



# Inventory Management System

Rayyana Suleiman

22AprEnable1

Week 5 Project



# Introduction

- Project Management
- Source Control Management
- Documentation – ERD, UML, Risks, MoSCoW
- Application MVP – CRUD methods for customers, items, orders
- Database persistence
- Unit Tests
- Build

# Risks

Risk Matrix

		Impact				
		Negligible	Minor	Moderate	Major	Catastrophic
Likelihood	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

Risks

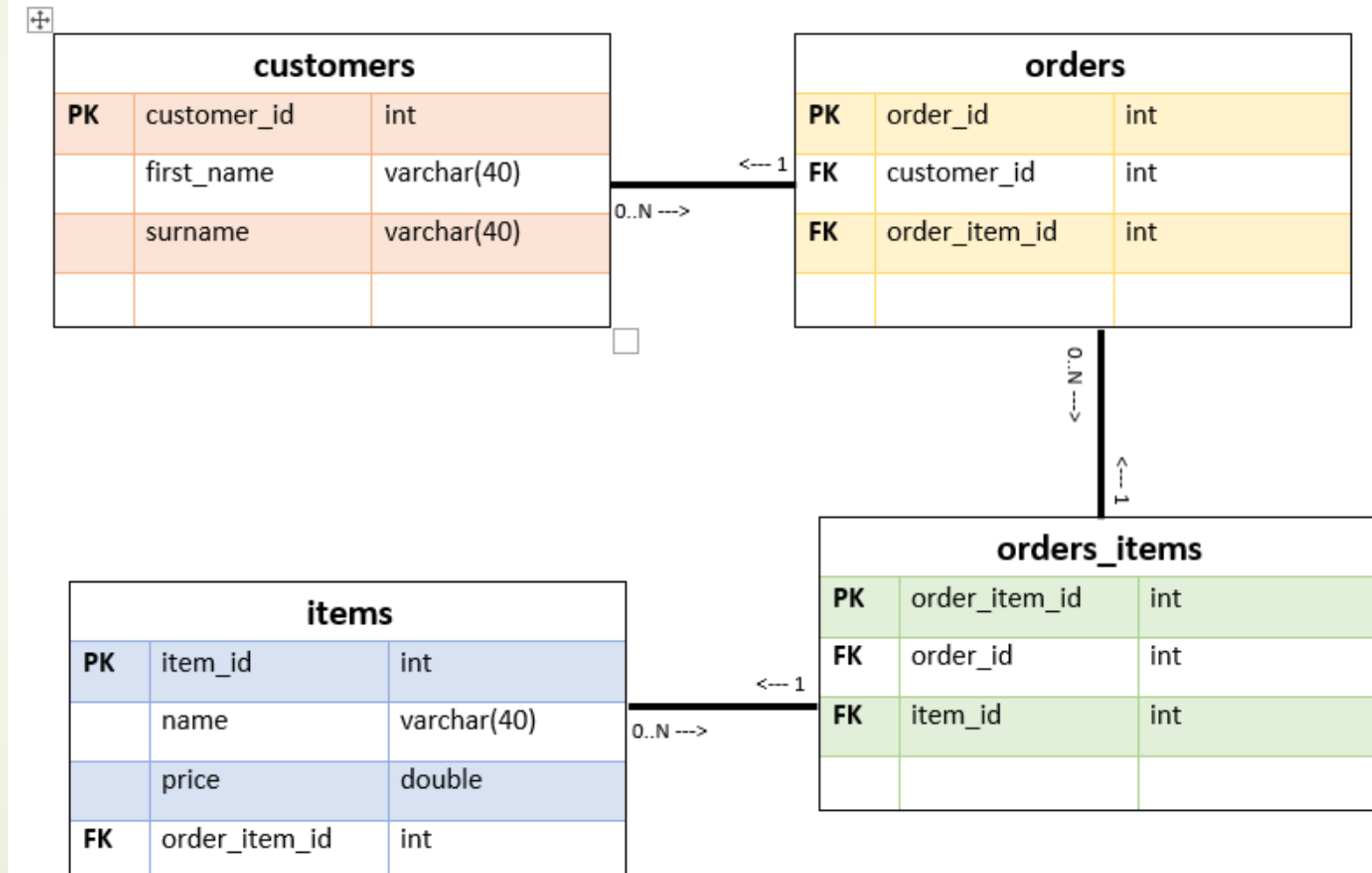
Risk	Evaluation	Unmitigated Response	Likelihood	Impact	Risk Level	Control Measures
RSI	Repetitive Strain Injury in wrist from excessive typing	Spend time waiting for wrists to heal which means time lost working on project	Very Unlikely	Minor	LOW	Regular breaks from typing to rest wrists and prevent RSI
Branch merging error	Incorrect branching workflow could lead to merge conflicts when merging	Revert to previous versions or reset commits to fix - could lose coding or files	Likely	Minor	LOW-MEDIUM	Follow correct branching workflow - merge feature branches to dev before opening new branch
Computer malfunction	Computer not responding or switching on means it can't be used for work	Unable to use computer for project - all unsaved work could be lost	Unlikely	Catastrophic	MEDIUM-HIGH	Monitor computer health and regularly save back-ups.
Broken version deployed	Packaged application may have bugs or missing requirements	Revert production back to most stable/ last working version	Moderate	Major	MEDIUM-HIGH	Automate tests before any deployment. Only merge working code to master branch
Poor Internet connection	Online resources such as GitHub or QA-community become unavailable	Temporarily spend time worrying and waiting for internet to stabilise	Unlikely	Minor	LOW-MEDIUM	Acceptance - unable to use new network provider. Wait for weak signal to strengthen
Physical injury	Dangling/unsecure wires/ equipment could cause injury	Incapable of working on project due to serious accidental injury	Very Unlikely	Catastrophic	MEDIUM	Keep workstation organised
GitHub servers down	Unable to access remote repository to push or pull code/files	Waste time waiting for server to become available so that code can be accessed and worked on	Unlikely	Major	MEDIUM	Make sure local repo is always up to date with GitHub repo so work can continue and be pushed when servers become available
Natural disaster	Flood or earthquake damaging home, person and computer	Unable to work at all, due to unstable living conditions	Very Unlikely	Catastrophic	MEDIUM	Acceptance. Unable to control Nature

