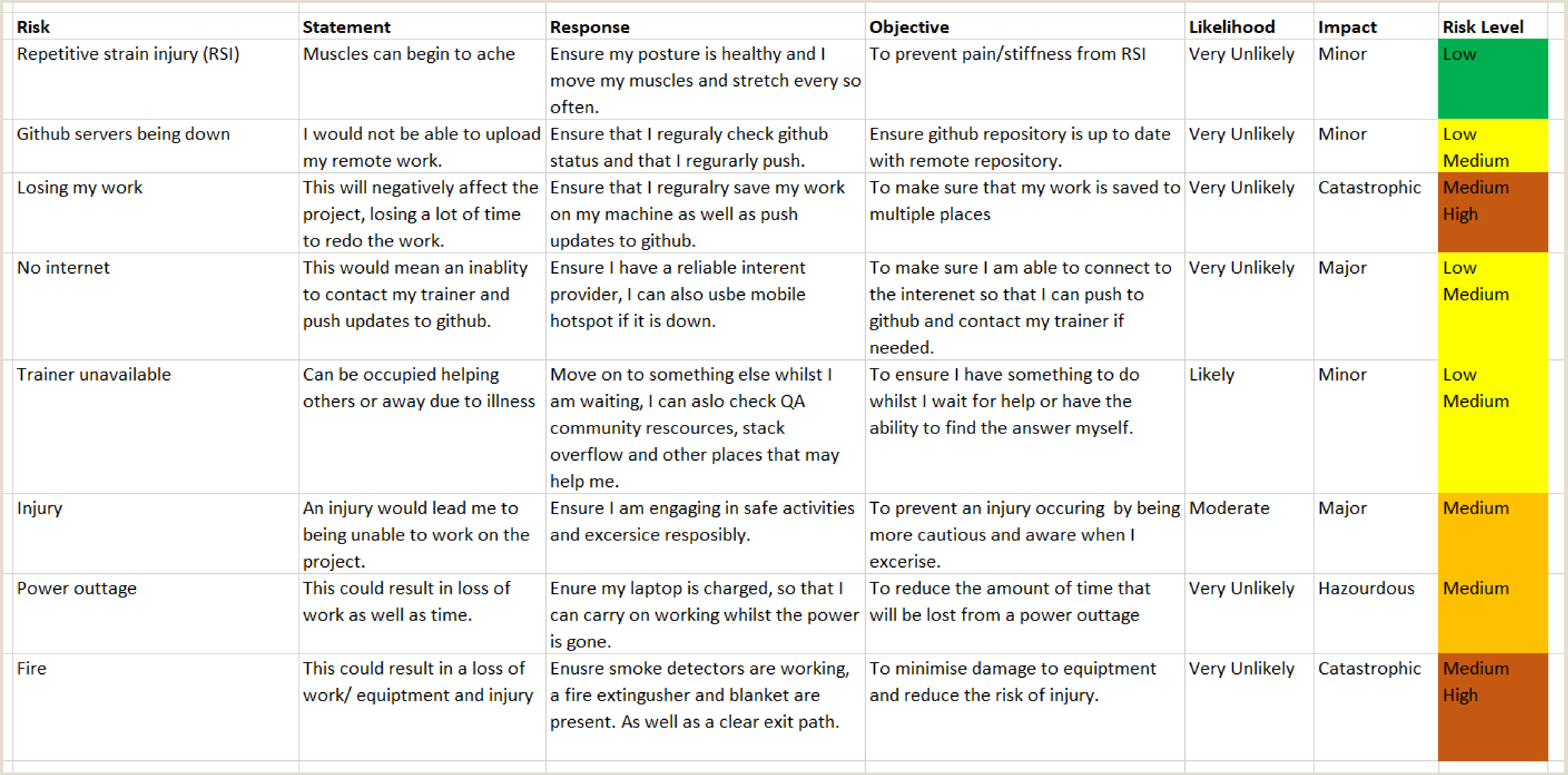
INVENTORY MANAGEMENT SYSTEM

Saad Ahmed
22AprEnable_1

Introduction

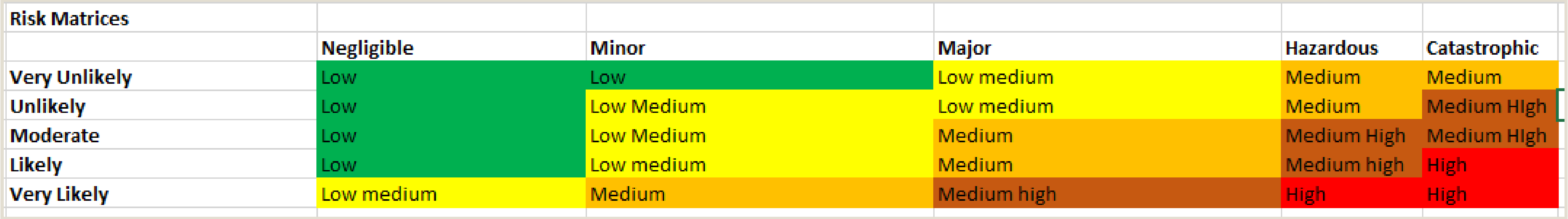
- Create an IMS with CRUD functionality.
- User can interact through a CLI.
- Testing
- Use tools learned over past 5 weeks:
  - Git
  - Jira
  - SQL
  - Java
  - Maven

Risk



Risk	Statement	Response	Objective	Likelihood	Impact	Risk Level
Repetitive strain injury (RSI)	Muscles can begin to ache	Ensure my posture is healthy and I move my muscles and stretch every so often.	To prevent pain/stiffness from RSI	Very Unlikely	Minor	Low
Github servers being down	I would not be able to upload my remote work.	Ensure that I reguraly check github status and that I regurarly push.	Ensure github repository is up to date with remote repository.	Very Unlikely	Minor	Low Medium
Losing my work	This will negatively affect the project, losing a lot of time to redo the work.	Ensure that I regularly save my work on my machine as well as push updates to github.	To make sure that my work is saved to multiple places	Very Unlikely	Catastrophic	Medium High
No internet	This would mean an inablity to contact my trainer and push updates to github.	Ensure I have a reliable interent provider, I can also usbe mobile hotspot if it is down.	To make sure I am able to connect to the interenet so that I can push to github and contact my trainer if needed.	Very Unlikely	Major	Low Medium
Trainer unavailable	Can be occupied helping others or away due to illness	Move on to something else whilst I am waiting, I can aslo check QA community rescources, stack overflow and other places that may help me.	To ensure I have something to do whilst I wait for help or have the ability to find the answer myself.	Likely	Minor	Low Medium
Injury	An injury would lead me to being unable to work on the project.	Ensure I am engaging in safe activities and excersice resposibly.	To prevent an injury occuring by being more cautious and aware when I excerise.	Moderate	Major	Medium
Power outtage	This could result in loss of work as well as time.	Enure my laptop is charged, so that I can carry on working whilst the power is gone.	To reduce the amount of time that will be lost from a power outtage	Very Unlikely	Hazourdous	Medium
Fire	This could result in a loss of work/ equiptment and injury	Enusre smoke detectors are working, a fire extingusher and blanket are present. As well as a clear exit path.	To minimise damage to equiptment and reduce the risk of injury.	Very Unlikely	Catastrophic	Medium High

Risk Matrices



Risk Matrices

	Negligible	**Minor**	**Major**	**Hazardous**	**Catastrophic**
Very Unlikely	Low	Low	Low medium	Medium	Medium
Unlikely	Low	Low Medium	Low medium	Medium	Medium High
Moderate	Low	Low Medium	Medium	Medium High	Medium High
Likely	Low	Low medium	Medium	Medium high	High
Very Likely	Low medium	Medium	Medium high	High	High

MoSCoW

Must have:

- A working application with a Command-Line Interface the end user can interact with.
- The ability to add/view/update/delete customers and items to the system.
- The ability to create/view/ delete orders in the system.
- The ability to add/delete an item to an order.

Should have:

- A customer should have a first name and last name
- Items should have a name and value
- Orders should have an order id.
- Each order should be connected to a customer and contains items
- Should have at least 80% test coverage.
- The ability to calculate a cost for an order.

