Fundamental Project (IMS)

Jools Arts

Contents

Introduction

Who are you?

- My name is Jools Arts, an aspiring software developer from Huntingdon, and this is the accompanying presentation for my inventory management system fundamental project

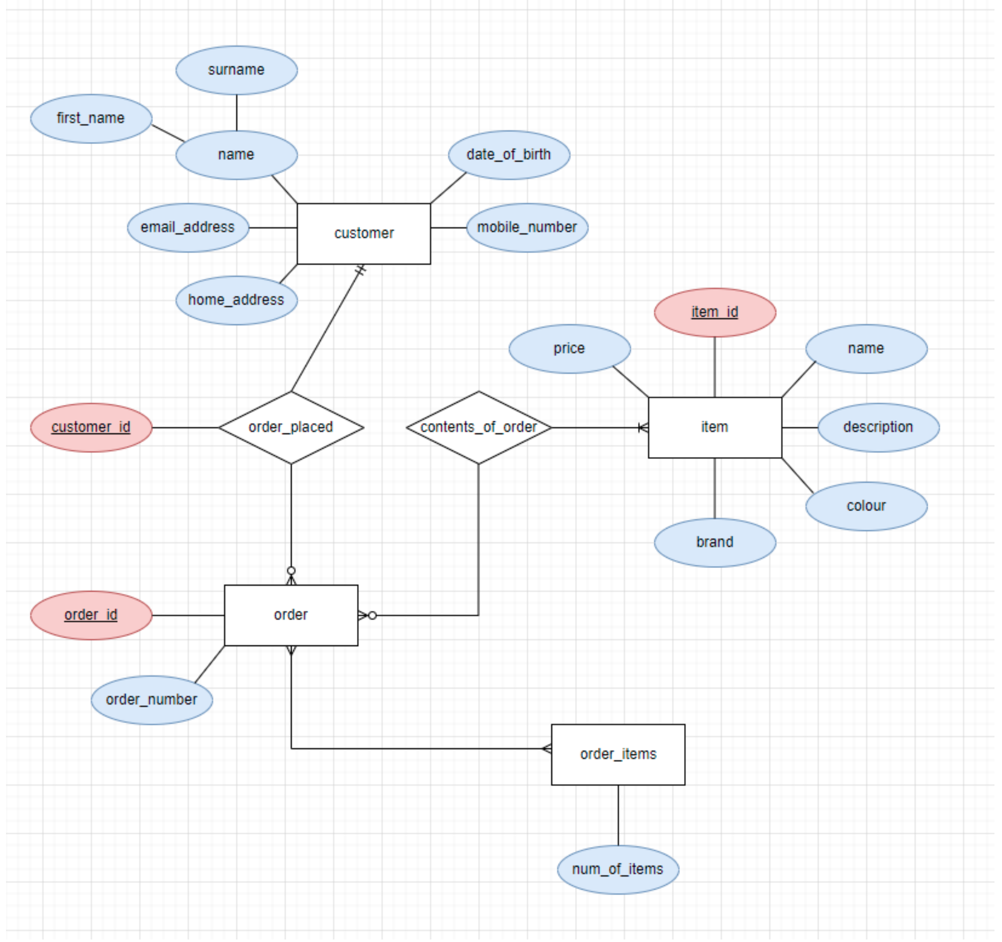
How did you approach the specification?

- I'm a firm believer of the popular mantra, "by failing to prepare, you are preparing to fail", and nowhere does this phrase ring more true than within the field of software development
- As tempting as it can be to dive feet first into the technical coding aspect of the project with your newly acquired skills and talents, I often find that planning your approach to tackling a large project can lead to less mistakes being made overall, resulting in a development process that is far more efficient, as well as being less likely to induce stress, as the developer is made far more aware of the tasks that he/she has been set, as well as the timeframe for completing them

Jira Software

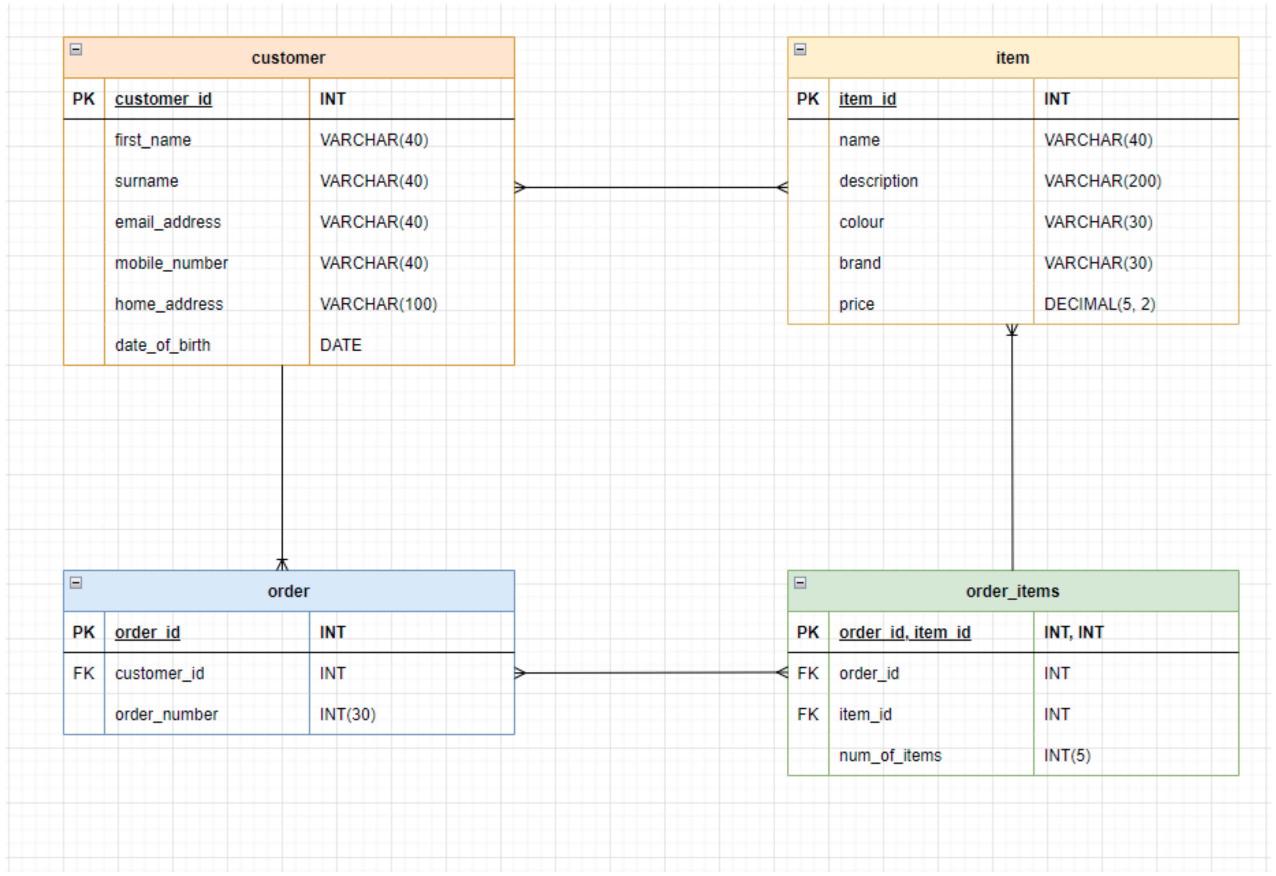
How did you approach the specification?

- One of the many tools we as developers are able to utilise to aid us in our planning for large scale projects are entity relationships diagrams, or ERDs for short
- They represent a fantastic method for visualising the structure of our code by mapping out various classes and interfaces, as well as highlighting the manner in which they interact with one another
- Two of the main diagrams we have been introduced to whilst studying the agile methodology are chen diagrams, and UML diagrams
- The diagram showcased to the right is that of a chen diagram, and is used to clearly display the classes that are intended to be utilised within the inventory management system
- The diagram also serves to highlight the different relationships between the entities for example, one customer can have many orders



How did you approach the specification?

- UML diagrams seek to offer a very similar functionality, only they can often provide greater detail in terms of the relevant attributes of the classes
- For example, within our customer entity, we are able to easily determine the primary key that is associated with that particular class (customer_id) which is of an INT type
- We can also determine the order_items class is formed of a composite key by way of it having two primary keys
- All of this aids us, as the developers, to visually map out the required contents of our various projects



	customer	
PK	customer_id	INT
	first_name	VARCHAR(40)
	surname	VARCHAR(40)
	email_address	VARCHAR(40)
	mobile_number	VARCHAR(40)
	home_address	VARCHAR(100)
	date_of_birth	DATE

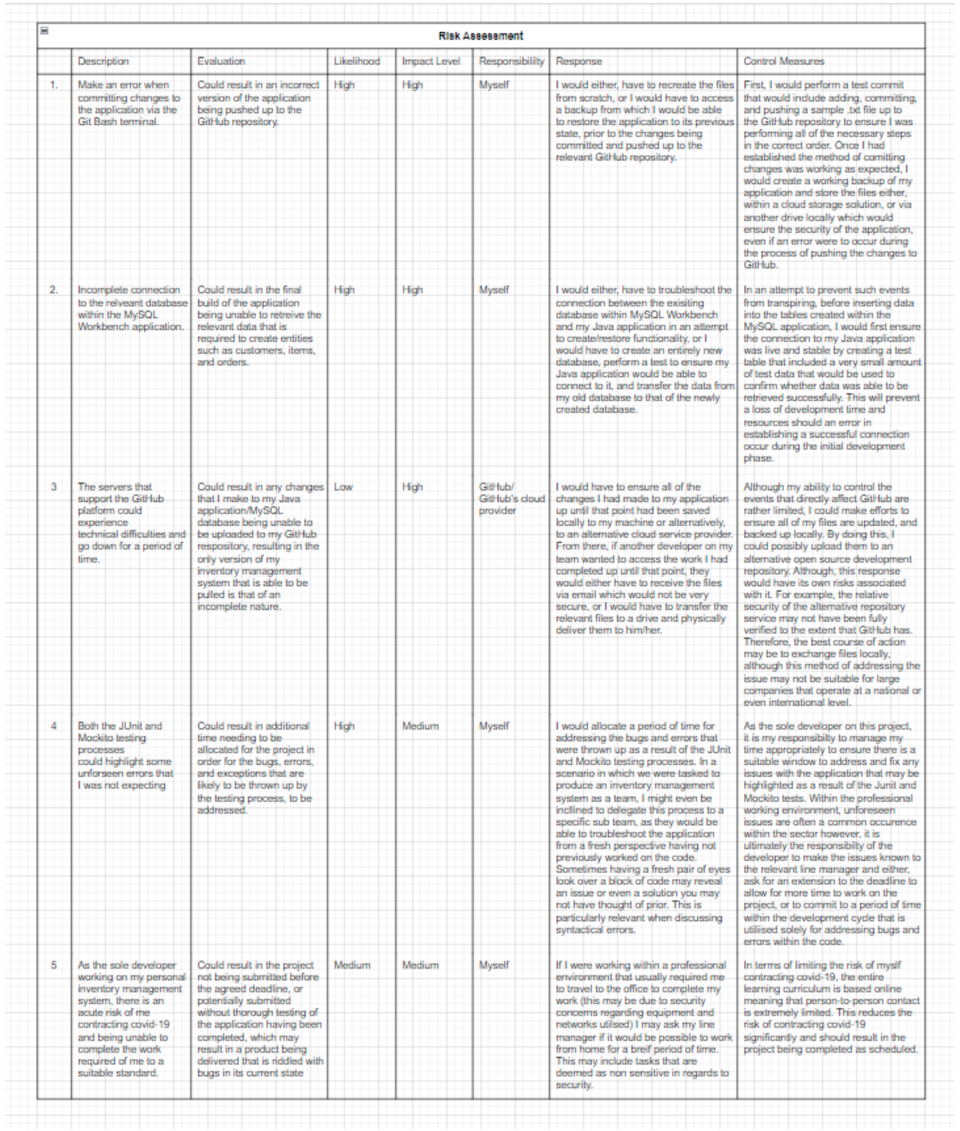
	item	
PK	item_id	INT
	name	VARCHAR(40)
	description	VARCHAR(200)
	colour	VARCHAR(30)
	brand	VARCHAR(30)
	price	DECIMAL(5, 2)

	order	
PK	order_id	INT
FK	customer_id	INT
	order_number	INT(30)

	order_items	
PK	order_id, item_id	INT, INT
FK	order_id	INT
FK	item_id	INT
	num_of_items	INT(5)

How did you approach the specification?