# MoSCoW Prioritisation

MoSCoW Prioritisation Table

Must Have	Should Have	Could Have	Won't Have
<ul style="list-style-type: none"><li>As an end user of the IMS, I want to add a customer to the system so that I can keep records of new customers.</li><li>As an end user of the IMS, I want to view all customers in the system so that I can have all the details of all customers written out.</li><li>As an end user of the IMS, I want to update a customer in the system so that I can still contact them if any of their details change.</li><li>As an end user of the IMS, I want to delete a customer from the system so that I am not holding customer details without their permission when they no longer wish to be on record.</li><li>As an end user of the IMS, I want to add an item to the system so that I can keep a record of all items, including new ones.</li></ul>	<ul style="list-style-type: none"><li>As a developer, I want to create a risk assessment so that I can evaluate scenarios that may impact the project in a negative way and mitigate the damage they cause.</li><li>As a developer, I want to create an initial ERD so that I can plan the relationships between the tables in the database.</li><li>As a developer, I want to create an initial UML diagram so that I can visualise the classes in the system before I actually code them.</li><li>As a developer, I want to create a final ERD so that I can accurately map out the relationships between the tables in the database.</li><li>As a developer, I want to update the README file so that the end user finds it easy to use the application.</li></ul>	<ul style="list-style-type: none"><li>As an end user I want to add a user account to the system so that every end user of the IMS has their own account.</li><li>As an end user I want to create a username and password so that my user account is private and cannot be accessed by other users.</li><li>As an end user I want to view all users so that I can select my own, correct account.</li><li>As an end user I want to log in as a user within the system so that I have permission to use and make changes to the IMS.</li><li>As an end user I want changes made to be tied to my user account so that all changes can be traced back to an individual, if needed.</li><li>As a developer, I want to create a final UML diagram so that I can</li></ul>	<ul style="list-style-type: none"><li>Graphical User Interface</li><li>Automatic stock calculator for the items in the inventory</li><li>Notifications for unavailable items in an order</li><li>Automatic receipt emailing to customers after placing order</li></ul>

