**Report on Analysis, Design, Testing and Code Artifacts**

**With critical evaluation of the failure of the application**

Link to Repository: https://github.com/lewis-holmes-98/com528\_ae2.git

### Introduction

The goal of the application is to provide a RESTful[[1]](#endnote-1) web-based shopping system which allows for users to search for specific items, add them to a shopping cart and then purchase the cart items. The purchasing will interact with a bank API which checks the user’s card is valid, detracts money from their bank account - or decline the transaction if there are not enough funds. The bank API will also be able to refund transactions to users. The refund would also 'restock' the refunded item in the catalogue to give the appearance the customer has returned the item.

Users will be able to log into the online shop. Depending on the credentials they provide, they will be given permissions associated with that account. For example, an Admin will have every privilege such as modifying orders as well as adding or removing items from the catalogue. Whereas a standard customer will only be able to add products to their cart, purchase items and view previous orders.

Utilising a RESTful interface greatly increases the effectiveness of the application, separating client and server reduces load of machines and allows for a greater division of time, skill and other assets between developers.

* New users will also be able to create a new account with provided details.
* The system will interact with a database in order to maintain track of stock.
* Logging must be suitably implemented throughout and detail any transaction, including failed transactions.

### Models and Use-case

Using the aforementioned requirements for reference, models and use-cases were created according to each of the main application features. Once a class had been created and populated, a class diagram and model can then be generated to provide a visual guide to how the class operates. The driving force behind the choices taken for these models and use-case was to be able to create and showcase a simple, yet effective and technology rich e-commerce application that other developers and users would be able to understand.

Furthermore, use-cases allow for a series of check-list events that a developer can carry out and ensure the correct result, much like code testing; a use-case allows for the developer to pre-plan for their Continuous Integration with making sure the basic structure and responses of the application are as expected.

### Code that did not match design

As is common with app development and Object-Orientated programming. There were instances where code had to be omitted, added, or changed despite the plan of integration. One such example is in the MVVController, where addition links and Lists were implemented in an attempt to rectify an issue with the storing of invoices one an order had been placed by the customer. The additional code included more calls to different models and repositories than previously expected. This added a layer of complexity to the code that was initially not supposed to be.

Additionally, by design the application was supposed to have a stock tracking feature and bank integration. Each item would have an associated, database-driven, stock value which could be incremented up or down depending on the users’ orders. This was not included alongside the charging to customers cards through a bank API. This was down to time-constraints and focusing on finishing core parts of the application.

### Testing

By making sure that all classes have Unit tests we can ensure our code has a great code coverage. This is especially important and useful when implementing Continuous Integration as it allows for the identification of any potential code breaking changes and gives the opportunity to rectify any issues prior to being pushed to the main branch. Continuous Integration with testing also creates a faster release schedule, this can be due to the reduction in time finding and fixing bugs, as many app breaking bugs will be identified in the testing process prior to push. “...calculated the release rate as the number of releases per month. Projects that use CI average .54 releases per month, while projects that do not use CI average .24 releases per month. That is more than double the release rate, and the difference is statistically significant (Wilcoxon, p < 0.00001).”[[2]](#endnote-2)

When designing the test plan, it was important that all main classes had these tests in place as well as further testing of functions to ensure the code was working and operable before pushing to a branch, or the release to a customer if applicable.

### Critical Evaluation of Code and Design

Designing and envisioning the application was successful, I have experience with front-end web applications and understand how they should work and which features are a priority, and which are nice-to-haves. Despite this, the implementation of the code in this assessment proved extremely difficult. Java is a very new language to myself and I struggled to understand its nuances in time to create a fully working application. The code included is operable, though unable to be demonstrated due to an issue with Tomcat serving the web-context. The included error “No qualifying bean of type 'org.solent.com504.oodd.cart.model.service.ShoppingService' available: expected single matching bean but found 2: shoppingServiceImpl, getShoppingService” allows me to believe I have mistakenly integrated a linked wire where I should not have.

After hours of investigation, I was left without an answer. The decision to keep the code as is, allow it to build without error and investigate the web serving error later. Poor time management and last-minute commitments meant that this issue was left unresolved and therefore the code is marked as a critical failure.

I am content with the code I had written, being a predominantly front-end developer, I was pleased with the progress I had made with the Java language and have gained a much wider understanding on how to use it. Though, as demonstrated in my submission, much more disciplined time and effort must be taken in order to be able to deliver respectable code that is fit for purpose and fits the brief.

To conclude – as a developer I must be more aware in the future of additional steps that need to be taken to ensure that both the design, and the implementation of the code and up-to standard. I entered the assignment with a presumption of success due to my prior web-application knowledge and did not dedicate enough time to design, learn and create the application. I have highlighted the key areas for improvement, as listed below:

1. Java classes, polymorphism and its uses in a dynamic web-application
2. Writing and implementing coherent and reliable class tests
3. Understanding multi-package Java applications and communication between them
4. Maximising the effectiveness of repositories and data access within class

If I were to repeat the assignment, I would first ensure understanding of the above prior, instead of attempting to learn during, the assessment period.

### References

1. Codecademy. (2022). *What is REST?.* Available: https://www.codecademy.com/article/what-is-rest. Last accessed 15/01/2022. [↑](#endnote-ref-1)
2. Timothy Tunnell, Michael Hilton, Kai Huang, Darko Marinov, Danny Dig. (2016). Usage, Costs, and Benefits of Continuous Integration in Open-Source Projects. *Usage, Costs, and Benefits of Continuous Integration in Open-Source Projects*. 1 (1), p1 - 10. [↑](#endnote-ref-2)