Year 2 Project – Music Store Sales and Stock Control

Music Store Sales and Stock Control

**Group Members:**

*James Lynam – X00073019*

*Craig Larkin – X00148375*

*Stephen Murphy – X00149904*

*Mark Kenny – X00091622*

**Second Year Project**

***Iteration 2 - Music Store Sales and Stock Control – The Soundgarden***

**Iteration 2 Report**

**Where we’re at:**

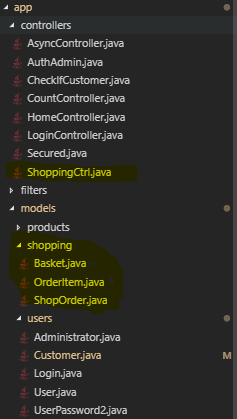
For iteration 2, our main goals were as follows:

* To eliminate any residual errors that were still present from the previous iteration.
* To add increased functionality, such as session management, stock management, image uploads, audio uploads and a shopping cart function.
* To maintain the testing practice, we had used in the previous iteration (with selenium).
* A search filter.

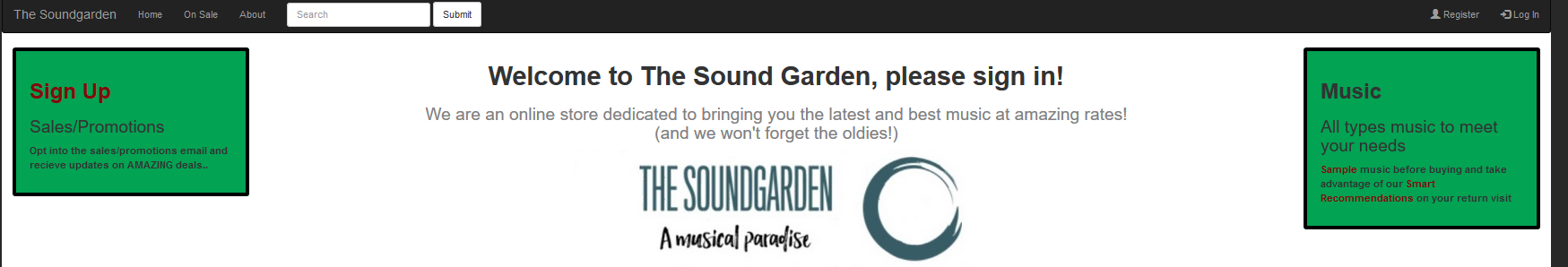
We began by expanding the CSS from what we originally had to make our site feel more unique.  
This allowed us to visualize in our minds what we wanted and where we wanted it, for a better user experience. Following the lab sheets from our SDev revision labs, we managed to successfully implement most of the things we had set out to do.

We wanted the users to be able to manage their own stock, also. They should be able to empty their cart and add to cart etc, with new values being reflected within the database.

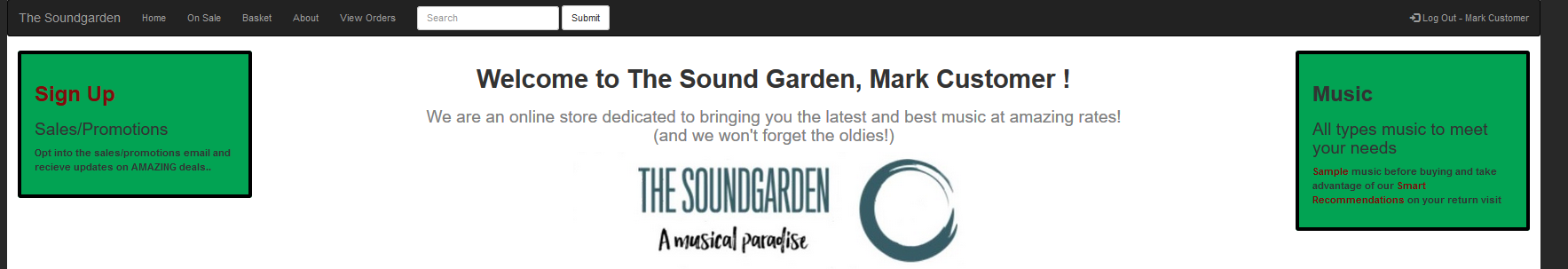
Many new classes were constructed in order achieve this functionality, such as Basket.java and ShoppingCtrl.java.

