Java Based CRUD

An Inventory Management System

By Anoush Lowton

TECHNOLOGIES LEARNED

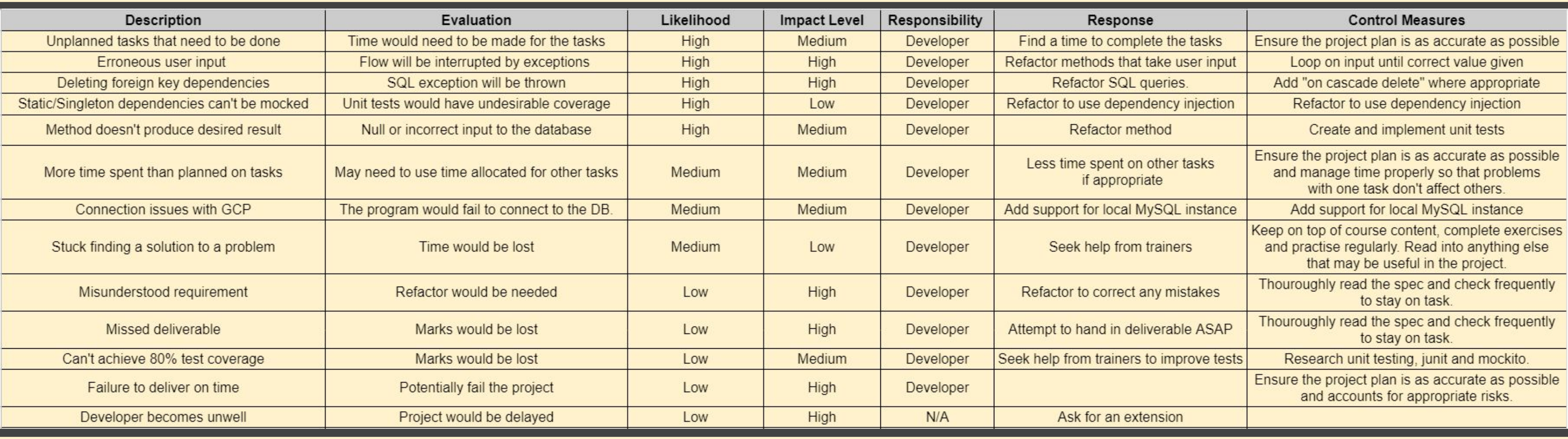
- Version Control System: **Git**
- Source Control Management: **GitHub**
- Project Management Board: **Jira**
- DBMS: **MySQL Server 8.0**
- Programming Language: **Java**
- Build Tool: **Maven**
- Unit Testing: **JUnit & Mockito**

APPROACH

- Plan the project using the project management board: Jira
    - Create epics, user stories and tasks
    - Decide on acceptance criteria
    - Give estimations using story points
    - Give prioritisations using the MoSCoW methodology

