QA IMS Project

Muddasir Ahmad

Approaching the specification

- Understanding the minimum requirements of the project as per the domain of the specification
- Read existing code to understand project
- Assess the strengths of each technology that is used and implement them effectively
- Adhere to good coding principals (SOLID, OOP)
- Continuously reflect retrospectively on work produced the previous day
- Break down tasks into smaller steps and emphasise simplicity

Risk assessment

- Identify any risks and complications that could arise during the duration of this course
- Measure the magnitude of the impact each risk carries and how it will influence my overall performance

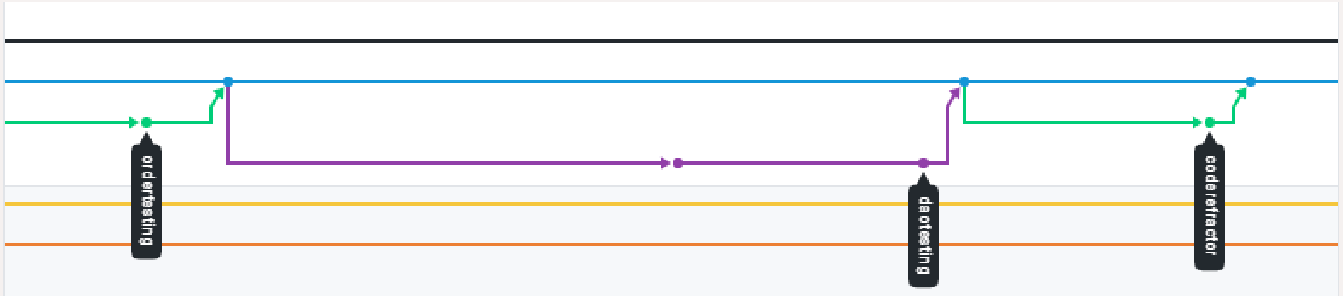
Prerequisites – my consultant journey

- Jira – a technology used to generate a storyboard in an agile style to estimate the difficulty, length, and description of stories and subtasks
- Version control – Git, manage changes to a project
- Knowledge of SQL – Interacts with java via JDBC for access to live database
- Knowledge of Java – for programming the core part of the project
- Maven and Dependencies – for crucial aspects of the project such as testing (JUNIT, Mockito) and packaging Java apps (.jar)

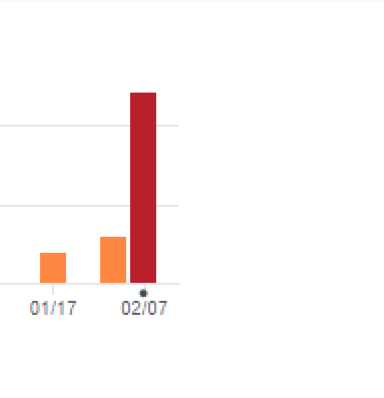
GitHub and Git Bash

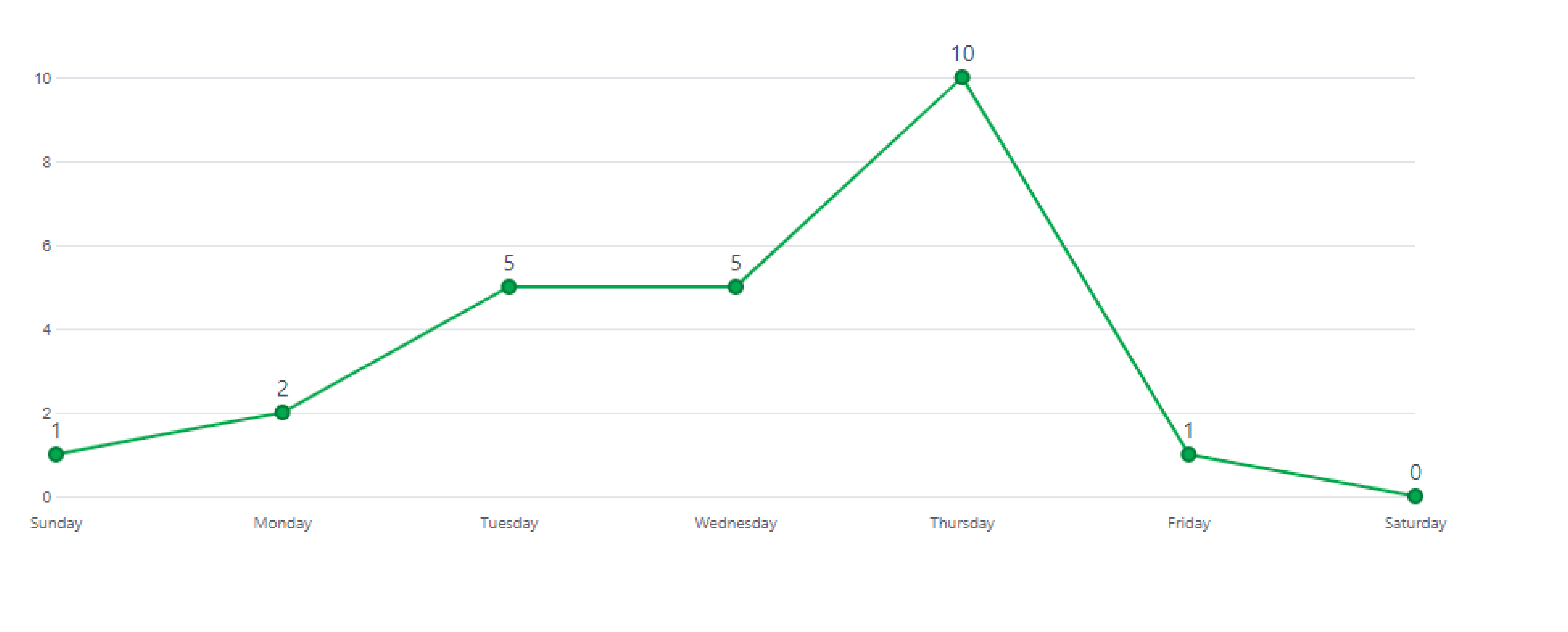
- Tracking and making changes to my codebase
- Should a catastrophic incident occur to my code, I would be able to revert changes
- Visualise the continuous changes made to my code from start to finish