# Entity-Relationship Diagram





# Class relationships

- Runner calls on Logger in IMS, which controls system output
- Controller - contains the methods that handle user input + Calls on DAO methods
  - All 3 controllers implement CrudController Interface
- DAO - allows us to connect to the database + Contains logic that interacts with database
- All 3 DAOs implement DAO interface
- Domain - Model of data stored in database + Called by DAO and Controller

# Sprint 1 and only

Projects / Week 5 IMS Project

## W5P Sprint 1

Complete Week 5 Project (Application and Documentation)

RS

Epic ▾

Label ▾

Type ▾

GROUP BY

None ▾

Insights

TO DO 9 ISSUES

Review Unit Test for CREATE METHOD

✓ W5P-16

Upload pdf to GitHub

✓ W5P-19

As an end user of the IMS, I want to view all items in the system so that I can get the details of the entire inventory

ITEMS MustHave

W5P-7 3

IN PROGRESS 2 ISSUES

As an end user of the IMS, I want to add a customer to the system so that I can keep records of new customers.

CUSTOMERS MustHave

W5P-1 3

As a developer, I want to create a risk assessment so that I can evaluate scenarios that may impact the project in a negative way and mitigate the damage they cause

DOCUMENTATION ShouldHave

DONE 3 ISSUES ✓

Review CREATE method in customer controller

✓ W5P-15 ✓

Create Risk Matrix

✓ W5P-17 ✓

Fill in Risks table

✓ W5P-18 ✓

+

### Edit sprint: W5P Sprint 1

W5P Sprint 1

Duration

custom ▾

Start date

5/3/2022 10:00 AM ✕

End date

5/6/2022 5:00 PM ✕

Sprint goal

Complete Week 5 Project (Application and Documentation)

Update Cancel



# User Stories

The screenshot displays the Jira Software interface for a project named "Week 5 IMS Project". The main content area shows a user story titled "As an end user of the IMS, I want to view all items in the system so that I can get the details of the entire inventory". The story is currently in the "To Do" status and has a story point estimate of 6. The description of the story is: "Given that the end user navigates to READ ITEMS, when they hit enter then all records of the items table should be printed out." Below the description, there is a section for "Child issues" which lists three tasks: "W5P-52 Create Item Domain class", "W5P-53 Create Item DAO class", and "W5P-54 Create Item Controller Class". The right sidebar shows the "Details" section with fields for Assignee (Unassigned), Labels (MustHave), Sprint (W5P Sprint 1), and Reporter (Rayyana Suleiman). The bottom of the interface shows a comment section with a "Pro tip: press M to comment" message.

**Jira Software** Your work ▾ Projects ▾ Filters ▾ Dashboards ▾ People ▾ Apps ▾ **Create**

Search

Projects / Week 5 IMS Project / Items / W5P-7

**Week 5 IMS Project**  
Software project

**PLANNING**

- Roadmap
- Backlog
- Board

**DEVELOPMENT**

- Code

**Project pages**

- Add shortcut
- Project settings

You're in a team-managed project  
[Learn more](#)

**As an end user of the IMS, I want to view all items in the system so that I can get the details of the entire inventory**

Attach Add a child issue Link issue ▾ ⋮

**Description**

Given that the end user navigates to READ ITEMS, when they hit enter then all records of the items table should be printed out.

