IMS Project presentation

AYUB YUSUF

SDET PROGRAMME (SOFTWARE DEVELOPMENT ENGINEER IN TEST)

Project Objective

IMS project:

- Build an application which interacts with a managed database
- The application must use supporting tools, methodologies and technologies that encapsulates all modules covered during training

Technologies used

- **Version Control System:** Git
- **Source Code Management:** GitHub
- **Kanban Board:** Jira
- **Database Management System:** MySQL
- **Back-End Programming Language:** Java
- **Build Tool:** Maven
- **Unit Testing**: JUnit

Scope

- A **risk assessment** which outlines the issues and risks faced during the project timeframe
- Code fully integrated into a **Version Control System**
- A **project management board**
- A **relational database** used to persist data for the project
- A functional application **back-end**
- A **build** of the application
- **Unit tests** for validation of the application

Risk Assessment

Key:

Likelihood:

Rare	Unlikely	Possible	Likely	Certain
1	2	3	4	5

Impact

Rare	Unlikely	Possible	Likely	Certain
1	2	3	4	5

Risk level

Low	Moderate	High	Extreme
(1-5)	(6-10)	(11-15)	(16-25)

Risk Assessment

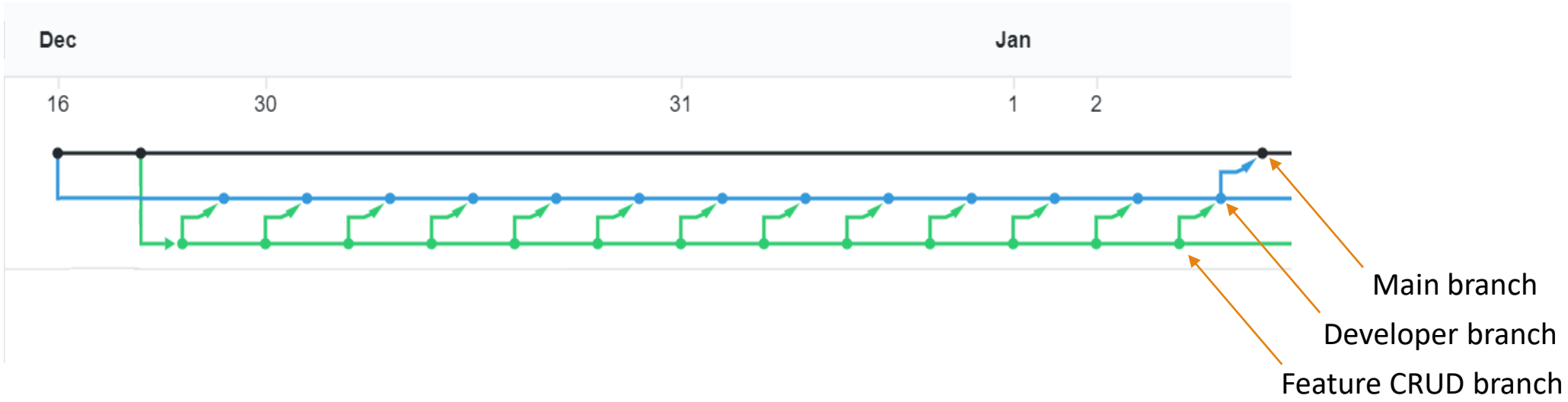
Risk	Description	Impact	Response Strategy	Forecasted Likelihood [1-5]	Forecasted Numerical Impact [1-5]	Forecasted Risk Level (Likelihood*Impact) [1-25]	Actual Likelihood [1-5]	Actual Numerical Impact [1-5]	Actual Risk Level (Likelihood*Impact) [1-25]
Insufficient time	Not managing time effectively leading to spending too much time on particular areas while neglecting others.	Project not being fully complete to the requirements within the given timeframe.	Plan daily/weekly sprints and assign time estimates to each sprint	3	5	15	1	1	1
Insufficient technical knowledge	Technology not covered at university	Project being completed to a suboptimal standard	Read through notes on QA Community. Ask trainers for help. Use Google to find solutions.	2	3	6	4	4	16
MySQL problems	Being unable to link tables together due to mySQL not supporting many-to-many relationships	Project will not function.	Construct an ERD diagram before creating tables/relationships to identify many-to-many relationships. Create intermediary tables to handle this.	4	5	20	4	1	4
COVID-19	Due to surging cases of COVID-19, myself or a family member could fall ill. This could result in myself needing to take time out.	Project will not be delivered on time.	I will ensure that I stay safe and minimise contact with members outside of my household. This will reduce the chances of me falling ill.	1	5	5	2	1	2

