Fundamental Project Specification: Inventory Management System (IMS)

Sam Goodland

Introduction

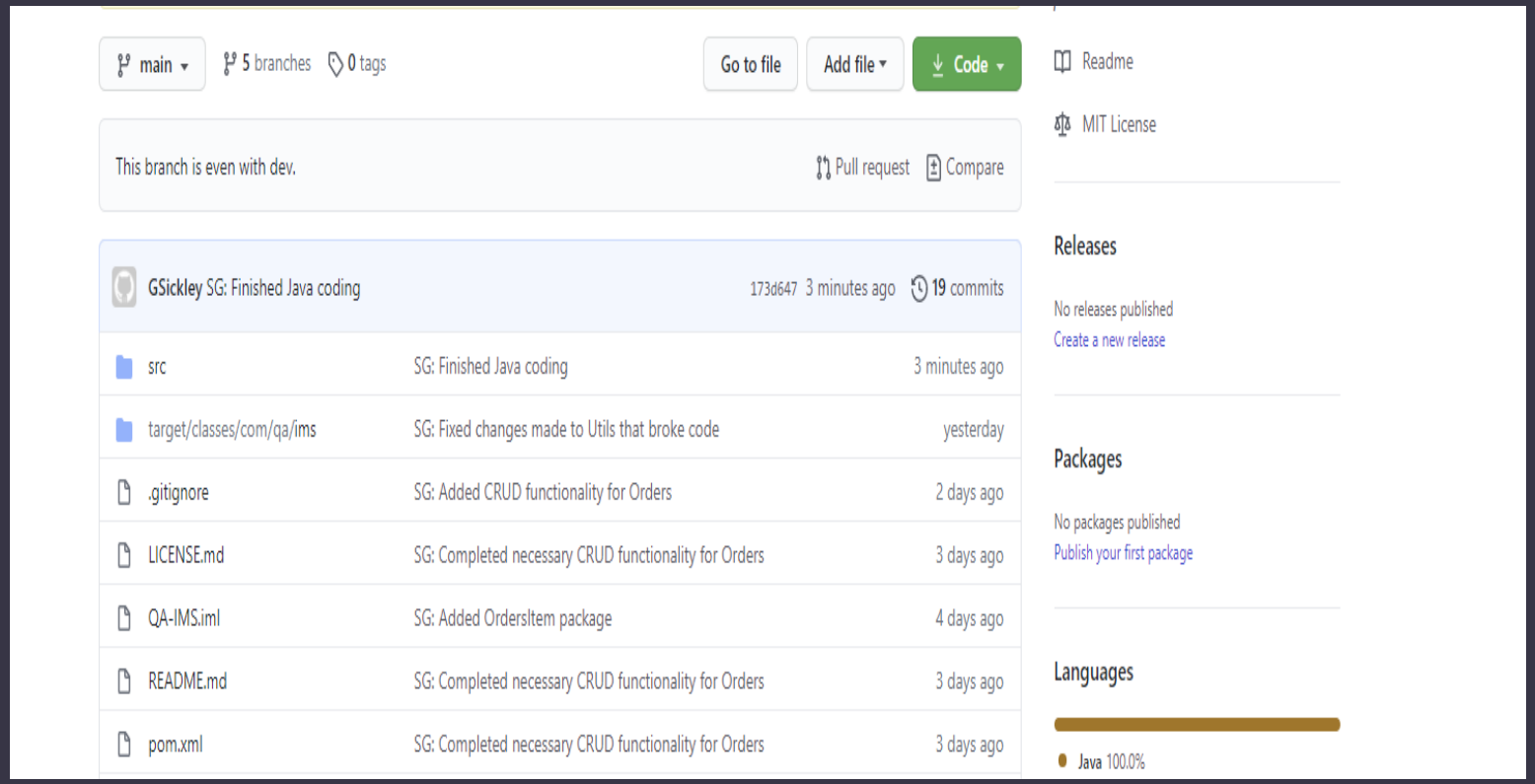
- Hello!
- BSc Grad. Applied Computing
- Approached domain requirements head-on
- In retrospect, additional planning would have aided

Consultant Journey

- Technologies I was already familiar with: Java, SQL
- Technologies I'd used but not mastered: Git, GitHub
- Technologies I'd never used before: Jira, Maven, JUnit