**Child issues** Order by ▾ ⋮ + 0% Done

W5P-52	Create Item Domain class	-	Unassigned	TO DO ▾
W5P-53	Create Item DAO class	-	Unassigned	TO DO ▾
W5P-54	Create Item Controller Class	-	Unassigned	TO DO ▾

**Details**

Assignee: Unassigned

Labels: MustHave

Sprint: W5P Sprint 1

Reporter: RS Rayyana Suleiman

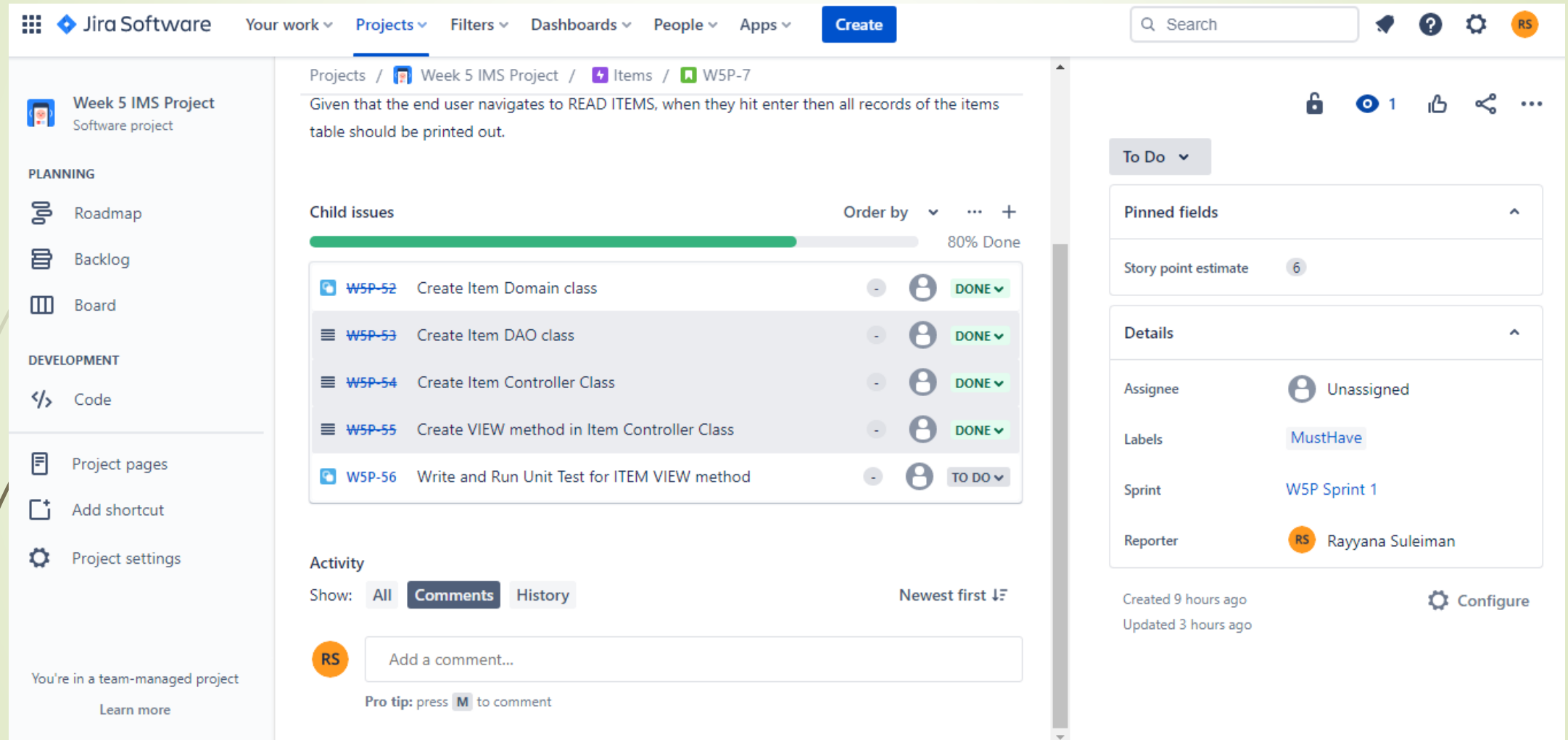
Created 9 hours ago Updated 3 hours ago [Configure](#)

RS Add a comment...