Risk Assessment

Not utilising Version control	Irreversible mistake is made or a file is deleted by mistake.	Valuable time wasted trying to recreate a file which could lead to project not being delivered on time.	I will make regular commits to my GitHub repository and utilise main-dev-feature branches. Rollbacks will then provide an invaluable time-saving safety net.	5	4	20	5	5	25
Concentration	Unable to concentrate due to neighbours carrying out building work.	Project being completed to a suboptimal standard.	Invest in a pair of noise-cancelling headphones. This will allow me to focus and be productive.	5	2	10	3	1	3
Insufficient testing	Application will be prone to errors/bugs.	Application will not function reliably.	Allocate time to ensure through testing is executed. Ensure a high test coverage (>80%) is achieved.	2	3	6	2	2	4

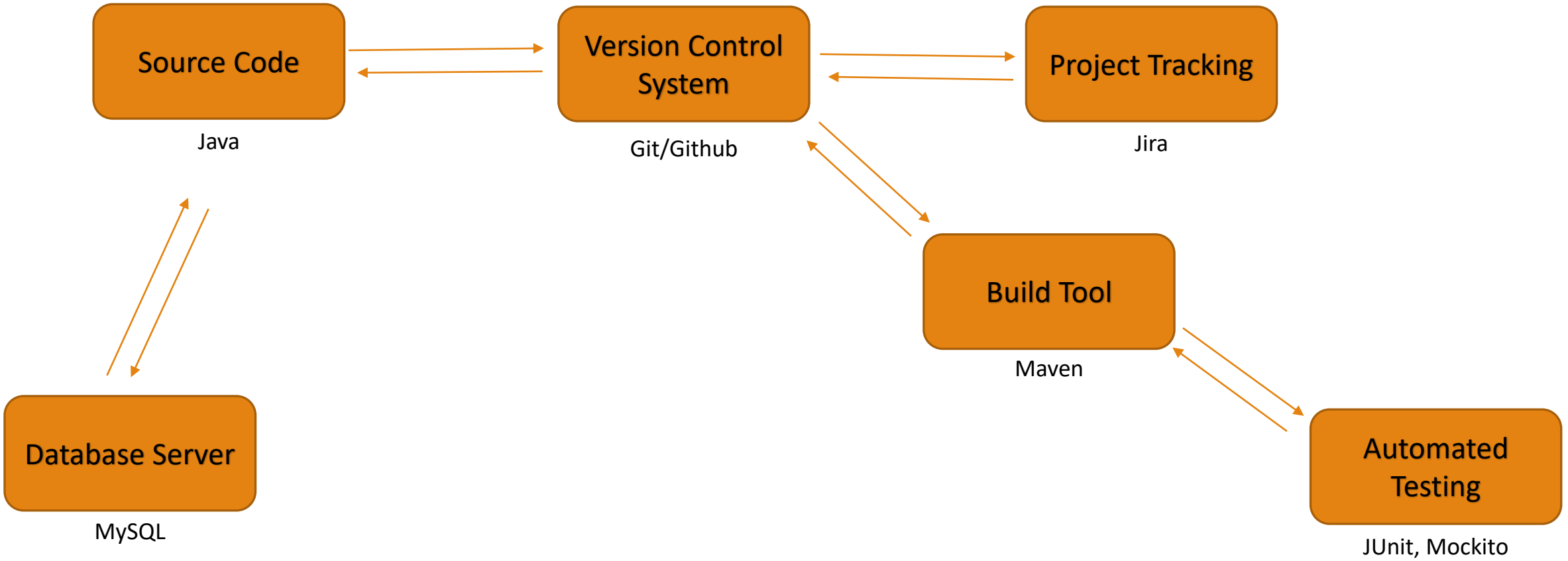
Scope

- A **risk assessment** which outlines the issues and risks faced during the project timeframe
- Code fully integrated into a **Version Control System**
- A **project management board**
- A **relational database** used to persist data for the project
- A functional application **back-end**
- A **build** of the application
- **Unit tests** for validation of the application