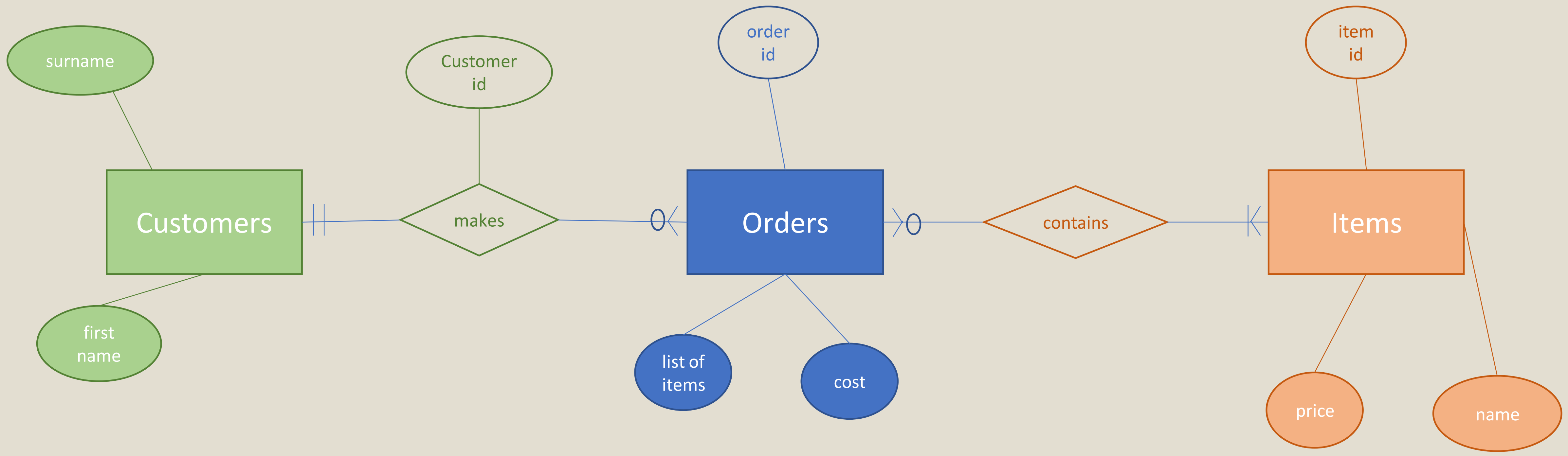
Could have:

- Items could have a description and number of stock left.
- Customers could have an email/ phone number/ address
- Orders could have a date placed and status

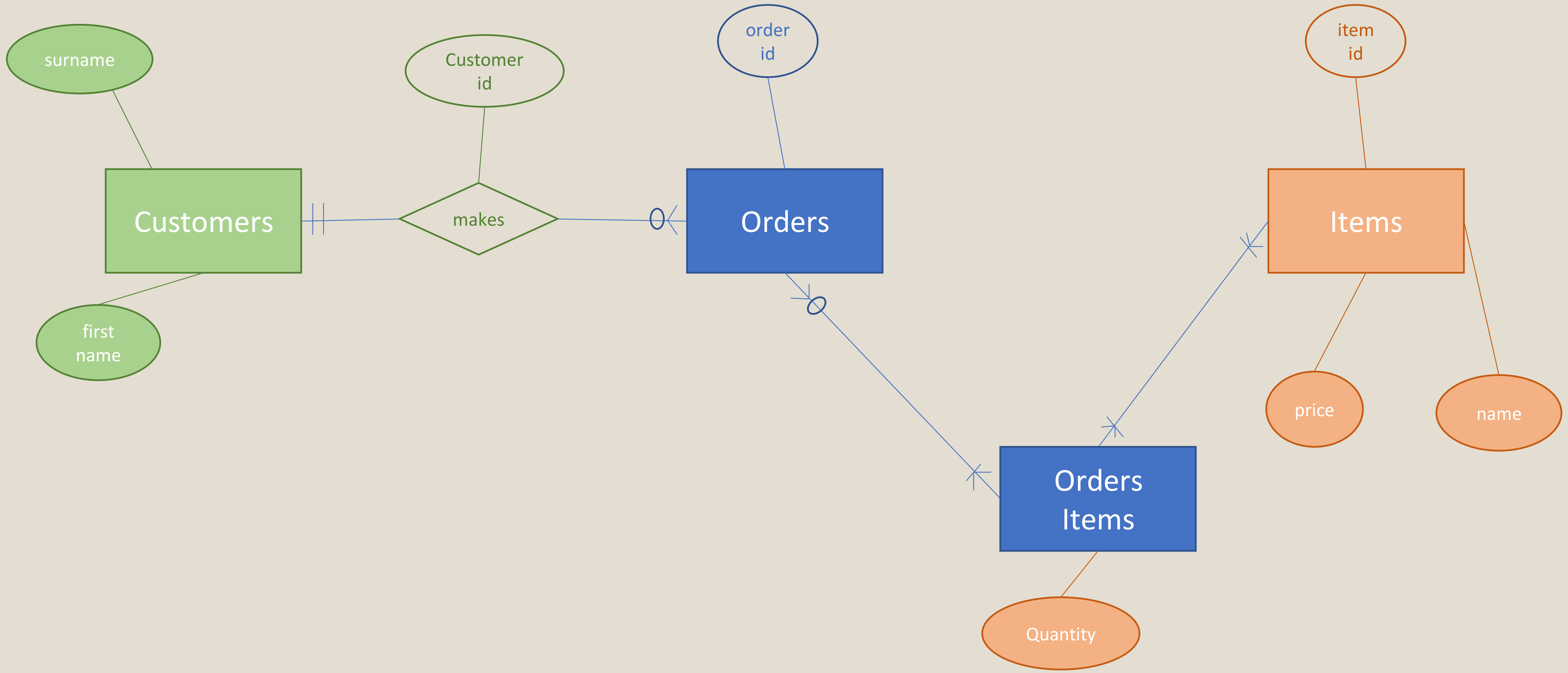
Won't have:

- Ads
- A login portal

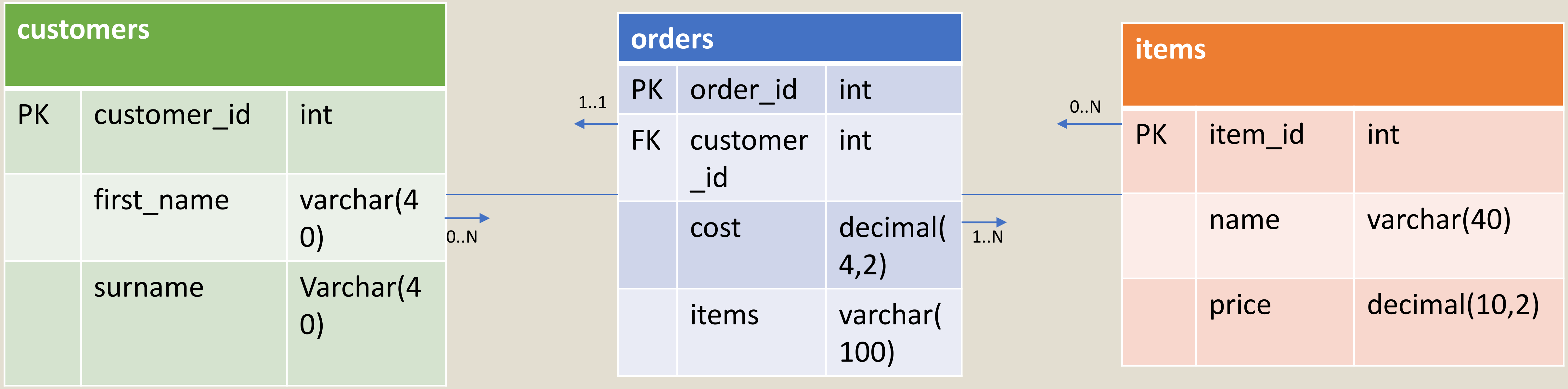
Entity Relationship Diagram



Entity Relationship Diagram - 2



UML

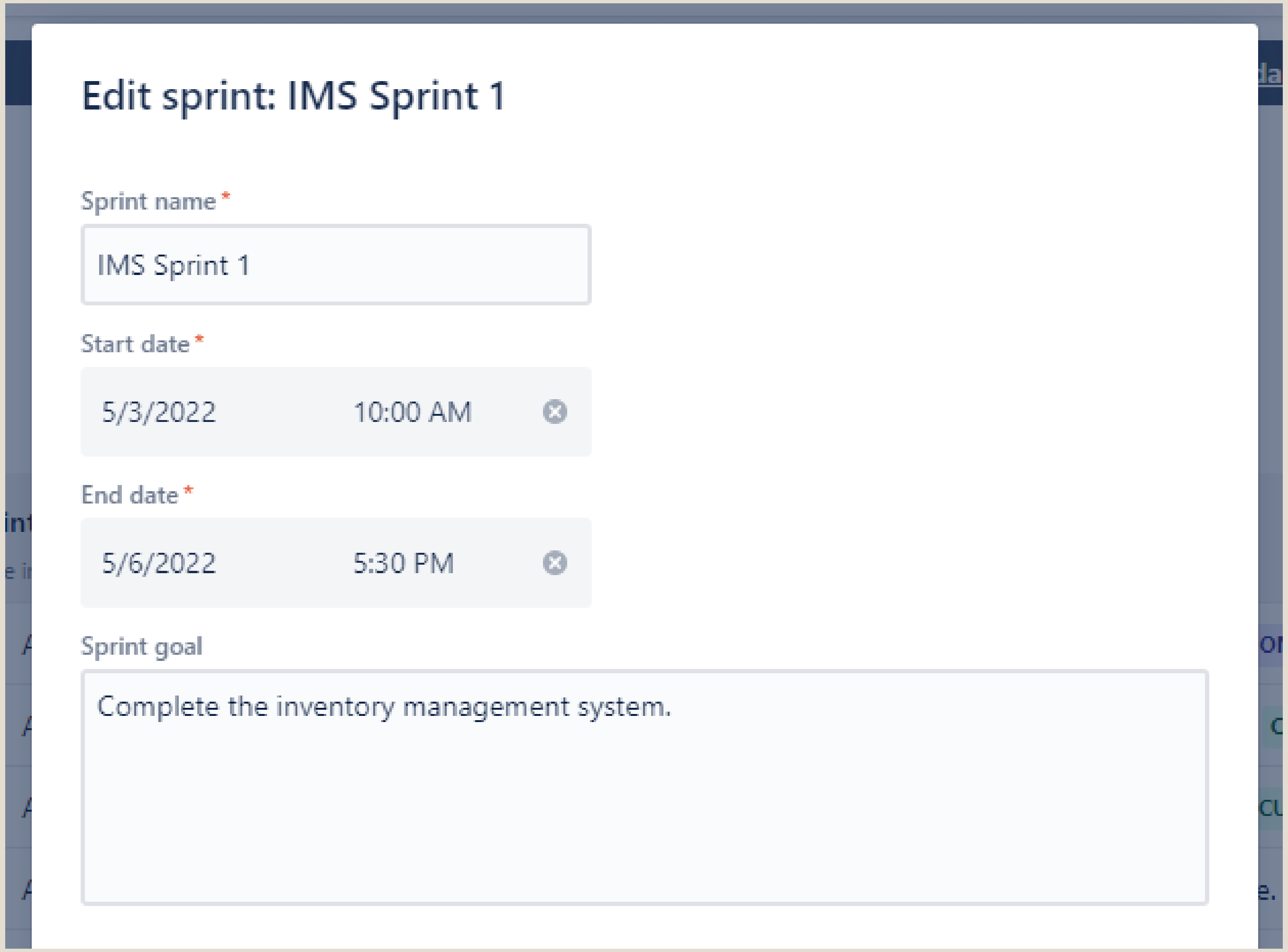


customers		
PK	customer_id	int
	first_name	varchar(40)
	surname	Varchar(40)

orders		
PK	order_id	int
FK	customer_id	int
	cost	decimal(4,2)
	items	varchar(100)

items		
PK	item_id	int
	name	varchar(40)
	price	decimal(10,2)

Sprint(s)

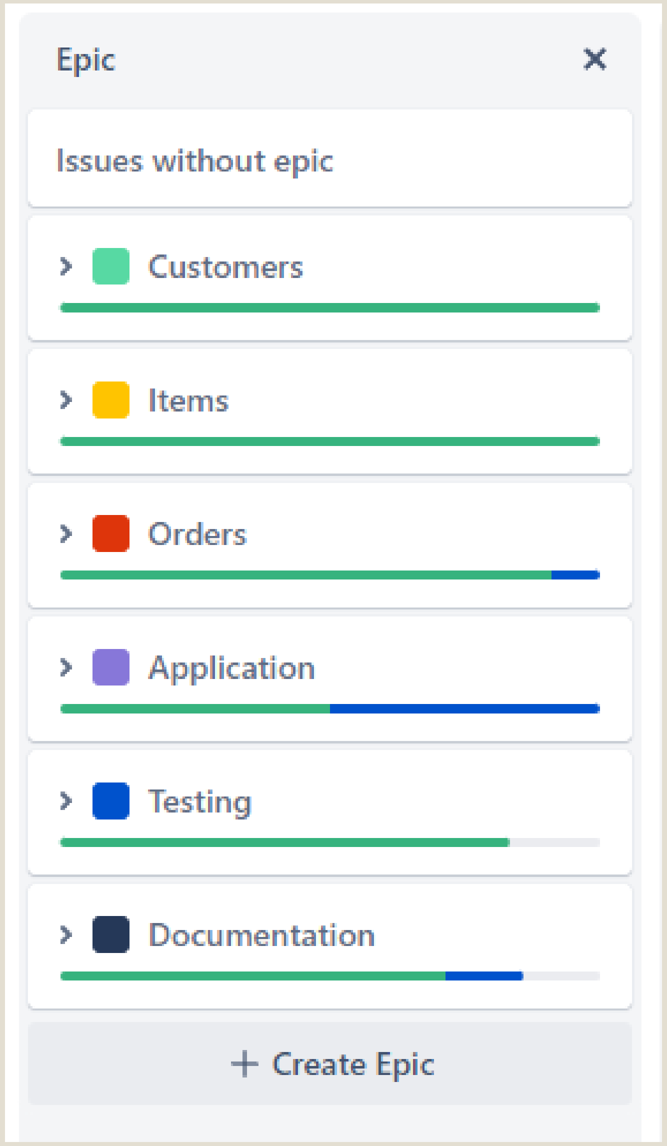


Edit sprint: IMS Sprint 1

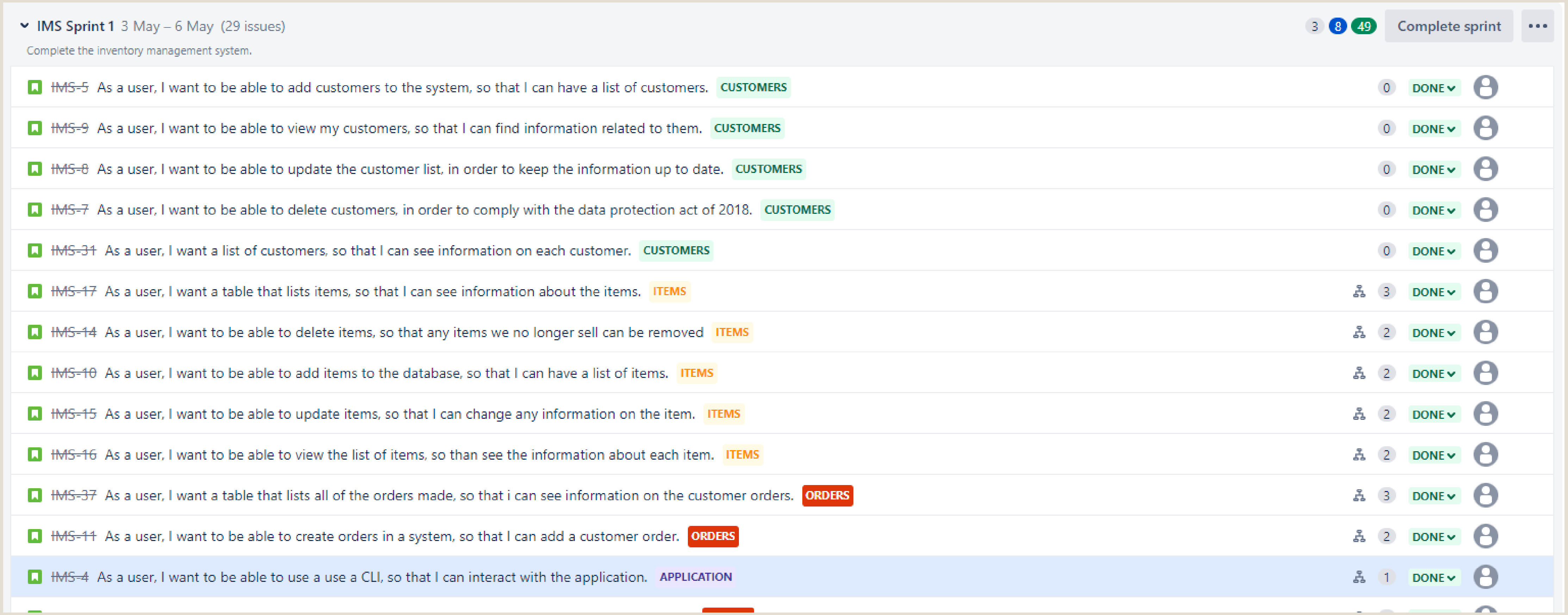
Sprint name*	IMS Sprint 1
Start date*	5/3/2022 10:00 AM
End date*	5/6/2022 5:30 PM
Sprint goal	Complete the inventory management system.

