MoSCoW Prioritisation

WON’T HAVE

COULD HAVE

SHOULD HAVE

MUST HAVE

MUST HAVE

* Code fully integrated into a Version Control System. (Git)
* A build of the application present in the root folder of your Git repository.
* A working application that is interactive with a front-end command-line interface (CLI)
* A **fat .jar** which can be deployed from the command-line
* A completed **project management board**, including user stories, acceptance criteria, estimations via story points, and prioritisation via MoSCoW methodology.
* A risk assessment which outlines the issues and risks faced during the project timeframe, utilising a matrix and in **.pdf** format.
* A functional application ‘back-end’ (Java), following best practices and design principles, meeting the requirements set on your project management board.
* A build of your application, including any dependencies it might need, produced using an integrated build tool (Maven).
* Unit tests for validation of the application. (Junit & Mockito)
* The project connecting via JDBC to a local-based MySQL instance
* CRUD functionality following the Enterprise Architecture Model for the **customers**, **items**, and **orders** entities.
* Item entity must have a name and value fields.
* The means to add/delete an item to and from an order.
* The means to calculate the cost of an order.
* A relational database used to persist data for the project, containing the **customers, items, orders, and orders\_items** tables.
* A primary key in each of the entities; **customers, items, orders, and orders\_items** tables.
* A sensible package structure.
* A completed **README.md**, explaining how to use and test the application.
* At least **one** **ERD** and **one** **UML** diagram, in **.png** format
* A copy of the presentation, in **.pdf** format (slides only – no notes)

SHOULD HAVE

* The Git repository utilising the feature-branch model: **main/dev/multiple features**.
* 5-7 risks with a matrix
* Relationships between tables modelled using an ERD.
* **Before-and-after ERD’s and UML**
* Unit test coverage of CRUD functionality of the **src/main/java folder,** aiming for **80%**
* A **fat .jar** which can be deployed from the command-line
* Adherence to best practice (OOP principles, SOLID, refactoring)
* A working **“.ignore”** for ignoring build-generated files and folders
* The means to calculate order total

COULD HAVE

* The project connecting via JDBC to a GCP-based MySQL instance
* Test coverage of CRUD Functionality **above 80%.**
* Data in the Orders table timestamped.
* A description and a stock count in the Items table.
* Evolution of ERD/UML Diagrams over time
* User-system; Username and password

WON’T HAVE

* Extra functionality not included in the specification
* No unnecessary imports/dependencies
* A Graphical User Interface
* Incomplete documentation
* Log-in function/portal