Version control system: Git

- feature-branch model: master/dev/multiple features


Source Code
Java
Version Control System
Git/Github
Project Tracking
Jira
Build Tool
Maven
Database Server
MySQL
Automated Testing
JUnit, Mockito

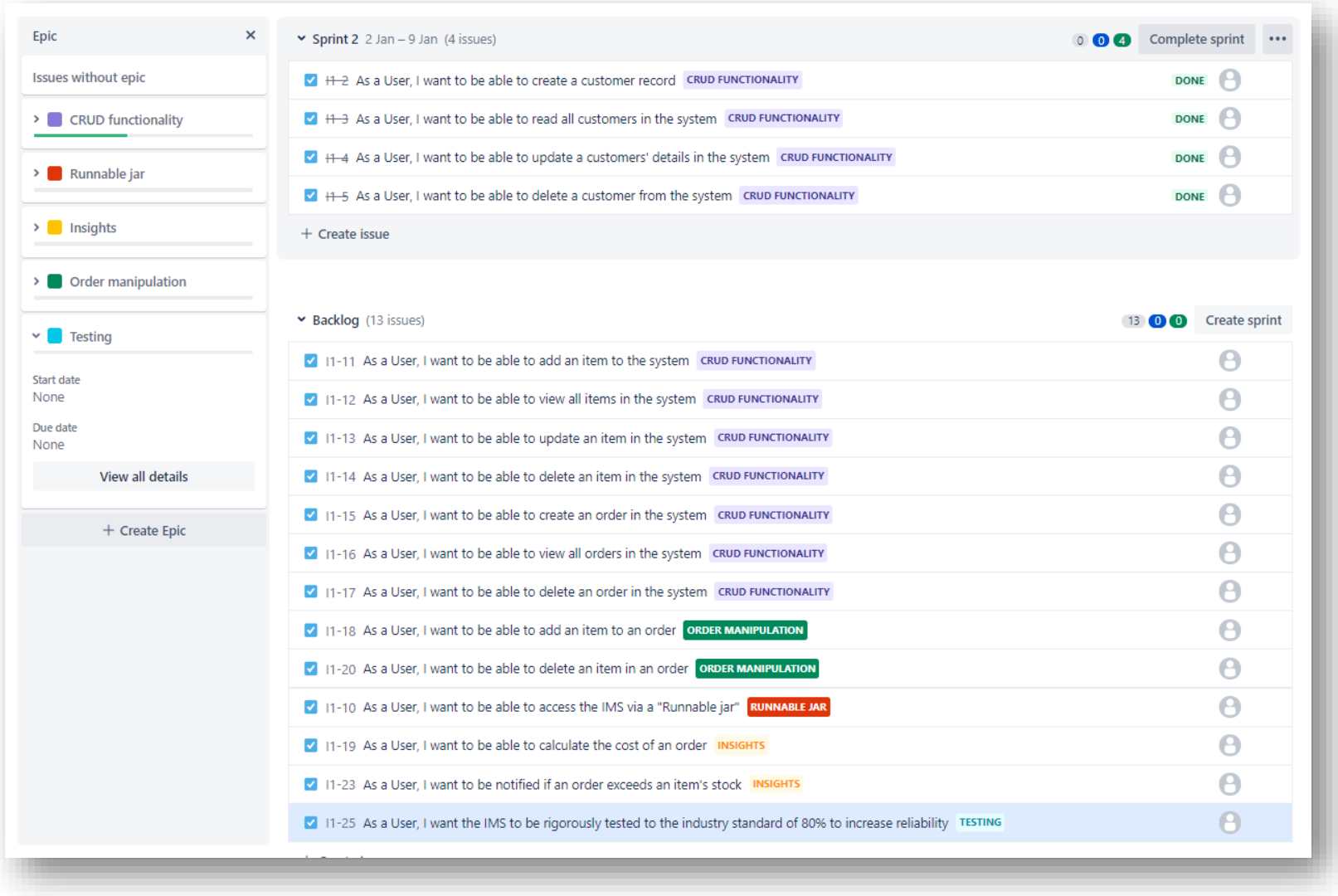
CI Pipeline

Scope

- A **risk assessment** which outlines the issues and risks faced during the project timeframe
- Code fully integrated into a **Version Control System**
- A **project management board**
- A **relational database** used to persist data for the project
- A functional application **back-end**
- A **build** of the application
- **Unit tests** for validation of the application