Epics

- Created 5 epics
- User stories
- Tasks
- Acceptance criteria



User Stories

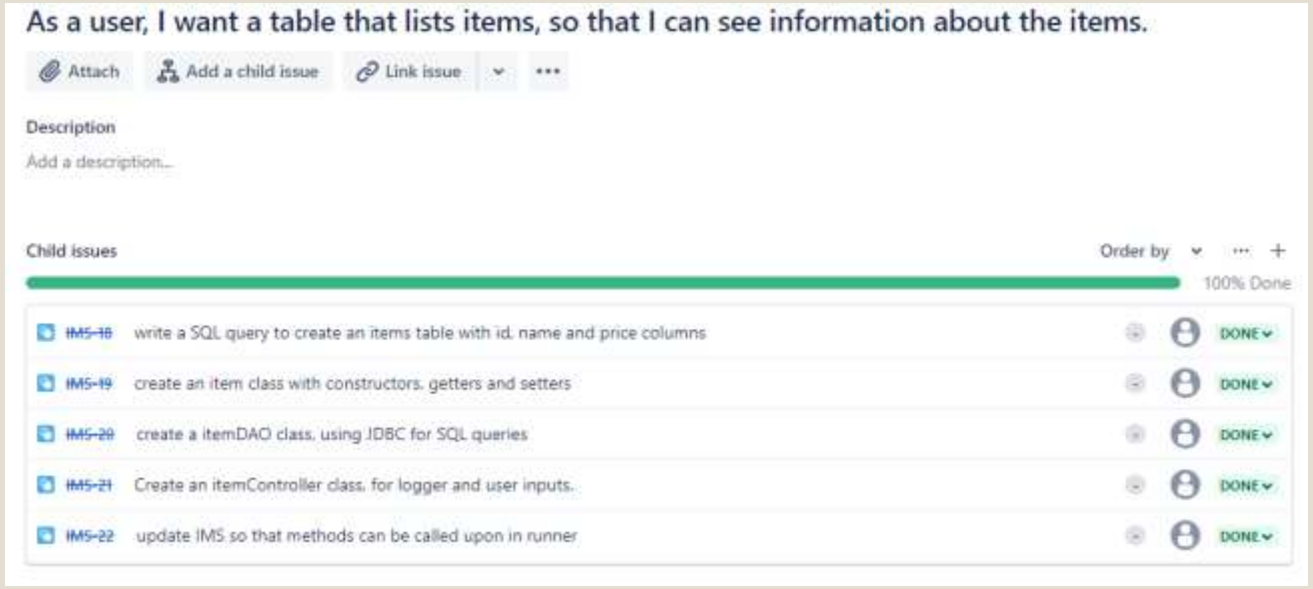


IMS Sprint 1 3 May – 6 May (29 issues) 3 8 49 Complete sprint

Complete the inventory management system.

~~IMS-5~~	As a user, I want to be able to add customers to the system, so that I can have a list of customers.	CUSTOMERS	0	DONE
~~IMS-9~~	As a user, I want to be able to view my customers, so that I can find information related to them.	CUSTOMERS	0	DONE
~~IMS-8~~	As a user, I want to be able to update the customer list, in order to keep the information up to date.	CUSTOMERS	0	DONE
~~IMS-7~~	As a user, I want to be able to delete customers, in order to comply with the data protection act of 2018.	CUSTOMERS	0	DONE
~~IMS-31~~	As a user, I want a list of customers, so that I can see information on each customer.	CUSTOMERS	0	DONE
~~IMS-17~~	As a user, I want a table that lists items, so that I can see information about the items.	ITEMS	3	DONE
~~IMS-14~~	As a user, I want to be able to delete items, so that any items we no longer sell can be removed	ITEMS	2	DONE
~~IMS-10~~	As a user, I want to be able to add items to the database, so that I can have a list of items.	ITEMS	2	DONE
~~IMS-15~~	As a user, I want to be able to update items, so that I can change any information on the item.	ITEMS	2	DONE
~~IMS-16~~	As a user, I want to be able to view the list of items, so than see the information about each item.	ITEMS	2	DONE
~~IMS-37~~	As a user, I want a table that lists all of the orders made, so that i can see information on the customer orders.	ORDERS	3	DONE
~~IMS-11~~	As a user, I want to be able to create orders in a system, so that I can add a customer order.	ORDERS	2	DONE
~~IMS-4~~	As a user, I want to be able to use a use a CLI, so that I can interact with the application.	APPLICATION	1	DONE

Tasks



As a user, I want a table that lists items, so that I can see information about the items.

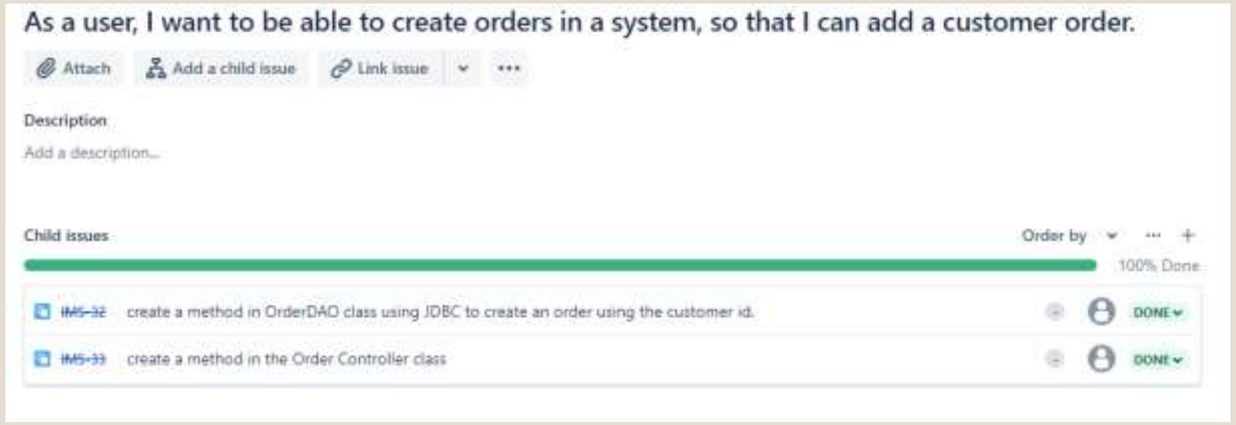
Description

Add a description...

Child issues

100% Done

~~IMS-18~~	write a SQL query to create an items table with id, name and price columns	DONE
~~IMS-19~~	create an item class with constructors, getters and setters	DONE
~~IMS-20~~	create a itemDAO class, using JDBC for SQL queries	DONE
~~IMS-21~~	Create an itemController class, for logger and user inputs.	DONE
~~IMS-22~~	update IMS so that methods can be called upon in runner	DONE



As a user, I want to be able to create orders in a system, so that I can add a customer order.