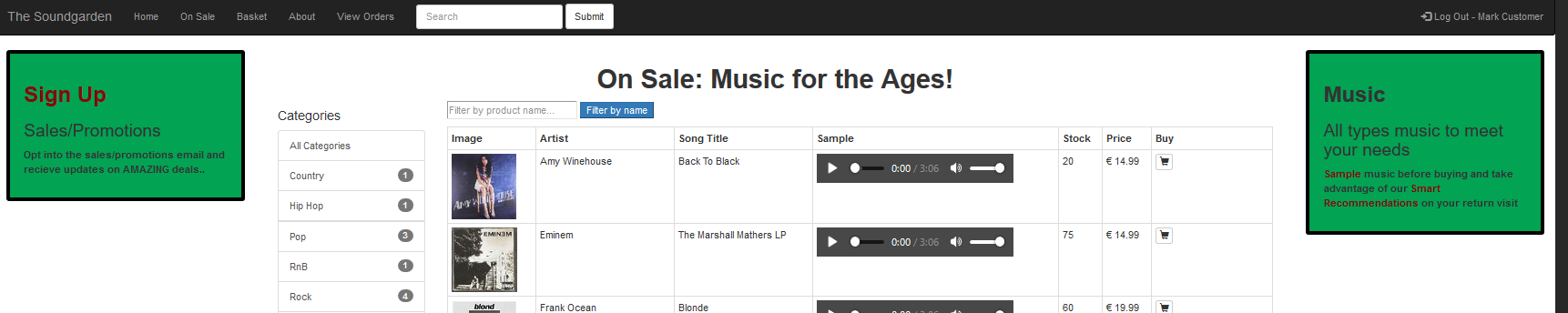


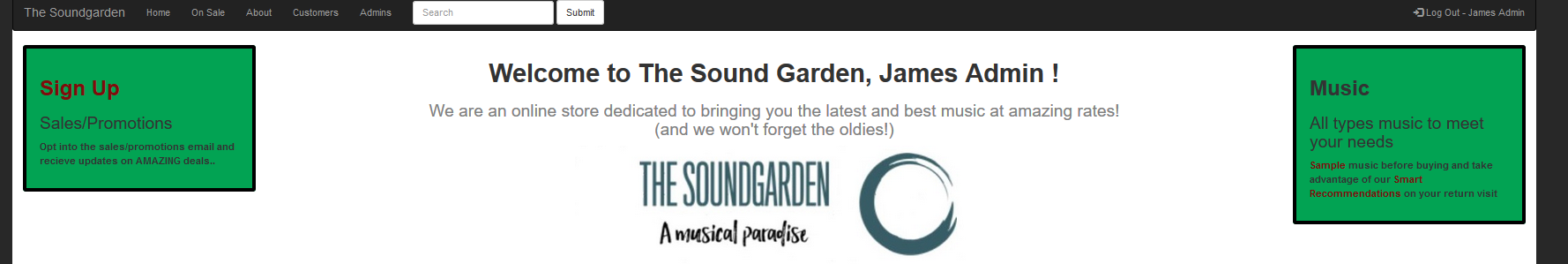
The site should also display the name of the user that is currently signed in, and a default message if nobody is signed in. This is to avoid confusion.