Project management board: Kanban

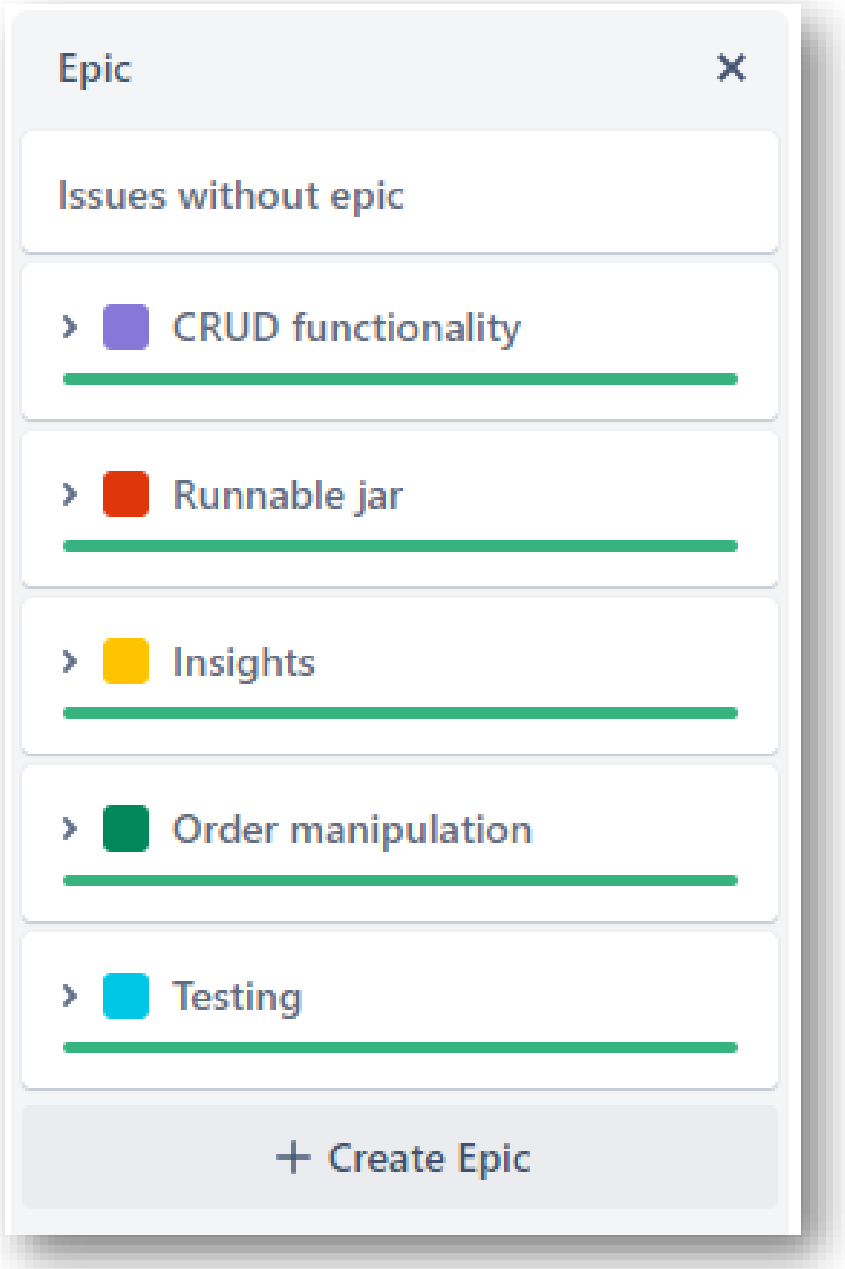
- To effectively manage my project, an Agile approach was adopted.
- A Kanban board (a feature of Jira) was used to manage the project.
- The first step of the planning was to add an exhaustive list of user stories to the backlog.