APPROACH

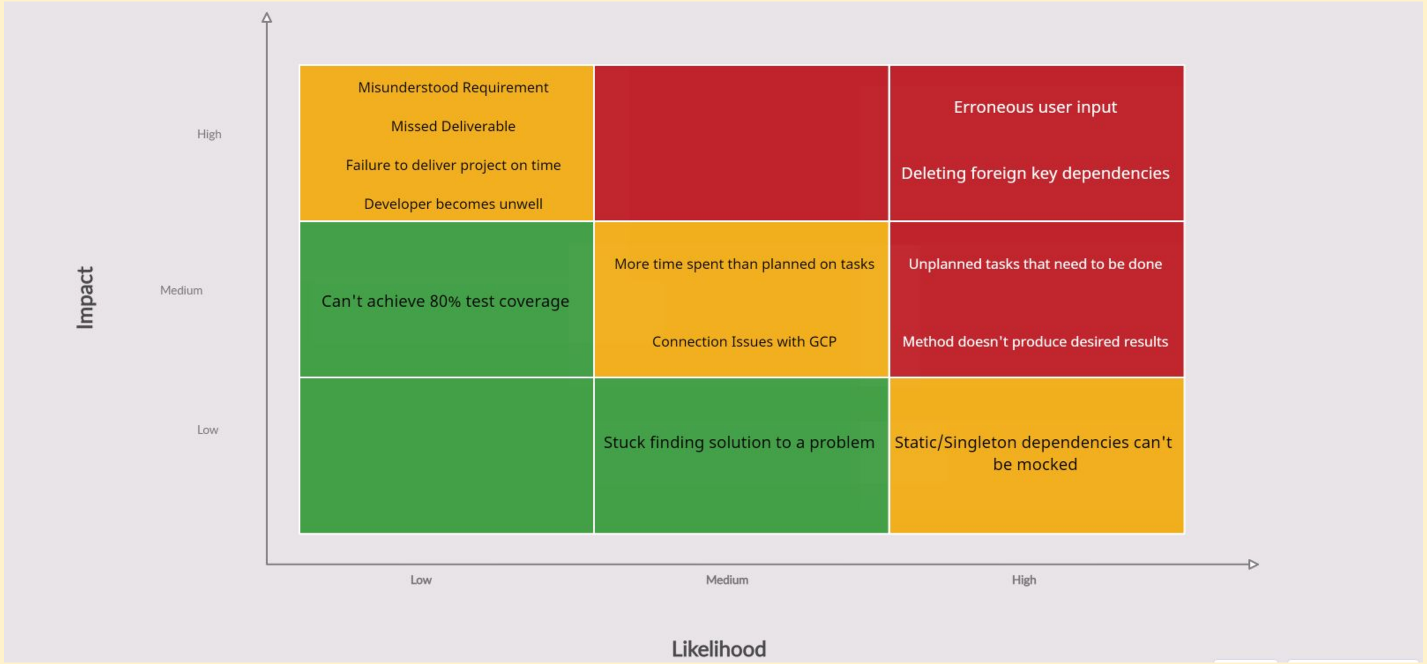
- Create a risk assessment table



Description	Evaluation	Likelihood	Impact Level	Responsibility	Response	Control Measures
Unplanned tasks that need to be done	Time would need to be made for the tasks	High	Medium	Developer	Find a time to complete the tasks	Ensure the project plan is as accurate as possible
Erroneous user input	Flow will be interrupted by exceptions	High	High	Developer	Refactor methods that take user input	Loop on input until correct value given
Deleting foreign key dependencies	SQL exception will be thrown	High	High	Developer	Refactor SQL queries.	Add "on cascade delete" where appropriate
Static/Singleton dependencies can't be mocked	Unit tests would have undesirable coverage	High	Low	Developer	Refactor to use dependency injection	Refactor to use dependency injection
Method doesn't produce desired result	Null or incorrect input to the database	High	Medium	Developer	Refactor method	Create and implement unit tests
More time spent than planned on tasks	May need to use time allocated for other tasks	Medium	Medium	Developer	Less time spent on other tasks if appropriate	Ensure the project plan is as accurate as possible and manage time properly so that problems with one task don't affect others.
Connection issues with GCP	The program would fail to connect to the DB.	Medium	Medium	Developer	Add support for local MySQL instance	Add support for local MySQL instance
Stuck finding a solution to a problem	Time would be lost	Medium	Low	Developer	Seek help from trainers	Keep on top of course content, complete exercises and practise regularly. Read into anything else that may be useful in the project.
Misunderstood requirement	Refactor would be needed	Low	High	Developer	Refactor to correct any mistakes	Thouroughly read the spec and check frequently to stay on task.
Missed deliverable	Marks would be lost	Low	High	Developer	Attempt to hand in deliverable ASAP	Thouroughly read the spec and check frequently to stay on task.
Can't achieve 80% test coverage	Marks would be lost	Low	Medium	Developer	Seek help from trainers to improve tests	Research unit testing, junit and mockito.
Failure to deliver on time	Potentially fail the project	Low	High	Developer		Ensure the project plan is as accurate as possible and accounts for appropriate risks.
Developer becomes unwell	Project would be delayed	Low	High	N/A	Ask for an extension	

APPROACH

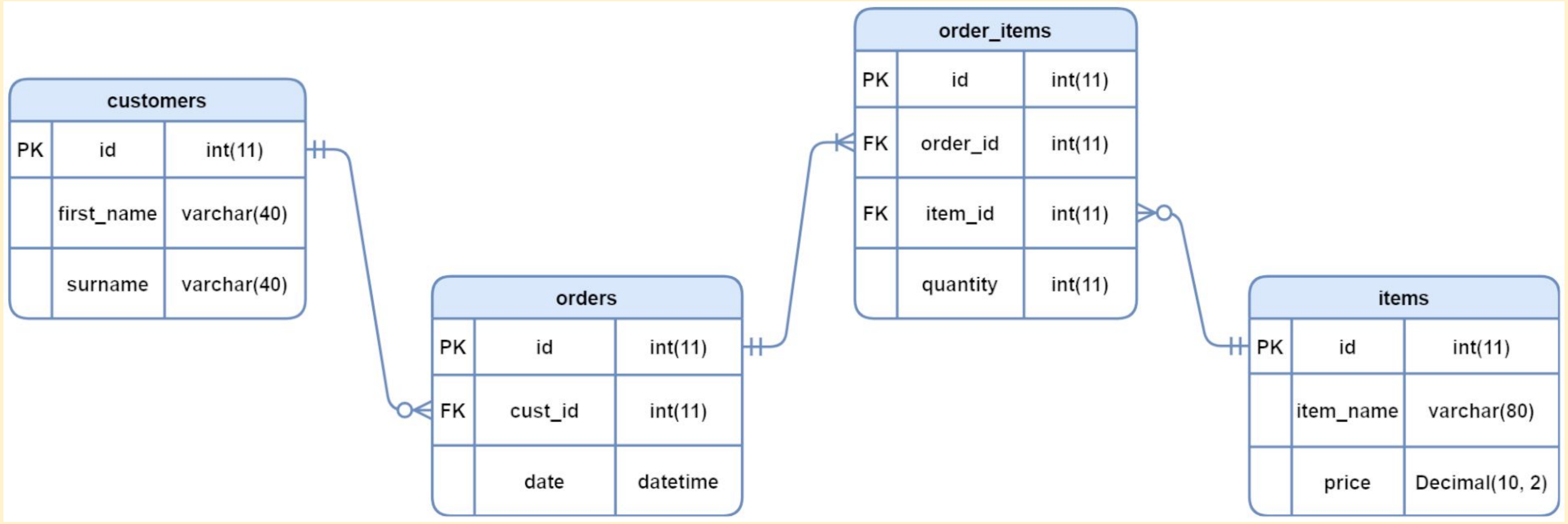
- Create a risk assessment matrix



Impact \ Likelihood	Low	Medium	High
High	🟧 Misunderstood Requirement Missed Deliverable Failure to deliver project on time Developer becomes unwell	🟥	🟥 Erroneous user input Deleting foreign key dependencies
Medium	🟩 Can't achieve 80% test coverage	🟧 More time spent than planned on tasks Connection Issues with GCP	🟥 Unplanned tasks that need to be done Method doesn't produce desired results
Low	🟩	🟩 Stuck finding solution to a problem	🟧 Static/Singleton dependencies can't be mocked