Description

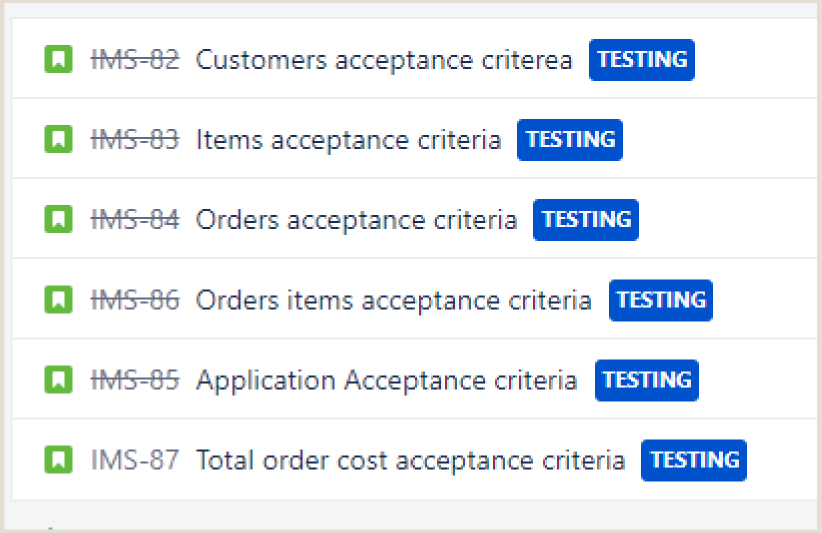
Add a description...

Child issues

100% Done

~~IMS-32~~	create a method in OrderDAO class using JDBC to create an order using the customer id.	DONE
~~IMS-33~~	create a method in the Order Controller class	DONE

Acceptance Criteria



~~IMS-82~~ Customers acceptance criterea TESTING

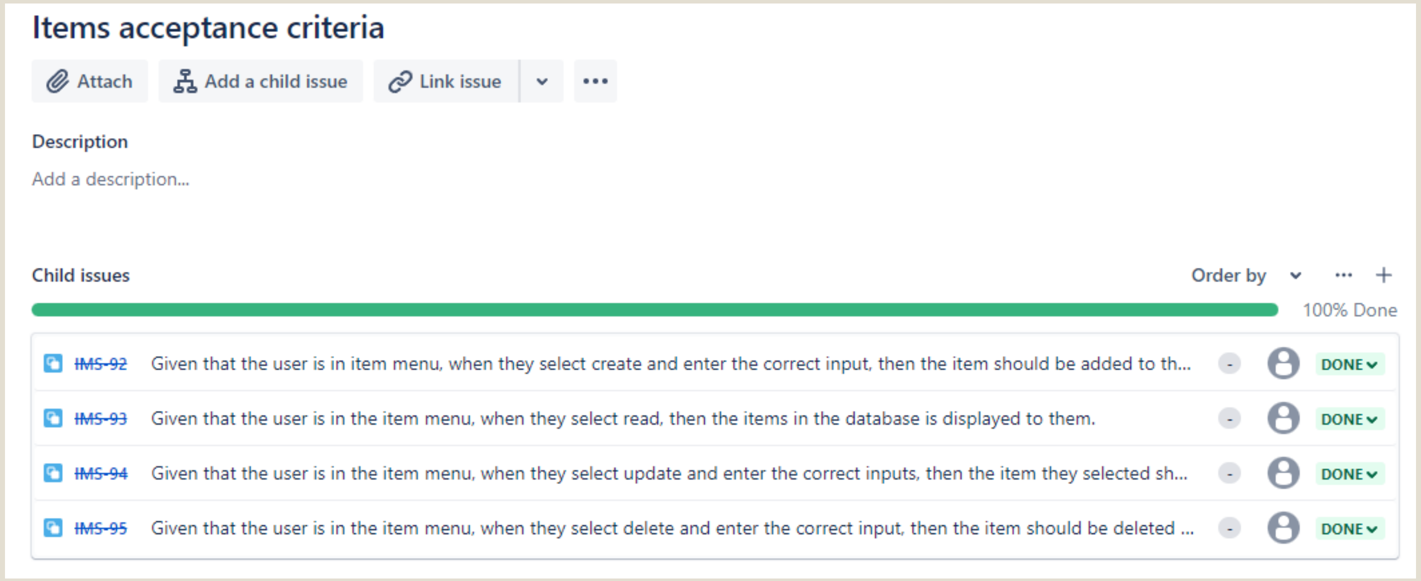
~~IMS-83~~ Items acceptance criteria TESTING

~~IMS-84~~ Orders acceptance criteria TESTING

~~IMS-86~~ Orders items acceptance criteria TESTING

~~IMS-85~~ Application Acceptance criteria TESTING

IMS-87 Total order cost acceptance criteria TESTING



Items acceptance criteria

Description

Add a description...

Child issues

100% Done

~~IMS-92~~ Given that the user is in item menu, when they select create and enter the correct input, then the item should be added to th... DONE

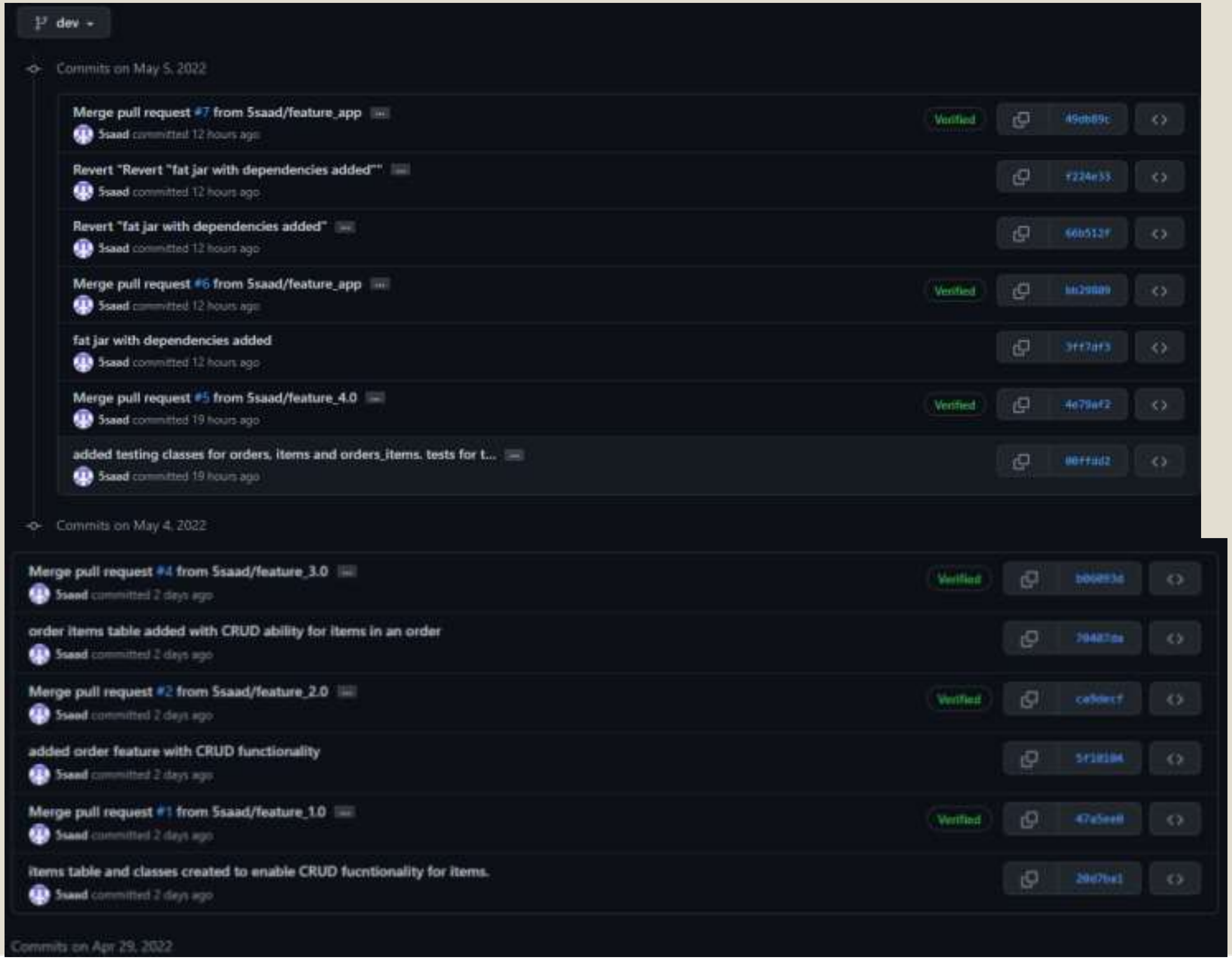
~~IMS-93~~ Given that the user is in the item menu, when they select read, then the items in the database is displayed to them. DONE