- Another excellent tool at the disposal of developers that can aid in the planning of large scale development projects is that of risk assessments
- Risk assessments, as the name suggests, allow the user to quickly establish various risks associated with the project that may potentially cause significant delay or disruption to the development cycle
- One particular area in which the risk assessment aided in the planning of my personal project was the way in which it highlighted my lack of experience in pushing changes to my GitHub repository within a formal setting therefore, I quickly surmised it would be beneficial for me to perform a test commit, to ensure I was well versed in the process before applying the same logic to my project files which were of great importance



Risk Assessment

	Description	Evaluation	Likelihood	Impact Level	Responsibility	Response	Control Measures
1.	Make an error when committing changes to the application via the Git Bash terminal.	Could result in an incorrect version of the application being pushed up to the GitHub repository.	High	High	Myself	I would either, have to recreate the files from scratch, or I would have to access a backup from which I would be able to restore the application to its previous state, prior to the changes being committed and pushed up to the relevant GitHub repository.	First, I would perform a test commit that would include adding, committing, and pushing a sample .txt file up to the GitHub repository to ensure I was performing all of the necessary steps in the correct order. Once I had established the method of comitting changes was working as expected, I would create a working backup of my application and store the files either, within a cloud storage solution, or via another drive locally which would ensure the security of the application, even if an error were to occur during the process of pushing the changes to GitHub.
2.	Incomplete connection to the relveant database within the MySQL Workbench application.	Could result in the final build of the application being unable to retreive the relevant data that is required to create entities such as customers, items, and orders.	High	High	Myself	I would either, have to troubleshoot the connection between the exisiting database within MySQL Workbench and my Java application in an attempt to create/restore functionality, or I would have to create an entirely new database, perform a test to ensure my Java application would be able to connect to it, and transfer the data from my old database to that of the newly created database.	In an attempt to prevent such events from transpiring, before inserting data into the tables created within the MySQL application, I would first ensure the connection to my Java application was live and stable by creating a test table that included a very small amount of test data that would be used to confirm whether data was able to be retrieved successfully. This will prevent a loss of development time and resources should an error in establishing a successful connection occur during the initial development phase.
3	The servers that support the GitHub platform could experience technical difficulties and go down for a period of time.	Could result in any changes that I make to my Java application/MySQL database being unable to be uploaded to my GitHub respository, resulting in the only version of my inventory management system that is able to be pulled is that of an incomplete nature.	Low	High	GitHub/ GitHub's cloud provider	I would have to ensure all of the changes I had made to my application up until that point had been saved locally to my machine or alternatively, to an alternative cloud service provider. From there, if another developer on my team wanted to access the work I had completed up until that point, they would either have to receive the files via email which would not be very secure, or I would have to transfer the relevant files to a drive and physically deliver them to him/her.	Although my ability to control the events that directly affect GitHub are rather limited, I could make efforts to ensure all of my files are updated, and backed up locally. By doing this, I could possibly upload them to an alternative open source development repository. Although, this response would have its own risks associated with it. For example, the relative security of the alternative repository service may not have been fully verified to the extent that GitHub has. Therefore, the best course of action may be to exchange files locally, although this method of addressing the issue may not be suitable for large companies that operate at a national or even international level.
4	Both the JUnit and Mockito testing processes could highlight some unforseen errors that I was not expecting	Could result in additional time needing to be allocated for the project in order for the bugs, errors, and exceptions that are likely to be thrown up by the testing process, to be addressed.	High	Medium	Myself	I would allocate a period of time for addressing the bugs and errors that were thrown up as a result of the JUnit and Mockito testing processes. In a scenario in which we were tasked to produce an inventory management system as a team, I might even be inclined to delegate this process to a specific sub team, as they would be able to troubleshoot the application from a fresh perspective having not previously worked on the code. Sometimes having a fresh pair of eyes look over a block of code may reveal an issue or even a solution you may not have thought of prior. This is particularly relevant when discussing syntactical errors.	As the sole developer on this project, it is my responsibilty to manage my time appropriately to ensure there is a suitable window to address and fix any issues with the application that may be highlighted as a result of the Junit and Mockito tests. Within the professional working environment, unforeseen issues are often a common occurence within the sector however, it is ultimately the responsibilty of the developer to make the issues known to the relevant line manager and either, ask for an extension to the deadline to allow for more time to work on the project, or to commit to a period of time within the development cycle that is utilised solely for addressing bugs and errors within the code.
5	As the sole developer working on my personal inventory management system, there is an acute risk of me contracting covid-19 and being unable to complete the work required of me to a suitable standard.	Could result in the project not being submitted before the agreed deadline, or potentially submitted without thorough testing of the application having been completed, which may result in a product being delivered that is riddled with bugs in its current state	Medium	Medium	Myself	If I were working within a professional environment that usually required me to travel to the office to complete my work (this may be due to security concerns regarding equipment and networks utilsed) I may ask my line manager if it would be possible to work from home for a breif period of time. This may include tasks that are deemed as non sensitive in regards to security.	In terms of limiting the risk of myslf contracting covid-19, the entire learning curriculum is based online meaning that person-to-person contact is extremely limited. This reduces the risk of contracting covid-19 significantly and should result in the project being completed as scheduled.

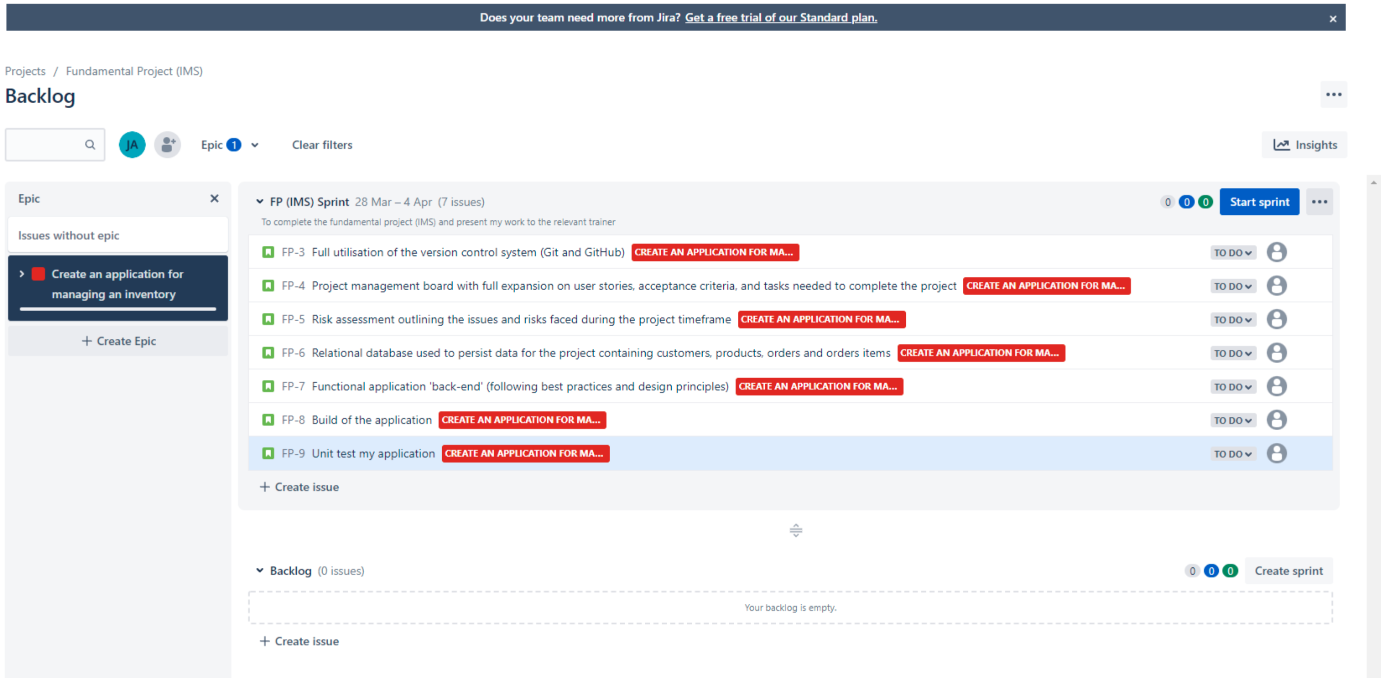
Consultant Journey

What technologies have you learned for this project?

- Throughout this project, we have utilised a variety of different technologies and applications to aid us in completing the inventory management system, with the main ones being:
- Jira
- MySQL Workbench
- Eclipse IDE
- Java

What technologies have you learned for this project?

- I personally used Jira Software and in particular, Jira boards to aid in the planning of my project by creating a series of user stories and issues that would need to be addressed in order to complete the project
- These were then placed within a sprint environment which served as a method for scheduling what tasks needed to be completed for that particular 1-4 week period
- As we were allotted a time frame of one week to complete our projects, we had one sprint comprised of a single week period to address all of the relevant issues


Does your team need more from Jira? Get a free trial of our Standard plan.

Projects / Fundamental Project (IMS)

Backlog

Epic 1 · Clear filters · Insights

Epic
Issues without epic
Create an application for managing an inventory
+ Create Epic

FP (IMS) Sprint 28 Mar – 4 Apr (7 issues) — 0 0 0 — Start sprint

To complete the fundamental project (IMS) and present my work to the relevant trainer

FP-3	Full utilisation of the version control system (Git and GitHub)	CREATE AN APPLICATION FOR MA...	TO DO
FP-4	Project management board with full expansion on user stories, acceptance criteria, and tasks needed to complete the project	CREATE AN APPLICATION FOR MA...	TO DO
FP-5	Risk assessment outlining the issues and risks faced during the project timeframe	CREATE AN APPLICATION FOR MA...	TO DO
FP-6	Relational database used to persist data for the project containing customers, products, orders and orders items	CREATE AN APPLICATION FOR MA...	TO DO
FP-7	Functional application 'back-end' (following best practices and design principles)	CREATE AN APPLICATION FOR MA...	TO DO
FP-8	Build of the application	CREATE AN APPLICATION FOR MA...	TO DO
FP-9	Unit test my application	CREATE AN APPLICATION FOR MA...	TO DO

+ Create issue

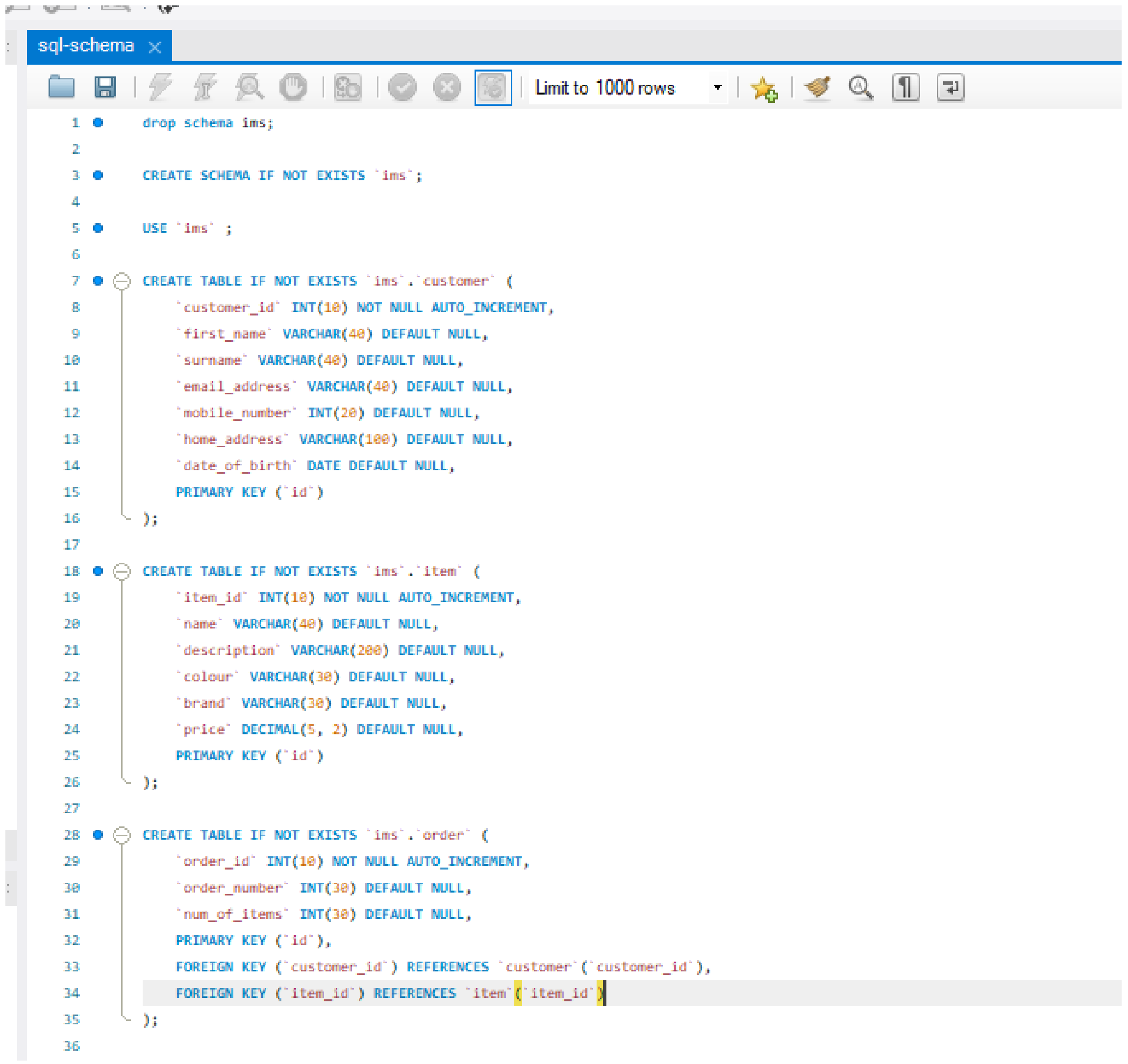
Backlog (0 issues) — 0 0 0 — Create sprint

Your backlog is empty.