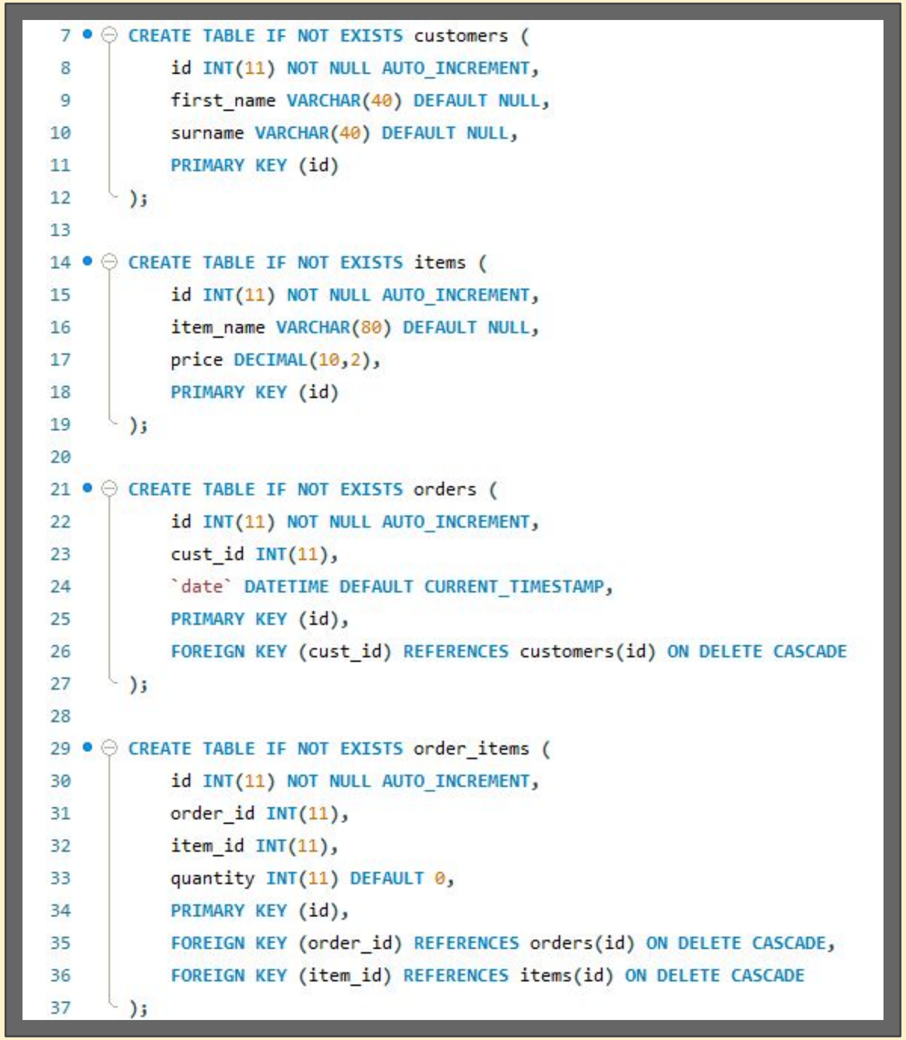
APPROACH

- Design table structure using an ERD



APPROACH

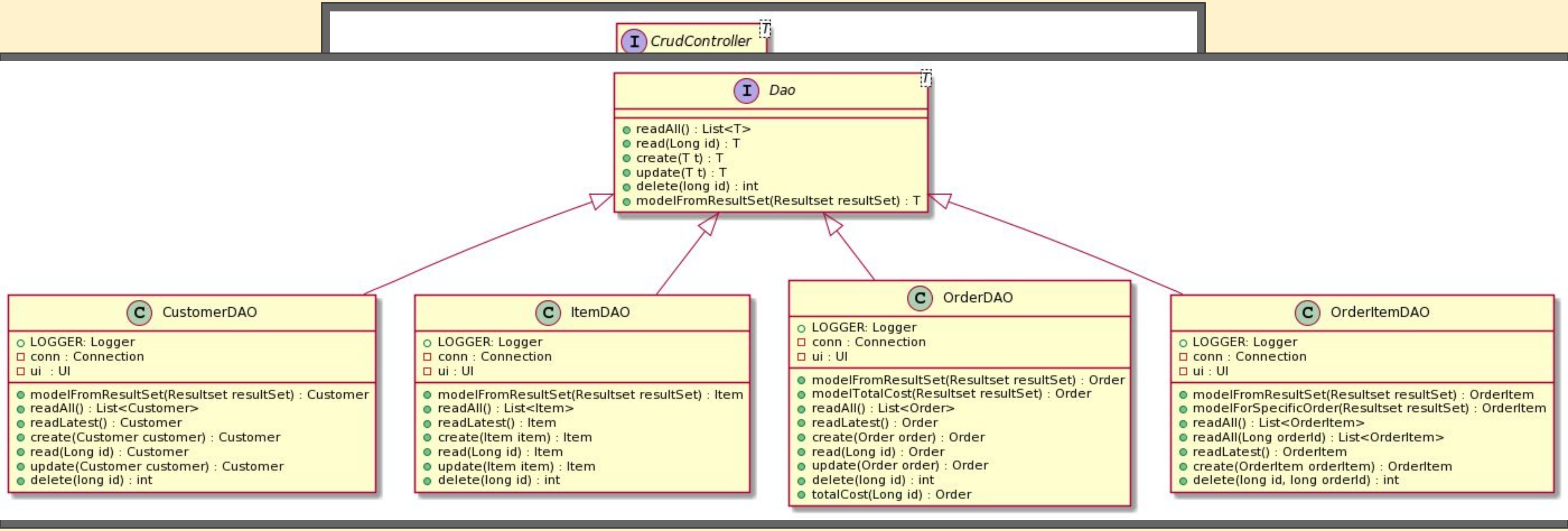
- Design tables in SQL



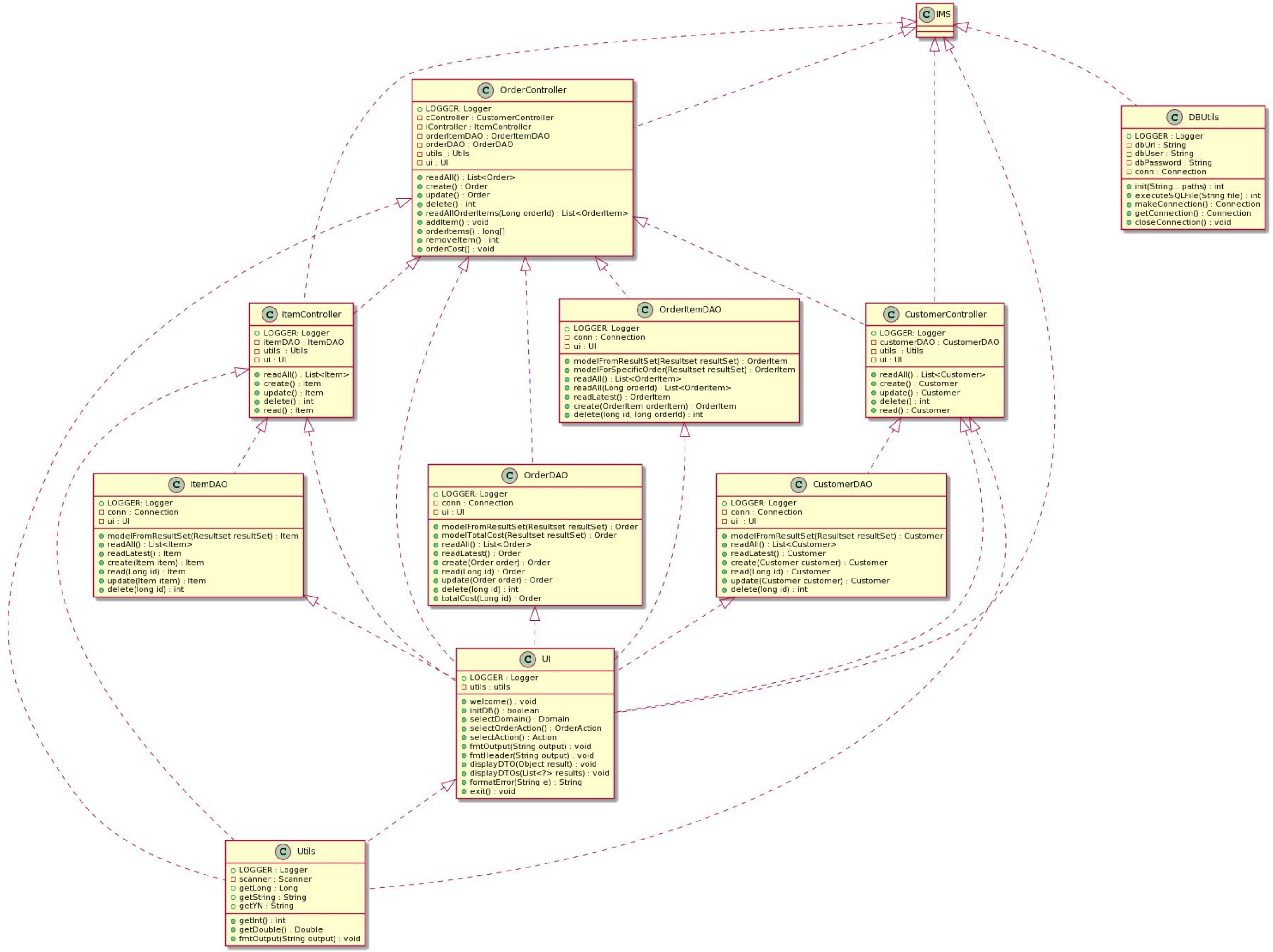
```sql
CREATE TABLE IF NOT EXISTS customers (
    id INT(11) NOT NULL AUTO_INCREMENT,
    first_name VARCHAR(40) DEFAULT NULL,
    surname VARCHAR(40) DEFAULT NULL,
    PRIMARY KEY (id)
);

CREATE TABLE IF NOT EXISTS items (
    id INT(11) NOT NULL AUTO_INCREMENT,
    item_name VARCHAR(80) DEFAULT NULL,
    price DECIMAL(10,2),
    PRIMARY KEY (id)
);

CREATE TABLE IF NOT EXISTS orders (
    id INT(11) NOT NULL AUTO_INCREMENT,
    cust_id INT(11),
    `date` DATETIME DEFAULT CURRENT_TIMESTAMP,
    PRIMARY KEY (id),
    FOREIGN KEY (cust_id) REFERENCES customers(id) ON DELETE CASCADE
);

CREATE TABLE IF NOT EXISTS order_items (
    id INT(11) NOT NULL AUTO_INCREMENT,
    order_id INT(11),
    item_id INT(11),
    quantity INT(11) DEFAULT 0,
    PRIMARY KEY (id),
    FOREIGN KEY (order_id) REFERENCES orders(id) ON DELETE CASCADE,
    FOREIGN KEY (item_id) REFERENCES items(id) ON DELETE CASCADE
);
```

APPROACH

- Design class structure using UML diagrams



I CrudController T
I Dao T
readAll() : List<T>
read(Long id) : T