~~IMS-94~~ Given that the user is in the item menu, when they select update and enter the correct inputs, then the item they selected sh... DONE

~~IMS-95~~ Given that the user is in the item menu, when they select delete and enter the correct input, then the item should be deleted ... DONE

GitHub

- Feature_1.0: Item, table/classes and CRUD
- Feature_2.0: Order table/classes and CRUD
- Feature_3.0: Order items table/classes and CRUD
- Feature_4.0: Testing tables/classes
- Feature_app: Created fat .jar
- Feature_readme: (coming soon)


dev

Commits on May 5, 2022

Merge pull request #7 from 5saad/feature_app
5saad committed 12 hours ago
Verified 49db89c

Revert "Revert "fat jar with dependencies added""
5saad committed 12 hours ago
f224e33

Revert "fat jar with dependencies added"
5saad committed 12 hours ago
66b512f

Merge pull request #6 from 5saad/feature_app
5saad committed 12 hours ago
Verified bb29889

fat jar with dependencies added
5saad committed 12 hours ago
3ff7df3

Merge pull request #5 from 5saad/feature_4.0
5saad committed 19 hours ago
Verified 4e79af2

added testing classes for orders, items and orders_items. tests for t...
5saad committed 19 hours ago
00ffdd2

Commits on May 4, 2022

Merge pull request #4 from 5saad/feature_3.0
5saad committed 2 days ago
Verified b060934

order items table added with CRUD ability for items in an order
5saad committed 2 days ago
70487da

Merge pull request #2 from 5saad/feature_2.0
5saad committed 2 days ago
Verified ce9decf

added order feature with CRUD functionality
5saad committed 2 days ago
5f101b4

Merge pull request #1 from 5saad/feature_1.0
5saad committed 2 days ago
Verified 47a5ee8

items table and classes created to enable CRUD fucntionality for items.
5saad committed 2 days ago
20d7ba1

Commits on Apr 29, 2022

Version Control

- Each epic = new feature
- Dev branch created>feature branch created
- After each feature is done:
    - Git add . > git commit –m "…" > git push
  - Merge with dev branch in GitHub
  - Git pull dev branch to local
  - Ready for new feature!

Developer Journey

5 weeks ago:

- No SQL experience
- No Java experience

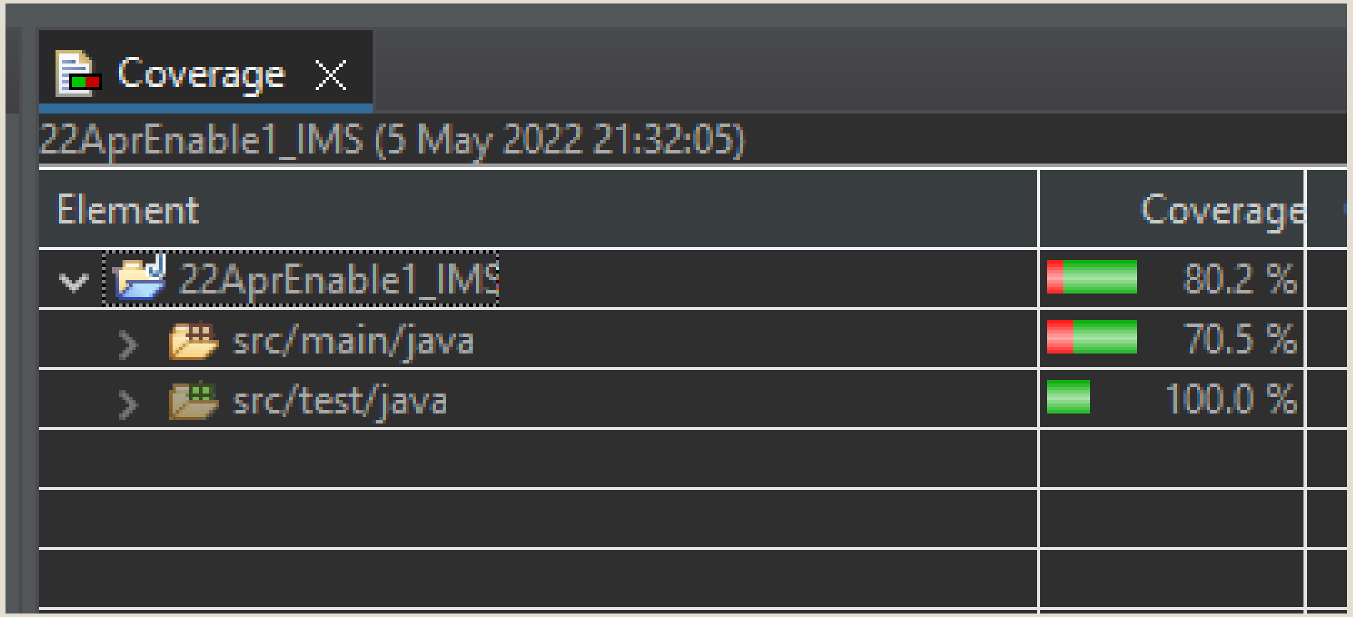
Today:

- Completed an Inventory Management App using SQL and JAVA
- Along with Jira and MoSCoW method

Thanks QA and trainers ☺

Test Coverage

- 70.5% coverage from tests
- Tests CRUD

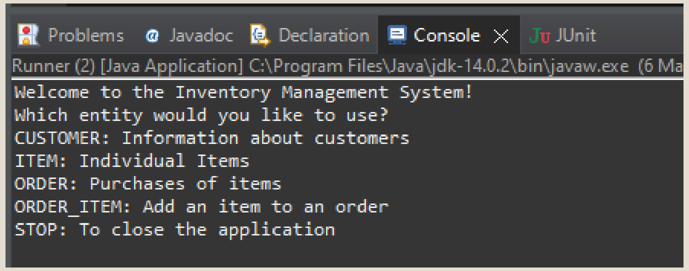


22AprEnable1_IMS (5 May 2022 21:32:05)

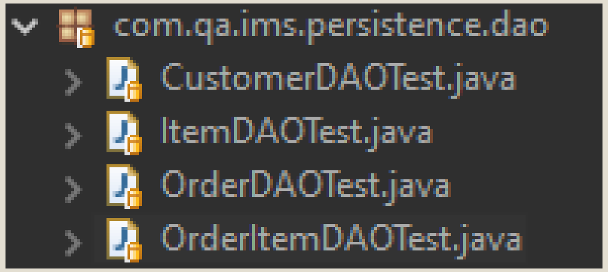
Element	Coverage
22AprEnable1_IMS	80.2 %
src/main/java	70.5 %
src/test/java	100.0 %

CRUD

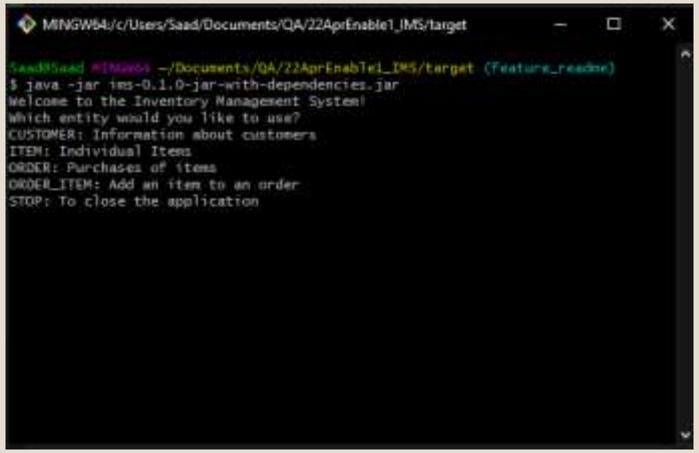
- Fulfilled CRUD user stories and acceptance criteria
- Tested through Eclipse console, application CLI and JUNIT testing classes.



