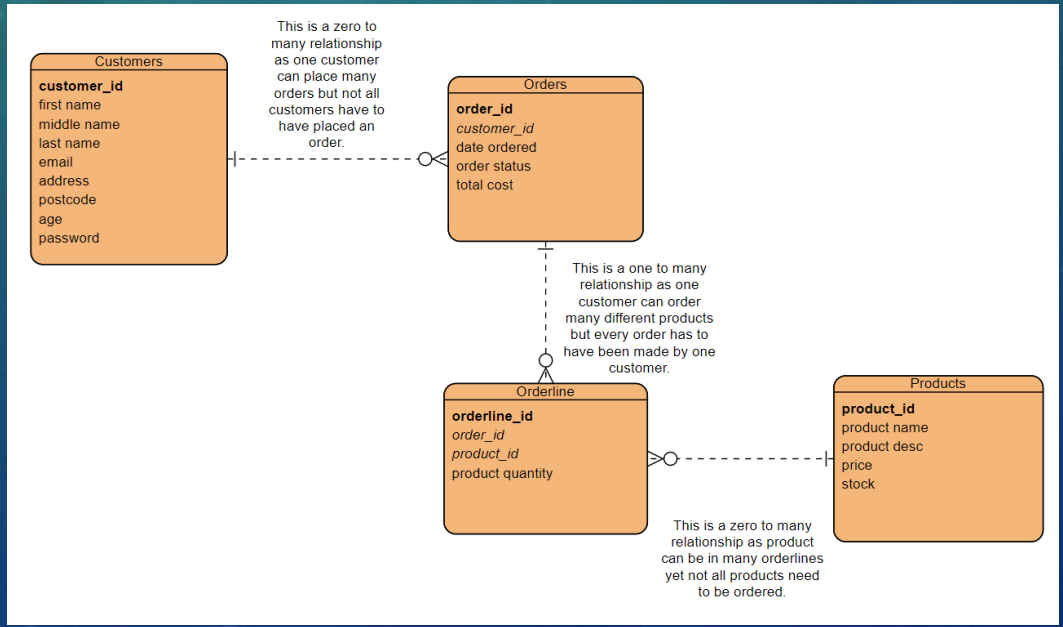
PROGRAMS/DESIGN METHODOLOGIES USED

JIRA – Kanban and Scrum boards for agile project planning.
GITHUB – Git GUI for repository and source management.
MYSQL WORKBENCH – Database creation and management tool.
ECLIPSE – Integrated development environment using Java, an object-oriented programming language.
MAVEN – Good practice build tool for Java projects.
JUNIT – Unit testing tool for Java projects.
MOCKITO – Extension of JUNIT, used to test methods that call many other methods efficiently.