create(T t) : T
update(T t) : T
delete(long id) : int
modelFromResultSet(Resultset resultSet) : T
C CustomerDAO
LOGGER: Logger
conn : Connection
ui : UI
modelFromResultSet(Resultset resultSet) : Customer
readAll() : List<Customer>
readLatest() : Customer
create(Customer customer) : Customer
read(Long id) : Customer
update(Customer customer) : Customer
delete(long id) : int
C ItemDAO
LOGGER: Logger
conn : Connection
ui : UI
modelFromResultSet(Resultset resultSet) : Item
readAll() : List<Item>
readLatest() : Item
create(Item item) : Item
read(Long id) : Item
update(Item item) : Item
delete(long id) : int
C OrderDAO
LOGGER: Logger
conn : Connection
ui : UI
modelFromResultSet(Resultset resultSet) : Order
modelTotalCost(Resultset resultSet) : Order
readAll() : List<Order>
readLatest() : Order
create(Order order) : Order
read(Long id) : Order
update(Order order) : Order
delete(long id) : int
totalCost(Long id) : Order
C OrderItemDAO
LOGGER: Logger
conn : Connection
ui : UI
modelFromResultSet(Resultset resultSet) : OrderItem
modelForSpecificOrder(Resultset resultSet) : OrderItem
readAll() : List<OrderItem>
readAll(Long orderId) : List<OrderItem>
readLatest() : OrderItem
create(OrderItem orderItem) : OrderItem
delete(long id, long orderId) : int