Implement the feature branch model



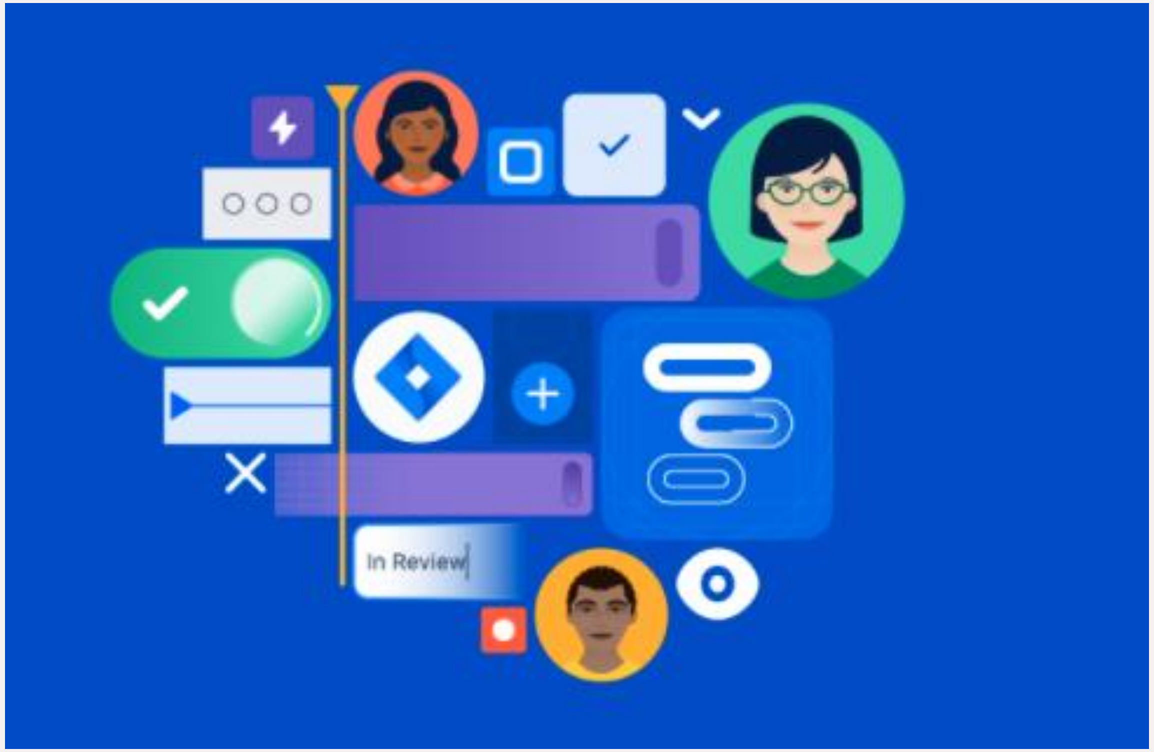
GitHub and Git Bash



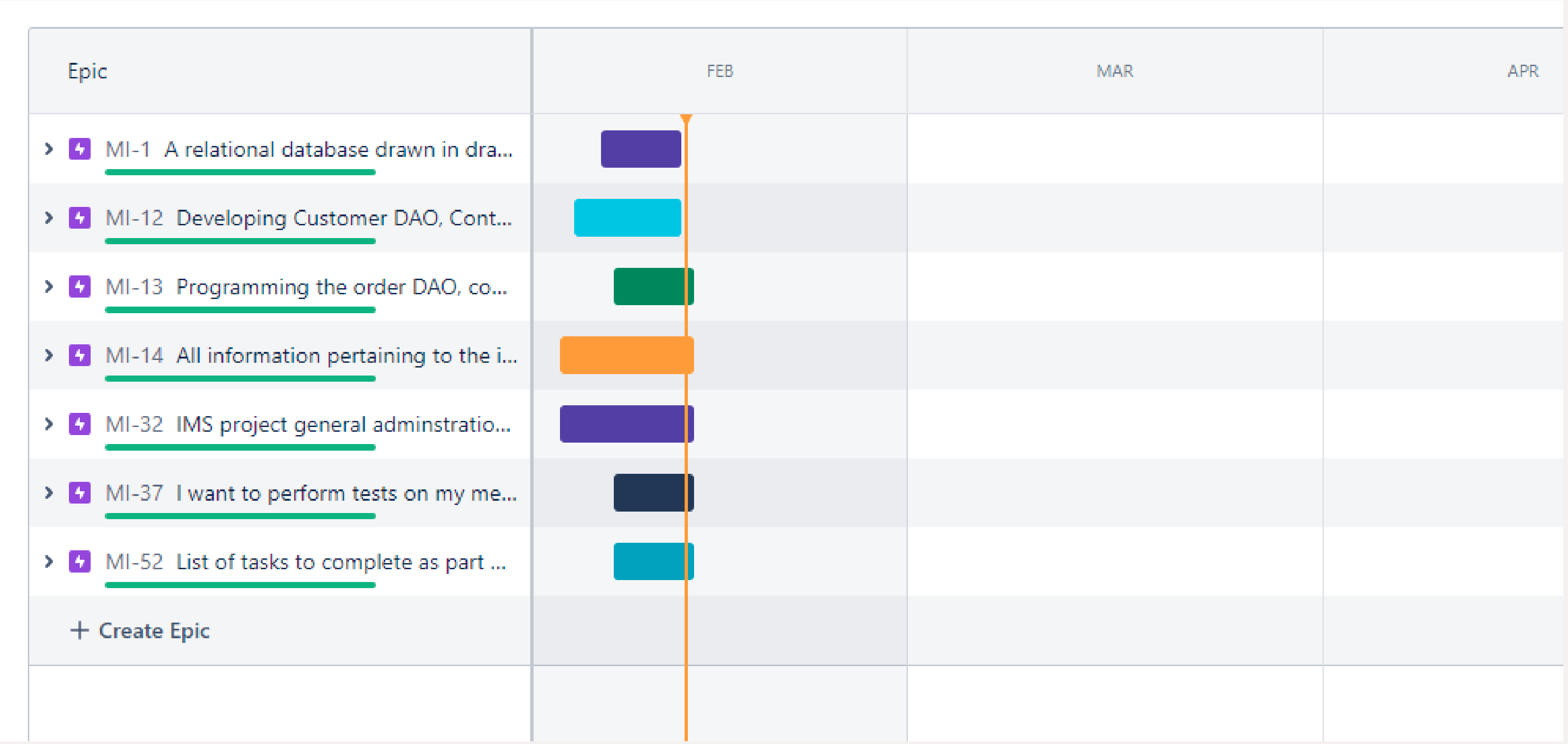


First step – Jiraboard

- Before any other task could be completed, I must ensure I have a thorough plan to execute this project.
- With reference to the specification, I planned a detailed, multifeatured storyboard which eventually consisted of 22 user stories, and 247 story points, to be completed in 2 sprints.
- The usage of sprints was integral to the planning of this project and allowed me to showcase some agile methodologies.



First step – Jiraboard – Roadmap



Epic | FEB | MAR | APR

MI-1 A relational database drawn in dra...

MI-12 Developing Customer DAO, Cont...

MI-13 Programming the order DAO, co...

MI-14 All information pertaining to the i...

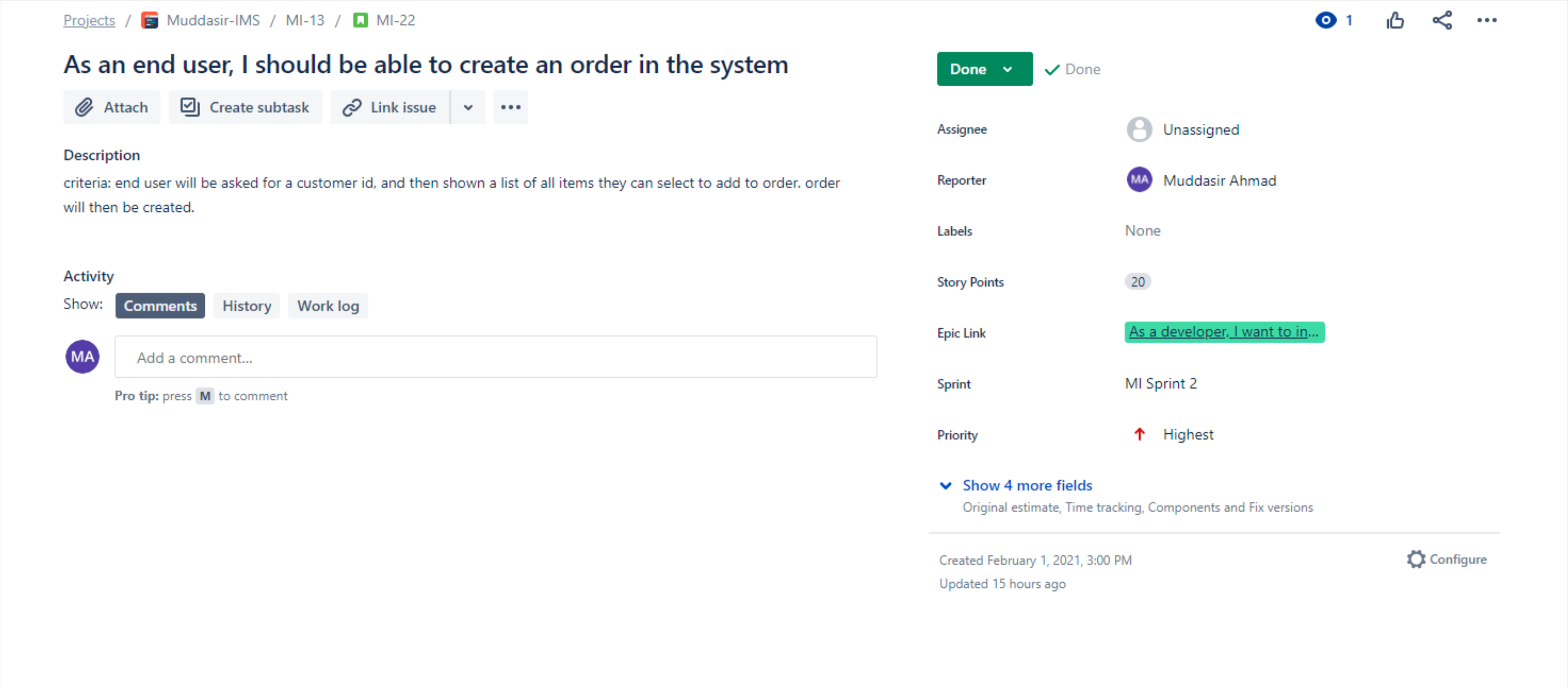
MI-32 IMS project general adminstratio...

MI-37 I want to perform tests on my me...

MI-52 List of tasks to complete as part ...

+ Create Epic

First step – Jiraboard



Projects / Muddasir-IMS / MI-13 / MI-22

As an end user, I should be able to create an order in the system

Attach
Create subtask
Link issue
Done
Done

Description

criteria: end user will be asked for a customer id, and then shown a list of all items they can select to add to order. order will then be created.

Activity

Show: Comments History Work log
MA
Add a comment...
Pro tip: press M to comment

Assignee: Unassigned

Reporter: Muddasir Ahmad

Labels: None

Story Points: 20

Epic Link: As a developer, I want to in...