+ Create issue

What technologies have you learned for this project?

- MySQL Workbench was used for formulating our tables and holding all of the relevant data that would be needed for the IMS



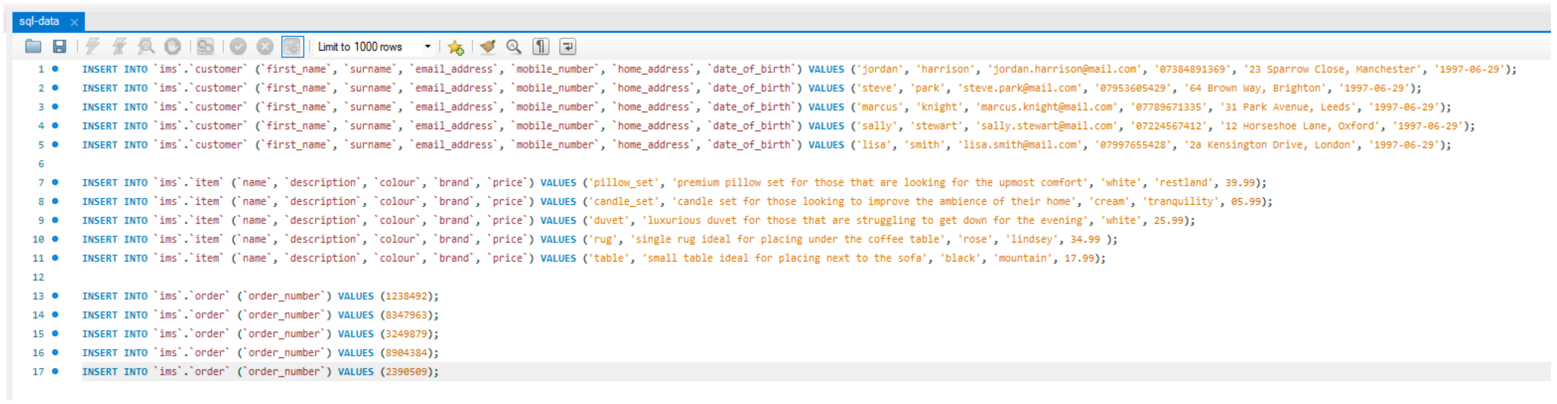
sql-schema

Limit to 1000 rows

```sql
drop schema ims;

CREATE SCHEMA IF NOT EXISTS `ims`;

USE `ims` ;

CREATE TABLE IF NOT EXISTS `ims`.`customer` (
    `customer_id` INT(10) NOT NULL AUTO_INCREMENT,
    `first_name` VARCHAR(40) DEFAULT NULL,
    `surname` VARCHAR(40) DEFAULT NULL,
    `email_address` VARCHAR(40) DEFAULT NULL,
    `mobile_number` INT(20) DEFAULT NULL,
    `home_address` VARCHAR(100) DEFAULT NULL,
    `date_of_birth` DATE DEFAULT NULL,
    PRIMARY KEY (`id`)
);

CREATE TABLE IF NOT EXISTS `ims`.`item` (
    `item_id` INT(10) NOT NULL AUTO_INCREMENT,
    `name` VARCHAR(40) DEFAULT NULL,
    `description` VARCHAR(200) DEFAULT NULL,
    `colour` VARCHAR(30) DEFAULT NULL,
    `brand` VARCHAR(30) DEFAULT NULL,
    `price` DECIMAL(5, 2) DEFAULT NULL,
    PRIMARY KEY (`id`)
);

CREATE TABLE IF NOT EXISTS `ims`.`order` (
    `order_id` INT(10) NOT NULL AUTO_INCREMENT,
    `order_number` INT(30) DEFAULT NULL,
    `num_of_items` INT(30) DEFAULT NULL,
    PRIMARY KEY (`id`),
    FOREIGN KEY (`customer_id`) REFERENCES `customer`(`customer_id`),
    FOREIGN KEY (`item_id`) REFERENCES `item`(`item_id`)
);
```

What technologies have you learned for this project?

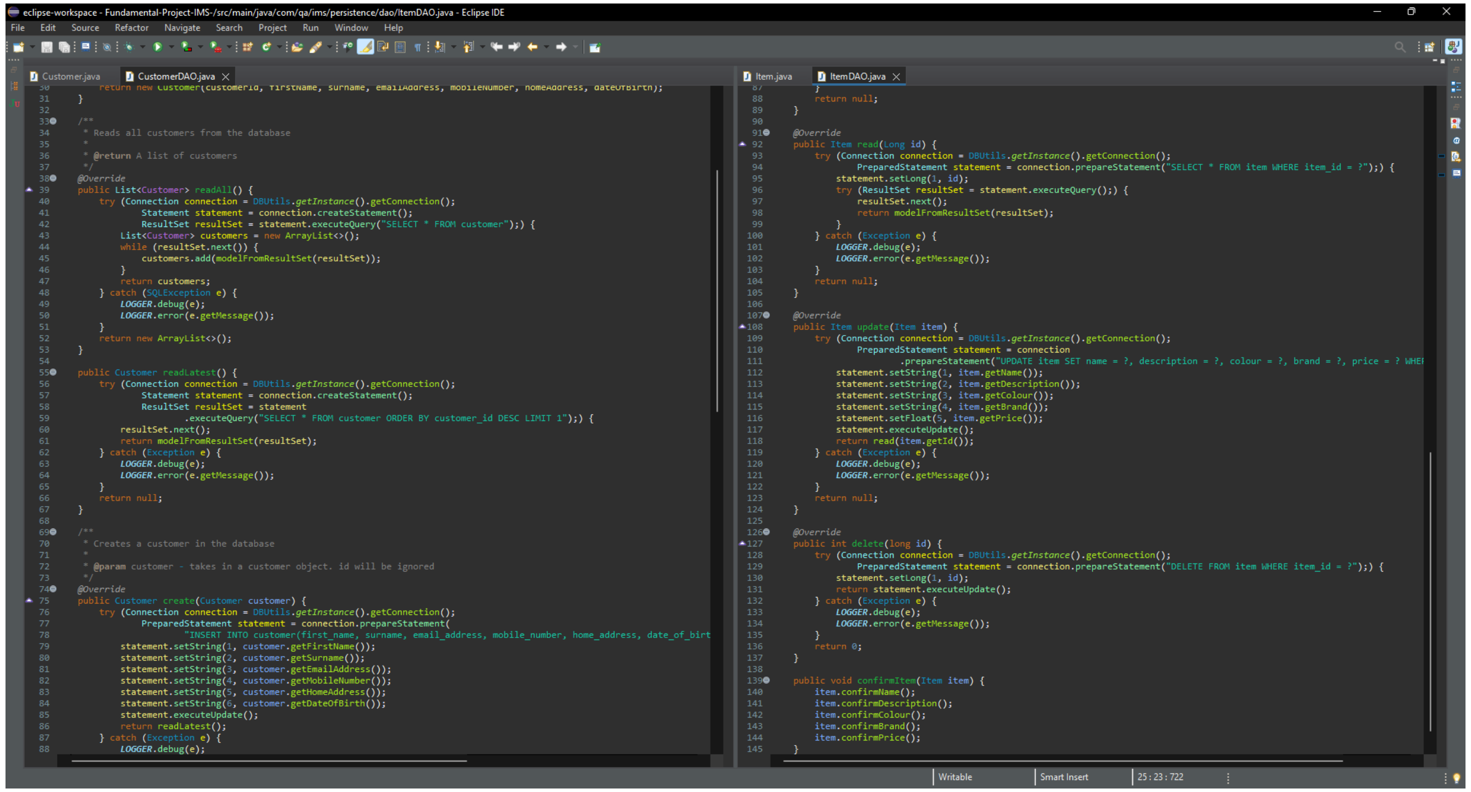
- It was primarily comprised of two separate files, one file for holding the tables themselves, and one table for holding the data that was due to be inserted



```sql
INSERT INTO `ims`.`customer` (`first_name`, `surname`, `email_address`, `mobile_number`, `home_address`, `date_of_birth`) VALUES ('jordan', 'harrison', 'jordan.harrison@mail.com', '07384891369', '23 Sparrow Close, Manchester', '1997-06-29');
INSERT INTO `ims`.`customer` (`first_name`, `surname`, `email_address`, `mobile_number`, `home_address`, `date_of_birth`) VALUES ('steve', 'park', 'steve.park@mail.com', '07953605429', '64 Brown Way, Brighton', '1997-06-29');
INSERT INTO `ims`.`customer` (`first_name`, `surname`, `email_address`, `mobile_number`, `home_address`, `date_of_birth`) VALUES ('marcus', 'knight', 'marcus.knight@mail.com', '07789671335', '31 Park Avenue, Leeds', '1997-06-29');
INSERT INTO `ims`.`customer` (`first_name`, `surname`, `email_address`, `mobile_number`, `home_address`, `date_of_birth`) VALUES ('sally', 'stewart', 'sally.stewart@mail.com', '07224567412', '12 Horseshoe Lane, Oxford', '1997-06-29');
INSERT INTO `ims`.`customer` (`first_name`, `surname`, `email_address`, `mobile_number`, `home_address`, `date_of_birth`) VALUES ('lisa', 'smith', 'lisa.smith@mail.com', '07997655428', '2a Kensington Drive, London', '1997-06-29');

INSERT INTO `ims`.`item` (`name`, `description`, `colour`, `brand`, `price`) VALUES ('pillow_set', 'premium pillow set for those that are looking for the upmost comfort', 'white', 'restland', 39.99);
INSERT INTO `ims`.`item` (`name`, `description`, `colour`, `brand`, `price`) VALUES ('candle_set', 'candle set for those looking to improve the ambience of their home', 'cream', 'tranquility', 05.99);
INSERT INTO `ims`.`item` (`name`, `description`, `colour`, `brand`, `price`) VALUES ('duvet', 'luxurious duvet for those that are struggling to get down for the evening', 'white', 25.99);
INSERT INTO `ims`.`item` (`name`, `description`, `colour`, `brand`, `price`) VALUES ('rug', 'single rug ideal for placing under the coffee table', 'rose', 'lindsey', 34.99 );
INSERT INTO `ims`.`item` (`name`, `description`, `colour`, `brand`, `price`) VALUES ('table', 'small table ideal for placing next to the sofa', 'black', 'mountain', 17.99);

INSERT INTO `ims`.`order` (`order_number`) VALUES (1238492);
INSERT INTO `ims`.`order` (`order_number`) VALUES (8347963);
INSERT INTO `ims`.`order` (`order_number`) VALUES (3249879);
INSERT INTO `ims`.`order` (`order_number`) VALUES (8904384);
INSERT INTO `ims`.`order` (`order_number`) VALUES (2390509);
```

What technologies have you learned for this project?