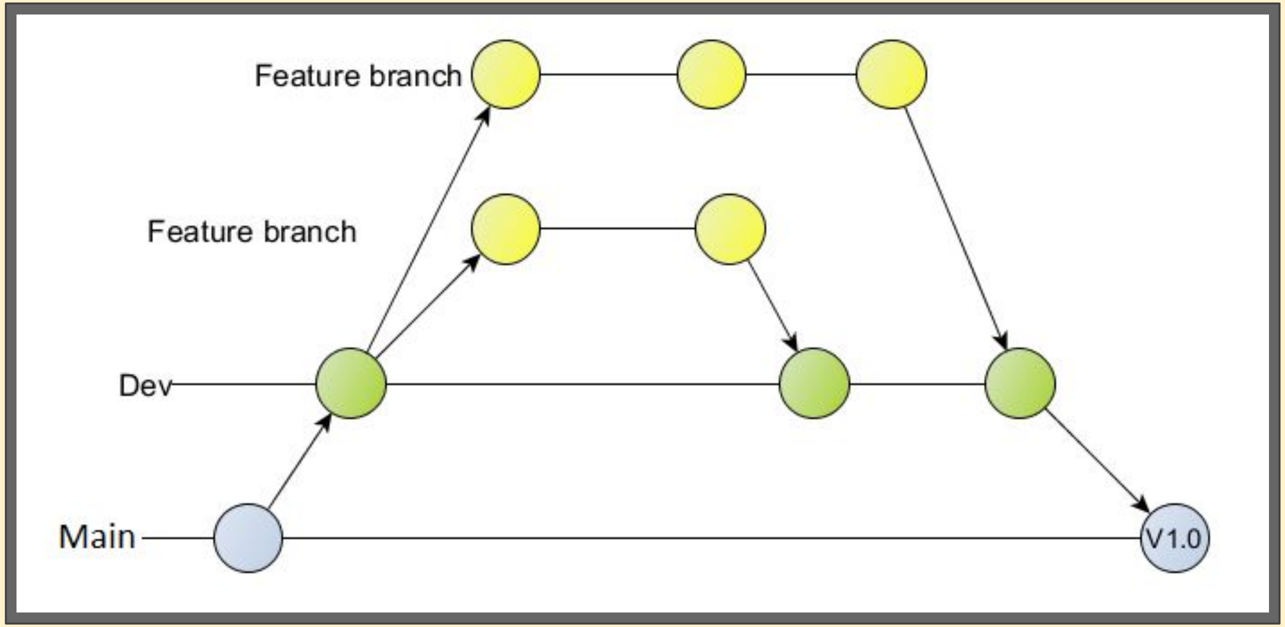

IMS
OrderController
+ LOGGER: Logger
- cController : CustomerController
- iController : ItemController
- orderItemDAO : OrderItemDAO
- orderDAO : OrderDAO
- utils : Utils
- ui : UI
+ readAll() : List<Order>
+ create() : Order
+ update() : Order
+ delete() : int
+ readAllOrderItems(Long orderId) : List<OrderItem>
+ addItem() : void
+ orderItems() : long[]
+ removeItem() : int
+ orderCost() : void
DBUtils
+ LOGGER : Logger
- dbUrl : String
- dbUser : String
- dbPassword : String
- conn : Connection
+ init(String... paths) : int
+ executeSQLFile(String file) : int
+ makeConnection() : Connection
+ getConnection() : Connection
+ closeConnection() : void
ItemController
+ LOGGER: Logger
- itemDAO : ItemDAO
- utils : Utils
- ui : UI
+ readAll() : List<Item>
+ create() : Item
+ update() : Item
+ delete() : int
+ read() : Item
OrderItemDAO
+ LOGGER: Logger
- conn : Connection
- ui : UI
+ modelFromResultSet(Resultset resultSet) : OrderItem
+ modelForSpecificOrder(Resultset resultSet) : OrderItem
+ readAll() : List<OrderItem>
+ readAll(Long orderId) : List<OrderItem>
+ readLatest() : OrderItem
+ create(OrderItem orderItem) : OrderItem
+ delete(long id, long orderId) : int
CustomerController
+ LOGGER: Logger
- customerDAO : CustomerDAO
- utils : Utils
- ui : UI
+ readAll() : List<Customer>
+ create() : Customer
+ update() : Customer
+ delete() : int
+ read() : Customer
ItemDAO
+ LOGGER: Logger
- conn : Connection
- ui : UI
+ modelFromResultSet(Resultset resultSet) : Item
+ readAll() : List<Item>
+ readLatest() : Item
+ create(Item item) : Item
+ read(Long id) : Item
+ update(Item item) : Item
+ delete(long id) : int
OrderDAO
+ LOGGER: Logger
- conn : Connection
- ui : UI
+ modelFromResultSet(Resultset resultSet) : Order
+ modelTotalCost(Resultset resultSet) : Order
+ readAll() : List<Order>
+ readLatest() : Order
+ create(Order order) : Order
+ read(Long id) : Order
+ update(Order order) : Order
+ delete(long id) : int
+ totalCost(Long id) : Order
CustomerDAO
+ LOGGER: Logger
- conn : Connection
- ui : UI
+ modelFromResultSet(Resultset resultSet) : Customer
+ readAll() : List<Customer>
+ readLatest() : Customer
+ create(Customer customer) : Customer
+ read(Long id) : Customer
+ update(Customer customer) : Customer
+ delete(long id) : int
UI
+ LOGGER : Logger
- utils : utils
+ welcome() : void
+ initDB() : boolean
+ selectDomain() : Domain
+ selectOrderAction() : OrderAction
+ selectAction() : Action
+ fmtOutput(String output) : void
+ fmtHeader(String output) : void
+ displayDTO(Object result) : void
+ displayDTOs(List<?> results) : void
+ formatError(String e) : String
+ exit() : void
Utils
+ LOGGER : Logger
- scanner : Scanner
+ getLong : Long
+ getString : String
+ getYN : String
+ getInt() : int
+ getDouble() : Double
+ fmtOutput(String output) : void

APPROACH

- Achieve MVP as soon as possible.
- Improve readability using a UI class and better formatting.
- Add QOL features to improve the UX.
- Handle all exceptions, erroneous inputs and edge cases.
- Improve class structure and apply better design principles where appropriate.
- Unit test the application to a minimum 80%.