Sprint: MI Sprint 2

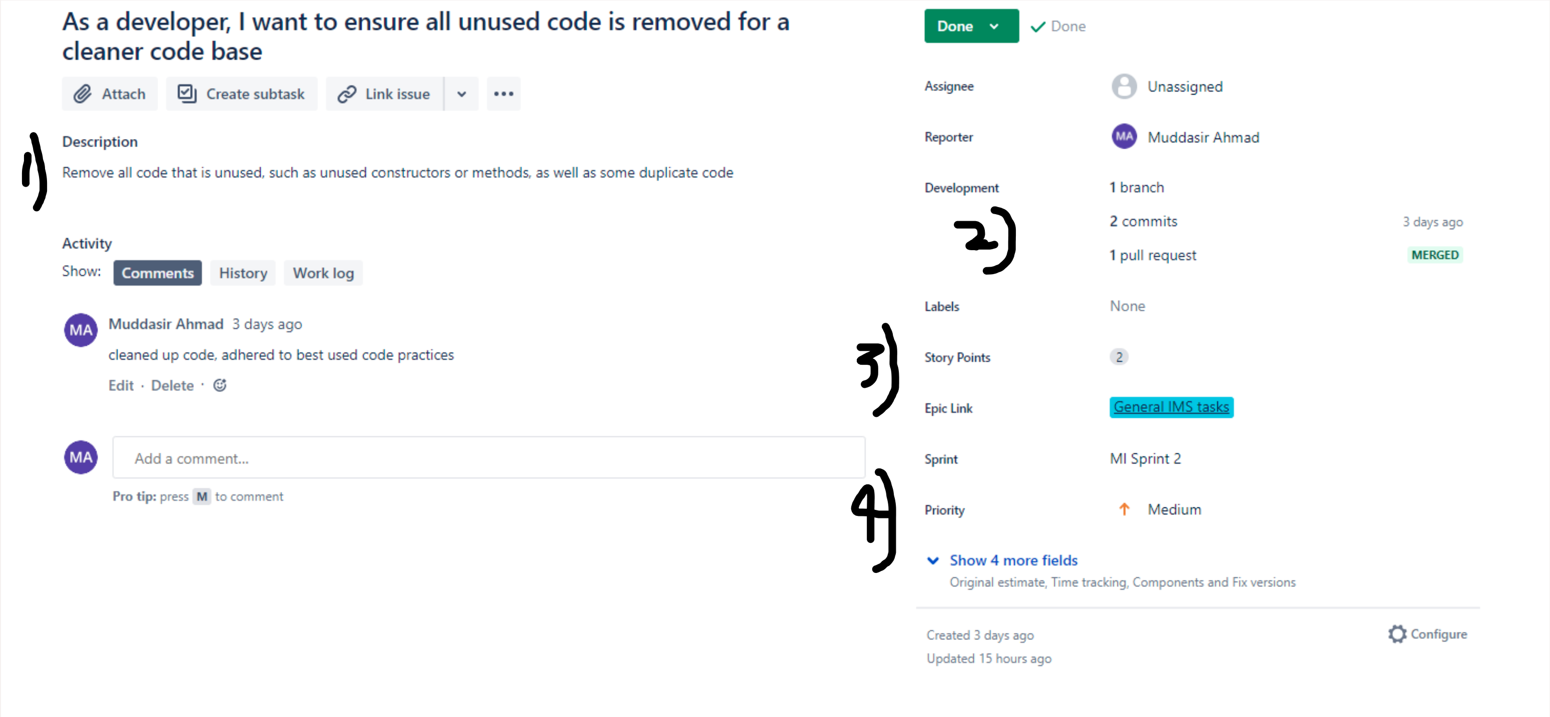
Priority: Highest

Show 4 more fields
Original estimate, Time tracking, Components and Fix versions
Configure

Created February 1, 2021, 3:00 PM
Updated 15 hours ago

User story example A

First step – Jiraboard



As a developer, I want to ensure all unused code is removed for a cleaner code base

Description

Remove all code that is unused, such as unused constructors or methods, as well as some duplicate code

Activity

Show: Comments History Work log

Muddasir Ahmad 3 days ago

cleaned up code, adhered to best used code practices

Edit · Delete ·

Add a comment...

Pro tip: press M to comment

Done ✓ Done

Assignee	Unassigned
Reporter	Muddasir Ahmad
Development	1 branch 2 commits 3 days ago 1 pull request MERGED
Labels	None
Story Points	2
Epic Link	General IMS tasks
Sprint	MI Sprint 2
Priority	↑ Medium

Show 4 more fields

Original estimate, Time tracking, Components and Fix versions

Created 3 days ago

Updated 15 hours ago

Configure

1) Acceptance criteria

2) Smart commits via Jira-GitHub integration

3) Story points assigned

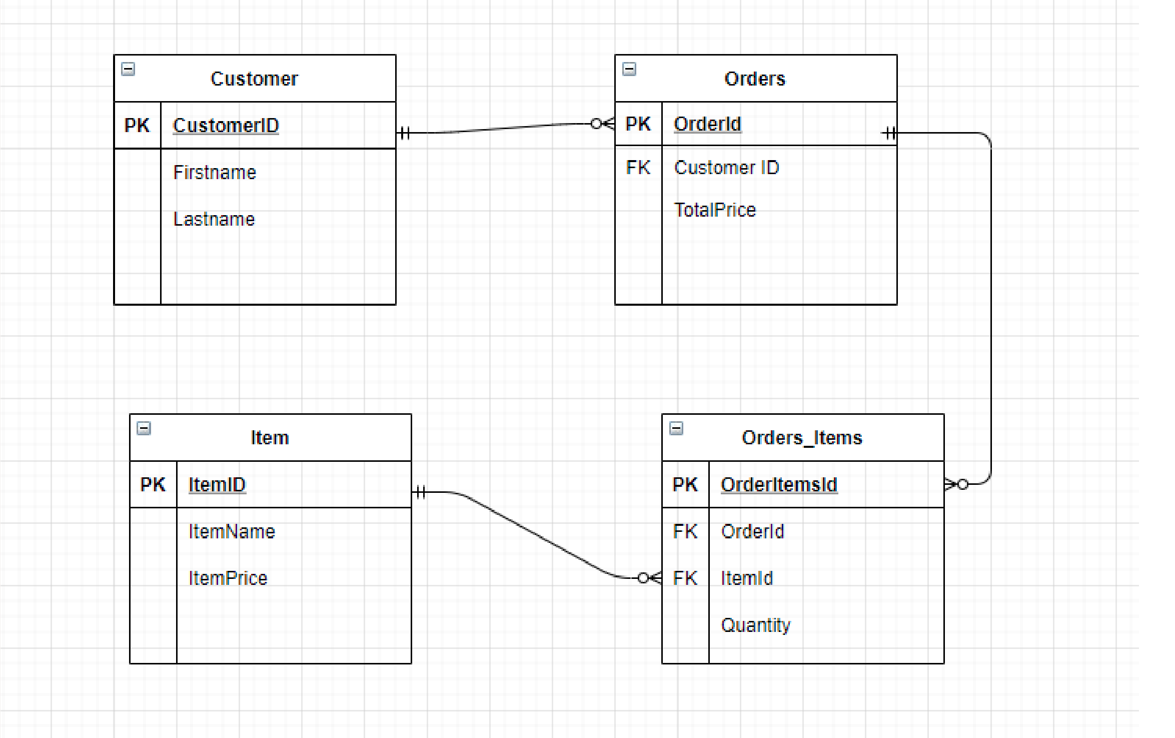
4) MoSCoW

User story example B

Entity Relationship Diagram

DRAWING AN ERD TO SHOWCASE THE RELATIONSHIPS BETWEEN THE DIFFERENT TABLES IN THE DATABASE

MOST IMPORTANTLY SHOW THE ORDER-ITEM INTERMEDIATE TABLE WITH FOREIGN KEYS AND ITS RELATIONSHIPS



Customer

PK	CustomerID
	Firstname
	Lastname

Orders

PK	OrderId
FK	Customer ID
	TotalPrice

Item

PK	ItemID
	ItemName
	ItemPrice

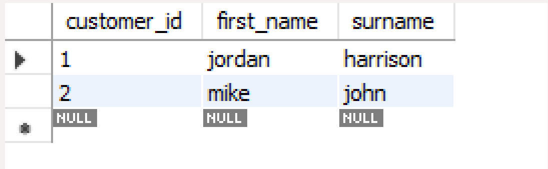
Orders_Items

PK	OrderItemsId
FK	OrderId
FK	ItemId
	Quantity

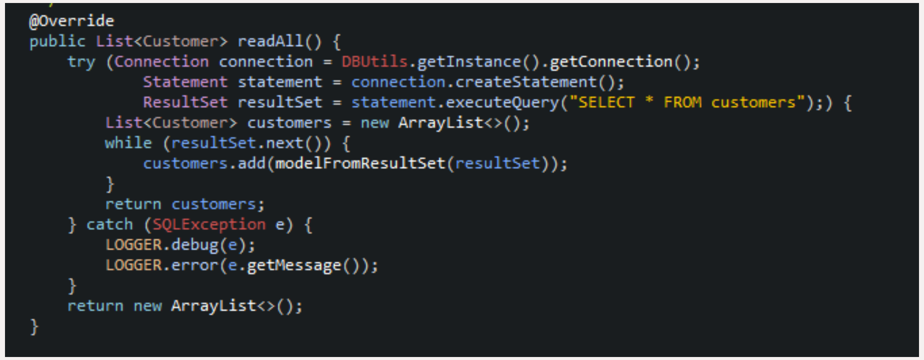
SQL workbench

- Workbench helped me visualise how my tables are intended to look like and made it easier to create SQL statements in my DAO methods.