Pro tip: press **M** to comment



# Tasks



The screenshot displays the Jira Software interface for a project named 'Week 5 IMS Project'. The left sidebar contains navigation options under 'PLANNING' (Roadmap, Backlog, Board) and 'DEVELOPMENT' (Code). The main content area shows a list of child issues for 'W5P-7', with a progress bar indicating 80% completion. The issues are: W5P-52 (Create Item Domain class, DONE), W5P-53 (Create Item DAO class, DONE), W5P-54 (Create Item Controller Class, DONE), W5P-55 (Create VIEW method in Item Controller Class, DONE), and W5P-56 (Write and Run Unit Test for ITEM VIEW method, TO DO). The right sidebar shows the 'To Do' section with pinned fields: Story point estimate (6), Details (Assignee: Unassigned, Labels: MustHave, Sprint: W5P Sprint 1, Reporter: Rayyana Suleiman), and a 'Configure' button.

**Jira Software** Your work ▾ Projects ▾ Filters ▾ Dashboards ▾ People ▾ Apps ▾ **Create**

Search

Projects / Week 5 IMS Project / Items / W5P-7

Given that the end user navigates to READ ITEMS, when they hit enter then all records of the items table should be printed out.

**Child issues** Order by ▾ ... + 80% Done

Issue ID	Task Description	Status
W5P-52	Create Item Domain class	DONE ✓
W5P-53	Create Item DAO class	DONE ✓
W5P-54	Create Item Controller Class	DONE ✓
W5P-55	Create VIEW method in Item Controller Class	DONE ✓
W5P-56	Write and Run Unit Test for ITEM VIEW method	TO DO ▾

**Activity** Show: All Comments History Newest first ⌵

RS Add a comment...