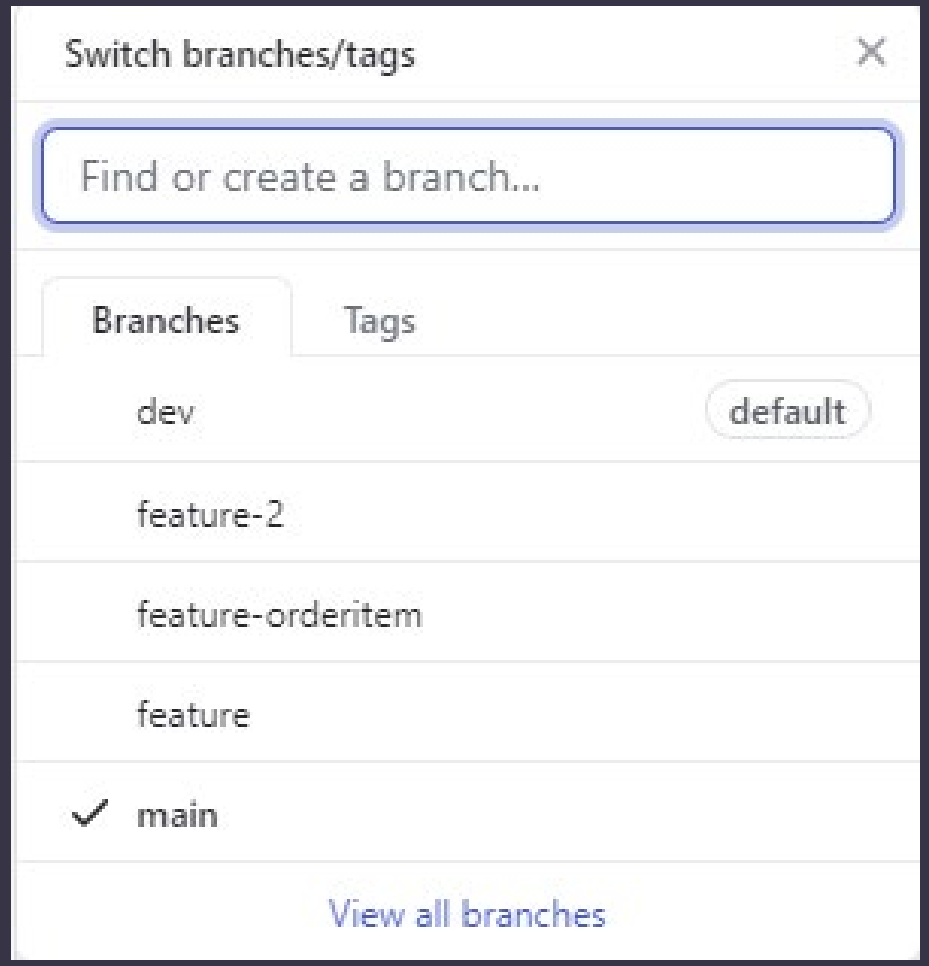
Continuous Integration (CI)

- Version control was approached with industry standards in mind



Continuous Integration (CI) – cont.