MY PRE-PROJECT RISK ASSESSMENT AND ERD BOARD

Project Name	Project Fundimentals (IMS)
Prepared By	Lloyd Low
Date	11-12-20

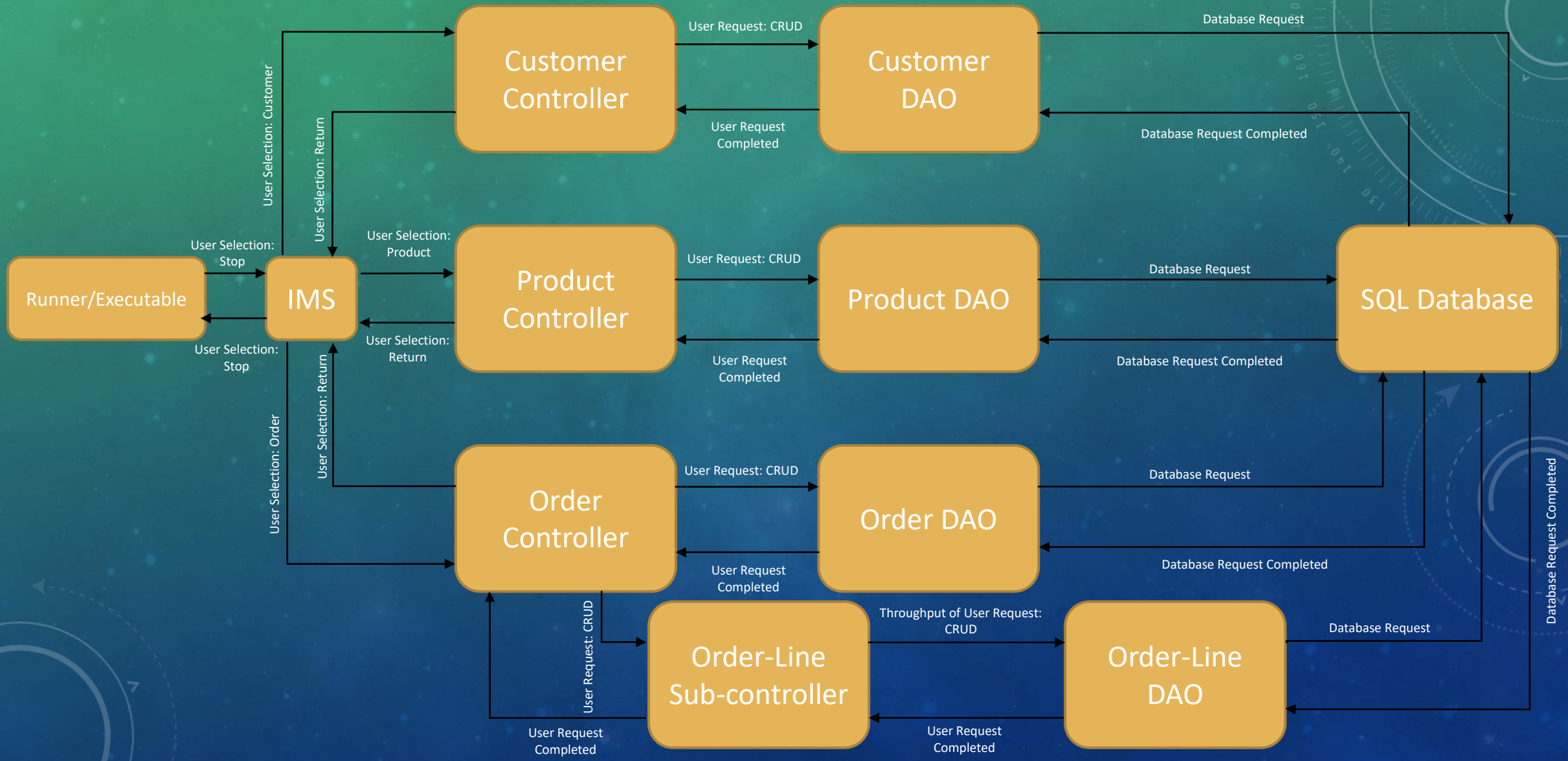
Problem Area or Activity	Risks Identified	Probability Of Occurance	Impact Intensity	Mitigation Strategy	Additional Measures
Overall Project	Risk of Covid-19	Low	High	Wear a mask when out in public	Observe social distancing
	Loss of Files/progress	Low	Extreme	Ensure regular commits	Save often
	Internet outages	Medium	Medium	Ensure a stable internet connection	Connect via ethernet cable where possible
	Leaving GCP running	Medium	Low/Medium	Always check the GCP server is spun down after use	Set up expenditure warnings
	Running out of time	Medium	Extreme	Properly utilise Jiras planning functionality to elect time to core issue	Take the approach that barebone yet functional code is better than complex and broken code.



KANBAN BOARD SETUP

FPI-9 *	As a customer I want to be added to the customer table so that I can order items	Story	Medium
FPI-10 *	As a store manager I want to be able to see all the customers in my table so that I can see how many customers the store has	Story	Medium
FPI-11 *	As a store owner I want to be able to change a customers details in the system so that I can keep customer information up to date	Story	Medium
FPI-12 *	As a store owner I want to be able to remove a customer from the system so that I can keep the list up to date	Story	Medium
FPI-13 *	As a store owner I want to be able to add a product so that customers can buy it	Story	Medium
FPI-14 *	As a customer I want to be able to view all products in the store so that I can choose what to buy	Story	Medium
FPI-15 *	As a store owner I want to be able to update products in order to keep the shop up to date	Story	Medium
FPI-16 *	As a store owner I want to be able to delete a product so that the product list is kept relevant	Story	Medium
FPI-24 *	Fork IMS-Starter repo	Task	Medium
FPI-25 *	Create developer branch	Task	Medium
FPI-26 *	Sprint risk assesment	Task	Medium
FPI-27 *	Add Boolean and integer parsing to utils.java	Task	Medium
FPI-30 *	Code a read method for the order table	Task	Medium
FPI-32 *	code a delete method for the orders table	Task	Medium
FPI-33 *	Code a create method for the product controller	Task	Medium
FPI-34 *	Code a read method for the products table	Task	Medium
FPI-35 *	Code an update method for the products table	Task	Medium
FPI-36 *	Code a delete method for the products table	Task	Medium
FPI-37 *	Create a feature branch in order to add the extra functionality	Task	Medium
FPI-38 *	Code a data access object for the Create product controller method	Task	Medium
FPI-39 *	Create a products class	Task	Medium
FPI-40 *	Update the Customer class to include the extra information within my table	Task	Medium
FPI-41 *	Update the Customer DAO to reflect my table	Task	Medium
FPI-37 *	Create a feature branch in order to add the extra functionality	Task	Medium
FPI-38 *	Code a data access object for the Create product controller method	Task	Medium
FPI-39 *	Create a products class	Task	Medium
FPI-40 *	Update the Customer class to include the extra information within my table	Task	Medium
FPI-41 *	Update the Customer DAO to reflect my table	Task	Medium
FPI-42 *	Update the Customer controller so as the reflect my table	Task	Medium
FPI-43 *	Update the user interface to ensure that customers are asked for all the necessary information	Task	Medium
FPI-44 *	Update Junit tests to incorporate all table columns.	Task	Medium
FPI-46 *	Create an orderline class	Task	Medium