Project management board: Kanban

- The following epics were created:
  - CRUD functionality
  - Runnable jar
  - Insights
  - Order manipulation
  - Testing

- All epics were successfully completed

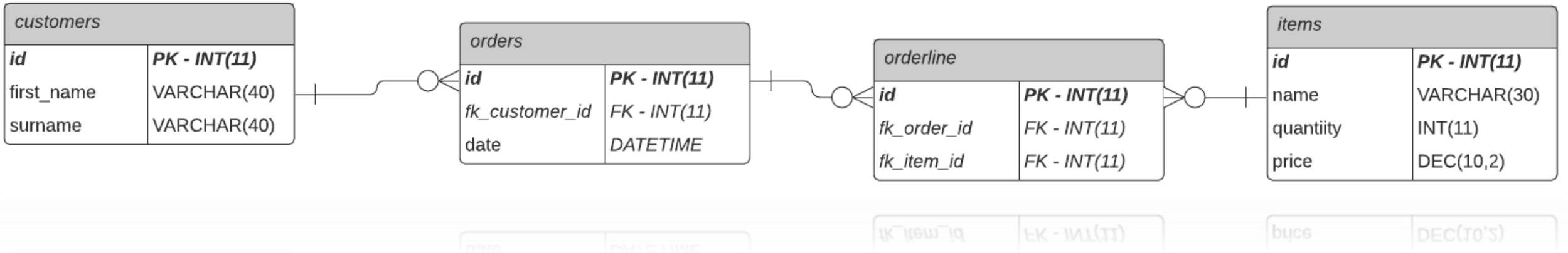


Sprint review

- Most of the Sprints were completed successfully, though some were left behind due to difficulty or lack of time:
  - Customer email
  - Orderline feature:
    - Multiple items with the same id can be contained in one line in an order

Scope

- A **risk assessment** which outlines the issues and risks faced during the project timeframe
- Code fully integrated into a **Version Control System**
- A **project management board**
- A **relational database** used to persist data for the project
- A functional application **back-end**
- A **build** of the application
- **Unit tests** for validation of the application



Relational database: MySQL

- A relational database (used to persist data for the project) was created using MySQL
- The above ERD diagram was implemented in the relational database
- An intermediary orderline table was created because mySQL does not support many-to-many relationships