- The Eclipse IDE served as our development environment for creating the Java application
- The development language utilised for this project was Java
- The selection of Java as our primary programming language for this project allowed us to become well versed within the realm of object oriented programming (OOP) principles, as well as the SOLID principles, which largely influence the manner in which our code is structured


eclipse-workspace - Fundamental-Project-IMS-/src/main/java/com/qa/ims/persistence/dao/ItemDAO.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Customer.java
CustomerDAO.java
Item.java
ItemDAO.java
Writable
Smart Insert
25 : 23 : 722

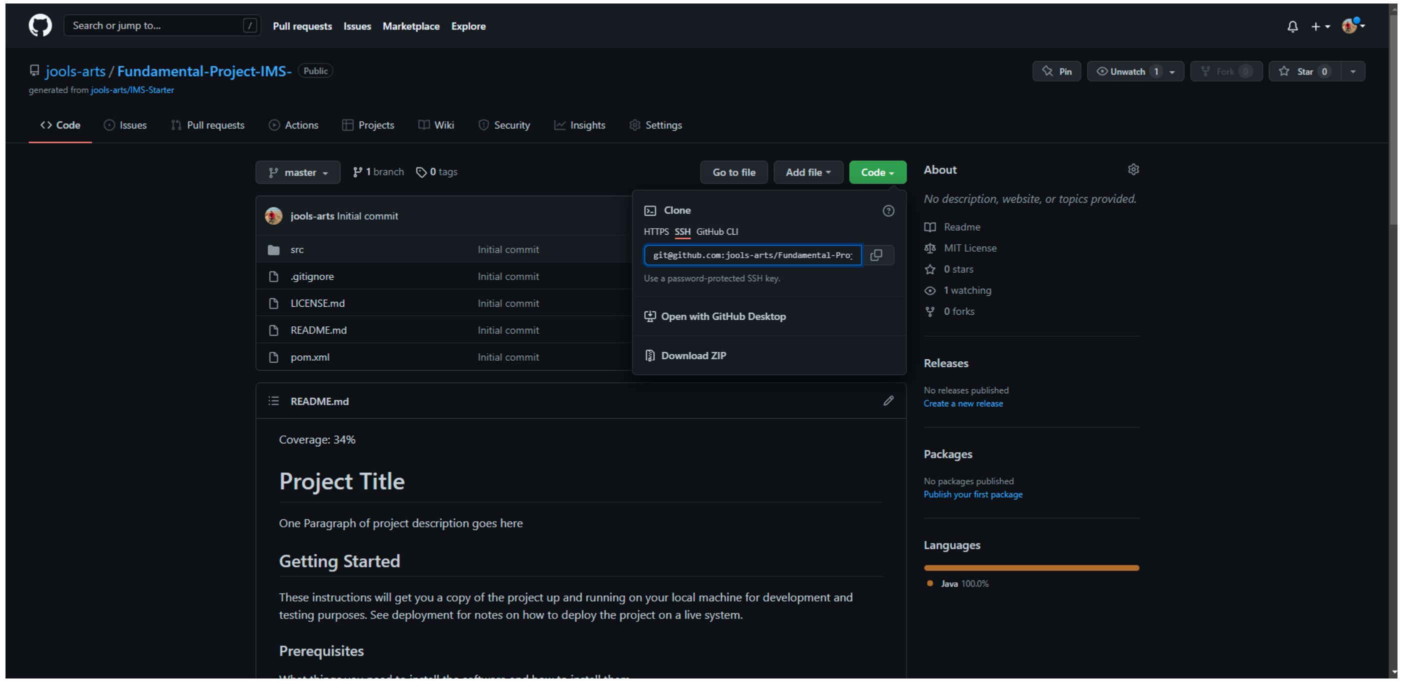
```java
        return new Customer(customerId, firstName, surname, emailAddress, mobileNumber, homeAddress, dateOfBirth);
    }

    /**
     * Reads all customers from the database
     *
     * @return A list of customers
     */
    @Override
    public List<Customer> readAll() {
        try (Connection connection = DBUtils.getInstance().getConnection();
                Statement statement = connection.createStatement();
                ResultSet resultSet = statement.executeQuery("SELECT * FROM customer");) {
            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<>();
    }

    public Customer readLatest() {
        try (Connection connection = DBUtils.getInstance().getConnection();
                Statement statement = connection.createStatement();
                ResultSet resultSet = statement
                        .executeQuery("SELECT * FROM customer ORDER BY customer_id DESC LIMIT 1");) {
            resultSet.next();
            return modelFromResultSet(resultSet);
        } catch (Exception e) {
            LOGGER.debug(e);
            LOGGER.error(e.getMessage());
        }
        return null;
    }

    /**
     * Creates a customer in the database
     *
     * @param customer - takes in a customer object. id will be ignored
     */
    @Override
    public Customer create(Customer customer) {
        try (Connection connection = DBUtils.getInstance().getConnection();
                PreparedStatement statement = connection.prepareStatement(
                        "INSERT INTO customer(first_name, surname, email_address, mobile_number, home_address, date_of_birt
            statement.setString(1, customer.getFirstName());
            statement.setString(2, customer.getSurname());
            statement.setString(3, customer.getEmailAddress());
            statement.setString(4, customer.getMobileNumber());
            statement.setString(5, customer.getHomeAddress());
            statement.setString(6, customer.getDateOfBirth());
            statement.executeUpdate();
            return readLatest();
        } catch (Exception e) {
            LOGGER.debug(e);
```

```java
        }
        return null;
    }

    @Override
    public Item read(Long id) {
        try (Connection connection = DBUtils.getInstance().getConnection();
                PreparedStatement statement = connection.prepareStatement("SELECT * FROM item WHERE item_id = ?");) {
            statement.setLong(1, id);
            try (ResultSet resultSet = statement.executeQuery();) {
                resultSet.next();
                return modelFromResultSet(resultSet);
            }
        } catch (Exception e) {
            LOGGER.debug(e);
            LOGGER.error(e.getMessage());
        }
        return null;
    }

    @Override
    public Item update(Item item) {
        try (Connection connection = DBUtils.getInstance().getConnection();
                PreparedStatement statement = connection
                        .prepareStatement("UPDATE item SET name = ?, description = ?, colour = ?, brand = ?, price = ? WHE
            statement.setString(1, item.getName());
            statement.setString(2, item.getDescription());
            statement.setString(3, item.getColour());
            statement.setString(4, item.getBrand());
            statement.setFloat(5, item.getPrice());
            statement.executeUpdate();
            return read(item.getId());
        } catch (Exception e) {
            LOGGER.debug(e);
            LOGGER.error(e.getMessage());
        }
        return null;
    }

    @Override
    public int delete(long id) {
        try (Connection connection = DBUtils.getInstance().getConnection();
                PreparedStatement statement = connection.prepareStatement("DELETE FROM item WHERE item_id = ?");) {
            statement.setLong(1, id);
            return statement.executeUpdate();
        } catch (Exception e) {
            LOGGER.debug(e);
            LOGGER.error(e.getMessage());
        }
        return 0;
    }

    public void confirmItem(Item item) {
        item.confirmName();
        item.confirmDescription();
        item.confirmColour();
        item.confirmBrand();
        item.confirmPrice();
    }
```

Continuous Integration

How did you approach version control?

- Before embarking upon completing the tasks as specified by the project specification, the first port of call was to clone the relevant project files from the GitHub repository
- This allowed me to work on all of the files I would need for the project, on my local machine, which I could then access via applications such as Eclipse, and MySQL Workbench



jools-arts / Fundamental-Project-IMS- Public

generated from jools-arts/IMS-Starter

jools-arts Initial commit

src	Initial commit
.gitignore	Initial commit
LICENSE.md	Initial commit
README.md	Initial commit
pom.xml	Initial commit

```
git@github.com:jools-arts/Fundamental-Pro;
```

Use a password-protected SSH key.

README.md

Coverage: 34%

Project Title

One Paragraph of project description goes here

Getting Started

These instructions will get you a copy of the project up and running on your local machine for development and testing purposes. See deployment for notes on how to deploy the project on a live system.

Prerequisites

About

No description, website, or topics provided.

Readme
MIT License
0 stars
1 watching
0 forks

Releases

No releases published
Create a new release

Packages

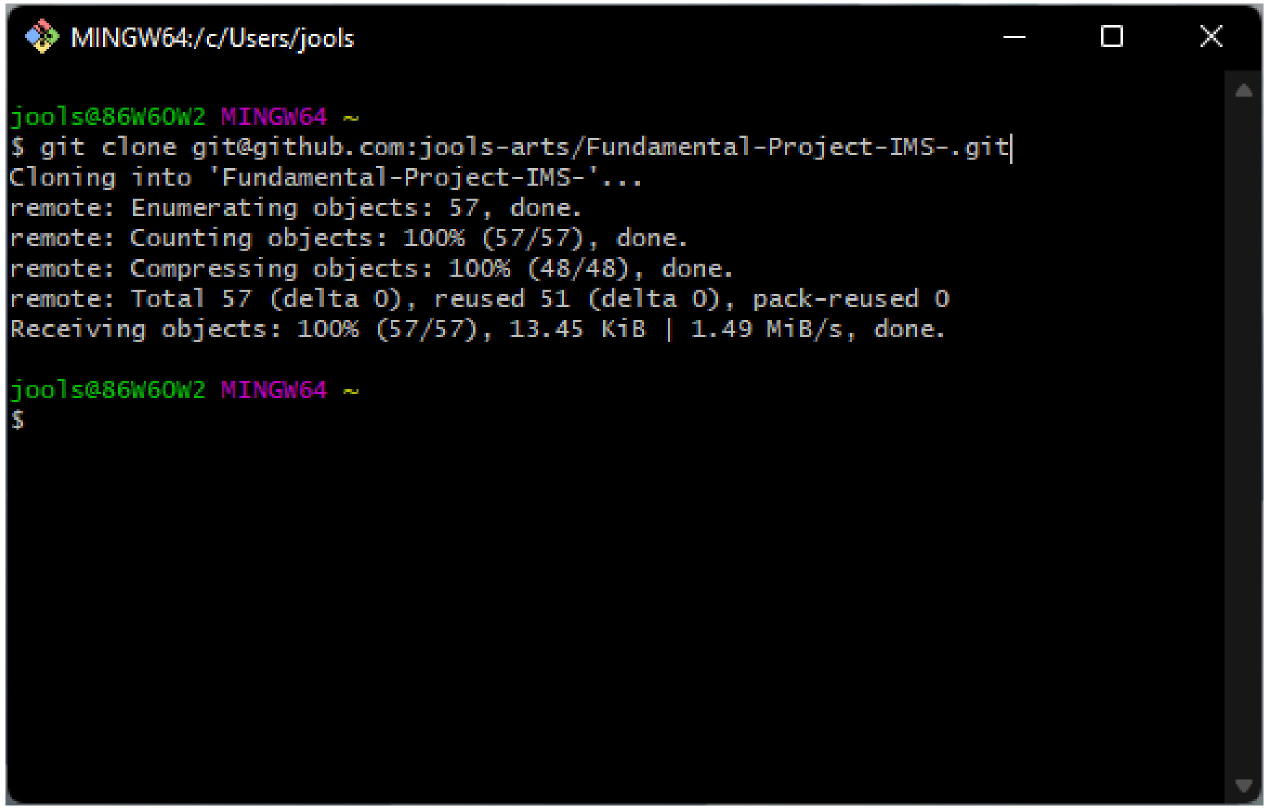
No packages published
Publish your first package

Languages

Java 100.0%

How did you approach version control?

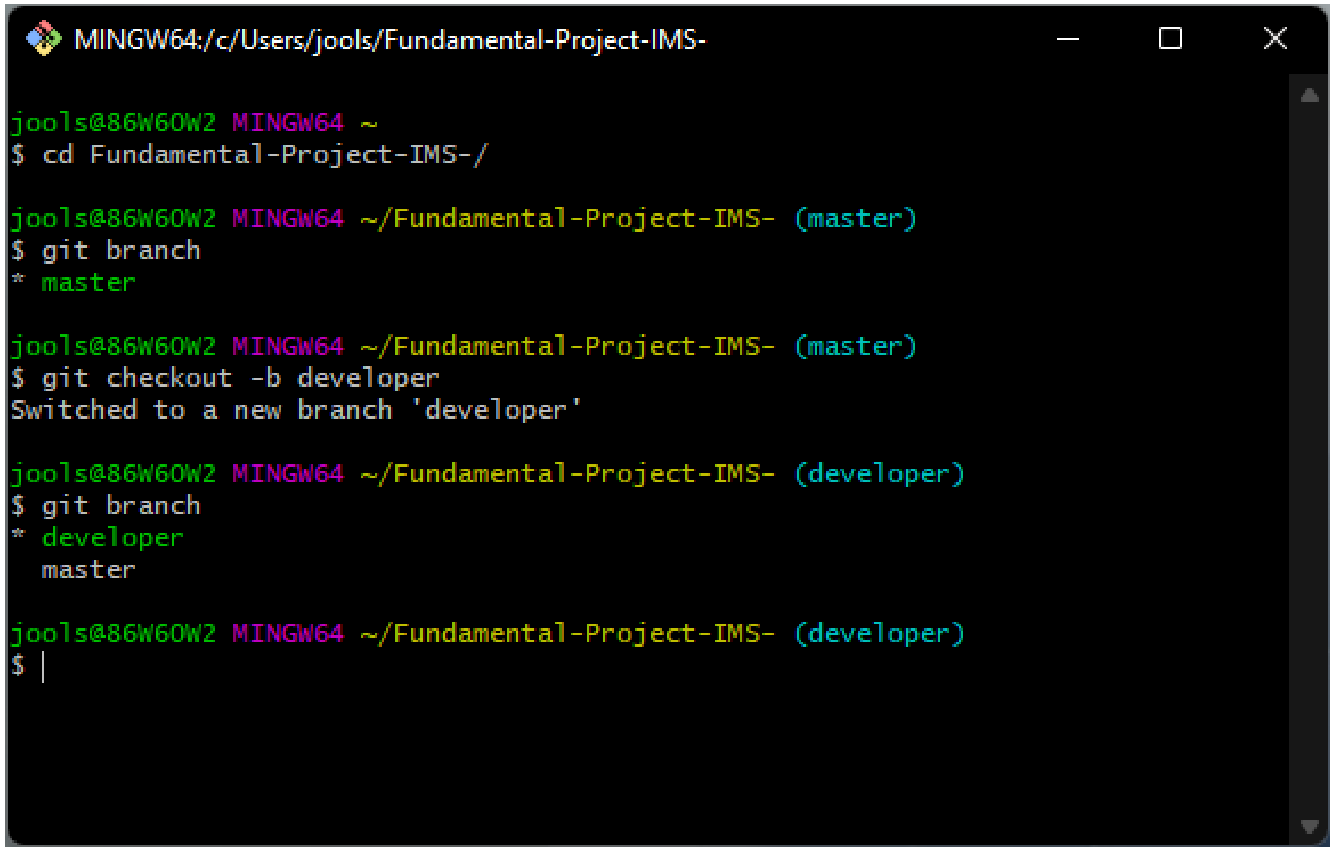
- I utilised the Git Bash terminal, along with the git clone command to pull a copy of the files I would need to work on for the project
- This involved using a password protected SSH key



```
jools@86W6OW2 MINGW64 ~
$ git clone git@github.com:jools-arts/Fundamental-Project-IMS-.git
Cloning into 'Fundamental-Project-IMS-'...
remote: Enumerating objects: 57, done.
remote: Counting objects: 100% (57/57), done.
remote: Compressing objects: 100% (48/48), done.
remote: Total 57 (delta 0), reused 51 (delta 0), pack-reused 0
Receiving objects: 100% (57/57), 13.45 KiB | 1.49 MiB/s, done.

jools@86W6OW2 MINGW64 ~
$
```

How did you approach version control?