IMS SETUP AND LINKS



SPRINT 1 REVIEW

14/Dec/20 4:30 PM - 14/Dec/20 5:31 PM

Completed Issues

View in Issue Navigator

Key	Summary	Issue Type	Priority	Status	Story Points (-)
FPI-1 *	Create an ERD board to correctly work out the links between various tables in my database	Task	Medium	DONE	-
FPI-2 *	Produce an overall project risk assessment	Task	Medium	DONE	-
FPI-3 *	produce a sprint specific risk assessment	Task	Medium	DONE	-
FPI-4 *	Create a new database	Task	Highest	DONE	-
FPI-5 *	Create a customer table	Task	Highest	DONE	-
FPI-6 *	Create an orders table	Task	Highest	DONE	-
FPI-7 *	Create an order line table	Task	Highest	DONE	-
FPI-8 *	Create a product table	Task	Highest	DONE	

Quickstart

What went well:

- Completed all issues within timeframe
- Established the framework of the project
- Good assessment of risks
- Good structuring of database table links

What Didn't:

- Later it became apparent that creating the database within SQL was not the correct way to build a reliant system and that instead building it within the repository and then invoking it was a better method

```sql
DROP schema IF EXISTS store_db;
CREATE SCHEMA IF NOT EXISTS `store_db`;
USE `store_db` ;
CREATE TABLE IF NOT EXISTS `store_db`.`customers` (
    `customer_id` BIGINT NOT NULL AUTO_INCREMENT,
    `fName` VARCHAR(45) NULL DEFAULT NULL,
    `lName` VARCHAR(45) NULL DEFAULT NULL,
    `age` INT NOT NULL,
    `email` VARCHAR(100) UNIQUE NOT NULL,
    `password` VARCHAR(45) NOT NULL,
    `address` VARCHAR(100) NOT NULL,
    `postcode` VARCHAR(8) NOT NULL,
    PRIMARY KEY (`customer_id`)
);
CREATE TABLE IF NOT EXISTS `store_db`.`products` (
  `product_id` BIGINT NOT NULL AUTO_INCREMENT,
  `product_name` varchar(45) NOT NULL,
  `product_desc` varchar(200) DEFAULT NULL,
  `price` DOUBLE(6,2) NOT NULL,
  `stock` int NOT NULL,
  PRIMARY KEY (`product_id`)
);
CREATE TABLE IF NOT EXISTS `store_db`.`orders` (
  `order_id` BIGINT NOT NULL AUTO_INCREMENT,
  `customer_id` BIGINT NOT NULL,
  `date_ordered` date NOT NULL,
  `total_cost` DOUBLE (6,2) DEFAULT NULL,
  PRIMARY KEY (`order_id`),
  KEY `customer_id_idx` (`customer_id`),
  CONSTRAINT `customer_id` FOREIGN KEY (`customer_id`) REFERENCES `customers` (`customer_id`)
);
CREATE TABLE IF NOT EXISTS `store_db`.`orderline` (
  `orderline_id` BIGINT NOT NULL AUTO_INCREMENT,
  `order_id` BIGINT DEFAULT NULL,
  `product_id` BIGINT DEFAULT NULL,
  `product_quantity` int(11) DEFAULT NULL,
  PRIMARY KEY (`orderline_id`),
  KEY `order_id_idx` (`order_id`),
  KEY `product_id_idx` (`product_id`),
  CONSTRAINT `order_id` FOREIGN KEY (`order_id`) REFERENCES `orders` (`order_id`),
  CONSTRAINT `product_id` FOREIGN KEY (`product_id`) REFERENCES `products` (`product_id`)
);
```

The SQL code that is called to initialize the shop database

SPRINT 2 REVIEW

16/Dec/20 9:00 AM - 17/Dec/20 5:50 PM

What went well:

- Was able to complete a large amount of the tasks
- Established most IMS links
- Fleshed out the project

What Didn't:

- Not all tasks were completed
- Unable to know if stories were completed fully as testing had not started

Status Report

* Issue added to sprint after start time

Completed Issues

View in Issue Navigator

Key	Summary	Issue Type	Priority	Status	Story Points (-)
FPI-9 *	As a customer I want to be added to the customer table so that I can order items	Story	High	DONE	-
FPI-10 *	As a store manager I want to be able to see all the customers in my table so that I can see how many customers the store has	Story	High	DONE	-
FPI-11 *	As a store owner I want to be able to change a customers details in the system so that I can keep customer information up to date	Story	High	DONE	-
FPI-12 *	As a store owner I want to be able to remove a customer from the system so that I can keep the list up to date	Story	High	DONE	-
FPI-13 *	As a store owner I want to be able to add a product so that customers can buy it	Story	High	DONE	-
FPI-14 *	As a customer I want to be able to view all products in the store so that I can choose what to buy	Story	High	DONE	-
FPI-15 *	As a store owner I want to be able to update products in order to keep the shop up to date	Story	High	DONE	-
FPI-16 *	As a store owner I want to be able to delete a product so that the product list is kept relevant	Story	High	DONE	-
FPI-24 *	Fork IMS-Starter repo	Task	Medium	DONE	-
FPI-25 *	Create developer branch	Task	Medium	DONE	-
FPI-26 *	Sprint risk assesment	Task	Medium	DONE	-
FPI-27 *	Add Boolean and integer parsing to utils.java	Task	Medium	DONE	-
FPI-30 *	Code a read method for the order table	Task	Medium	DONE	-
FPI-32 *	code a delete method for the orders table	Task	Medium	DONE	-
FPI-33 *	Code a create method for the product controller	Task	Medium	DONE	-
FPI-34 *	Code a read method for the products table	Task	Medium	DONE	-
FPI-35 *	Code an update method for the products table	Task	Medium	DONE	-
FPI-36 *	Code a delete method for the products table	Task	Medium	DONE	-
FPI-37 *	Create a feature branch in order to add the extra functionality	Task	Medium	DONE	-
FPI-38 *	Code a data access object for the Create product controller method	Task	Medium	DONE	-
FPI-39 *	Create a products class	Task	Medium	DONE	-
FPI-40 *	Update the Customer class to include the extra information within my table	Task	Medium	DONE	-
FPI-41 *	Update the Customer DAO to reflect my table	Task	Medium	DONE	-
FPI-42 *	Update the Customer controller so as the reflect my table	Task	Medium	DONE	-
FPI-43 *	Update the user interface to ensure that customers are asked for all the necessary information	Task	Medium	DONE	-
FPI-44 *	Update Junit tests to incorporate all table columns.	Task	Medium	DONE	-
FPI-46 *	Create an orderline class	Task	Medium	DONE	-