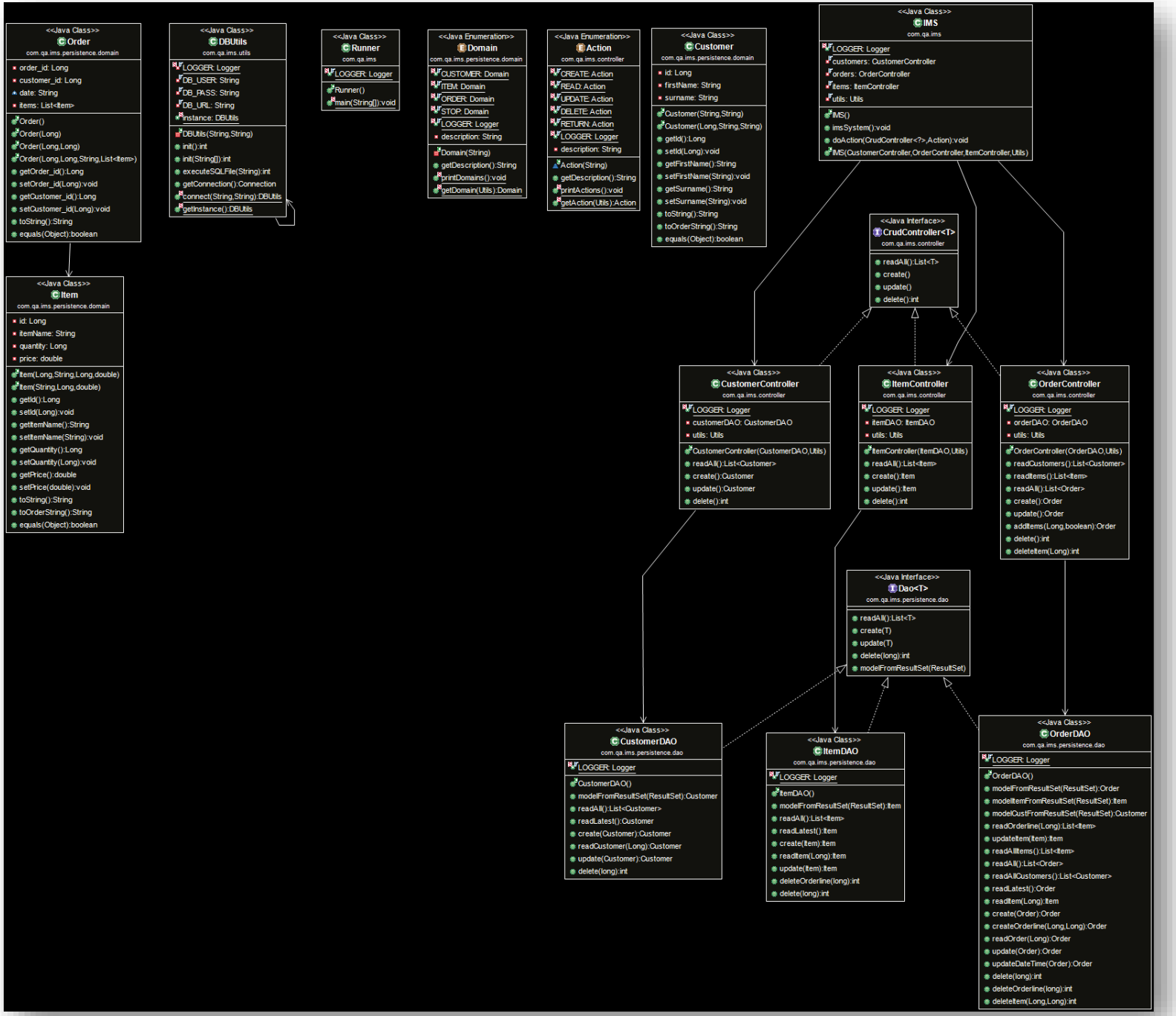
Scope

- A **risk assessment** which outlines the issues and risks faced during the project timeframe
- Code fully integrated into a **Version Control System**
- A **project management board**
- A **relational database** used to persist data for the project
- A functional application **back-end**
- A **build** of the application
- **Unit tests** for validation of the application

Back-end: UML diagram

Good practices and design principles were followed:

- ObjectControllers only instantiate their own ObjectDAO
- ObjectControllers do not instantiate other ObjectControllers
- DAOs do not instantiate other DAOs



Extra features

- When updating an order, the following extra features have been implemented:
- Stock:
  - An item is not added to an order if the quantity required exceeds the item stock
- DateTime:
  - Each time an item is added to or deleted from an order, the date and time for that order is updated

```
if(stock < quantity) {
    LOGGER.info("\nERROR: insufficient stock!");
}
else {
    for(int i=0; i<quantity; i++) {
        updated_order = orderDAO.createOrderline(order_id, item_id);
    }
    stock -= quantity;
    item.setQuantity(stock);
    orderDAO.updateItem(item);
    orderDAO.updateDateTime(orderDAO.readOrder(order_id));
}
```

Scope

- A **risk assessment** which outlines the issues and risks faced during the project timeframe
- Code fully integrated into a **Version Control System**
- A **project management board**
- A **relational database** used to persist data for the project
- A functional application **back-end**
- A **build** of the application
- **Unit tests** for validation of the application