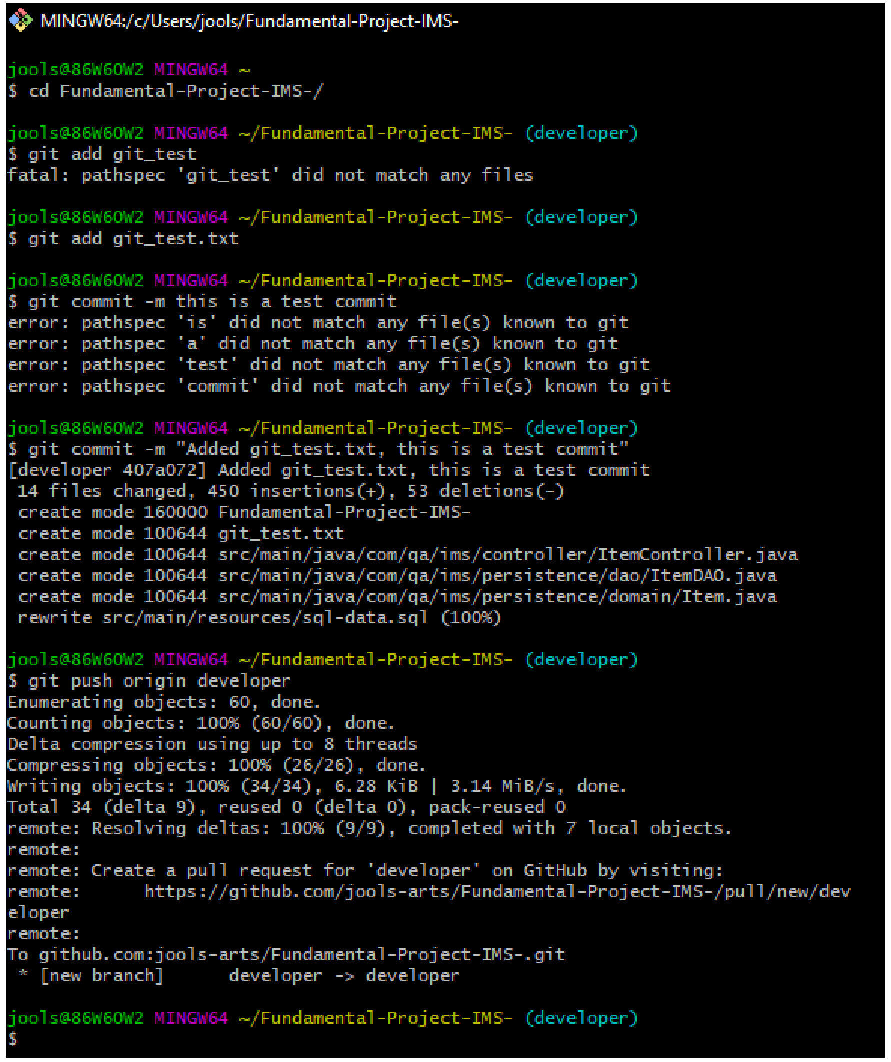
- The next stage involved creating a developer branch which I would be able add, commit, and push all of the changes to my code base to, before then merging them into my master branch
- Once I had checked the requested changes had been verified by myself, I would then create a pull request which would then allow me to merge my new changes into my master branch
- The feature branch model allows the user to upload changes to a specific version of the codebase, one which doesn't affect the master branch, this allows for a great degree of freedom as one can trial new features and changes without the fear of upsetting the stability of the main source code


MINGW64:/c/Users/jools/Fundamental-Project-IMS-

```
jools@86W60W2 MINGW64 ~
$ cd Fundamental-Project-IMS-/

jools@86W60W2 MINGW64 ~/Fundamental-Project-IMS- (master)
$ git branch
* master

jools@86W60W2 MINGW64 ~/Fundamental-Project-IMS- (master)
$ git checkout -b developer
Switched to a new branch 'developer'

jools@86W60W2 MINGW64 ~/Fundamental-Project-IMS- (developer)
$ git branch
* developer
  master

jools@86W60W2 MINGW64 ~/Fundamental-Project-IMS- (developer)
$ |
```

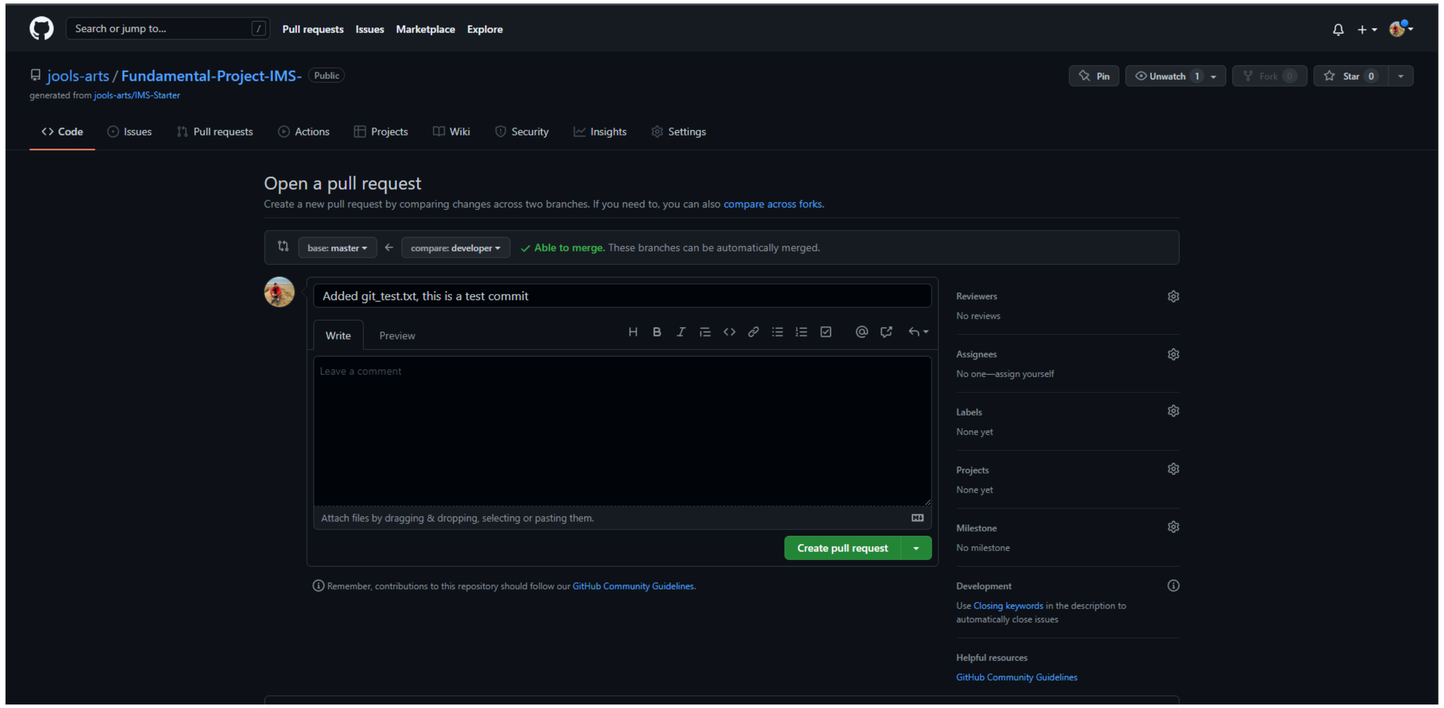
How did you approach version control?

- Once I had the feature branch model set up and ready to use, it was time to make my first commit
- With both the Git and GitHub technologies still being relatively new to myself, I thought it would be sensible to first conduct a test commit to ensure I was performing the process correctly
- This would alleviate some of the risk of potentially sabotaging my main project if something were to go wrong, or I was to make an error
- I therefore created a very simple .txt file containing the phrase "this is a test file", added it to the staging area, committed the file, and then pushed it to the developer branch of my GitHub repository


```
MINGW64:/c/Users/jools/Fundamental-Project-IMS-

jools@86W60W2 MINGW64 ~
$ cd Fundamental-Project-IMS-/

jools@86W60W2 MINGW64 ~/Fundamental-Project-IMS- (developer)
$ git add git_test
fatal: pathspec 'git_test' did not match any files

jools@86W60W2 MINGW64 ~/Fundamental-Project-IMS- (developer)
$ git add git_test.txt

jools@86W60W2 MINGW64 ~/Fundamental-Project-IMS- (developer)
$ git commit -m this is a test commit
error: pathspec 'is' did not match any file(s) known to git
error: pathspec 'a' did not match any file(s) known to git
error: pathspec 'test' did not match any file(s) known to git
error: pathspec 'commit' did not match any file(s) known to git

jools@86W60W2 MINGW64 ~/Fundamental-Project-IMS- (developer)
$ git commit -m "Added git_test.txt, this is a test commit"
[developer 407a072] Added git_test.txt, this is a test commit
 14 files changed, 450 insertions(+), 53 deletions(-)
 create mode 160000 Fundamental-Project-IMS-
 create mode 100644 git_test.txt
 create mode 100644 src/main/java/com/qa/ims/controller/ItemController.java
 create mode 100644 src/main/java/com/qa/ims/persistence/dao/ItemDAO.java
 create mode 100644 src/main/java/com/qa/ims/persistence/domain/Item.java
 rewrite src/main/resources/sql-data.sql (100%)

jools@86W60W2 MINGW64 ~/Fundamental-Project-IMS- (developer)
$ git push origin developer
Enumerating objects: 60, done.
Counting objects: 100% (60/60), done.
Delta compression using up to 8 threads
Compressing objects: 100% (26/26), done.
Writing objects: 100% (34/34), 6.28 KiB | 3.14 MiB/s, done.
Total 34 (delta 9), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (9/9), completed with 7 local objects.
remote:
remote: Create a pull request for 'developer' on GitHub by visiting:
remote:      https://github.com/jools-arts/Fundamental-Project-IMS-/pull/new/dev
eloper
remote:
To github.com:jools-arts/Fundamental-Project-IMS-.git
 * [new branch]      developer -> developer

jools@86W60W2 MINGW64 ~/Fundamental-Project-IMS- (developer)
$
```

How did you approach version control?

- I then launched GitHub and noticed a small notification notifying me of my recently proposed changes
- I then opted to create a pull request which would then allow me to merge my new test file with that of the master branch



jools-arts / Fundamental-Project-IMS- Public
generated from jools-arts/IMS-Starter

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: master ← compare: developer ✓ Able to merge. These branches can be automatically merged.