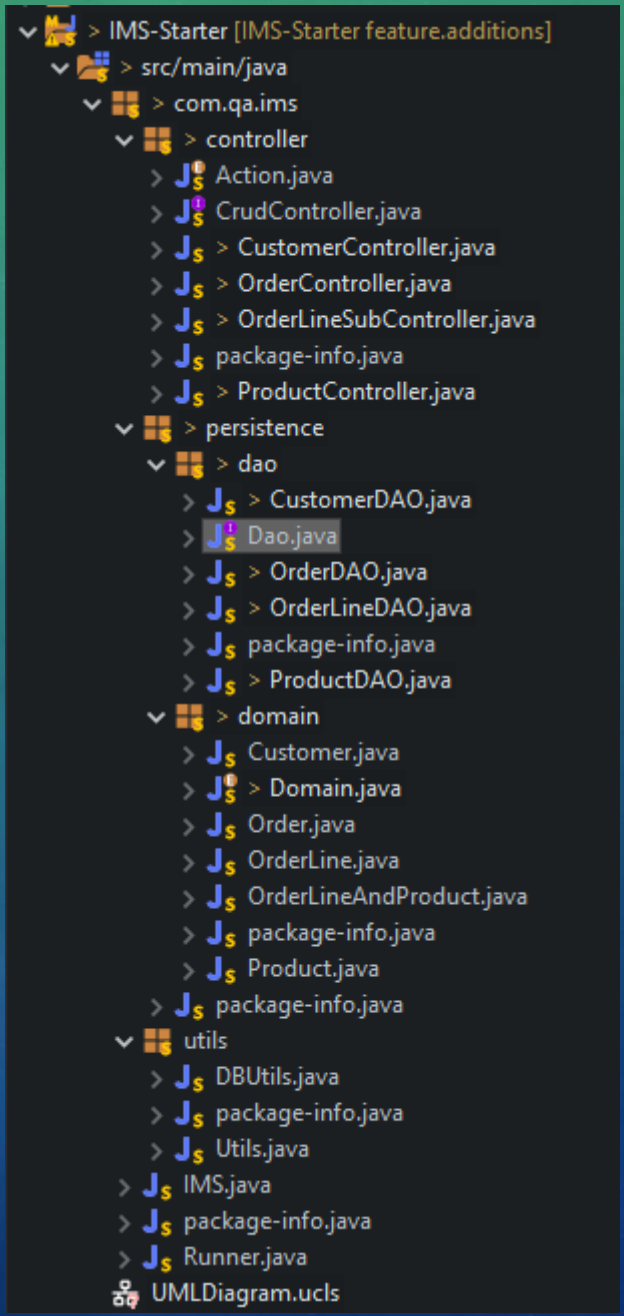
Issues Not Completed

View in Issue Navigator

Key	Summary	Issue Type	Priority	Status	Story Points (-)
FPI-29 *	Code a create method for orders table	Task	Medium	IN PROGRESS	-
FPI-31 *	Code an update method for the orders table	Task	Medium	BACKLOG	

Quickstart

The fully fleshed out project repository



SPRINT 3 REVIEW

18/Dec/20 9:00 AM - 19/Dec/20 1:29 AM

Completed Issues

View in Issue Navigator

Key	Summary	Issue Type	Priority	Status	Story Points (-)
FPI-29	Code a create method for orders table	Task	Medium	DONE	-
FPI-47 *	create a controller for orderline	Task	Medium	DONE	-
FPI-48 *	create a DAO for orderline	Task	Medium	DONE	-
FPI-49 *	code in CRUD functionality for orderline	Task	Medium	DONE	-

Issues Not Completed

View in Issue Navigator

Key	Summary	Issue Type	Priority	Status	Story Points (-)
FPI-31	Code an update method for the orders table	Task	Medium	BACKLOG	

Quickstart

What went well:

- Had full database interaction coded
- Used creative methods to solve various problems

What Didn't:

- CRUD functionality later produced fatal flaws in the order line code
- Not all tasks were completed

```java
    @Override
    public Order create() {

        boolean more = true;

        Long customerID = emailCheck();

        if (customerID == null) {

            return null;

        }else {

            LocalDate date = LocalDate.now();
            Date orderDate = Date.valueOf(date);

            double total = 0;

            orderDAO.create(new Order(customerID,orderDate,total));
            Long orderID = orderDAO.readLatest().getOrderID();

            while (more) {

                ordSub.createOrderLine(orderID);
                LOGGER.info("Would you like to add more items to you order? (y/n)");
                more = utils.getBool();

        }


        Order orders = orderDAO.updateTotalPriceCreate(orderID);
        LOGGER.info("Order created");
        return orders;
        //END

    }

    }
```

```java
    public OrderLine createOrderLine(Long ID) {

        Long orderID = ID;

        productDAO.listToString();

        LOGGER.info('\n' + "Which product would you like to order?");
        String userChoice = utils.getString();
        Long productID = productDAO.returnProductID(userChoice);

        if (productID == null) {

            LOGGER.info("Error - Please enter a valid product name.");
            createOrderLine(ID);

        }else {

            LOGGER.info("How many would you like the purchase?");
            int quant = utils.getInt();

            int stockTotal = productDAO.stockCheck(productID, quant);
            if (stockTotal < 0 ) {

                int availableStock = productDAO.readProduct(productID).getStock();
                LOGGER.info("Error - Only " + availableStock + " items were added as that is all we have in stock. \n "
                        + "Would you still like to proceed? (y/n)");
                boolean decision = utils.getBool();

                if (decision) {

                    productDAO.stockCheck(productID, availableStock);
                    OrderLine orderLine = orderLineDAO.create(new OrderLine(orderID, productID, availableStock));
                    return orderLine;

                }else {

                    createOrderLine(ID);

                }

            }else {

            OrderLine orderLine = orderLineDAO.create(new OrderLine(orderID, productID, quant));
            return orderLine;
            }

        }

        return null;

    }
```

Visual demonstration of the role of OrderLineSubController.java within OrderController.java

SPRINT 4 REVIEW

22/Dec/20 9:00 AM - 23/Dec/20 9:22 PM

Completed Issues

View in Issue Navigator

Key	Summary	Issue Type	Priority	Status	Story Points (-)
FPI-31 *	Code an update method for the orders table	Task	Medium	DONE	-
FPI-51 *	Perform full scale testing of CustomerDAO	Task	Medium	DONE	-