- Below is an example of a SQL query which displays all data in the customer's table

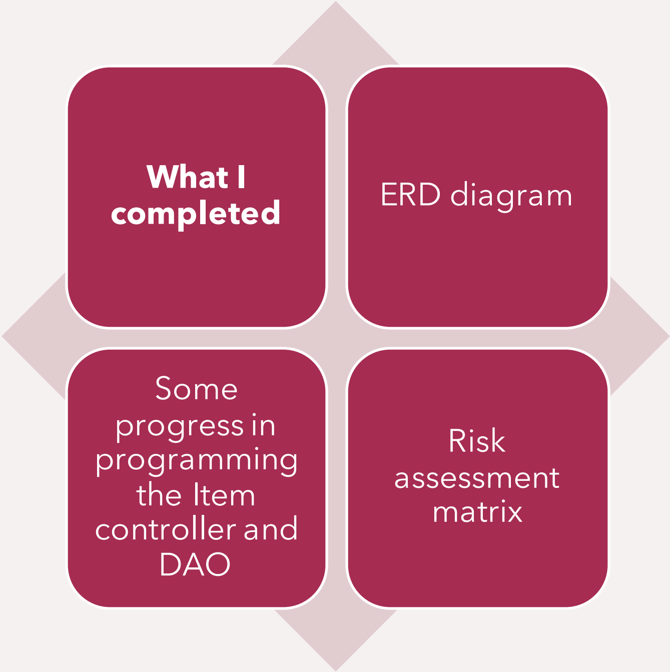


customer_id	first_name	surname
1	jordan	harrison
2	mike	john
NULL	NULL	NULL



```
@Override
public List<Customer> readAll() {
    try (Connection connection = DBUtils.getInstance().getConnection();
            Statement statement = connection.createStatement();
            ResultSet resultSet = statement.executeQuery("SELECT * FROM customers");) {
        List<Customer> customers = new ArrayList<>();
        while (resultSet.next()) {
            customers.add(modelFromResultSet(resultSet));
        }
        return customers;
    } catch (SQLException e) {
        LOGGER.debug(e);
        LOGGER.error(e.getMessage());
    }
    return new ArrayList<>();
}
```

Sprint 1 review



What I completed

ERD diagram

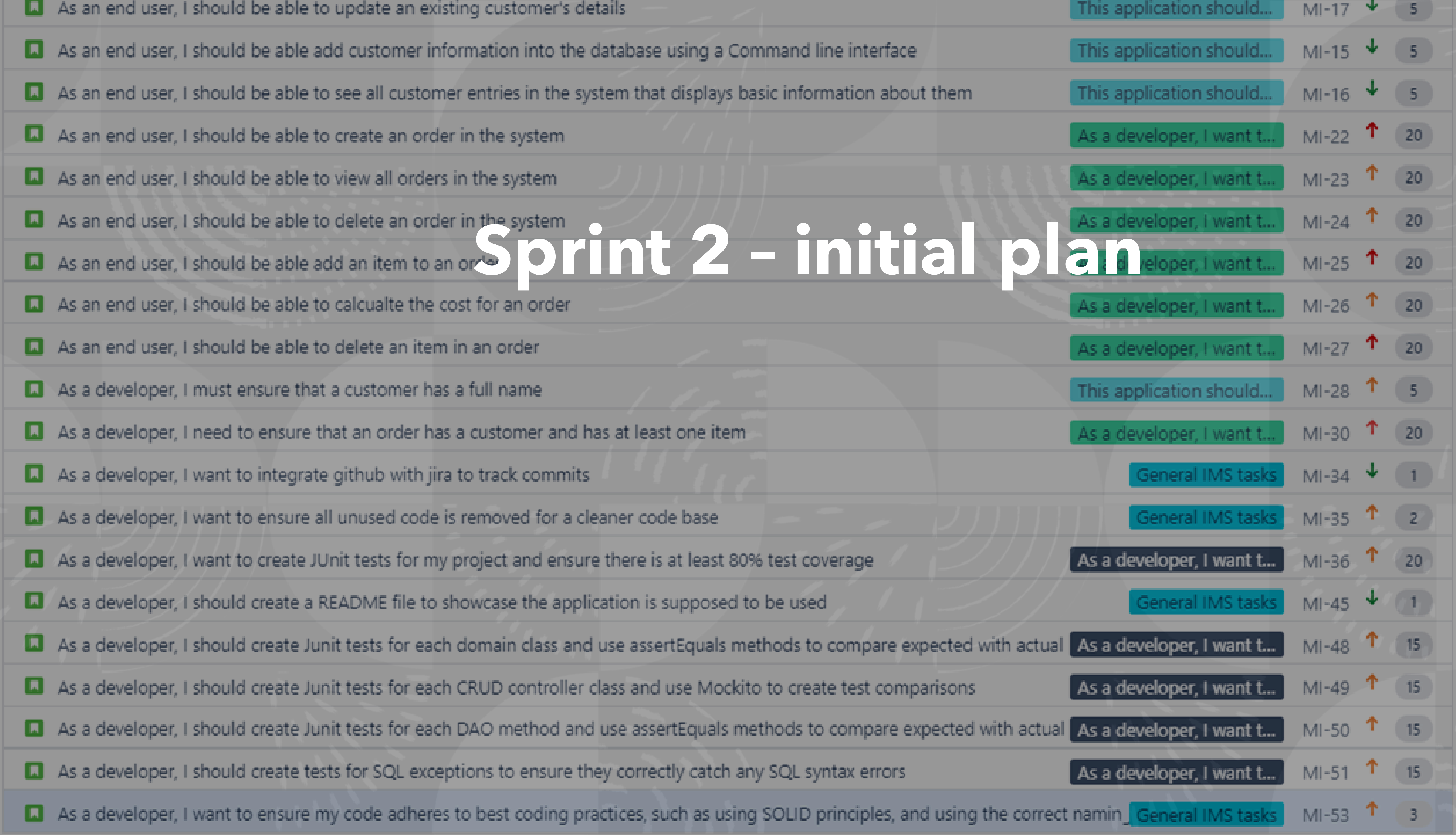
Some progress in programming the Item controller and DAO

Risk assessment matrix

What was carried forward to Sprint 2 from Sprint 1

- Continuation on item DAO and domain coding
- UML diagram

Sprint 2 – initial plan



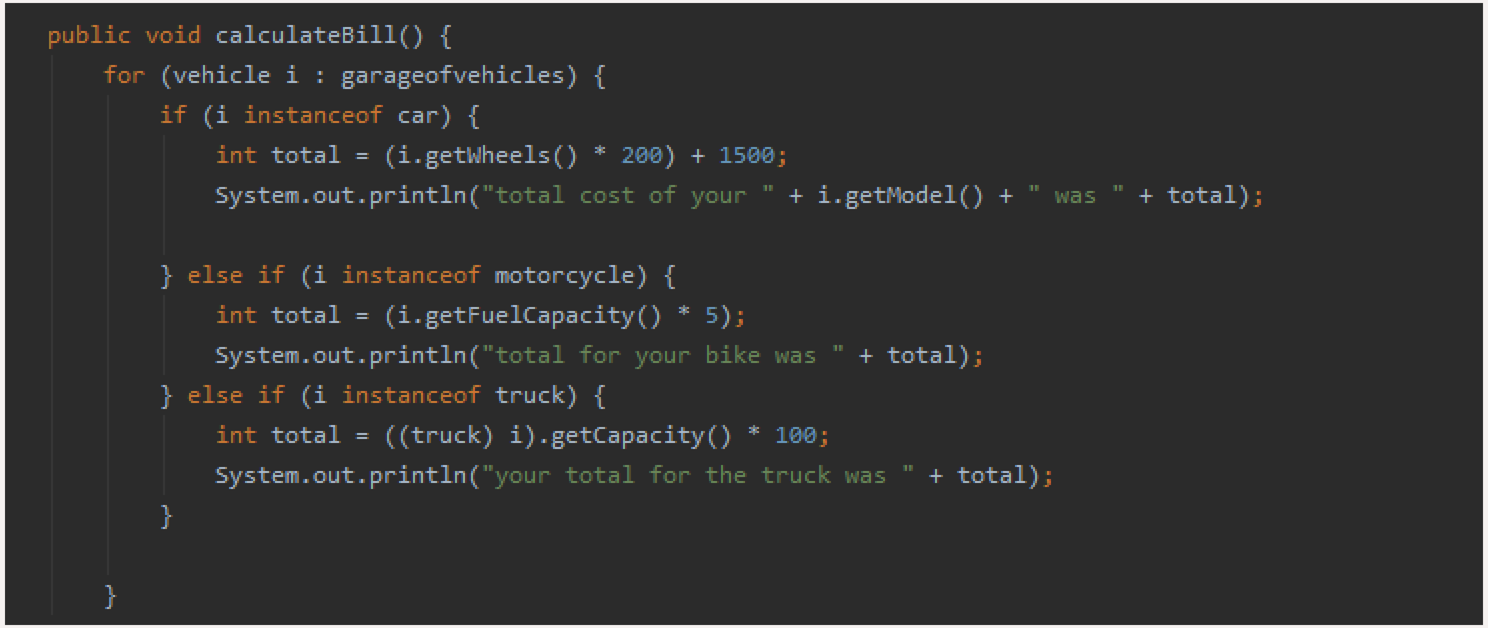
As an end user, I should be able to update an existing customer's details	This application should...	MI-17	↓	5
As an end user, I should be able add customer information into the database using a Command line interface	This application should...	MI-15	↓	5
As an end user, I should be able to see all customer entries in the system that displays basic information about them	This application should...	MI-16	↓	5
As an end user, I should be able to create an order in the system	As a developer, I want t...	MI-22	↑	20
As an end user, I should be able to view all orders in the system	As a developer, I want t...	MI-23	↑	20
As an end user, I should be able to delete an order in the system	As a developer, I want t...	MI-24	↑	20
As an end user, I should be able add an item to an or	a developer, I want t...	MI-25	↑	20
As an end user, I should be able to calcualte the cost for an order	As a developer, I want t...	MI-26	↑	20
As an end user, I should be able to delete an item in an order	As a developer, I want t...	MI-27	↑	20
As a developer, I must ensure that a customer has a full name	This application should...	MI-28	↑	5
As a developer, I need to ensure that an order has a customer and has at least one item	As a developer, I want t...	MI-30	↑	20
As a developer, I want to integrate github with jira to track commits	General IMS tasks	MI-34	↓	1
As a developer, I want to ensure all unused code is removed for a cleaner code base	General IMS tasks	MI-35	↑	2
As a developer, I want to create JUnit tests for my project and ensure there is at least 80% test coverage	As a developer, I want t...	MI-36	↑	20
As a developer, I should create a README file to showcase the application is supposed to be used	General IMS tasks	MI-45	↓	1
As a developer, I should create Junit tests for each domain class and use assertEquals methods to compare expected with actual	As a developer, I want t...	MI-48	↑	15
As a developer, I should create Junit tests for each CRUD controller class and use Mockito to create test comparisons	As a developer, I want t...	MI-49	↑	15
As a developer, I should create Junit tests for each DAO method and use assertEquals methods to compare expected with actual	As a developer, I want t...	MI-50	↑	15
As a developer, I should create tests for SQL exceptions to ensure they correctly catch any SQL syntax errors	As a developer, I want t...	MI-51	↑	15
As a developer, I want to ensure my code adheres to best coding practices, such as using SOLID principles, and using the correct namin_	General IMS tasks	MI-53	↑	3