Added git_test.txt, this is a test commit

Leave a comment

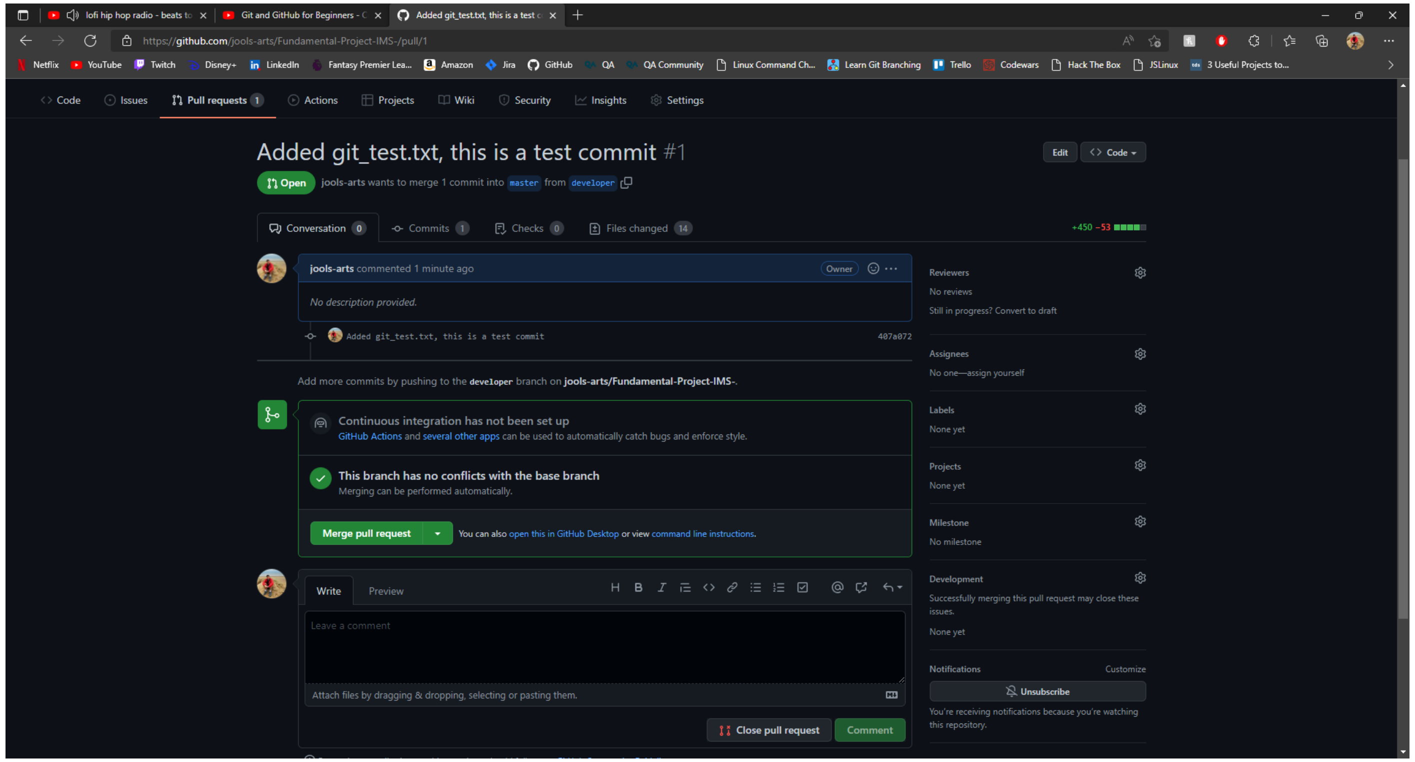
Attach files by dragging & dropping, selecting or pasting them.

Remember, contributions to this repository should follow our GitHub Community Guidelines.

Reviewers: No reviews
Assignees: No one—assign yourself
Labels: None yet
Projects: None yet
Milestone: No milestone
Development: Use Closing keywords in the description to automatically close issues
Helpful resources: GitHub Community Guidelines

How did you approach version control?

- Upon creating the pull request, I was then presented with the message notifying me there were no merge conflicts present or in need of rectifying
- This was of course, to be expected, given this was only a test to make sure I was performing the process correctly
- I was thus given the all clear to proceed with merging the pull request, which would generate a request for the test file to be merged into the master branch



Added git_test.txt, this is a test commit #1

Open jools-arts wants to merge 1 commit into `master` from `developer`

+450 −53

jools-arts commented 1 minute ago

No description provided.

`Added git_test.txt, this is a test commit` 407a072

Add more commits by pushing to the **developer** branch on **jools-arts/Fundamental-Project-IMS-**.

Continuous integration has not been set up
GitHub Actions and several other apps can be used to automatically catch bugs and enforce style.

This branch has no conflicts with the base branch
Merging can be performed automatically.

You can also open this in GitHub Desktop or view command line instructions.

Leave a comment

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

Reviewers
No reviews
Still in progress? Convert to draft

Assignees
No one—assign yourself

Labels
None yet

Projects
None yet

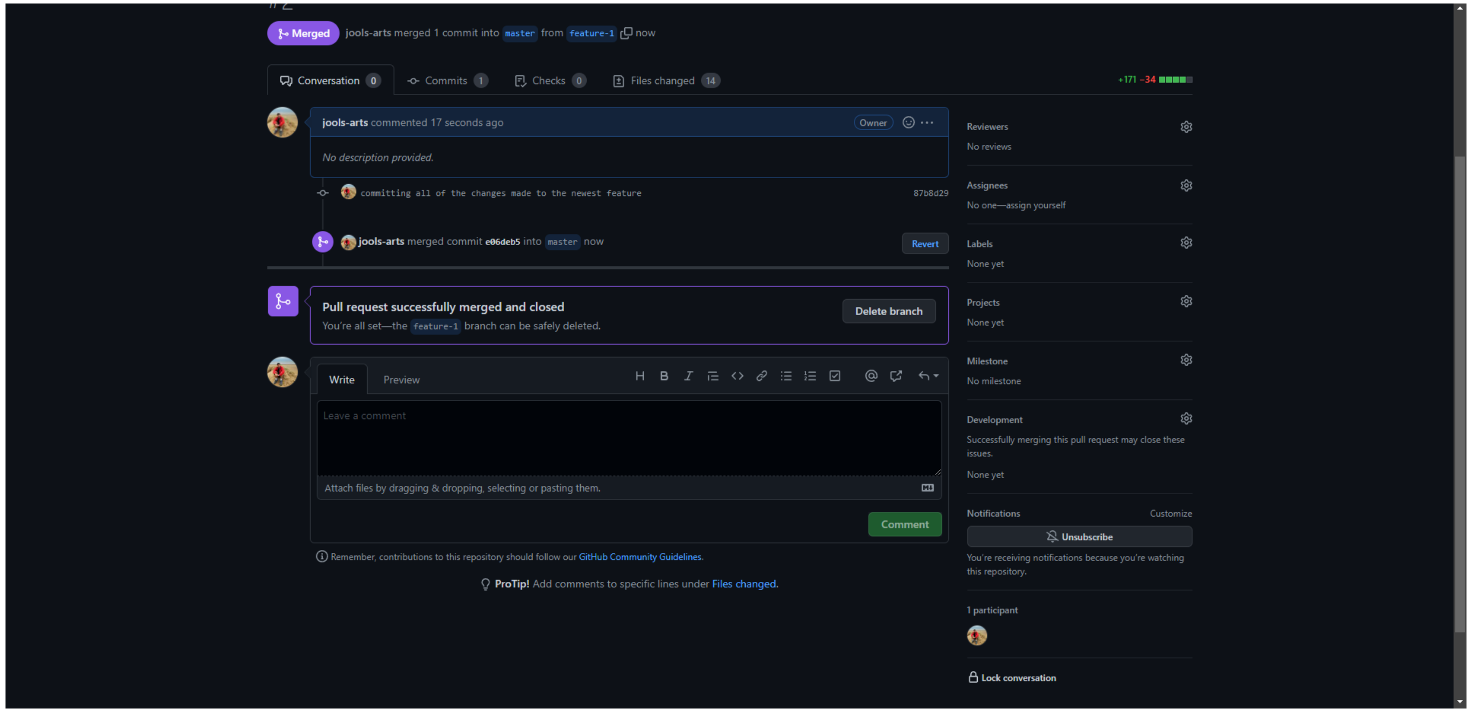
Milestone
No milestone

Development
Successfully merging this pull request may close these issues.
None yet

Notifications
You're receiving notifications because you're watching this repository.

How did you approach version control?

- Unfortunately I don’t have a screenshot of my test file being successfully merged into my master branch, however I do have evidence of another merge being successfully completed
- This was a merge that included changes that were committed to my feature branch being merged with my master branch



Merged jools-arts merged 1 commit into `master` from `feature-1` now

Conversation 0 · Commits 1 · Checks 0 · Files changed 14 · +171 −34

jools-arts commented 17 seconds ago · Owner

No description provided.

`committing all of the changes made to the newest feature` 87b8d29

jools-arts merged commit **e06deb5** into `master` now · Revert

Pull request successfully merged and closed
You’re all set—the `feature-1` branch can be safely deleted. · Delete branch

Write · Preview

Leave a comment

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

Comment

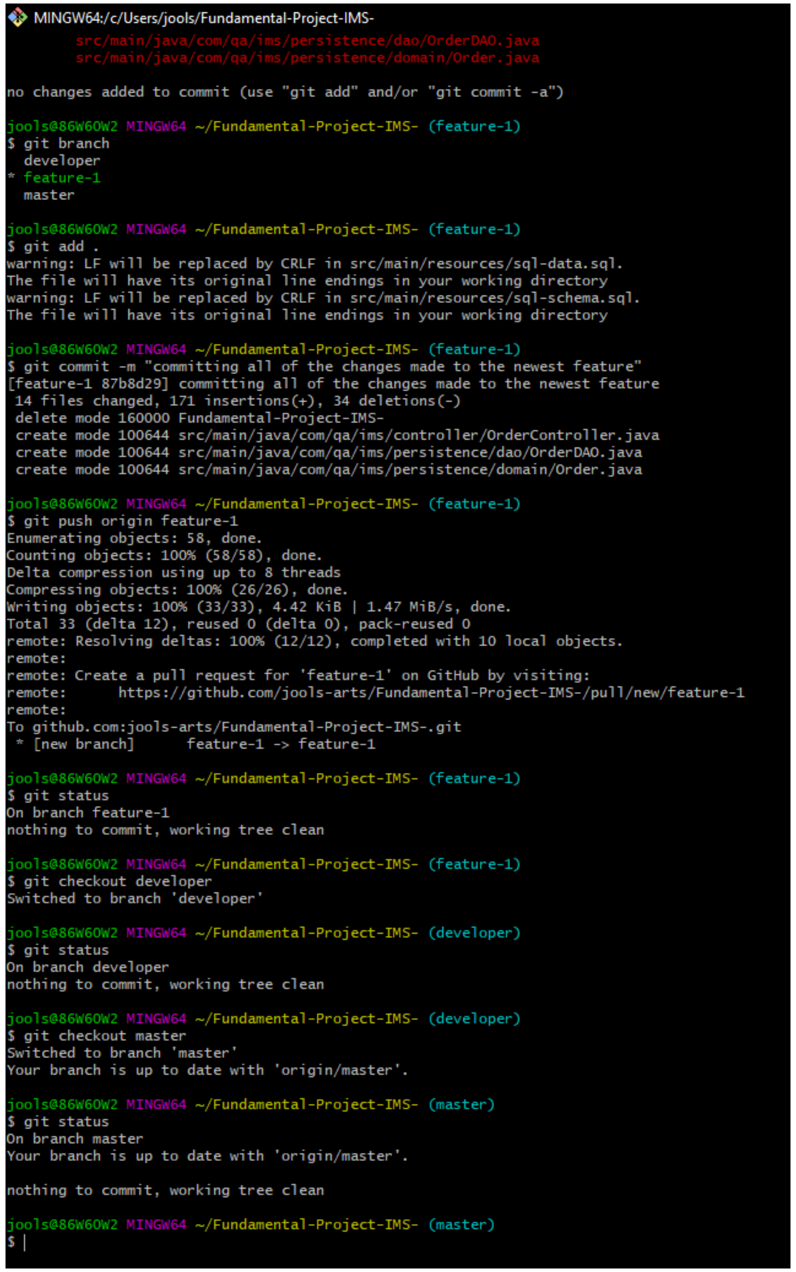
Remember, contributions to this repository should follow our GitHub Community Guidelines.

ProTip! Add comments to specific lines under Files changed.

Reviewers: No reviews
Assignees: No one—assign yourself
Labels: None yet
Projects: None yet
Milestone: No milestone
Development: Successfully merging this pull request may close these issues. None yet
Notifications: Customize · Unsubscribe · You’re receiving notifications because you’re watching this repository.
1 participant
Lock conversation

How did you approach version control?