Build of application: Maven

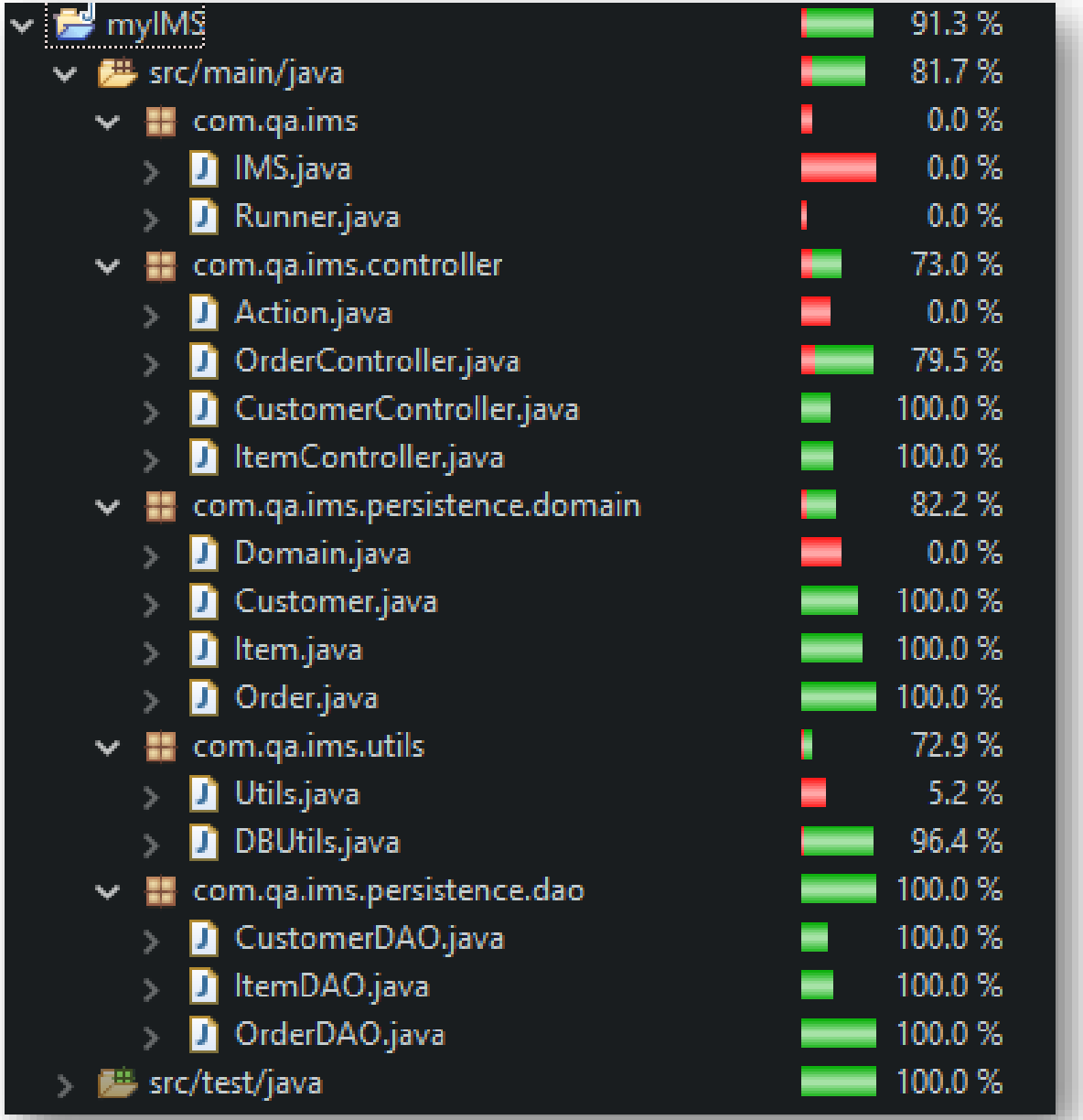
- The application was built using the build tool Maven.
- I will now demonstrate the following user stories in the application:
  - As a user, I want to create an order
  - As a user, I want to add an item to an order
  - As a user, I want to delete an item from an order

Scope

- A **risk assessment** which outlines the issues and risks faced during the project timeframe
- Code fully integrated into a **Version Control System**
- A **project management board**
- A **relational database** used to persist data for the project
- A functional application **back-end**
- A **build** of the application
- **Unit tests** for validation of the application

Testing: Coverage

- A coverage of **81.7%** was achieved.
- 101/101 Tests ran successfully
- The following Tests files were created:
  - Controllers:
    - CustomerControllerTest.java
    - ItemControllerTest.java
    - OrderControllerTest.java
  - DAOs:
    - CustomerDAOTest.java
    - ItemDAOTest.java
    - OrderDAOTest.java
    - CustomerDAOTestFAIL.java
    - ItemDAOTestFAIL.java
    - OrderDAOTestFAIL.java
  - Domain:
    - CustomerTest.java
    - CustomerEqualsTest.java
    - ItemTest.java
    - ItemEqualsTest.java
    - OrderTest.java
    - OrderEqualsTest.java



Conclusion

- Better commits should be used, eg “feature-customer-crud”
- Commits should be made regularly to avoid dumping lots of changes in one go
- User stories should continuously be added to the backlog throughout the project timeline so it is clear what is left to be done when new user stories surface

Questions?