Approaching Orders

- With items and customers completed, the only main aspect of the program left to code was orders
- Review key lessons learned from previous coding exercises
- ArrayLists<> and iterations…

Approaching Orders

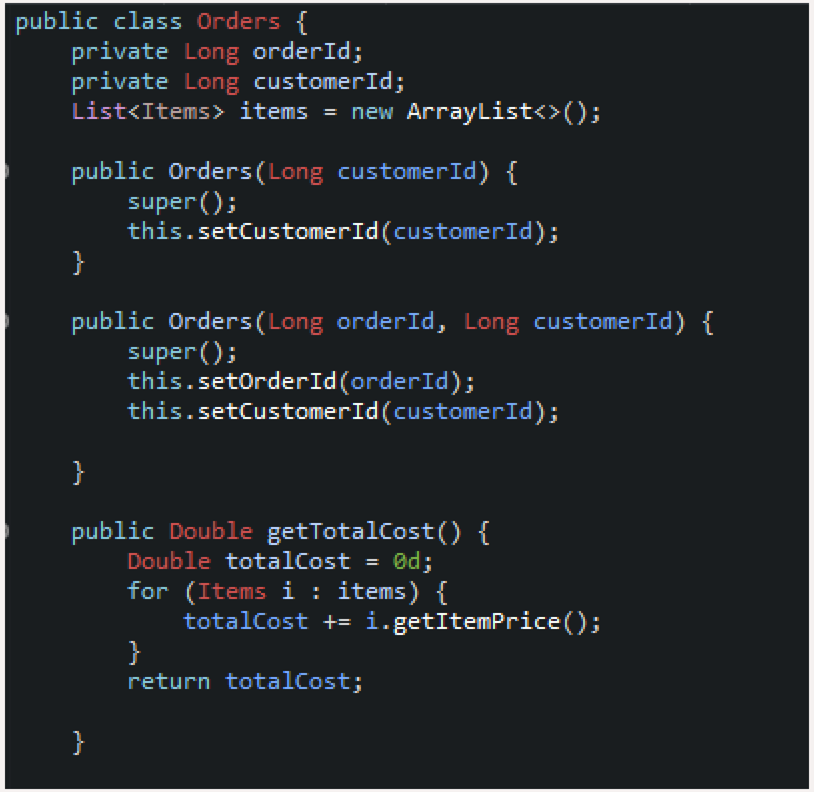
Garage coding exercise



```
public void calculateBill() {
    for (vehicle i : garageofvehicles) {
        if (i instanceof car) {
            int total = (i.getWheels() * 200) + 1500;
            System.out.println("total cost of your " + i.getModel() + " was " + total);

        } else if (i instanceof motorcycle) {
            int total = (i.getFuelCapacity() * 5);
            System.out.println("total for your bike was " + total);
        } else if (i instanceof truck) {
            int total = ((truck) i).getCapacity() * 100;
            System.out.println("your total for the truck was " + total);
        }

    }
```

Approaching Orders

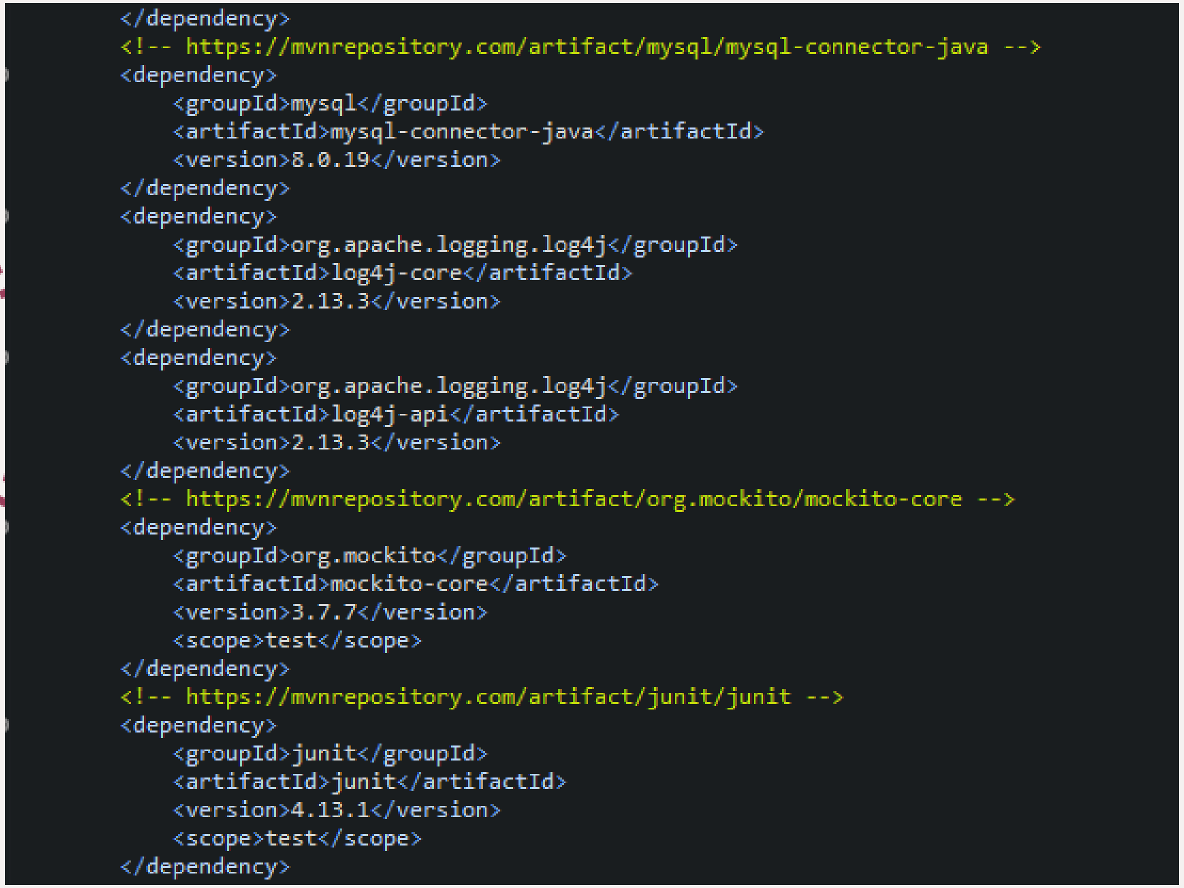
Order code snippet for calculating total cost



```
public class Orders {
    private Long orderId;
    private Long customerId;
    List<Items> items = new ArrayList<>();

    public Orders(Long customerId) {
        super();
        this.setCustomerId(customerId);
    }

    public Orders(Long orderId, Long customerId) {
        super();
        this.setOrderId(orderId);
        this.setCustomerId(customerId);

    }

    public Double getTotalCost() {
        Double totalCost = 0d;
        for (Items i : items) {
            totalCost += i.getItemPrice();
        }
        return totalCost;

    }
```