Issues Not Completed

View in Issue Navigator

Key	Summary	Issue Type	Priority	Status	Story Points (-)
FPI-17 *	As a customer I want to be able to order products so that I can buy products	Story	Medium	SELECTED FOR DEVELOPMENT	-
FPI-18 *	As a store owner I want to be able to view all orders so that I know what's being ordered	Story	Medium	SELECTED FOR DEVELOPMENT	-
FPI-21 *	As a customer I want to know how much my order costs so that I can work out if I can pay for it.	Story	Medium	BACKLOG	-
FPI-50 *	Perform full scale testing of CustomerController	Task	Medium	IN PROGRESS	-
FPI-52 *	perform full scale testing of ProductController	Task	Medium	IN PROGRESS	-

What went well:

- Started testing sections of code
- Learnt how to utilise JUNIT and Mockito
- Managed to code a user friendly method for updating

What Didn't:

- Many issues with how I thought coding update for this table would work causing delays in the sprint
- Integration errors due to my lack of knowledge on controller interactions
- Not all tasks were completed

```java
    @Test
    public void testCreate() {
        final Customer created = new Customer(5L, "chris", "perrins",72,"reach@ufgs.net", "g", "343 red", "DC3 4rd");
        assertEquals(created, DAO.create(created));
    }

    @Test
    public void testReturningCustomerID() {
        final String email = "ll@qa.com";
        final Long id = 1L;
        assertEquals(id,DAO.returningCustomerID(email));
    }

    @Test
    public void testReadAll() {
        List<Customer> expected = new ArrayList<>();
        expected.add(new Customer(1L, "jordan", "harrison",22,"ll@qa.com", "dgsfhaf8g", "343 fasd sf", "DC3 4rd"));
        expected.add(new Customer(2L,"James","Pierson",29,"jp@gmail.com","Morg123","23 Word Street","SD23 3GH"));
        expected.add(new Customer(3L,"Hannah","Wardwell",65,"hwe@gmail.com","Matgag3","12 Faor Loop","PE13 5RH"));
        expected.add(new Customer(4L,"Andre","Harlow",40,"igas.43@aol.com","grafac5","1A Lawn Road","DO03 9KF"));

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

    @Test
    public void testReadLatest() {
        assertEquals(new Customer(4L,"Andre","Harlow",40,"igas.43@aol.com","grafac5","1A Lawn Road","DO03 9KF"), DAO.readLatest());
    }

    @Test
    public void testRead() {
        final long ID = 1L;
        assertEquals(new Customer(ID, "jordan", "harrison",22,"ll@qa.com", "dgsfhaf8g", "343 fasd sf", "DC3 4rd"), DAO.readCustomer(ID));
    }

    @Test
    public void testUpdate() {
        final Customer updated = new Customer(1L, "chris", "perrins",22,"rech@ufgs.net", "g", "343 red", "DC3 4rd");
        assertEquals(updated, DAO.update(updated));

    }

    @Test
    public void testDelete() {
        assertEquals(1, DAO.delete(1));
    }


    @Test
    public void testReturningCustomers() {
        final String email = "ll@qa.com";
        assertTrue(DAO.returningCustomer(email));
    }

    @Test
    public void testReturningCustomersFalse() {
        final String email = "NOTANEMAIL";
        assertFalse(DAO.returningCustomer(email));
    }
```

CustomerDAO.java	100.0 %	436	0	436

Demonstration of 100% coverage within CustomerDAO.java

SPRINT 5 REVIEW

27/Dec/20 12:00 PM - 05/Jan/21 10:47 AM

Completed Issues

View in Issue Navigator

Key	Summary	Issue Type	Priority	Status	Story Points (-)
FPI-50 *	Perform full scale testing of CustomerController	Task	Medium	DONE	-
FPI-52 *	perform full scale testing of ProductController	Task	Medium	DONE	-
FPI-53 *	Perform full scale testing of ProductDAO	Task	Medium	DONE	-
FPI-54 *	Perform full scale testing of Order Controller	Task	Medium	DONE	-
FPI-55 *	perform full scale testing of Order Line Controller	Task	Medium	DONE	-
FPI-56 *	perform full scale testing of Order DAO	Task	Medium	DONE	-
FPI-57 *	Perform full scale testing of Order line DAO	Task	Medium	DONE	-

What went well:

- Achieved 100% test coverage across all DAOs and 96.6% across the Controllers
- Completed all tasks within timeframe
- Used errors as an excuse to streamline and simplify overcomplicated code
- Used testing as a justification of new methods that reduced unnecessary copy-paste coding

What Didn't:

- Errors slowed down process and caused rewriting of certain sections
- SQL syntax errors that obviously aren't picked up by eclipse generating weird result sets

```
<terminated> java (1) [JUnit] C:\Program Files\Java\jdk-14.0.1\bin\javaw.exe  (Jan 7, 2021, 3:20:30 PM – 3:24:29 PM)
Access denied for user 'Loot'@'localhost' (using password: YES)
Access denied for user 'Loot'@'localhost' (using password: YES)
Access denied for user 'Loot'@'localhost' (using password: YES)
Access denied for user 'Loot'@'localhost' (using password: YES)
Access denied for user 'Loot'@'localhost' (using password: YES)
Access denied for user 'Loot'@'localhost' (using password: YES)
Access denied for user 'Loot'@'localhost' (using password: YES)
Access denied for user 'Loot'@'localhost' (using password: YES)
Access denied for user 'Loot'@'localhost' (using password: YES)
Access denied for user 'Loot'@'localhost' (using password: YES)
Customer: Customer ID = 1, First name = jordan, Surname = harrison, Age = 22, Email = ll@qa.com, Password = dgsfhaf8g, Address = 343 fasd sf, Postcode = DC3 4rd
Please enter your first name:
Please enter your surname:
Please enter your age:
Please enter your email:
Please enter a password:
Please enter your address:
Please enter your postcode:
Customer created.
Please enter the id of the customer you would like to delete:
Customer deleted.
Please enter the id of the customer you would like to update:
Please update your first name:
Please update your surname:
Please update your age:
Please update your email:
Please update a password:
Please update your address:
Please update your postcode:
Customer updated.

Which product would you like to order?
How many would you like the purchase?
Product in orders basket: OrderLine ID = 1, Product name = Apple, Product Quantity = 2, product Price = 1.99


Product in orders basket: OrderLine ID = 1, Product name = Apple, Product Quantity = 2, product Price = 1.99


Please select the ID you would like to change:
Please update which product to would you like to order:
Please update the order quantity:
Error - Only 102 items were added as that is all we have in stock.
 Would you still like to proceed? (y/n)
null
Do you want to make any more updates? y/n
Product in orders basket: OrderLine ID = 1, Product name = Apple, Product Quantity = 2, product Price = 1.99


Please enter the id of the orderline you would like to delete:
Product in orders basket: OrderLine ID = 1, Product name = Apple, Product Quantity = 2, product Price = 1.99


Please select the ID you would like to change:
Please update which product to would you like to order:
Please update the order quantity:
null
Do you want to make any more updates? y/n
```

Terminal snippet of the tests being carried out

SPRINT 6 REVIEW

06/Jan/21 9:00 AM - 08/Jan/21 12:00 PM

Key	Summary	Issue Type	Priority	Status	Story Points (-)
FPI-17 *	As a customer I want to be able to order products so that I can buy products	Story	High	DONE	-
FPI-18 *	As a store owner I want to be able to view all orders so that I know what's being ordered	Story	High	DONE	-
FPI-19 *	As a store owner I want to be able to delete an order that a customer no longer wants to make	Story	High	DONE	-
FPI-20 *	As a customer I want to be able to add an item to my order so that I can buy it.	Story	High	DONE	-
FPI-21 *	As a customer I want to know how much my order costs so that I can work out if I can pay for it.	Story	High	DONE	-
FPI-22 *	As a customer I want to be able to remove an item from my order so that I don't order it	Story	High	DONE	-
FPI-58 *	Have stock reduce	Story	Low	DONE	-
FPI-60 *	Add in a method to track stock levels and alert when stock drops to zero	Task	Medium	DONE	-
FPI-61 *	Test IMS	Task	Medium	DONE	-

Issues Not Completed

View in Issue Navigator

Key	Summary	Issue Type	Priority	Status	Story Points (-)
FPI-59 *	add extra functionality	Task	Lowest	BACKLOG	

Quickstart

What went well:

- Learnt how to effectively test void methods
- Managed to achieve 88.3% test coverage
- Fulfilled all story points set out in the specifications

What Didn't:

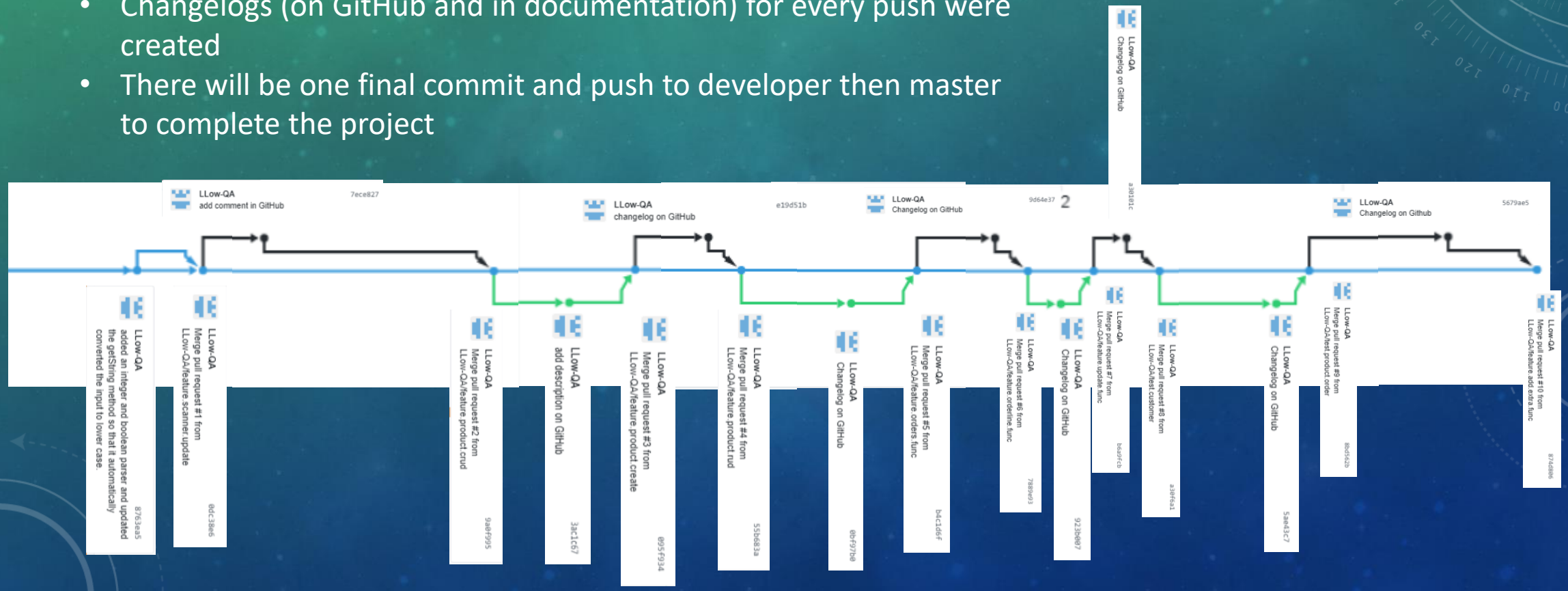
- Removing redundant or counter-intuitive pieces of code that I had previously spent time building
- No extra functionality was added due to time constraints

Element	Coverage	Covered Instructions	Missed Instructions	Total Instructions
IMS-Starter	94.4 %	8,716	521	9,237
src/main/java	88.3 %	3,939	521	4,460
com.qa.ims.persistence.domain	71.3 %	737	296	1,033
Customer.java	71.2 %	208	84	292
OrderLine.java	58.3 %	91	65	156
OrderLineAndProduct.java	62.3 %	86	52	138
Order.java	72.7 %	120	45	165
Product.java	74.6 %	132	45	177
Domain.java	95.2 %	100	5	105
com.qa.ims.utils	56.7 %	161	123	284
Utils.java	9.8 %	12	110	122
DBUtils.java	92.0 %	149	13	162
com.qa.ims	66.5 %	123	62	185
IMS.java	72.8 %	123	46	169
Runner.java	0.0 %	0	16	16
com.qa.ims.controller	96.3 %	1,046	40	1,086
OrderLineSubController.java	90.2 %	333	36	369
Action.java	96.6 %	115	4	119
CustomerController.java	100.0 %	193	0	193
OrderController.java	100.0 %	261	0	261
ProductController.java	100.0 %	144	0	144
com.qa.ims.persistence.dao	100.0 %	1,872	0	1,872
CustomerDAO.java	100.0 %	436	0	436
OrderDAO.java	100.0 %	434	0	434
OrderLineDAO.java	100.0 %	517	0	517
ProductDAO.java	100.0 %	485	0	485
src/test/java	100.0 %	4,777	0	4,777

88.3% coverage and all 106 tests ran successfully

GITHUB FORKING AND BRANCHING

- All new methods, features and tests were created on branches off the developer
- Changelogs (on GitHub and in documentation) for every push were created
- There will be one final commit and push to developer then master to complete the project



PROJECT DEMONSTRATION

OVERALL PROJECT RETROSPECTIVE

What would I do next time:

- Utilise my better understanding of project planning to apply more stringent acceptance criteria and linking of issues/tasks to give myself a more informed view of the project as I had a few but not enough.
- Use story points and weightings to give myself more certainty when it comes to fully completing a sprint.
- Fully integrate my code with Jira to immerse it in the sprints
- Unit test as each method or class is implemented to reduce testing at the end of the project.

FINISHING REMARKS