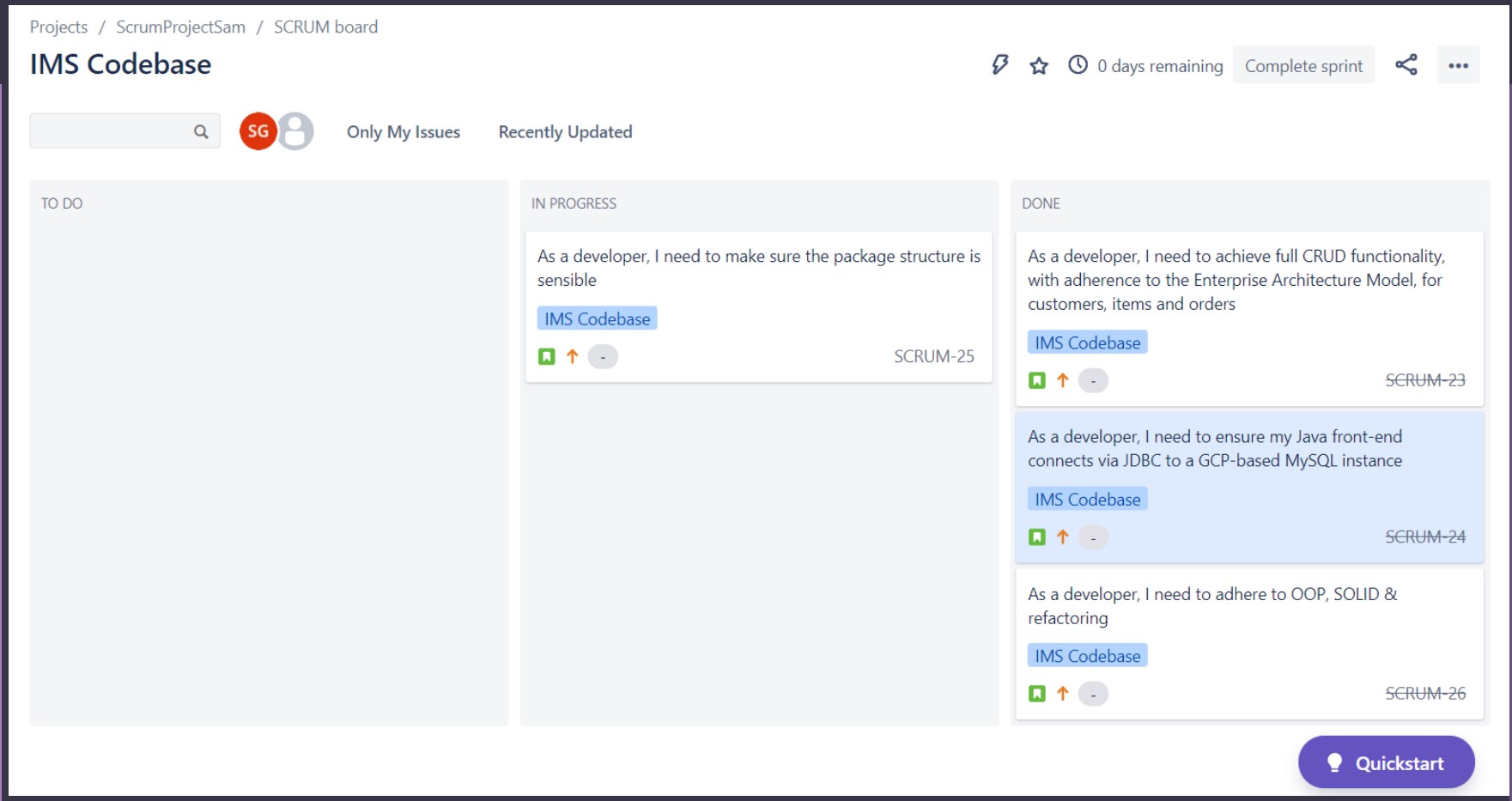
- Feature-dev-main branches utilised; new feature for every new part

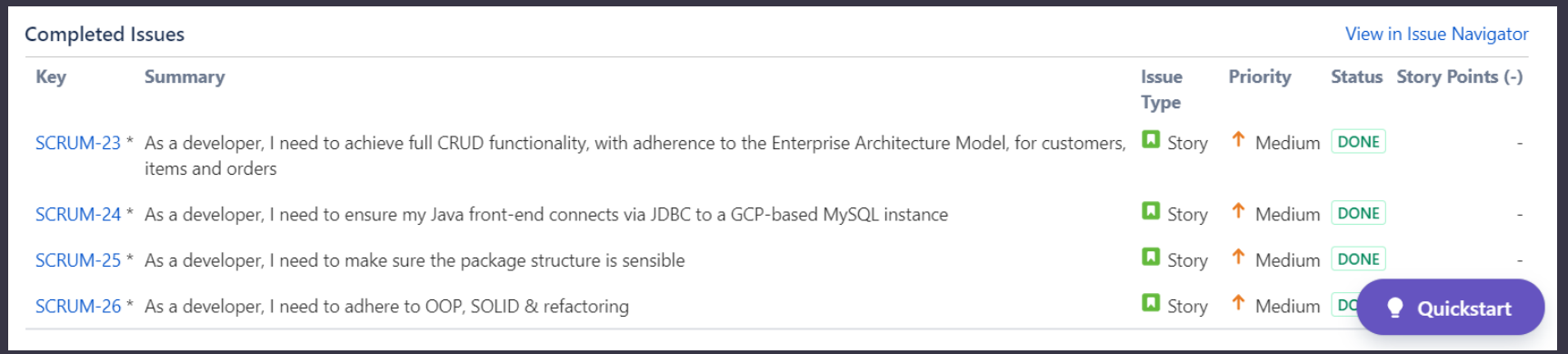


Testing

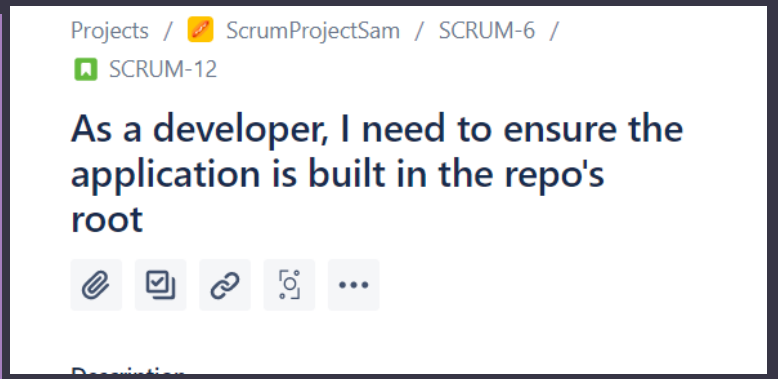
- Hardest part of the project
- Had the least amount of experience with