Dependencies

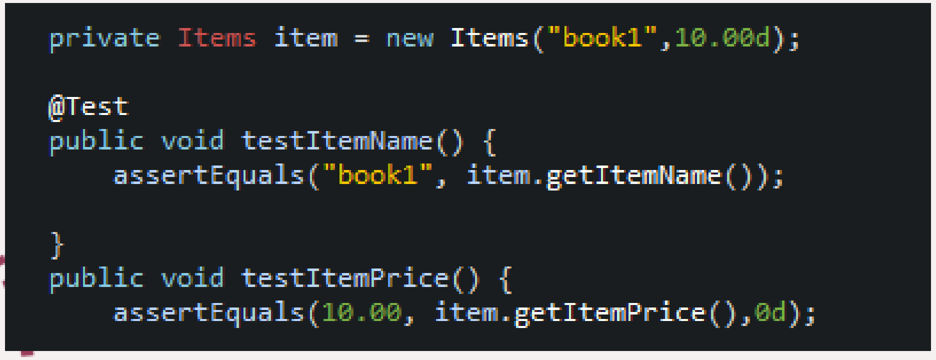
Pom file



```
        </dependency>
        <!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java -->
        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
            <version>8.0.19</version>
        </dependency>
        <dependency>
            <groupId>org.apache.logging.log4j</groupId>
            <artifactId>log4j-core</artifactId>
            <version>2.13.3</version>
        </dependency>
        <dependency>
            <groupId>org.apache.logging.log4j</groupId>
            <artifactId>log4j-api</artifactId>
            <version>2.13.3</version>
        </dependency>
        <!-- https://mvnrepository.com/artifact/org.mockito/mockito-core -->
        <dependency>
            <groupId>org.mockito</groupId>
            <artifactId>mockito-core</artifactId>
            <version>3.7.7</version>
            <scope>test</scope>
        </dependency>
        <!-- https://mvnrepository.com/artifact/junit/junit -->
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>4.13.1</version>
            <scope>test</scope>
        </dependency>
```

Dependencies - JUNT

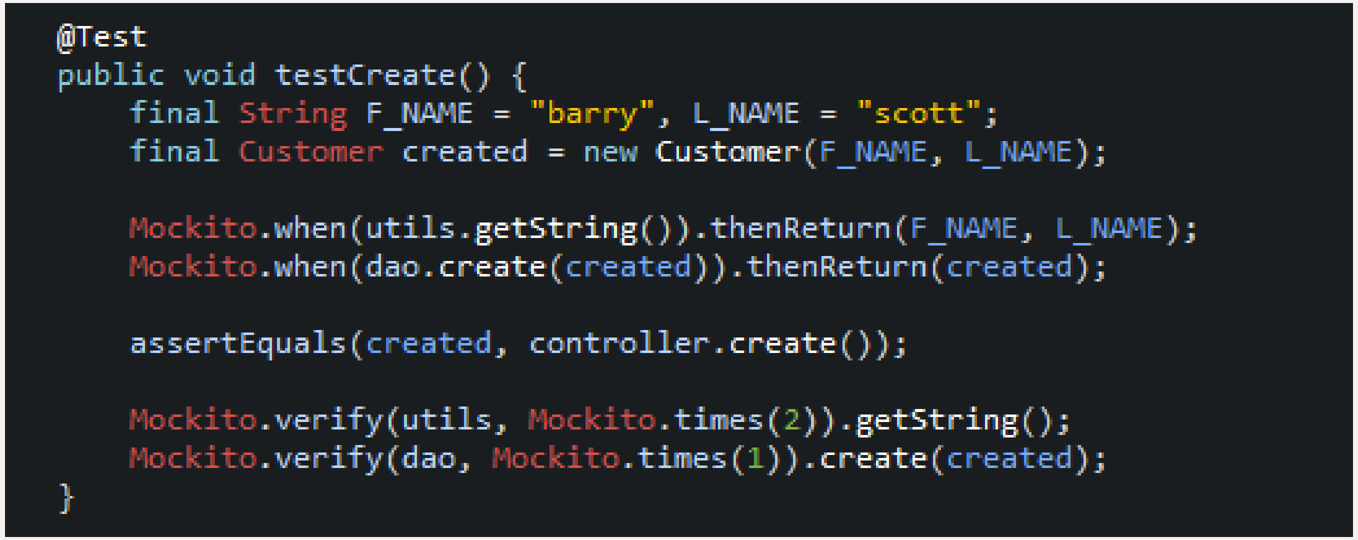
JUNIT test cases



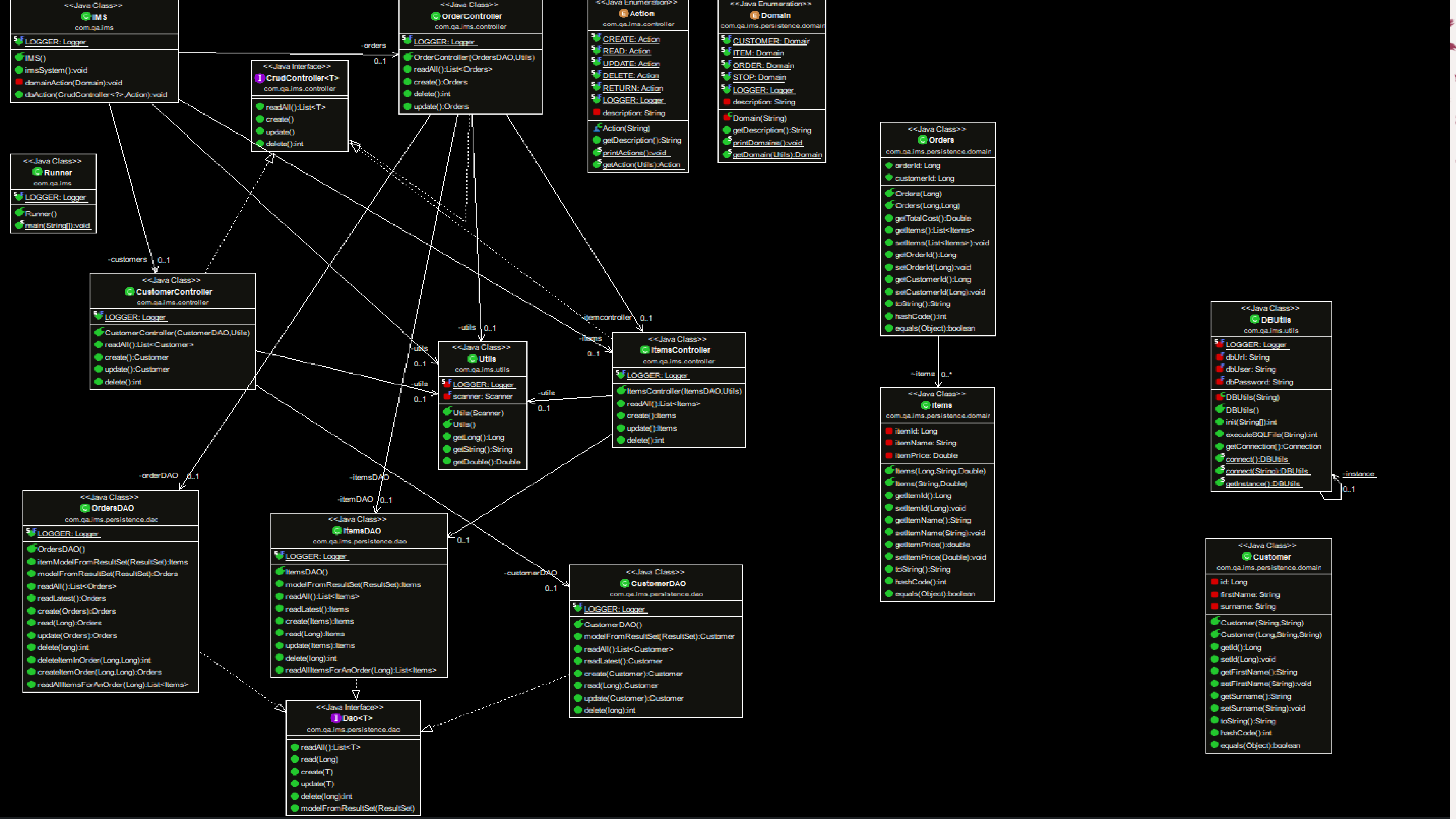
```
private Items item = new Items("book1",10.00d);

@Test
public void testItemName() {
    assertEquals("book1", item.getItemName());

}
public void testItemPrice() {
    assertEquals(10.00, item.getItemPrice(),0d);
```