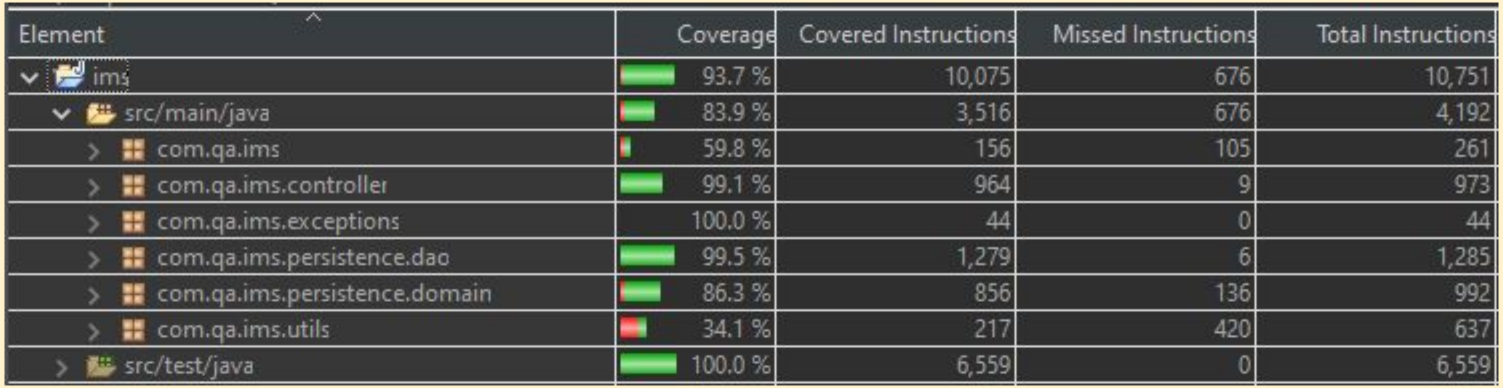
Version Control

- Git: feature-branch model



Unit Testing

- Achieve at least an 80% coverage of the src/main/java folder.
- Aim was to unit test every complex class in as isolated manner as possible.



Element	Coverage	Covered Instructions	Missed Instructions	Total Instructions
ims	93.7 %	10,075	676	10,751
src/main/java	83.9 %	3,516	676	4,192
com.qa.ims	59.8 %	156	105	261
com.qa.ims.controller	99.1 %	964	9	973
com.qa.ims.exceptions	100.0 %	44	0	44
com.qa.ims.persistence.dao	99.5 %	1,279	6	1,285
com.qa.ims.persistence.domain	86.3 %	856	136	992
com.qa.ims.utils	34.1 %	217	420	637
src/test/java	100.0 %	6,559	0	6,559

Unit Testing

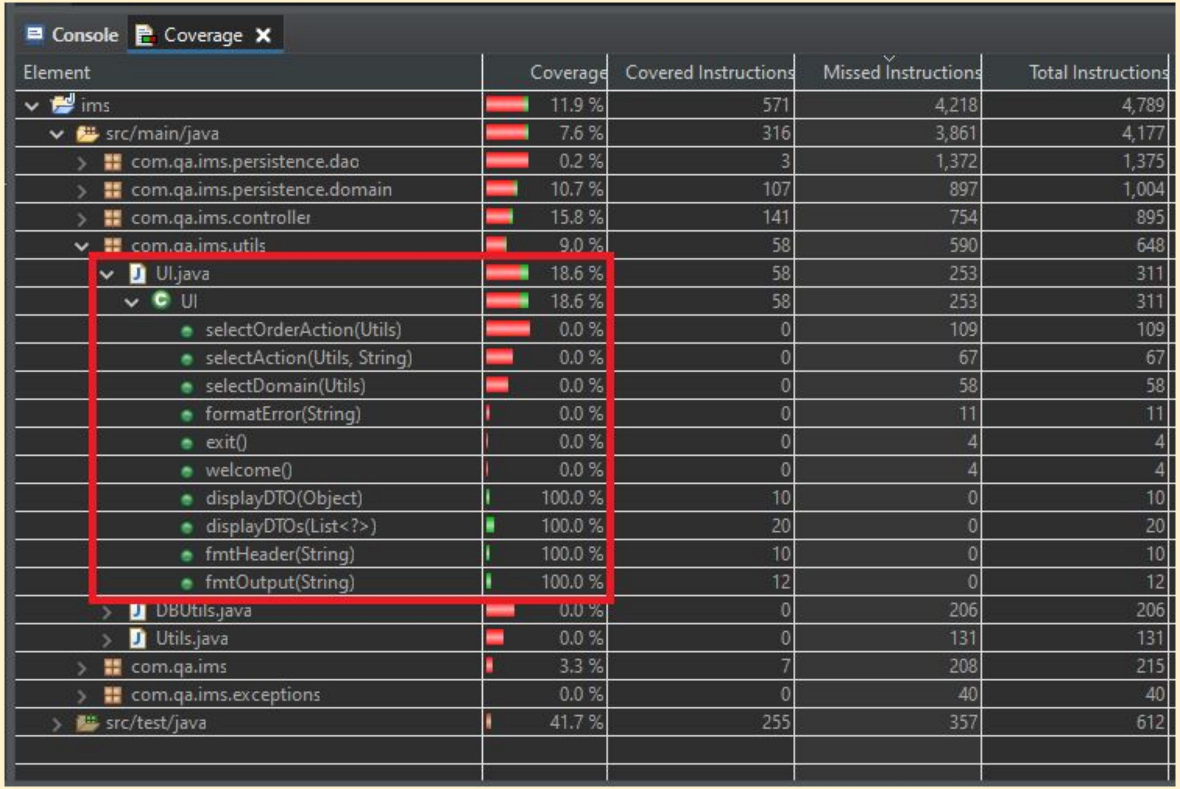
- Initial design led to discovering that static/singleton classes can’t be mocked.

- Static instances and singleton classes introduce tight coupling of classes, and removes the ability to mock those external dependencies.

- This leads to undesirable coverage of classes outside of the system under test, violating the core principle of unit testing, which is to test in isolation.

Unit Testing

- Testing the ItemController revealed unintended coverage of the UI class, due to the UI being accessed statically.