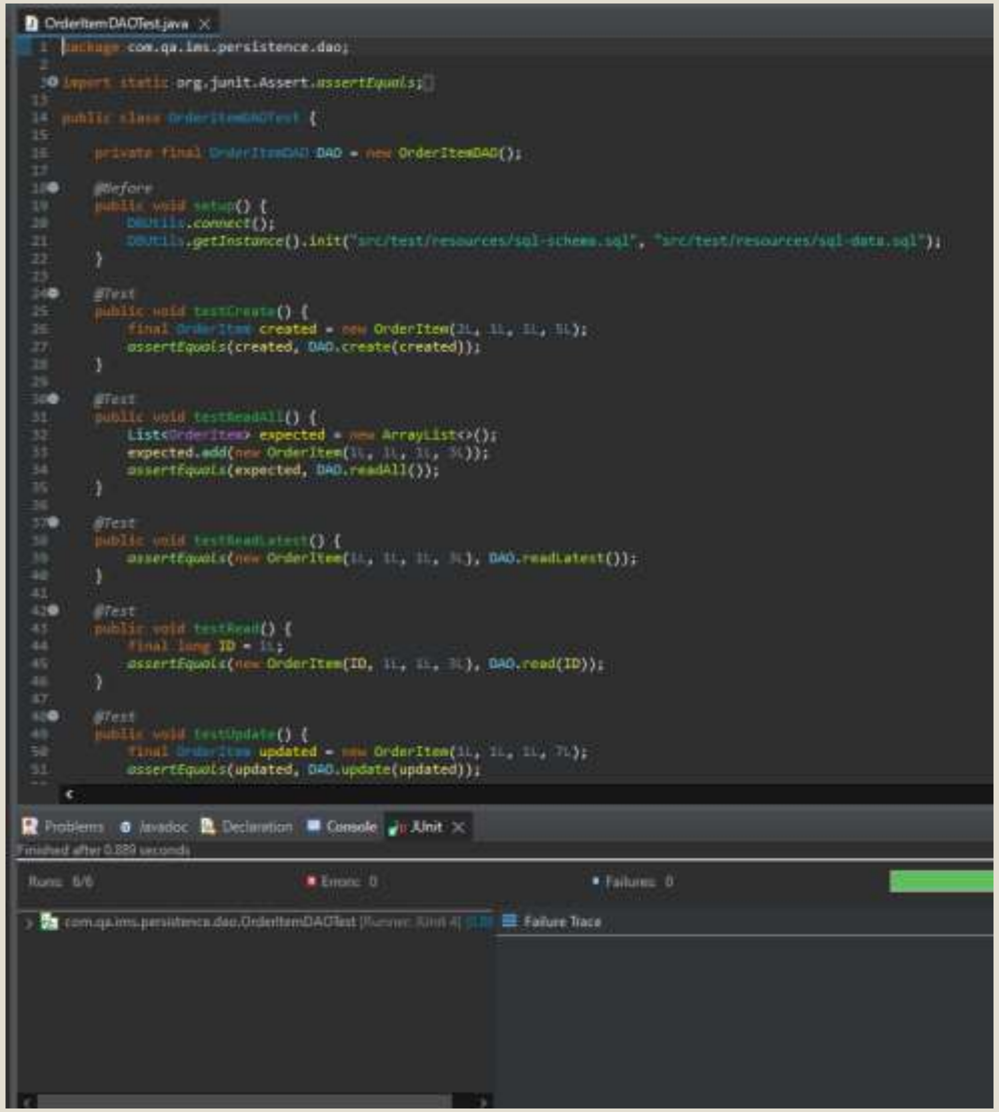
```
Runner (2) [Java Application] C:\Program Files\Java\jdk-14.0.2\bin\javaw.exe (6 Ma
Welcome to the Inventory Management System!
Which entity would you like to use?
CUSTOMER: Information about customers
ITEM: Individual Items
ORDER: Purchases of items
ORDER_ITEM: Add an item to an order
STOP: To close the application
```



```
com.qa.ims.persistence.dao
    CustomerDAOTest.java
    ItemDAOTest.java
    OrderDAOTest.java
    OrderItemDAOTest.java
```



```
MINGW64:/c/Users/Saad/Documents/QA/22AprEnable1_IMS/target
Saad@Saad MINGW64 ~/Documents/QA/22AprEnable1_IMS/target (feature_readme)
$ java -jar ims-0.1.0-jar-with-dependencies.jar
Welcome to the Inventory Management System!
Which entity would you like to use?
CUSTOMER: Information about customers
ITEM: Individual Items
ORDER: Purchases of items
ORDER_ITEM: Add an item to an order
STOP: To close the application
```

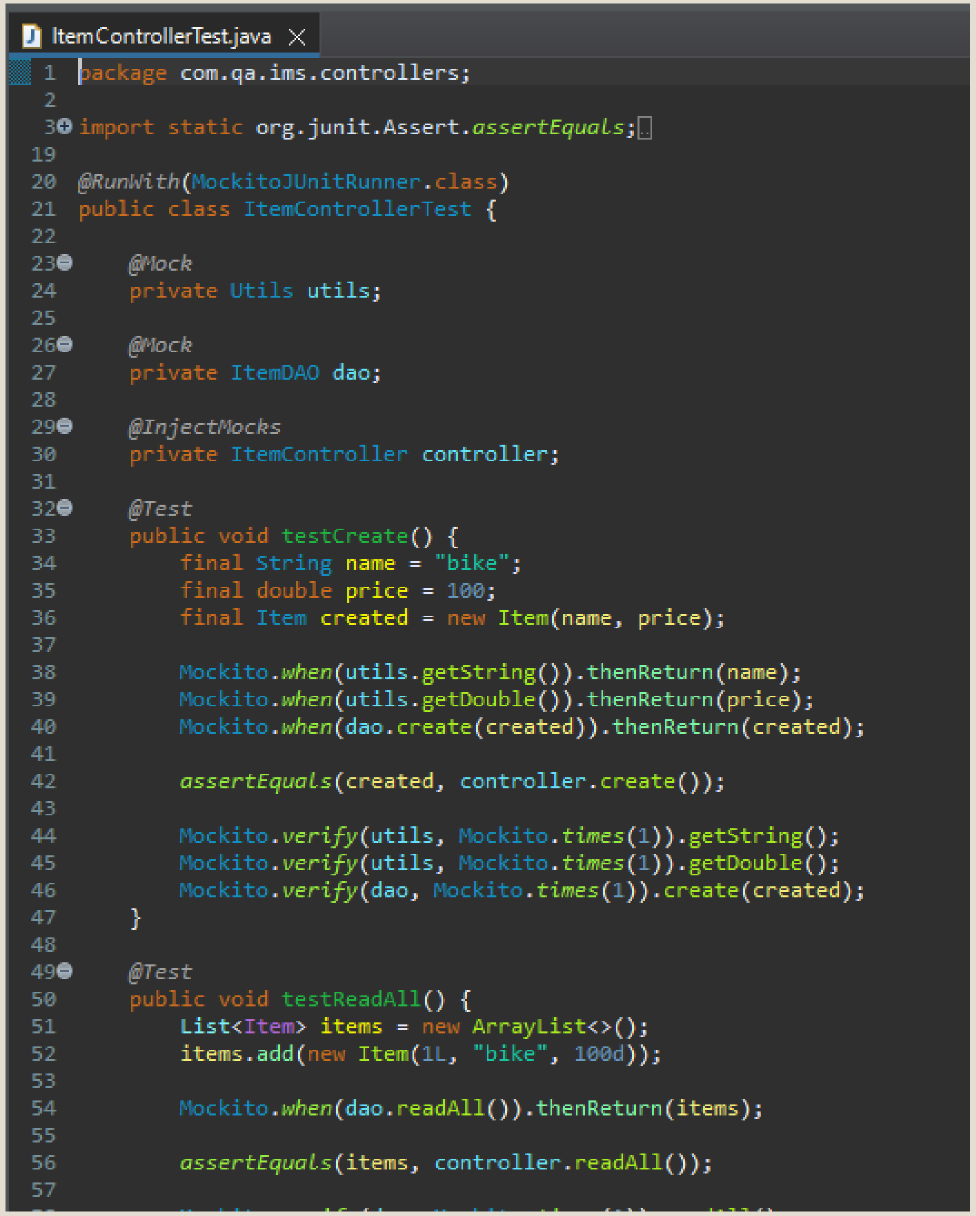


```java
package com.qa.ims.persistence.dao;

import static org.junit.Assert.assertEquals;

public class OrderItemDAOTest {

    private final OrderItemDAO DAO = new OrderItemDAO();

    @Before
    public void setup() {
        DBUtils.connect();
        DBUtils.getInstance().init("src/test/resources/sql-schema.sql", "src/test/resources/sql-data.sql");
    }

    @Test
    public void testCreate() {
        final OrderItem created = new OrderItem(2L, 1L, 1L, 5L);
        assertEquals(created, DAO.create(created));
    }

    @Test
    public void testReadAll() {
        List<OrderItem> expected = new ArrayList<>();
        expected.add(new OrderItem(1L, 1L, 1L, 5L));
        assertEquals(expected, DAO.readAll());
    }

    @Test
    public void testReadLatest() {
        assertEquals(new OrderItem(1L, 1L, 1L, 3L), DAO.readLatest());
    }

    @Test
    public void testRead() {
        final long ID = 1L;
        assertEquals(new OrderItem(ID, 1L, 1L, 3L), DAO.read(ID));
    }

    @Test
    public void testUpdate() {
        final OrderItem updated = new OrderItem(1L, 1L, 1L, 7L);
        assertEquals(updated, DAO.update(updated));
```

```
Finished after 0.889 seconds
Runs: 6/6        Errors: 0        Failures: 0
com.qa.ims.persistence.dao.OrderItemDAOTest [Runner: JUnit 4] (0.0
```

