**COM528 AE2 REPORT**

Ben Lever

GITHUB: <https://github.com/BenLever/COM528-AE2-BenLever>

Contents

[1. Design Decisions 2](#_Toc92466728)

[2. Diagram Decisions 2](#_Toc92466729)

[3. Design And Code Discrepancies 3](#_Toc92466730)

[4. Test Plan Strategy 3](#_Toc92466731)

[5. Evaluation 3](#_Toc92466732)

[6. Bibliography 4](#_Toc92466733)

# Design Decisions

During the design process I considered many different approaches, I decided on altering the given template for my model. I added a catalogue page that could add new products to the stock, also implementing a system to monitor the stock and how many items were available at a given time. After this, I added a cart page that included the payment screen to purchase the items.

To better organise the project the system was built using MVC – Model, View & Controller. Using this architecture it made the development of supplementary features much easier to add into my application. Specifically, I used the MVC framework known as Spring MVC (17. Web MVC framework, 2022). For my build system, I decided upon using Maven (Porter, Zyl and Lamy, 2022). Maven is very useful as the dependencies are manually added to the project just by reading the POM files. Another important feature of Maven is its multi-module projects, this makes it so I can compile my web project to a WAR to be run by Tomcat all whilst my other projects are compiling to a JAR file.

I created suitable use cases to account for these newly added features from a range of POVs. For example:

* Login from all POVs
* The removal of items from the cart
* The purchase of items
* Admin - adding a new item to the catalogue
* Admin – viewing and modification of users

I believe the use cases provided show a comprehensive usage for the application and show the system working in the way it is intended to be used. I would have liked to have added additional features such as the ability to search for items or sort items by category or price providing use cases for each.

# Diagram Decisions

Diagram

Description automatically generatedWhen creating the diagrams to plan my project, I had to carefully consider how each page would interact with one another. Using the template robustness diagram, I changed and added my new features to the diagram allowing me to see clearly how the project would run but more importantly to see where the challenges and errors may lie before beginning the project.

To aid with the creation of the class diagrams I used the NetBeans plugin called Easy UML (GitHub - mgeee35/easyUML: easyUML plug-in with .nbm files, 2021). Showing every part of the system from the bank, web, model, service and DAO.

I then moved onto the sequence diagram where I could implement my added features and see how they and their requests would work in conjunction with the other layers of the application including the session, service layer and data access layer.

# Design And Code Discrepancies

I was unable to incorporate the orders page fully showing the customers orders and invoices. I would have also spent more time on the catalogue, as of now you can only add an item accompanied by its price and quantity. I would have liked to be able to modify and update old items for a seamless CRUD operation.

As of now, the stock level does not decrease when the order completes, and the user’s cart doesn’t clear upon purchase either.

# Test Plan Strategy

For my application I wanted to offer Unit Tests that covered every component such as Models, Services, and repositories, I did this by using JUnit (Stefan Bechtold, 2021). However, this alone is not enough as it only provides testing for individual elements. To ensure full testing I performed user tests manually.

Using manual tests allowed me to know whether the UI is functional or not. I did this using multiple POVs who use the system.

# Evaluation

Overall, this module challenged my coding knowledge but also my ability to design a project from the bottom up. I found it very rewarding seeing my plan come together. However, this was my first time coding in Java so I know the structure of my code and design could be improved if I were to attempt this task again. If I had more time I would also like to add additional features to the application such as the ability to search and the ability for a user to save their card details to their account. The vulnerability of the shopkeeper’s card is also in question as it is stored in a properties file and could very easily be uncovered. In conclusion, I was unable to complete the more advanced areas of the module however I am very happy with how my application has turned out and my knowledge of sound coding practices and workflow has greatly increased.

# Bibliography

Docs.spring.io. 2022. *17. Web MVC framework*. [online] Available at: <https://docs.spring.io/spring-framework/docs/3.2.x/spring-framework-reference/html/mvc.html> [Accessed 7 January 2022].

GitHub. 2021. *GitHub - mgeee35/easyUML: easyUML plug-in with .nbm files*. [online] Available at: <https://github.com/mgeee35/easyUML> [Accessed 7 January 2022].

Porter, B., Zyl, J. and Lamy, O., 2022. *Maven – Welcome to Apache Maven*. [online] Maven.apache.org. Available at: <https://maven.apache.org/> [Accessed 7 January 2022].

Stefan Bechtold, C., 2021. *JUnit 5 User Guide*. [online] Junit.org. Available at: <https://junit.org/junit5/docs/current/user-guide/> [Accessed 7 January 2022].