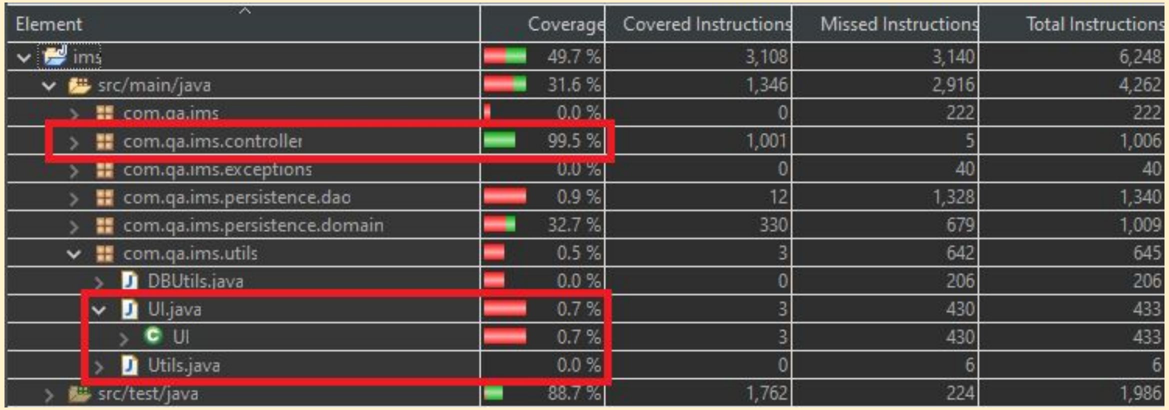


Console | Coverage

Element	Coverage	Covered Instructions	Missed Instructions	Total Instructions
ims	11.9 %	571	4,218	4,789
src/main/java	7.6 %	316	3,861	4,177
com.qa.ims.persistence.dao	0.2 %	3	1,372	1,375
com.qa.ims.persistence.domain	10.7 %	107	897	1,004
com.qa.ims.controller	15.8 %	141	754	895
com.qa.ims.utils	9.0 %	58	590	648
UI.java	18.6 %	58	253	311
UI	18.6 %	58	253	311
selectOrderAction(Utils)	0.0 %	0	109	109
selectAction(Utils, String)	0.0 %	0	67	67
selectDomain(Utils)	0.0 %	0	58	58
formatError(String)	0.0 %	0	11	11
exit()	0.0 %	0	4	4
welcome()	0.0 %	0	4	4
displayDTO(Object)	100.0 %	10	0	10
displayDTOs(List<?>)	100.0 %	20	0	20
fmtHeader(String)	100.0 %	10	0	10
fmtOutput(String)	100.0 %	12	0	12
DBUtils.java	0.0 %	0	206	206
Utils.java	0.0 %	0	131	131
com.qa.ims	3.3 %	7	208	215
com.qa.ims.exceptions	0.0 %	0	40	40
src/test/java	41.7 %	255	357	612

Unit Testing

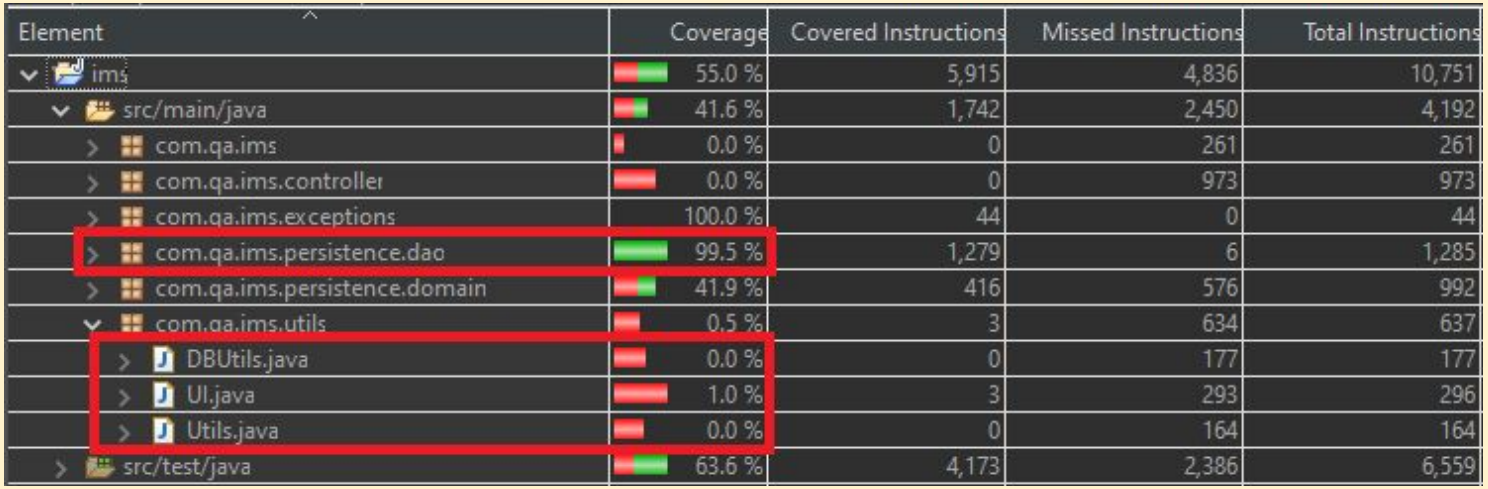
- After refactoring to use the dependency injection technique, the classes became loosely coupled, and I was able to Mock the UI and fully isolate the Controller classes for unit testing.



Element	Coverage	Covered Instructions	Missed Instructions	Total Instructions
ims	49.7 %	3,108	3,140	6,248
src/main/java	31.6 %	1,346	2,916	4,262
com.qa.ims	0.0 %	0	222	222
com.qa.ims.controller	99.5 %	1,001	5	1,006
com.qa.ims.exceptions	0.0 %	0	40	40
com.qa.ims.persistence.dao	0.9 %	12	1,328	1,340
com.qa.ims.persistence.domain	32.7 %	330	679	1,009
com.qa.ims.utils	0.5 %	3	642	645
DBUtils.java	0.0 %	0	206	206
UI.java	0.7 %	3	430	433
UI	0.7 %	3	430	433
Utils.java	0.0 %	0	6	6
src/test/java	88.7 %	1,762	224	1,986

Unit Testing

- My DBUtils class containing the database connection suffered from a similar issue due to being a singleton.
- When testing the DAO classes, they made real connections to the database, and as such had coverage of the DBUtils class.
- By refactoring to the dependency injection technique, the DAOs could be tested in isolation.