Pro tip: press **M** to comment

**To Do ▾**

**Pinned fields** ^

Story point estimate 6

**Details** ^

Assignee Unassigned

Labels MustHave

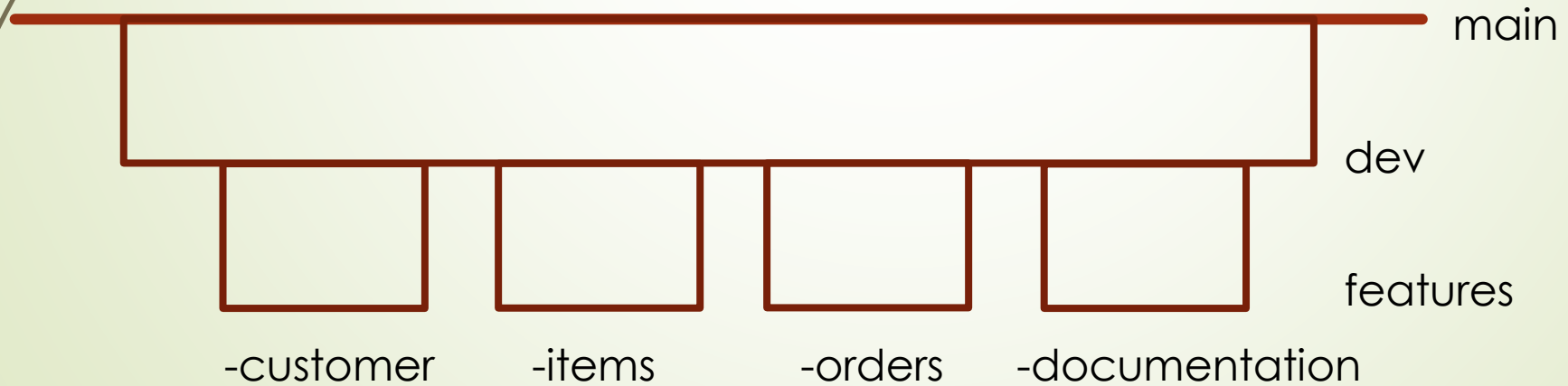
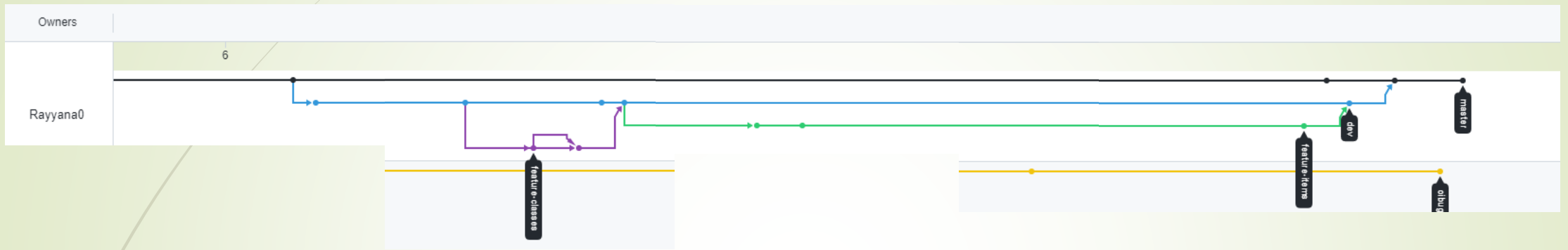
Sprint W5P Sprint 1

Reporter RS Rayyana Suleiman

Created 9 hours ago Updated 3 hours ago **Configure**

You're in a team-managed project [Learn more](#)