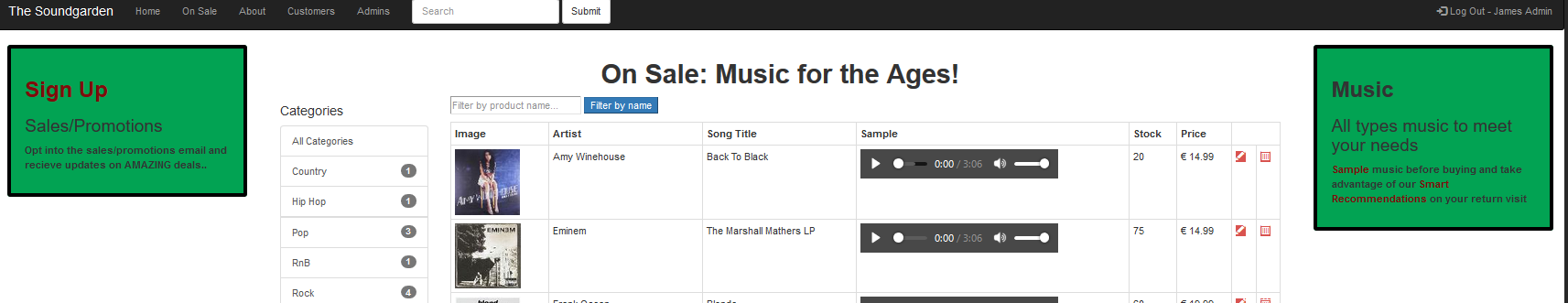


Much like the previous iteration where only admins could see the CRUD icons, only customers can now use the shopping cart features.