Unit Testing

- All CRUD methods tested using JUNIT tests
- Mockito was used to test the controller classes.
- Incorrect inputs not tested



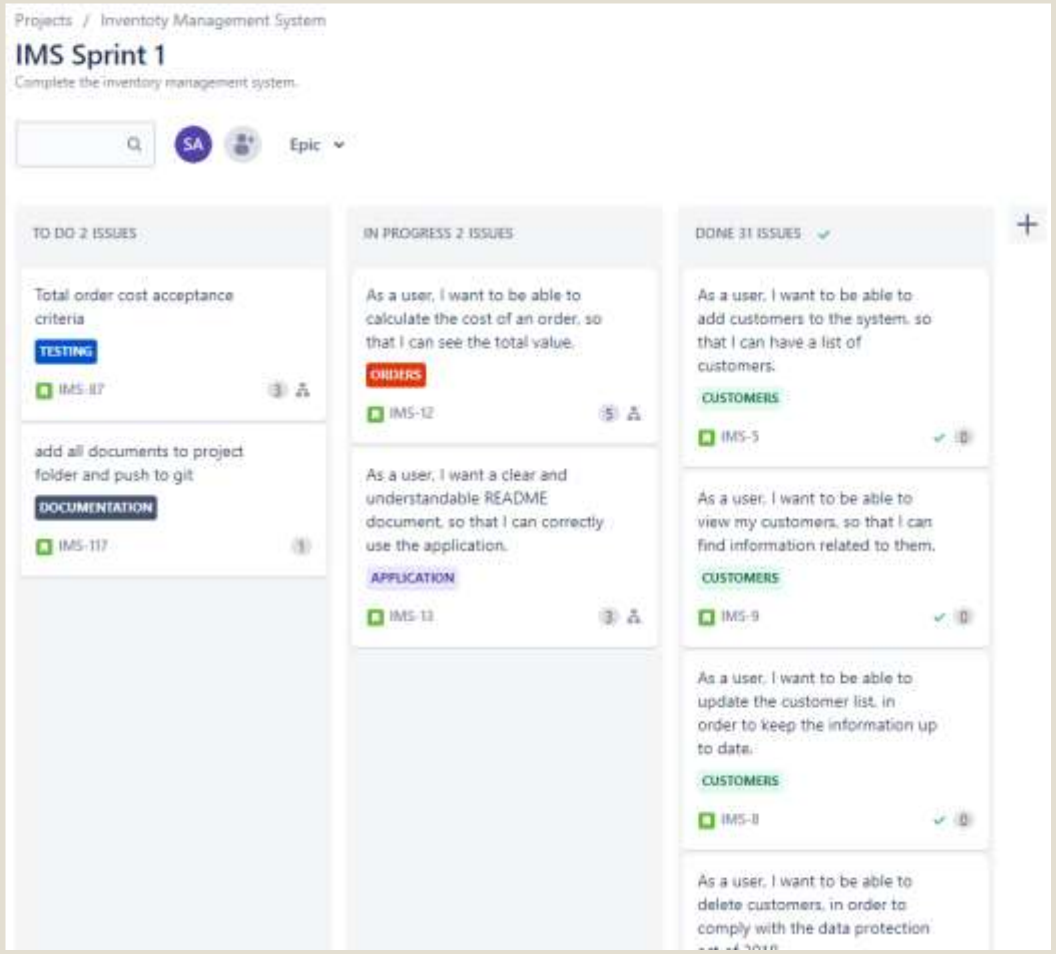
ItemControllerTest.java

```java
package com.qa.ims.controllers;

import static org.junit.Assert.assertEquals;

@RunWith(MockitoJUnitRunner.class)
public class ItemControllerTest {

    @Mock
    private Utils utils;

    @Mock
    private ItemDAO dao;

    @InjectMocks
    private ItemController controller;

    @Test
    public void testCreate() {
        final String name = "bike";
        final double price = 100;
        final Item created = new Item(name, price);

        Mockito.when(utils.getString()).thenReturn(name);
        Mockito.when(utils.getDouble()).thenReturn(price);
        Mockito.when(dao.create(created)).thenReturn(created);

        assertEquals(created, controller.create());

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

    @Test
    public void testReadAll() {
        List<Item> items = new ArrayList<>();
        items.add(new Item(1L, "bike", 100d));

        Mockito.when(dao.readAll()).thenReturn(items);

        assertEquals(items, controller.readAll());

```

LIVE DEMO!

Sprint Review

- 33 out of 35 issue complete by end of sprint.
- MVP achieved
- Must haves/some should haves achieved
- Ability to view order total incomplete:
  - SQL query done
  - Java implementation incomplete



Projects / Inventoty Management System

IMS Sprint 1

Complete the inventory management system.

SA Epic

TO DO 2 ISSUES

- Total order cost acceptance criteria — TESTING — IMS-117 — 3
- add all documents to project folder and push to git — DOCUMENTATION — IMS-117 — 1

IN PROGRESS 2 ISSUES

- As a user, I want to be able to calculate the cost of an order, so that I can see the total value. — ORDERS — IMS-12 — 5
- As a user, I want a clear and understandable README document, so that I can correctly use the application. — APPLICATION — IMS-13 — 3

DONE 31 ISSUES ✓

- As a user, I want to be able to add customers to the system, so that I can have a list of customers. — CUSTOMERS — IMS-5 ✓ 0
- As a user, I want to be able to view my customers, so that I can find information related to them. — CUSTOMERS — IMS-9 ✓ 0
- As a user, I want to be able to update the customer list, in order to keep the information up to date. — CUSTOMERS — IMS-8 ✓ 0
- As a user, I want to be able to delete customers, in order to comply with the data protection

SELECT oi.order_id, o.customer_id, oi.item_id, oi.quantity, i.name, i.price, SUM(i.price * oi.quantity) as total FROM orders o JOIN orders_items oi ON o.id = oi.order_id JOIN items i ON i.id = oi.item_id GROUP BY order_id ORDER BY oi.order_id DESC;

Thank You

Questions?