Dependencies - Mockito

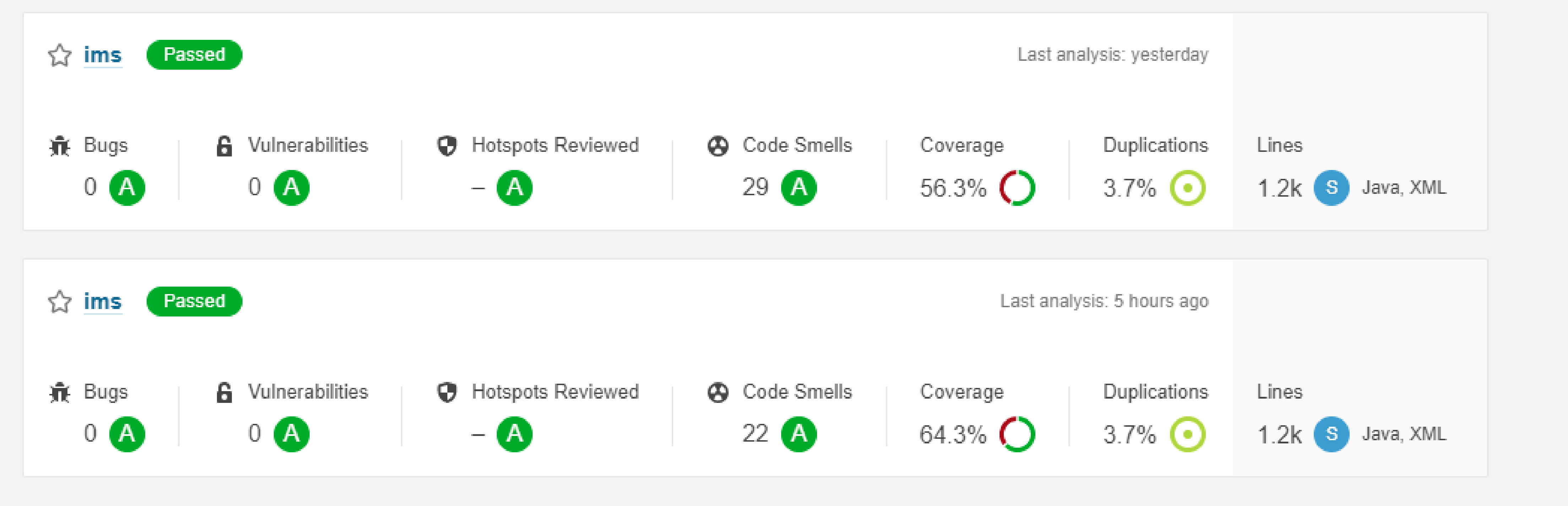
Mockito



```
@Test
public void testCreate() {
    final String F_NAME = "barry", L_NAME = "scott";
    final Customer created = new Customer(F_NAME, L_NAME);

    Mockito.when(utils.getString()).thenReturn(F_NAME, L_NAME);
    Mockito.when(dao.create(created)).thenReturn(created);

    assertEquals(created, controller.create());

    Mockito.verify(utils, Mockito.times(2)).getString();
    Mockito.verify(dao, Mockito.times(1)).create(created);
}
```


<<Java Class>>
IMS
com.qa.ims
LOGGER: Logger
IMS()
imsSystem():void
domainAction(Domain):void
doAction(CrudController<?>,Action):void
<<Java Class>>
OrderController
com.qa.ims.controller
LOGGER: Logger
OrderController(OrdersDAO,Utils)
readAll():List<Orders>
create():Orders
delete():int
update():Orders
<<Java Enumeration>>
Action
com.qa.ims.controller
CREATE: Action
READ: Action
UPDATE: Action
DELETE: Action
RETURN: Action
LOGGER: Logger
description: String
Action(String)
getDescription():String
printActions():void
getAction(Utils):Action
<<Java Enumeration>>
Domain
com.qa.ims.persistence.domain
CUSTOMER: Domain
ITEM: Domain
ORDER: Domain
STOP: Domain
LOGGER: Logger
description: String
Domain(String)
getDescription():String
printDomains():void
getDomain(Utils):Domain
<<Java Interface>>
CrudController<T>
com.qa.ims.controller
readAll():List<T>
create()
update()
delete():int
-orders
0..1
<<Java Class>>
Runner
com.qa.ims
LOGGER: Logger
Runner()
main(String[]):void
<<Java Class>>
Orders
com.qa.ims.persistence.domain
orderId: Long
customerId: Long
Orders(Long)
Orders(Long,Long)
getTotalCost():Double
getItems():List<Items>
setItems(List<Items>):void
getOrderId():Long
setOrderId(Long):void
getCustomerId():Long
setCustomerId(Long):void
toString():String
hashCode():int
equals(Object):boolean
-customers
0..1
<<Java Class>>
CustomerController
com.qa.ims.controller
LOGGER: Logger
CustomerController(CustomerDAO,Utils)
readAll():List<Customer>
create():Customer
update():Customer
delete():int
-utils
0..1
-itemcontroller
0..1
-items
0..1
<<Java Class>>
Utils
com.qa.ims.utils
LOGGER: Logger
scanner: Scanner
Utils(Scanner)
Utils()
getLong():Long
getString():String
getDouble():Double
-utils
0..1
-utils
0..1
-utils
0..1
<<Java Class>>
ItemsController
com.qa.ims.controller
LOGGER: Logger
ItemsController(ItemsDAO,Utils)
readAll():List<Items>
create():Items
update():Items
delete():int
<<Java Class>>
DBUtils
com.qa.ims.utils
LOGGER: Logger
dbUrl: String
dbUser: String
dbPassword: String
DBUtils(String)
DBUtils()
init(String[]):int
executeSQLFile(String):int
getConnection():Connection
connect():DBUtils
connect(String):DBUtils
getInstance():DBUtils
-instance
0..1
-items
0..*
<<Java Class>>
Items
com.qa.ims.persistence.domain
itemId: Long
itemName: String
itemPrice: Double
Items(Long,String,Double)
Items(String,Double)
getItemId():Long
setItemId(Long):void
getItemName():String
setItemName(String):void
getItemPrice():double
setItemPrice(Double):void
toString():String
hashCode():int
equals(Object):boolean
-orderDAO
0..1
-itemsDAO
-itemDAO
0..1
0..1
-customerDAO
0..1
<<Java Class>>
OrdersDAO
com.qa.ims.persistence.dao
LOGGER: Logger
OrdersDAO()
itemModelFromResultSet(ResultSet):Items
modelFromResultSet(ResultSet):Orders
readAll():List<Orders>
readLatest():Orders
create(Orders):Orders
read(Long):Orders
update(Orders):Orders
delete(long):int
deleteItemInOrder(Long,Long):int
createItemOrder(Long,Long):Orders
readAllItemsForAnOrder(Long):List<Items>
<<Java Class>>
ItemsDAO
com.qa.ims.persistence.dao
LOGGER: Logger
ItemsDAO()
modelFromResultSet(ResultSet):Items
readAll():List<Items>
readLatest():Items
create(Items):Items
read(Long):Items
update(Items):Items
delete(long):int
readAllItemsForAnOrder(Long):List<Items>
<<Java Class>>
CustomerDAO
com.qa.ims.persistence.dao
LOGGER: Logger
CustomerDAO()
modelFromResultSet(ResultSet):Customer
readAll():List<Customer>
readLatest():Customer
create(Customer):Customer
read(Long):Customer
update(Customer):Customer
delete(long):int
<<Java Class>>
Customer
com.qa.ims.persistence.domain
id: Long
firstName: String
surname: String
Customer(String,String)
Customer(Long,String,String)
getId():Long
setId(Long):void
getFirstName():String
setFirstName(String):void
getSurname():String
setSurname(String):void
toString():String
hashCode():int
equals(Object):boolean
<<Java Interface>>
Dao<T>
com.qa.ims.persistence.dao
readAll():List<T>
read(Long)
create(T)
update(T)
delete(long):int
modelFromResultSet(ResultSet)

SonarQube – Test Coverage



☆ ims Passed

Last analysis: yesterday

Bugs	Vulnerabilities	Hotspots Reviewed	Code Smells	Coverage	Duplications	Lines
0 A	0 A	– A	29 A	56.3%	3.7%	1.2k S Java, XML

☆ ims Passed

Last analysis: 5 hours ago

Bugs	Vulnerabilities	Hotspots Reviewed	Code Smells	Coverage	Duplications	Lines
0 A	0 A	– A	22 A	64.3%	3.7%	1.2k S Java, XML

Sprint review

What was completed

- Detailed storyboard with story point estimations, acceptance criteria and prioritisation
- IMS project completed with full CRUD functionality (with one exception)
- Continuous integration of project and creation of feature branches when appropriate, to be merged into develop branch once the segment of code was completed
- Adhered to SOLID and OOP principles with the help of SonarQube
- All necessary documents in the documentation folder
- A readily deployable fat-JAR file that can be used in any terminal
- Majority of code tested with JUNIT and Mockito tests at 65% coverage