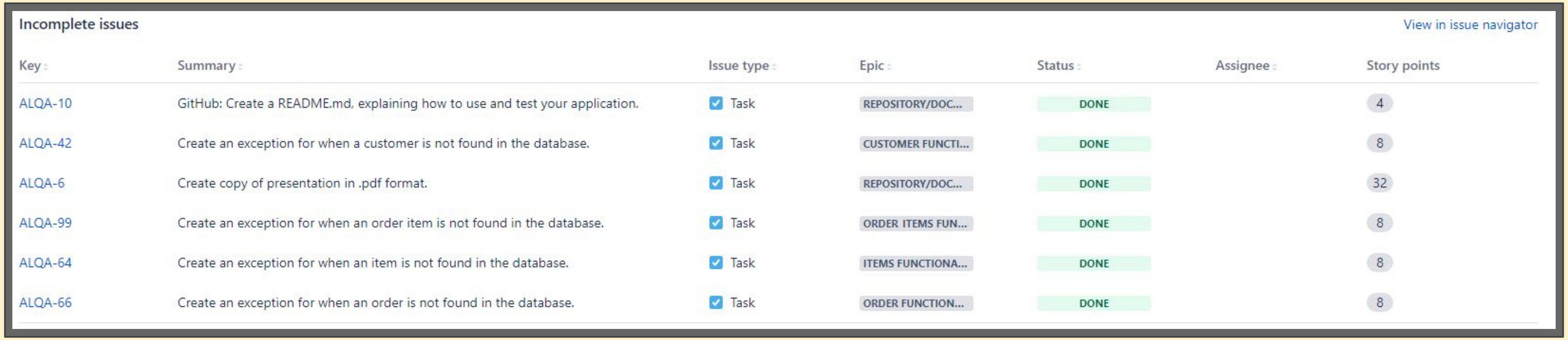
Element	Coverage	Covered Instructions	Missed Instructions	Total Instructions
ims	55.0 %	5,915	4,836	10,751
src/main/java	41.6 %	1,742	2,450	4,192
com.qa.ims	0.0 %	0	261	261
com.qa.ims.controller	0.0 %	0	973	973
com.qa.ims.exceptions	100.0 %	44	0	44
com.qa.ims.persistence.dao	99.5 %	1,279	6	1,285
com.qa.ims.persistence.domain	41.9 %	416	576	992
com.qa.ims.utils	0.5 %	3	634	637
DBUtils.java	0.0 %	0	177	177
UI.java	1.0 %	3	293	296
Utils.java	0.0 %	0	164	164
src/test/java	63.6 %	4,173	2,386	6,559

Demonstration

A few minutes to demonstrate the application

Sprint Review

- Sprint 1 - 9 April 2021 - 19 April 2021
  - 81/87 issues completed
  - 528/596 story points completed
- The following 6 issues were not completed and were moved to a new sprint



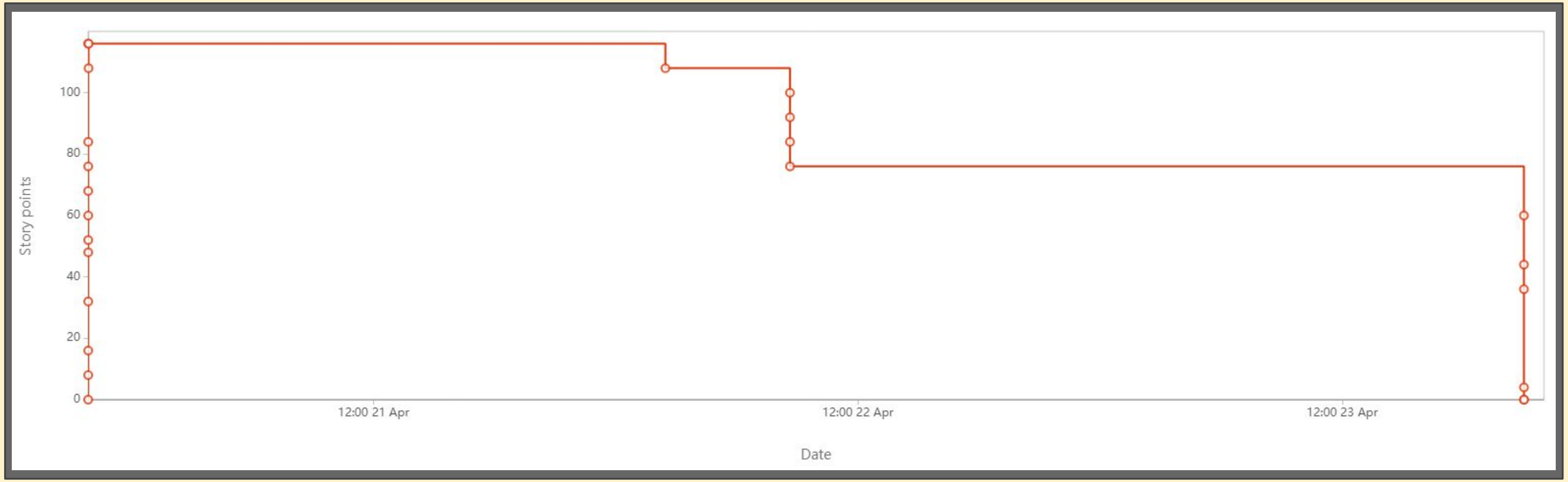
Incomplete issues

View in issue navigator

Key	Summary	Issue type	Epic	Status	Assignee	Story points
ALQA-10	GitHub: Create a README.md, explaining how to use and test your application.	✓ Task	REPOSITORY/DOC...	DONE		4
ALQA-42	Create an exception for when a customer is not found in the database.	✓ Task	CUSTOMER FUNCTI...	DONE		8
ALQA-6	Create copy of presentation in .pdf format.	✓ Task	REPOSITORY/DOC...	DONE		32
ALQA-99	Create an exception for when an order item is not found in the database.	✓ Task	ORDER ITEMS FUN...	DONE		8
ALQA-64	Create an exception for when an item is not found in the database.	✓ Task	ITEMS FUNCTIONA...	DONE		8
ALQA-66	Create an exception for when an order is not found in the database.	✓ Task	ORDER FUNCTION...	DONE		8

Sprint Review

- Sprint 2 - 20 April 2021 - 23 April 2021
  - Goal: Finish application and prepare presentation



Sprint Retrospective

- What went well?
- What could be improved?

Conclusion

...and 5 minutes for questions