# Git FBM



# Version Control

```
Hussein@LAPTOP-76OLIO71 MINGW64 ~/Documents/1. Rayyana/QA Training Academy/IMS Project/22AprEnable1_IMS (dev)
$ git branch feature-items

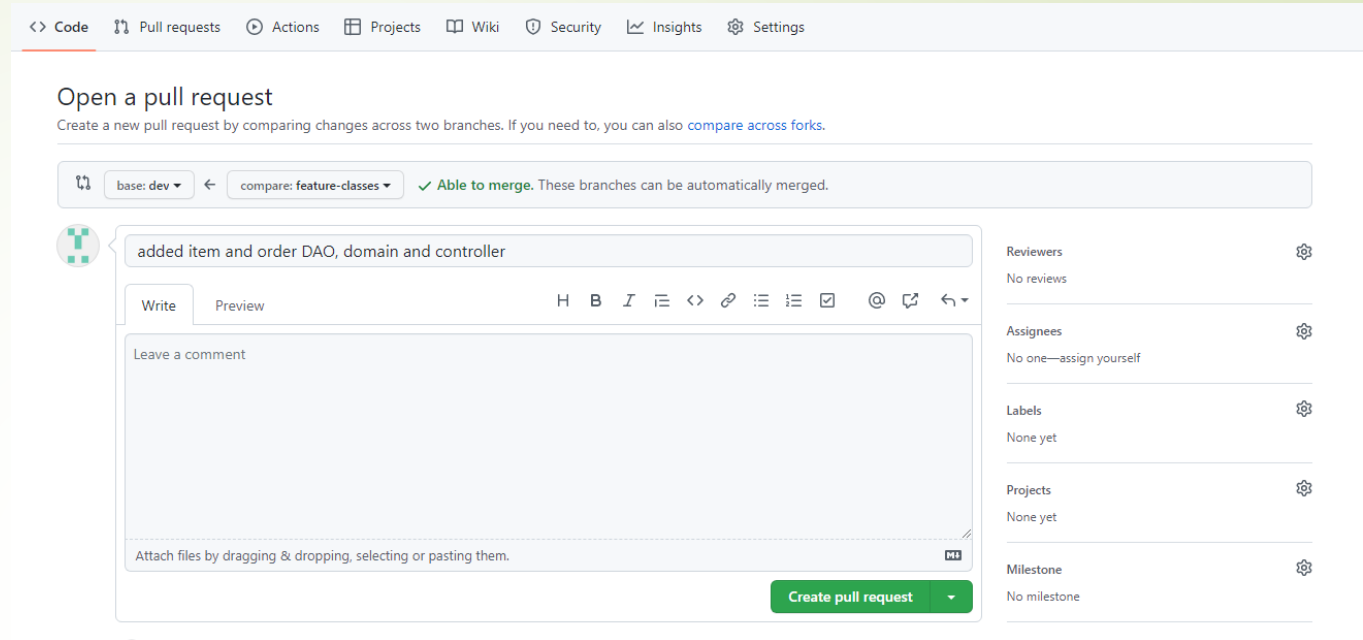
Hussein@LAPTOP-76OLIO71 MINGW64 ~/Documents/1. Rayyana/QA Training Academy/IMS Project/22AprEnable1_IMS (dev)
$ git checkout feature-items
Switched to branch 'feature-items'

Hussein@LAPTOP-76OLIO71 MINGW64 ~/Documents/1. Rayyana/QA Training Academy/IMS Project/22AprEnable1_IMS (feature-items)
$ git add .

Hussein@LAPTOP-76OLIO71 MINGW64 ~/Documents/1. Rayyana/QA Training Academy/IMS Project/22AprEnable1_IMS (feature-items)
$ git commit -m "added Item domain getters, setters and constructors"
[feature-items 9a75dba] added Item domain getters, setters and constructors
1 file changed, 39 insertions(+)
```

```
Hussein@LAPTOP-76OLIO71 MINGW64 ~/Documents/1. Rayyana/QA Training Academy/IMS Project/22AprEnable1_IMS (feature-items)
$ git push --set-upstream origin feature-items
Enumerating objects: 21, done.
Counting objects: 100% (21/21), done.
Delta compression using up to 4 threads
Compressing objects: 100% (8/8), done.
Writing objects: 100% (11/11), 1.02 KiB | 173.00 KiB/s, done.
Total 11 (delta 3), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (3/3), completed with 3 local objects.
remote:
remote: Create a pull request for 'feature-items' on GitHub by visiting:
remote:   https://github.com/Rayyana0/22AprEnable1_IMS/pull/new/feature-items
remote:
To github.com:Rayyana0/22AprEnable1_IMS.git
 * [new branch]      feature-items -> feature-items
branch 'feature-items' set up to track 'origin/feature-items'.

Hussein@LAPTOP-76OLIO71 MINGW64 ~/Documents/1. Rayyana/QA Training Academy/IMS Project/22AprEnable1_IMS (feature-items)
$
```



Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also [compare across forks](#).

base: dev ← compare: feature-items ✓ Able to merge. These branches can be automatically merged.

added item and order DAO, domain and controller

Write Preview H B I ≡ <> 🔗 ≡ ≡ ☑ @ 📎 ↶

Leave a comment

Attach files by dragging & dropping, selecting or pasting them.