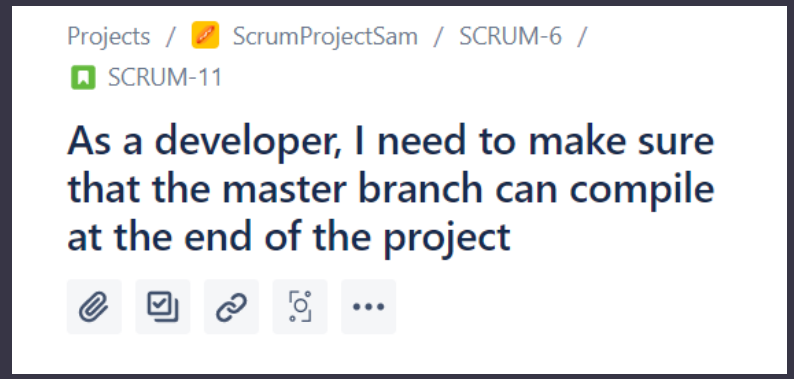
Demonstration – the Codebase

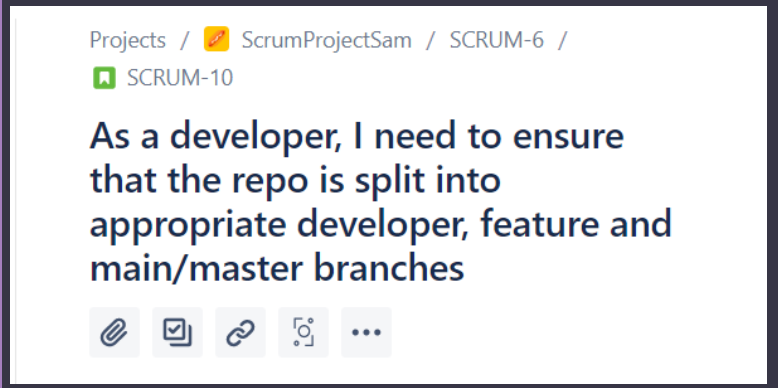


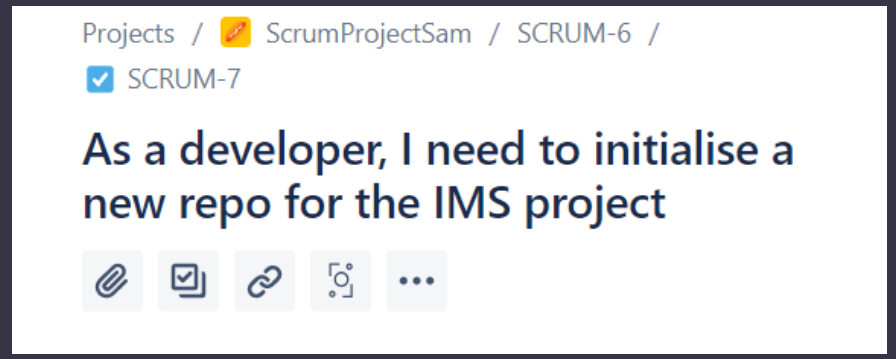


Demonstration – the IMS repo







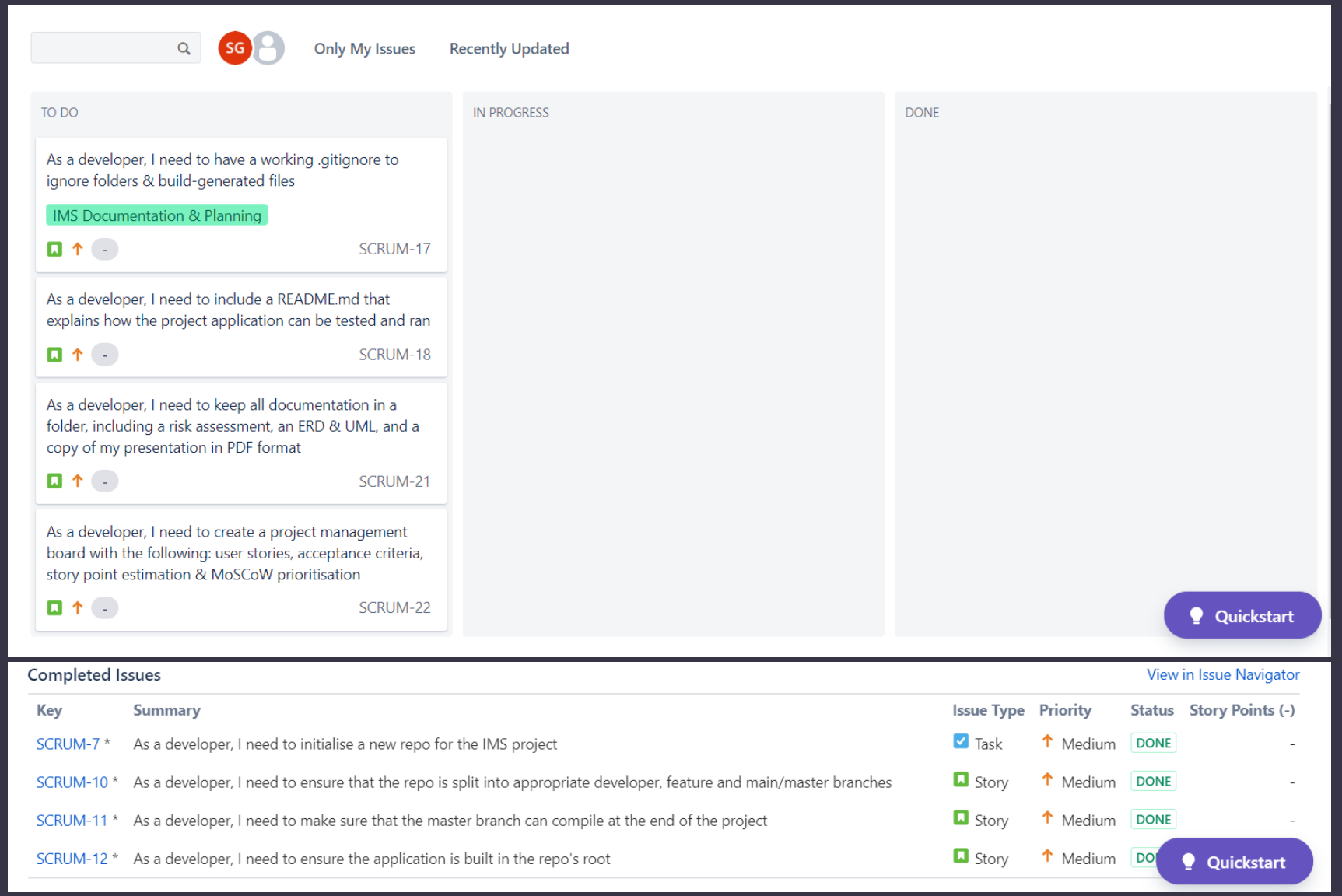


Completed Issues — View in Issue Navigator

Key	Summary	Issue Type	Priority	Status	Story Points (-)
SCRUM-7 *	As a developer, I need to initialise a new repo for the IMS project	Task	Medium	DONE	-
SCRUM-10 *	As a developer, I need to ensure that the repo is split into appropriate developer, feature and main/master branches	Story	Medium	DONE	-
SCRUM-11 *	As a developer, I need to make sure that the master branch can compile at the end of the project	Story	Medium	DONE	-
SCRUM-12 *	As a developer, I need to ensure the application is built in the repo's root	Story	Medium	DO	

Quickstart

Demonstration – the Documentation



Sprint Review – what was completed

- All Codebase aspects, save the calculation of the total cost
- All Repository & Documentation aspects
- Continuous Integration aspects

Sprint Review – What was abandoned

```
public double calculateOrderCost(){
  double cost = 0.0;
  for (Item item : this.items){
    cost += item.getItemValue();
  }
  return cost;
}
```

- The total calculation of a cost
- The test coverage 80% threshhold

Sprint retrospective – what went well

- MySQL
- Java
- Git
- GitHub
- Maven

Sprint retrospective – what could have been done better

- Junit
- Jira

Conclusions drawn

- 1. Time allocation is key
- 2. Knowledge of Java, GitHub, Git & SQL currently meets or is close to meeting expected standards
- 3. Additional JUnit practising will need to be conducted in spare time
- 4. Scope creep was biggest reason for project shortcomings

Conclusion

- Thank you for listening!
- Are there any questions?