Ticket Machine Web App Report

Tutor: Craig Gallen

Unit: COM 528

Jacob Dale

# Contents

# Decisions Made During Creation of Diagrams

# Where the Code doesn’t Match the Design

# Rationale for Test Strategy

The test strategy I created for this piece of work has a simplified format of the one I use for testing at work. It has a basic design because the tester would usually be someone who doesn’t know what the code behind the app looks like. The tester might not even have a working knowledge of programming in Java or any other language so, I tried to make it as fool poof as possible. I am concerned that the language I used may be too informal for a real-world test. That would be addressed in the next version of the document.

Instead of writing the test document against the use cases I have designed, I wrote the test as if I was a user using the system and checked I had tested against all the software requirements once the first version was complete.

I tried to work in an agile manner, which allowed me to write my test document before creating the use cases. I feel that this helped me when creating my use cases as it gave me the experience with what the app needed to do.

# Critical Evaluation of Code Implementation and Design

Overall, I am content with the implementation of the code. I am aware there are many improvements that can be made to improve the efficiency and user friendliness of the app. One of the big improvements I would make is to abstract the embedded Java code away from the JSPs. This would decouple the app a little which - in theory - should improve the scalability, help reduce development time in future and make the app more understandable for fresh eyes. I am seeing 100+ lines of embedded Java a little daunting.

Abstracting the Java form the JSPs would ease unit testing because if I wanted to unit test the current embed Java I would have to copy and paste it into a unit test and every time that code gets a slight change, the unit test would or it would become invalid. If the code was abstracted all I would have to do is reference the corresponding classes and test their function. Therefore, I have not implemented many unit tests; I received the project with unit tests already written to test the backend of the app. In future versions I would add to these unit tests as I got a better understanding of the backend. The test procedure should cover for the lack of frontend unit tests making it an unworthy investment of time to write said front end unit tests. In a future sprint abstracting the Java and creating frontend unit tests would be the top priority, I would use Test Driven Development to ensure it.

As for the design, there wasn’t much for me to add to the class diagram as I only added JSPs and edited pre-existing functions. The Use cases were simple to create as I could use my test document as a guide for what a user would do in the app. Robustness diagrams have always been a weakness of mine as they are a new concept and there isn’t much information about them easily accessible on the internet. I used a previous robustness diagram as a reference for what one should look like.