Sprint review

What was left behind

- One aspect of CRUD functionality was left behind (updating orders)
- Increase in code coverage
- More unit tests to cover all methods
- Additional, optional, features I wanted to include in the IMS project

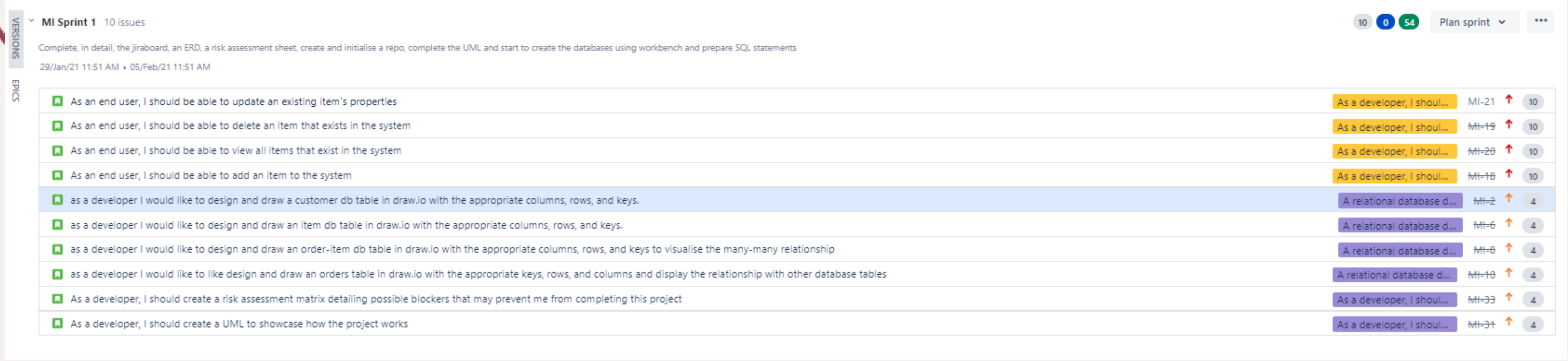
Such as offering discount price to existing customer, or a relative of customer

Sprint retrospective

The usage of all technologies learned, both new and old, has reinfornced my knowledge and honed my skills exceptionally

Continuously challenged in many different ways, but persevering through adversity to overcome these problems

Sprint retrospective

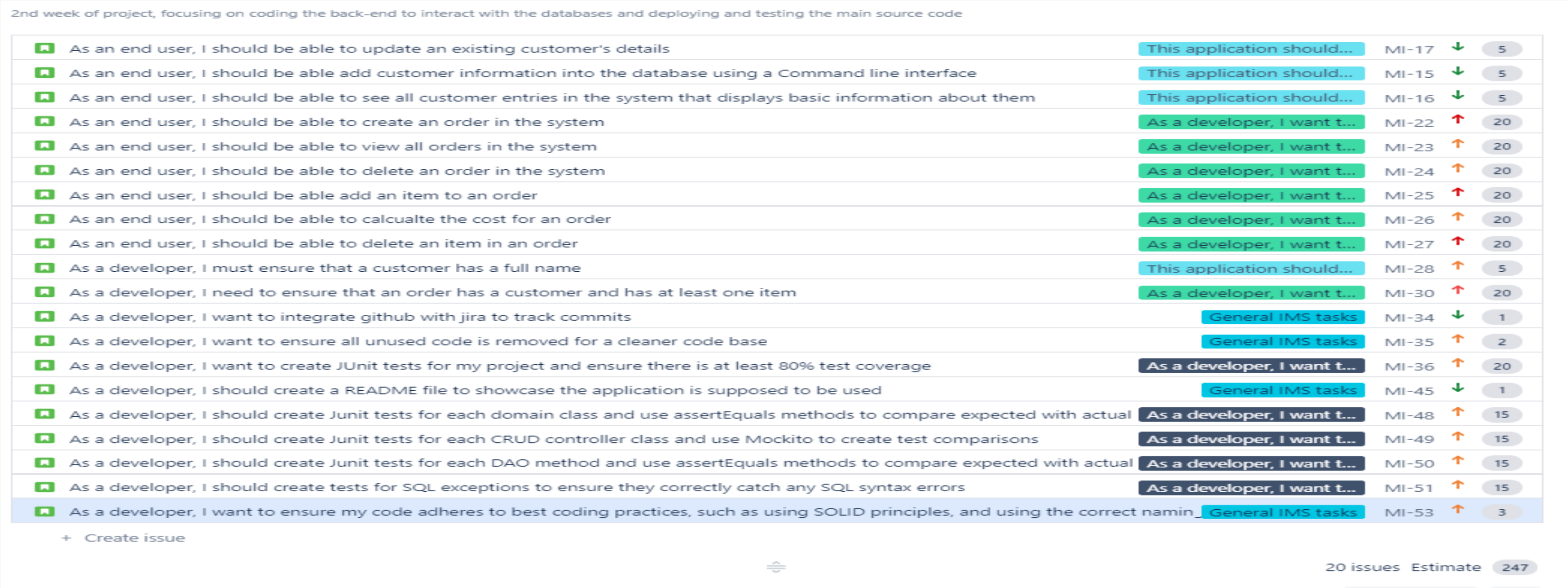


MI Sprint 1 10 issues

Complete, in detail, the jiraboard, an ERD, a risk assessment sheet, create and initialise a repo, complete the UML and start to create the databases using workbench and prepare SQL statements

29/Jan/21 11:51 AM • 05/Feb/21 11:51 AM

As an end user, I should be able to update an existing item's properties	As a developer, I shoul...	MI-21	↑	10
As an end user, I should be able to delete an item that exists in the system	As a developer, I shoul...	~~MI-19~~	↑	10
As an end user, I should be able to view all items that exist in the system	As a developer, I shoul...	~~MI-20~~	↑	10
As an end user, I should be able to add an item to the system	As a developer, I shoul...	~~MI-18~~	↑	10
as a developer I would like to design and draw a customer db table in draw.io with the appropriate columns, rows, and keys.	A relational database d...	~~MI-2~~	↑	4
as a developer I would like to design and draw an item db table in draw.io with the appropriate columns, rows, and keys.	A relational database d...	~~MI-6~~	↑	4
as a developer I would like to design and draw an order-item db table in draw.io with the appropriate columns, rows, and keys to visualise the many-many relationship	A relational database d...	~~MI-8~~	↑	4
as a developer I would like to like design and draw an orders table in draw.io with the appropriate keys, rows, and columns and display the relationship with other database tables	A relational database d...	~~MI-10~~	↑	4
As a developer, I should create a risk assessment matrix detailing possible blockers that may prevent me from completing this project	As a developer, I shoul...	~~MI-33~~	↑	4
As a developer, I should create a UML to showcase how the project works	As a developer, I shoul...	~~MI-31~~	↑	4



2nd week of project, focusing on coding the back-end to interact with the databases and deploying and testing the main source code

As an end user, I should be able to update an existing customer's details	This application should...	MI-17	↓	5
As an end user, I should be able add customer information into the database using a Command line interface	This application should...	MI-15	↓	5
As an end user, I should be able to see all customer entries in the system that displays basic information about them	This application should...	MI-16	↓	5
As an end user, I should be able to create an order in the system	As a developer, I want t...	MI-22	↑	20
As an end user, I should be able to view all orders in the system	As a developer, I want t...	MI-23	↑	20
As an end user, I should be able to delete an order in the system	As a developer, I want t...	MI-24	↑	20
As an end user, I should be able add an item to an order	As a developer, I want t...	MI-25	↑	20
As an end user, I should be able to calcualte the cost for an order	As a developer, I want t...	MI-26	↑	20
As an end user, I should be able to delete an item in an order	As a developer, I want t...	MI-27	↑	20
As a developer, I must ensure that a customer has a full name	This application should...	MI-28	↑	5
As a developer, I need to ensure that an order has a customer and has at least one item	As a developer, I want t...	MI-30	↑	20
As a developer, I want to integrate github with jira to track commits	General IMS tasks	MI-34	↓	1
As a developer, I want to ensure all unused code is removed for a cleaner code base	General IMS tasks	MI-35	↑	2
As a developer, I want to create JUnit tests for my project and ensure there is at least 80% test coverage	As a developer, I want t...	MI-36	↑	20
As a developer, I should create a README file to showcase the application is supposed to be used	General IMS tasks	MI-45	↓	1
As a developer, I should create Junit tests for each domain class and use assertEquals methods to compare expected with actual	As a developer, I want t...	MI-48	↑	15
As a developer, I should create Junit tests for each CRUD controller class and use Mockito to create test comparisons	As a developer, I want t...	MI-49	↑	15
As a developer, I should create Junit tests for each DAO method and use assertEquals methods to compare expected with actual	As a developer, I want t...	MI-50	↑	15
As a developer, I should create tests for SQL exceptions to ensure they correctly catch any SQL syntax errors	As a developer, I want t...	MI-51	↑	15
As a developer, I want to ensure my code adheres to best coding practices, such as using SOLID principles, and using the correct namin	General IMS tasks	MI-53	↑	3

+ Create issue

20 issues Estimate 247

Questions?