- Once I had established the test commit had worked and I had a full and complete understanding of the feature branch model, I then felt comfortable enough to begin adding, committing and pushing my project files up to my GitHub repository
- Here you can see evidence of myself adding files to the staging area of my feature-1 branch, before committing the changes, and then pushing said changes up to the feature-1 branch of my GitHub repository
- From there, I would repeat the process of creating a pull request for the changes that had been recently committed to be merged into the master branch

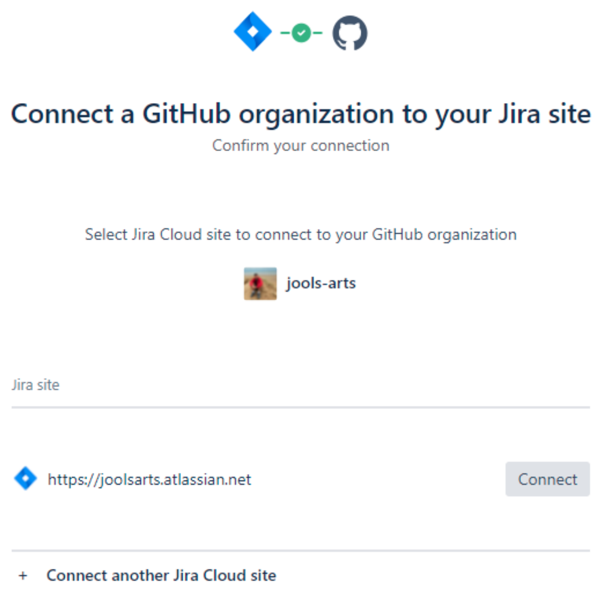


MINGW64:/c/Users/jools/Fundamental-Project-IMS-

```
        src/main/java/com/qa/ims/persistence/dao/OrderDAO.java
        src/main/java/com/qa/ims/persistence/domain/Order.java

no changes added to commit (use "git add" and/or "git commit -a")

jools@86W60W2 MINGW64 ~/Fundamental-Project-IMS- (feature-1)
$ git branch
  developer
* feature-1
  master

jools@86W60W2 MINGW64 ~/Fundamental-Project-IMS- (feature-1)
$ git add .
warning: LF will be replaced by CRLF in src/main/resources/sql-data.sql.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in src/main/resources/sql-schema.sql.
The file will have its original line endings in your working directory

jools@86W60W2 MINGW64 ~/Fundamental-Project-IMS- (feature-1)
$ git commit -m "committing all of the changes made to the newest feature"
[feature-1 87b8d29] committing all of the changes made to the newest feature
 14 files changed, 171 insertions(+), 34 deletions(-)
 delete mode 160000 Fundamental-Project-IMS-
 create mode 100644 src/main/java/com/qa/ims/controller/OrderController.java
 create mode 100644 src/main/java/com/qa/ims/persistence/dao/OrderDAO.java
 create mode 100644 src/main/java/com/qa/ims/persistence/domain/Order.java

jools@86W60W2 MINGW64 ~/Fundamental-Project-IMS- (feature-1)
$ git push origin feature-1
Enumerating objects: 58, done.
Counting objects: 100% (58/58), done.
Delta compression using up to 8 threads
Compressing objects: 100% (26/26), done.
Writing objects: 100% (33/33), 4.42 KiB | 1.47 MiB/s, done.
Total 33 (delta 12), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (12/12), completed with 10 local objects.
remote:
remote: Create a pull request for 'feature-1' on GitHub by visiting:
remote:      https://github.com/jools-arts/Fundamental-Project-IMS-/pull/new/feature-1
remote:
To github.com:jools-arts/Fundamental-Project-IMS-.git
 * [new branch]      feature-1 -> feature-1

jools@86W60W2 MINGW64 ~/Fundamental-Project-IMS- (feature-1)
$ git status
On branch feature-1
nothing to commit, working tree clean

jools@86W60W2 MINGW64 ~/Fundamental-Project-IMS- (feature-1)
$ git checkout developer
Switched to branch 'developer'

jools@86W60W2 MINGW64 ~/Fundamental-Project-IMS- (developer)
$ git status
On branch developer
nothing to commit, working tree clean

jools@86W60W2 MINGW64 ~/Fundamental-Project-IMS- (developer)
$ git checkout master
Switched to branch 'master'
Your branch is up to date with 'origin/master'.

jools@86W60W2 MINGW64 ~/Fundamental-Project-IMS- (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

nothing to commit, working tree clean

jools@86W60W2 MINGW64 ~/Fundamental-Project-IMS- (master)
$ |
```

How did you approach version control?

- I also made a specific effort to setup and utilise smart commits throughout my project
- The first part of the process was to connect my Jira account to my GitHub account



Connect a GitHub organization to your Jira site

Confirm your connection

Select Jira Cloud site to connect to your GitHub organization

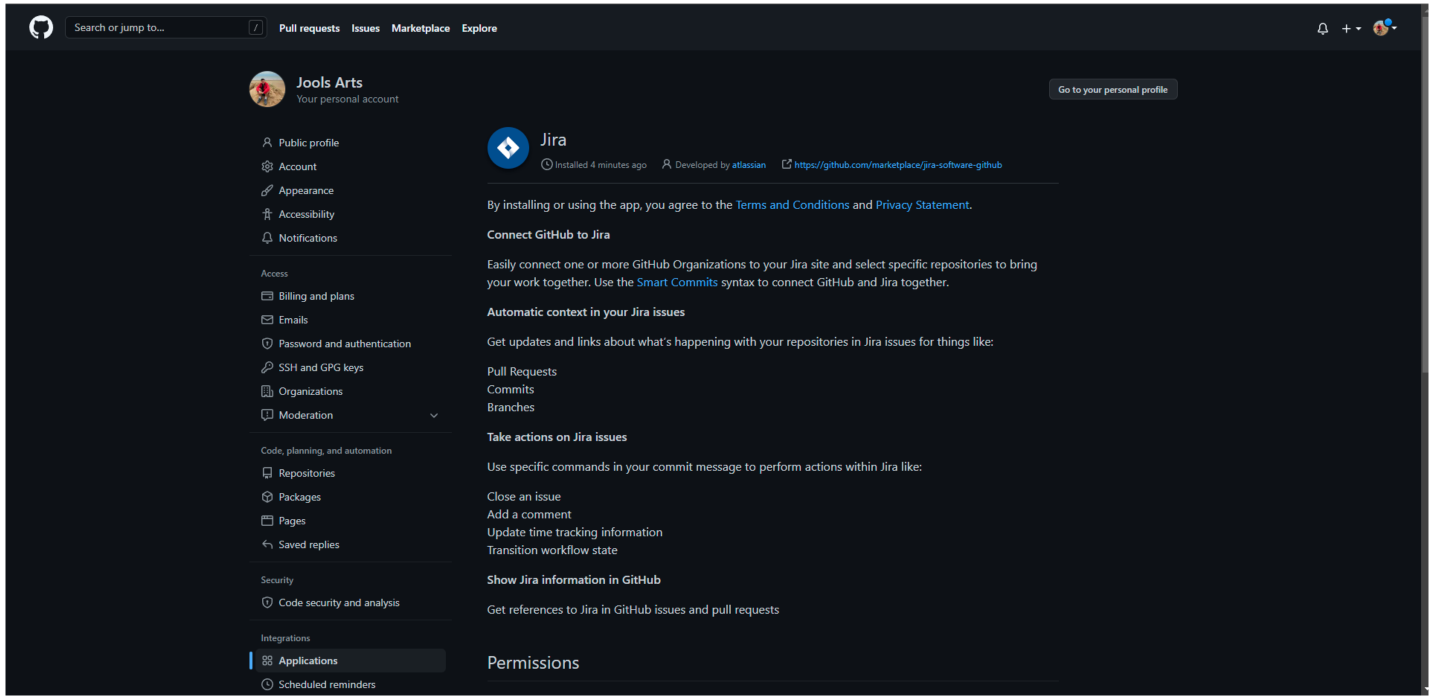
jools-arts

Jira site

https://joolsarts.atlassian.net

How did you approach version control?

- Once I had successfully connected my Jira account to my GitHub account, I was then able to start maximising the advantages that are associated with the synergy of the two technologies working in tandem with one another
- For example, smart commits allow the user to comment on issues directly, which in a professional environment, would allow multiple developers to communicate their ideas and changes far more freely than they would be able otherwise



Jira

Installed 4 minutes ago Developed by atlassian https://github.com/marketplace/jira-software-github

By installing or using the app, you agree to the Terms and Conditions and Privacy Statement.

Connect GitHub to Jira

Easily connect one or more GitHub Organizations to your Jira site and select specific repositories to bring your work together. Use the Smart Commits syntax to connect GitHub and Jira together.

Automatic context in your Jira issues

Get updates and links about what's happening with your repositories in Jira issues for things like:

Pull Requests
Commits
Branches

Take actions on Jira issues

Use specific commands in your commit message to perform actions within Jira like:

Close an issue
Add a comment
Update time tracking information
Transition workflow state

Show Jira information in GitHub

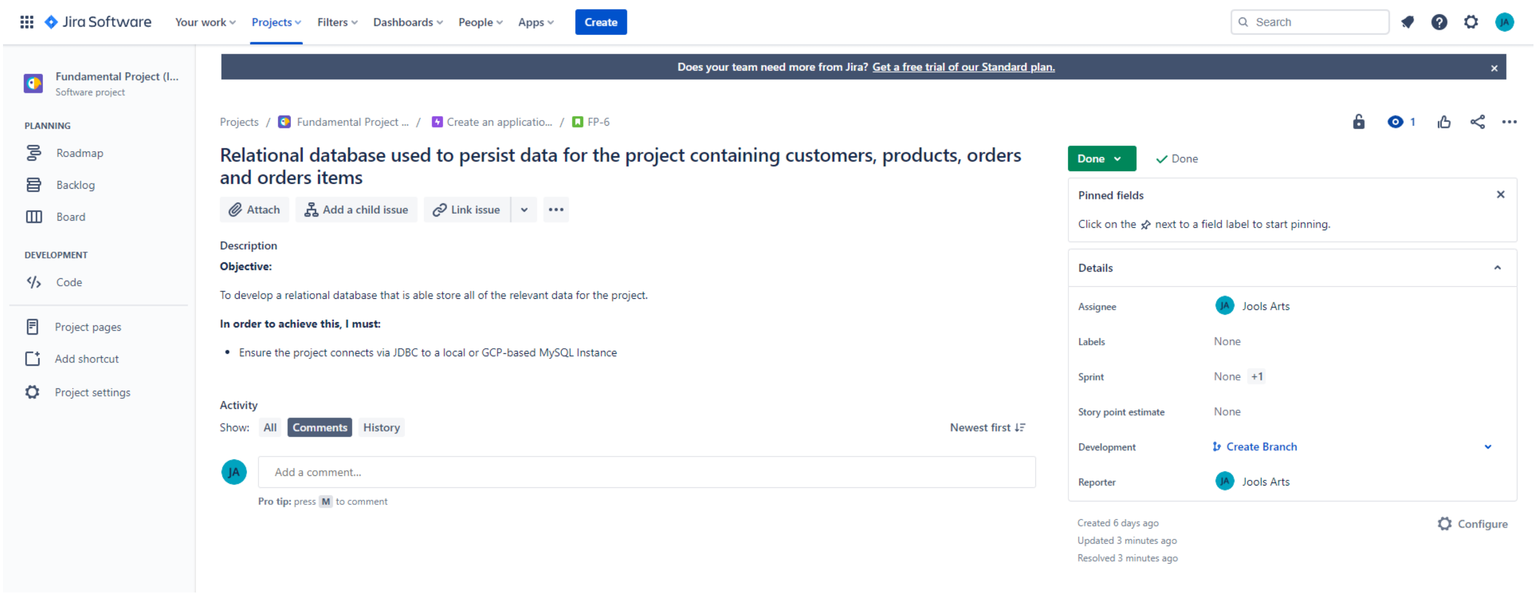
Get references to Jira in GitHub issues and pull requests

Permissions

Demonstration

Run through a couple of user stories – relational database

- I tasked myself with developing a relational database that is able to store all of the relevant data for the project
- In order to achieve this, I had to ensure my project connected to a local or GCP-based MySQL instance



Does your team need more from Jira? Get a free trial of our Standard plan.

Projects / Fundamental Project ... / Create an applicatio... / FP-6

Relational database used to persist data for the project containing customers, products, orders and orders items

Description

Objective:

To develop a relational database that is able store all of the relevant data for the project.

In order to achieve this, I must:

- Ensure the project connects via JDBC to a local or GCP-based MySQL Instance

Activity

Show: All Comments History — Newest first

Add a comment...

Pro tip: press M to comment

Pinned fields

Click on the next to a field label to start pinning.