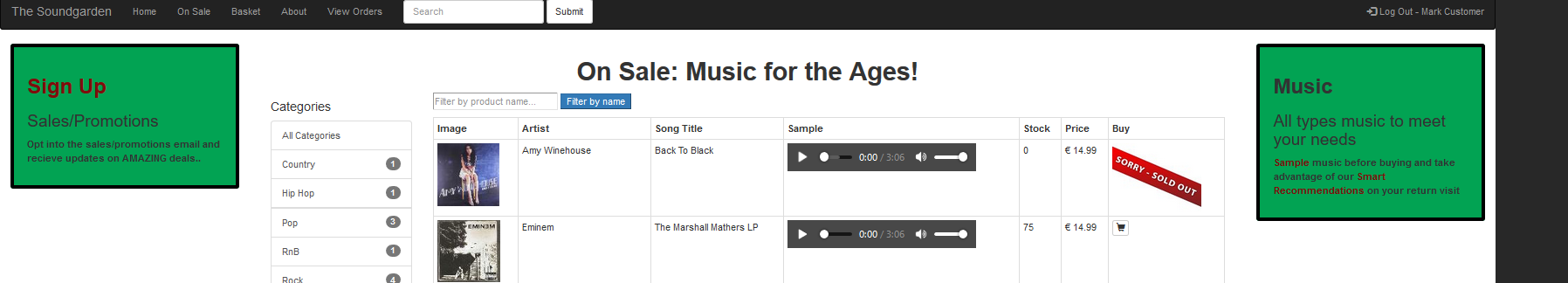




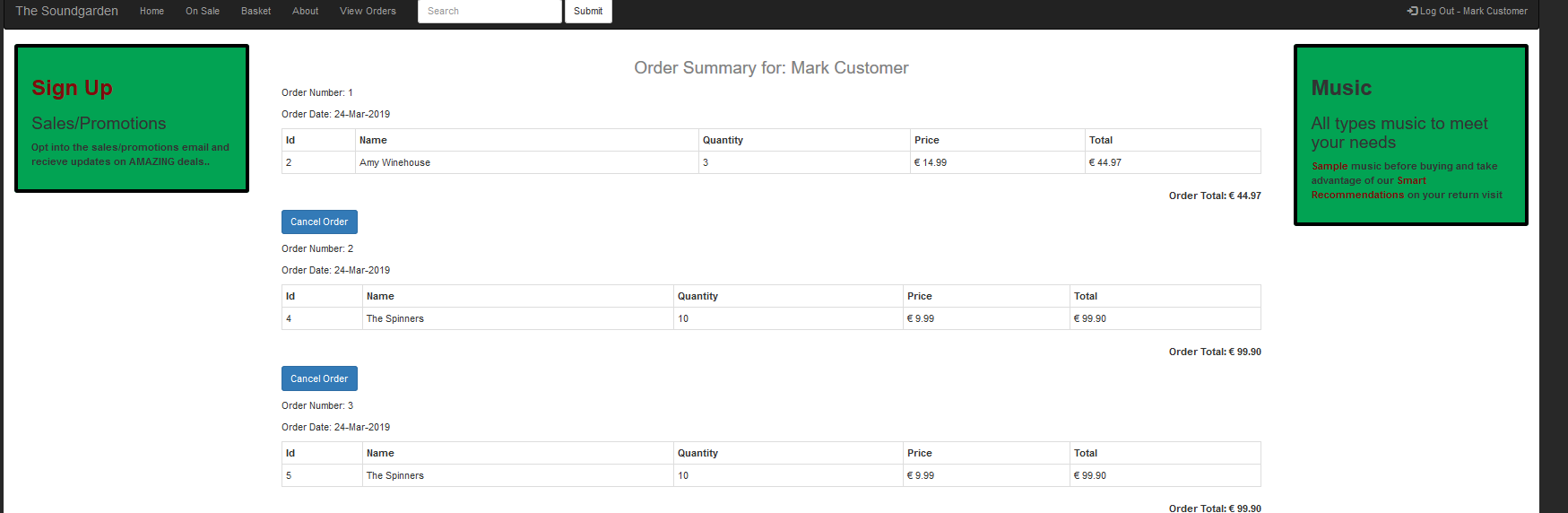


Also seen in the previous screenshots are examples of the relationship mappings of the items, as well as the images and music samples within the form.

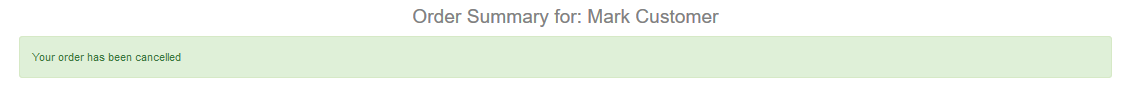
We also wanted the user to be shown an “out of stock” message if the item runs out:



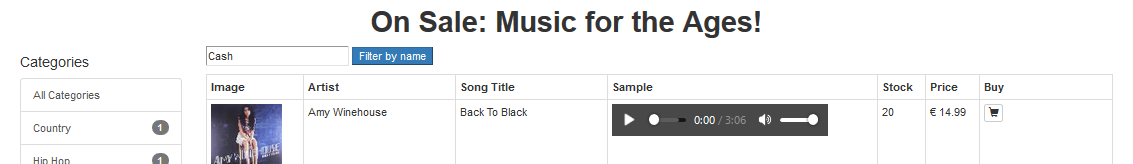
We gave the user the option to view all the current orders with an option to cancel them if they so desired:

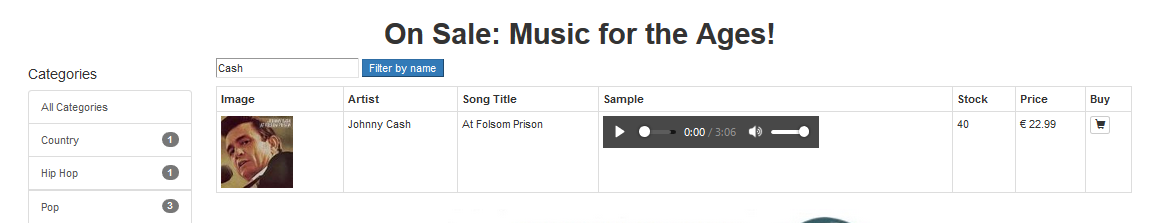


Upon cancellation:

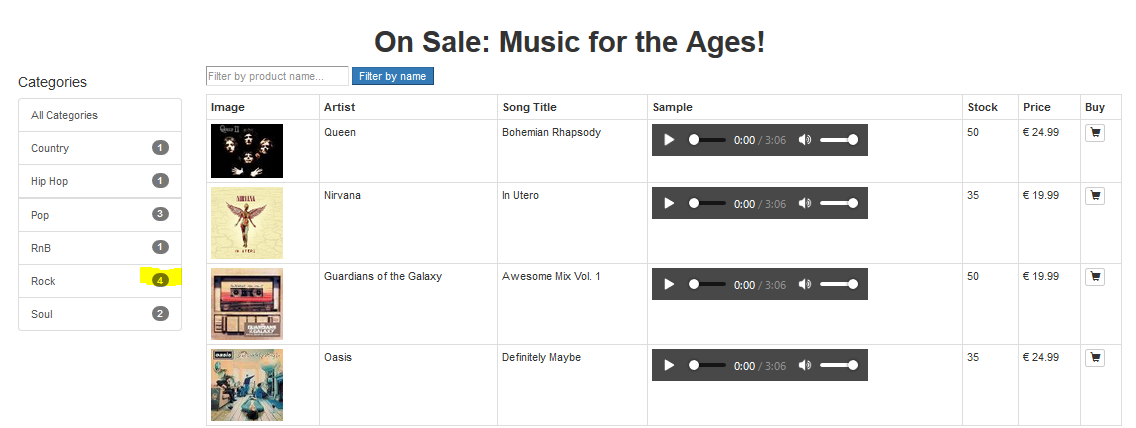


As far as our search filter goes on the On Sale page, right now it is limited to the Artist’s name. We hope to improve this for iteration 3.





And an example of the categories filter (click on a genre):



The search bar inside of the nav is currently for show only, we hope to implement functionality soon.

**Testing:**

We created a series of tests to test all this functionality in Selenium.

**Moving Forward / Next Steps:**

As we go forwards we need to work more on implanting the subcategories and creating a form in which the admins can upload a music sample for each item.

As far as the subcategories are concerned, we would like it to look like this:

**ItemOnSale > Music  
 > Apparel > Hats  
 > Clothes etc…**

For a start at least, more could be added if we have time.

We also plan to further improve the CSS and make it a truly unique visual experience.  
Currently, with the audio sample within the site, they are just all sampling the same song by default.

We don’t anticipate many problems with these desired implementations, as we feel we would have had them done if we didn’t have to do the many other CAs that we had been assigned that overlapped with this project. We hope to have the time to do it all.

We also need to implement a checkout system more efficiently. This would involve creating new routes and a view for a check out confirmation page which would prompt the user for payment information and display a receipt.

Finally, we want to introduce some sort of review system, perhaps a star rating or comment section and to add a payment form for checking the user out. We would use Stripe for this.

**Errors Encountered**

**SQL Errors**

**Error:** When attempting to insert many of our SQL statements, we kept receiving many SQL insertion errors (and everybody loves those!).

**Our understanding of the error:** Basically, because we had the errors originally, and our addItem form worked, we shelved the SQL aspect for a brief period until we had a deeper understanding of the system. Once we had figured out how resolve the issue, we returned the insertions.

**Solution**: At this point we received another error, but this was merely due to the fact there had been an ID conflict between the items that we had added manually, and the items the SQL was inserting. We deleted the items and re-ran, and it worked.

**NullPointer exceptions as Customers**

**Error:** We were receiving a repeated NullPointerException whenever we logged in as a customer and click the Basket tab.

**Our understanding of the error:** The Basket tab had no data to display.

**Solution:** We realised that there had been an item still in the cart from before we fixed the SQL as described in the previous error. We fixed it by simply adding an item the cart and clicking the Basket tab once again.

**Registering a new user**

**Error:** When we register a new user, a role is not assigned.

**Our understanding of the error:** Whenever a new user is created through the form provided, it is added to the database as a “user”, and not a “customer” or “admin”. This is an issue because only these 2 roles have the permissions and data associated with them to render the information.

For example, a Customer can click the On-Sale page to view the items, but a “User” cannot as their role is null, so a NullPointerException is thrown. We should note that a user that is not logged in can still view the products, but they cannot interact with them.

**Solution:** No solution yet, but we feel like we could figure it out relatively easy. We would just need to adjust the form so that a user is assigned a role before the save.

**Unable to cancel “legacy” orders**

**Error and Understanding:** This is somewhat a “pseudo-error” in that a customer cannot cancel an order if too much time has passed (which is what we want!). However, as we were testing our functionality, we discovered that if one of us made an order, saved the state and pushed to our repo, that anyone who would clone it would not be able to cancel these orders due to “too much time passing”, when it hadn’t.

**Solution** – Simply remember to cancel an order we do not to be permanent before saving the state of the project. A further future solution has yet to be devised.