Create pull request

Reviewers: No reviews

Assignees: No one—assign yourself

Labels: None yet

Projects: None yet

Milestone: No milestone

- Create feature branch from dev branch
- Make changes to code, git add, commit and push to feature branch in remote repo
- Merge feature branch into dev with pull request on GitHub
- Fetch changes from remote dev to local dev
- Create next feature branch

# Developer Journey

- Git
- Agile
- Databases
- Java
- Maven
- Testing

 Jira Software



**Maven**<sup>TM</sup>

JUnit 

# CRUD methods

- Methods in Item DAO that interact with the database – contains the statements that MySQL can execute

```
61     return null;
62 }
63
64 /**
65  * Creates an item in the database
66  *
67  * @param item - takes in an item object. id will be ignored
68  */
69 @Override
70 public Item create(Item item) {
71     try (Connection connection = DBUtils.getInstance().getConnection();
72         PreparedStatement statement = connection
73             .prepareStatement("INSERT INTO items(name, price) VALUES (?, ?)");) {
74         statement.setString(1, item.getName());
75         statement.setDouble(2, item.getPrice());
76         statement.executeUpdate();
77         return readLatest();
78     } catch (Exception e) {
79         LOGGER.debug(e);
80         LOGGER.error(e.getMessage());
81     }
82     return null;
83 }
84
85 @Override
86 public Item read(Long id) {
```

```
ItemDAO.java x IMS.java ItemDAOTest... ItemTest.java CustomerCon... ItemControll... »6
80
81     }
82     return null;
83 }
84
85 @Override
86 public Item read(Long id) {
87     try (Connection connection = DBUtils.getInstance().getConnection();
88         PreparedStatement statement = connection.prepareStatement("SELECT * FROM items WHERE id = ?");) {
89         statement.setLong(1, id);
90         try (ResultSet resultSet = statement.executeQuery();) {
91             resultSet.next();
92             return modelFromResultSet(resultSet);
93         }
94     } catch (Exception e) {
95         LOGGER.debug(e);
96         LOGGER.error(e.getMessage());
97     }
98     return null;
99 }
100
101 /**
102  * Updates an item in the database
103  *
104  * @param item - takes in an item object, the id field will be used to
105  *               update that item in the database
106  * @return
```

# Unit Testing

The screenshot shows an IDE with the following components:

- Package Explorer:** Shows the project structure with a tree view of test classes. `com.qa.ims.controllers.ItemControllerTest` is selected.
- JUnit Runner:** Shows the test results for `ItemControllerTest`. It indicates 4 runs, 0 errors, and 2 failures. The failed tests are `testReadAll` (2.293 s), `testCreate` (0.051 s), `testDelete` (0.008 s), and `testUpdate` (0.010 s).
- Failure Trace:** Displays the exception for the failed test: `org.mockito.exceptions.verification.TooFewActualInvocations`. The message states: "utils.getString(); Wanted 2 times: -> at com.qa.ims.controllers.ItemControllerTest.testCreate() But was 1 time: -> at com.qa.ims.controller.ItemController.create(ItemControllerTest.java:35)".
- Source Editor:** Shows the code for `ItemDAO.java`. It includes a `@InjectMocks` annotation for `ItemController controller;` and several `@Test` methods: `testCreate()`, `testReadAll()`, and `testDelete()`. The `testCreate()` method uses `Mockito.when()` to stub `utils.getString()`, `utils.getDouble()`, and `dao.create()`, and `assertEquals()` to verify the result.
- Console:** Shows the output of the test run, including the prompt "Please enter a price(£)" and the response "Item created".

The screenshot shows an IDE with the following components:

- Package Explorer:** Shows the project structure with a tree view of test classes. `ItemTest.java` is selected.
- JUnit Runner:** Shows the test results for `ItemTest`. It indicates 4 runs, 0 errors, and 0 failures.
- Source Editor:** Shows the code for `ItemTest.java`. It includes a `@Test` method `testUpdate()` and a `@Test` method `testDelete()`. The `testUpdate()` method uses `Mockito.when()` to stub `utils.getLong()`, `utils.getString()`, and `utils.getDouble()`, and `assertEquals()` to verify the result. The `testDelete()` method uses `Mockito.when()` to stub `utils.getLong()` and `dao.delete()`, and `assertEquals()` to verify the result.
- Console:** Shows the output of the test run, including the prompt "Please enter a price(£)" and the response "Item Updated".





The End