Details

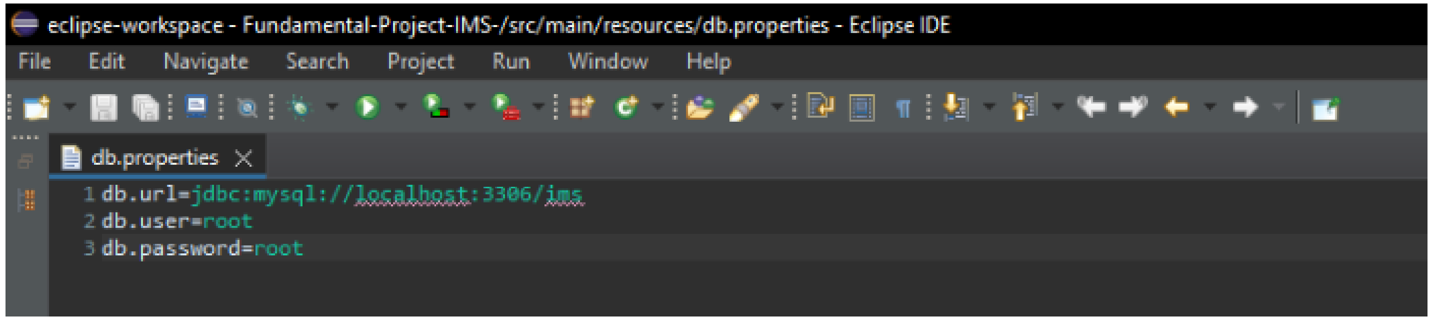
Assignee	Jools Arts
Labels	None
Sprint	None +1
Story point estimate	None
Development	Create Branch
Reporter	Jools Arts

Created 6 days ago
Updated 3 minutes ago
Resolved 3 minutes ago

Configure

Run through a couple of user stories – relational database

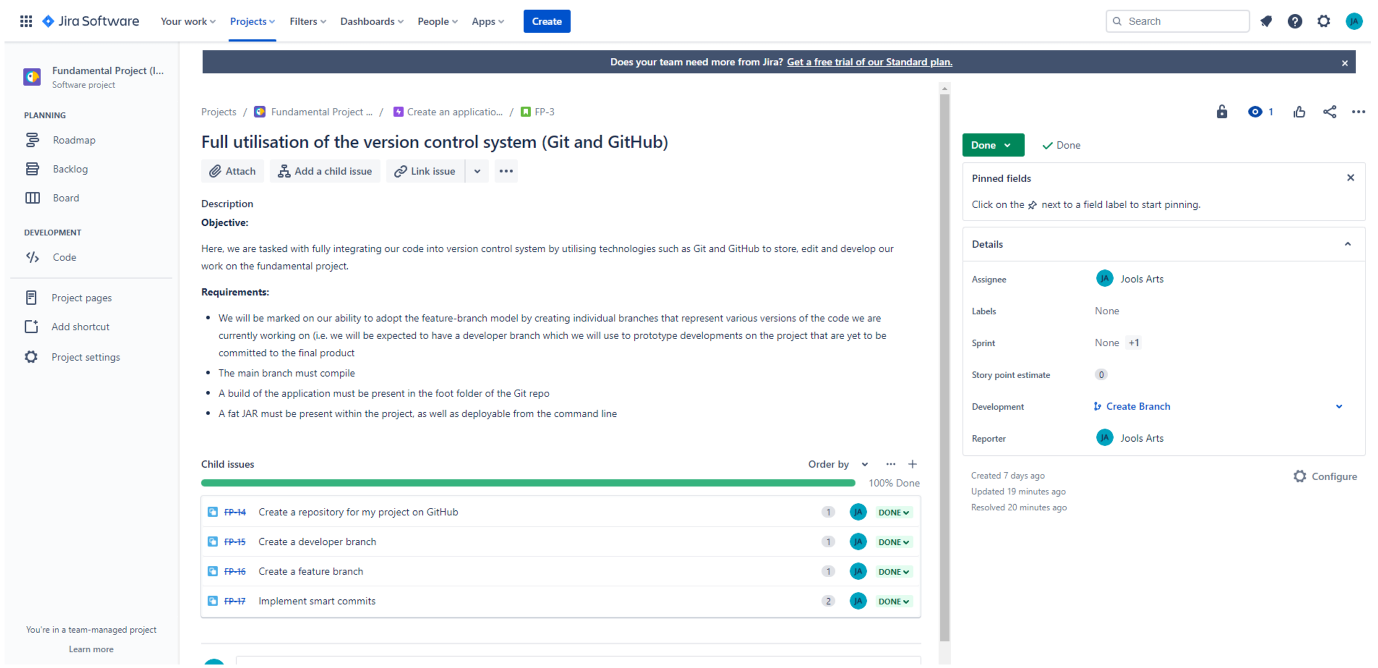
- I was able to achieve this via by the creation of tables and the insertion of relevant data, as showcased within previous slides
- I was also able to connect my table to a local MySQL instance, meaning I completed all of the requirements of the user story


eclipse-workspace - Fundamental-Project-IMS-/src/main/resources/db.properties - Eclipse IDE
File Edit Navigate Search Project Run Window Help
db.properties

```
db.url=jdbc:mysql://localhost:3306/ims
db.user=root
db.password=root
```

Run through a couple of user stories – relational database

- I was tasked with fully utilising the version control system (Git and GitHub)
- I was able to continually add, commit and push new changes to my GitHub repository via the feature branch model, therefore satisfying this particular user story



Full utilisation of the version control system (Git and GitHub)

Description

Objective:

Here, we are tasked with fully integrating our code into version control system by utilising technologies such as Git and GitHub to store, edit and develop our work on the fundamental project.

Requirements:

- We will be marked on our ability to adopt the feature-branch model by creating individual branches that represent various versions of the code we are currently working on (i.e. we will be expected to have a developer branch which we will use to prototype developments on the project that are yet to be committed to the final product
- The main branch must compile
- A build of the application must be present in the foot folder of the Git repo
- A fat JAR must be present within the project, as well as deployable from the command line

Child issues — Order by — 100% Done

~~FP-14~~	Create a repository for my project on GitHub	1	JA	DONE
~~FP-15~~	Create a developer branch	1	JA	DONE
~~FP-16~~	Create a feature branch	1	JA	DONE
~~FP-17~~	Implement smart commits	2	JA	DONE

✓ Done

Pinned fields

Click on the next to a field label to start pinning.

Details

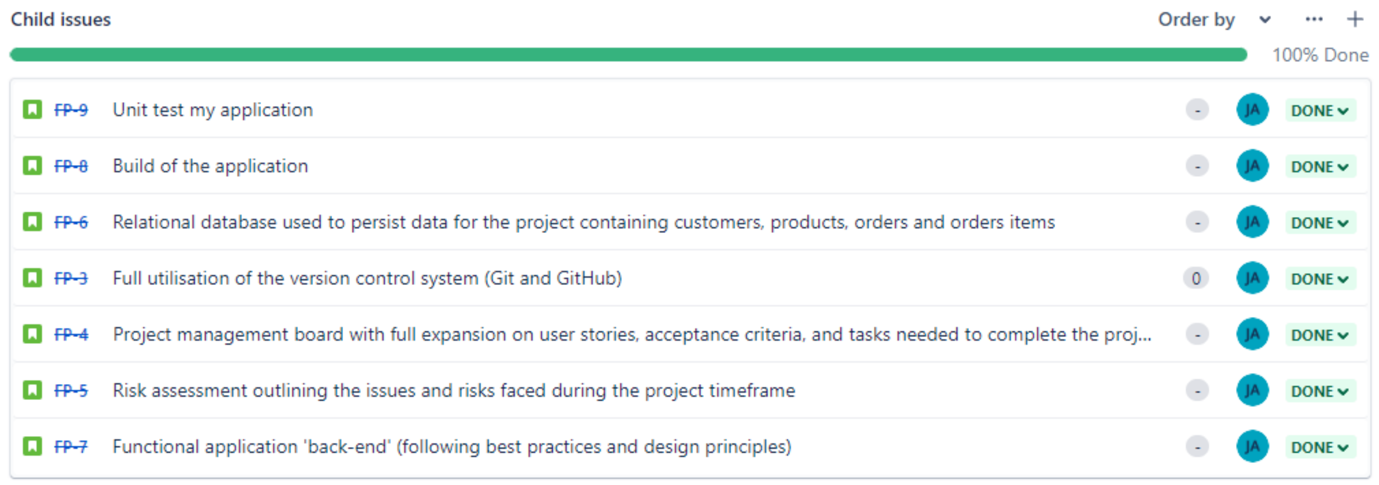
Assignee	Jools Arts
Labels	None
Sprint	None +1
Story point estimate	0
Development	Create Branch
Reporter	Jools Arts

Created 7 days ago
Updated 19 minutes ago
Resolved 20 minutes ago

Sprint Review

What did you complete?

- I completed all of user stories within my main epic

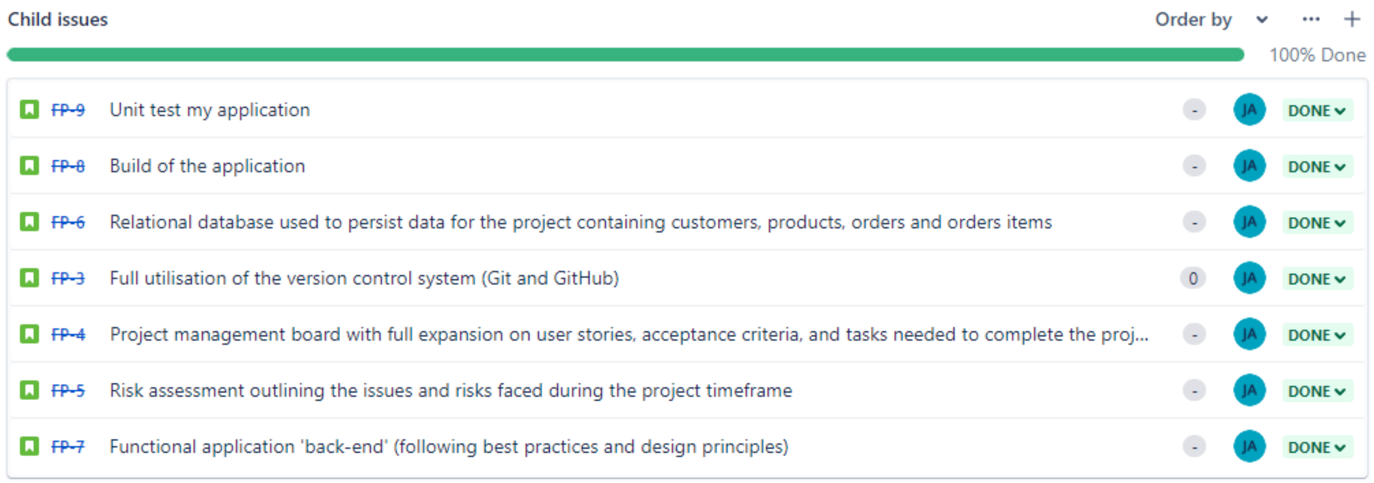


Child issues Order by 100% Done

~~FP-9~~	Unit test my application	-	JA	DONE
~~FP-8~~	Build of the application	-	JA	DONE
~~FP-6~~	Relational database used to persist data for the project containing customers, products, orders and orders items	-	JA	DONE
~~FP-3~~	Full utilisation of the version control system (Git and GitHub)	0	JA	DONE
~~FP-4~~	Project management board with full expansion on user stories, acceptance criteria, and tasks needed to complete the proj...	-	JA	DONE
~~FP-5~~	Risk assessment outlining the issues and risks faced during the project timeframe	-	JA	DONE
~~FP-7~~	Functional application 'back-end' (following best practices and design principles)	-	JA	DONE

What got left behind?

- However, there were certain smaller issues with some rather persistent little niggles that I would aim to go back and address if we were afforded some further time in the future
- For example, creating an order remained a constant issue throughout the project however, creating an item and a customer featured no such issues



Child issues

Order by

100% Done

~~FP-9~~	Unit test my application	-	JA	DONE
~~FP-8~~	Build of the application	-	JA	DONE
~~FP-6~~	Relational database used to persist data for the project containing customers, products, orders and orders items	-	JA	DONE
~~FP-3~~	Full utilisation of the version control system (Git and GitHub)	0	JA	DONE
~~FP-4~~	Project management board with full expansion on user stories, acceptance criteria, and tasks needed to complete the proj...	-	JA	DONE
~~FP-5~~	Risk assessment outlining the issues and risks faced during the project timeframe	-	JA	DONE
~~FP-7~~	Functional application 'back-end' (following best practices and design principles)	-	JA	DONE

Sprint Retrospective

What went well?

- I believe all of the aspects involving the planning of the project, integrating the relevant MySQL databases, and fully utilising version control technology went well

What could be improved?

- I believe there is definitely some room for improvement in regards to creating orders, as well as providing greater coverage for my testing procedures

Conclusion

Reflections on the project, and possible future steps moving forward

- On the whole, I'm really pleased with my overall performance throughout this project
- Looking back at where I was only two weeks ago, I can see evidence of massive improvement in my Java programming skills, as well as my implementation of MySQL databases
- The inventory management system project definitely represented a steep learning curve in term of its overall difficulty, and that was mainly due to the need for implementing a great many number of different skills and forms of logic within a single cohesive application
- However, undertaking the task has highlighted several areas in which I can feel proud in regards to the progress I've made over the previous five weeks of teaching, as well as a few areas in which further study is required
- The beautiful thing about software development is that there are always new technologies to learn, and a plethora of skills to master, and what this project has shown me is that the process is always rewarding, so long may it continue

Thank you