Team Proj18

Project Deliverable G: **Prototype II and Customer Feedback**

GNG 1103 – Engineering Design

Team Members:

Shuyuan Bai (300023989)

Grace Buchardt (300236838)

Craig Bush (300251044)

Simon Situ (7761503)

Steven Wu (7883953)

Faculty of Engineering – University of Ottawa

2021/11/11

Abstract

In this deliverable, our second prototype is presented with all current features and planned revisions. The purpose of the second prototype was to create a functional UI that could process transactions, the selling, and buying of points. The prototype was tested by simulating transactions on the platform and comparing the cash and loyalty points balance before and after transactions. If the values after the transaction are consistent with predicted values, and if only valid transactions were processed then we can confirm our prototype was successful. We found that our prototype could perform simple valid transactions but did not have to ability to filter and prevent invalid inputs. These functions will be revised in future iterations.

Contents

1. Introduction

From the previous deliverable, we outlined the testing procedures and objects for the second prototype. In this deliverable, we will present the model for the second prototype and show the results from the testing. Potential refinements of our prototype will be discussed, along with the test plan for prototype III.

2. Prototype II Design

The objective for the second prototype was to create a prototype that could facilitate the buying and selling of points on the platform. Tests were performed using a test profile with a pre-existing balance for loyalty points and money. A buy and sell option was used to run a transaction to add or subtract points to the wallet. Executing this transaction would also change the cash balance. To validate the transaction balances were compared from before and after the transaction. If the final values after a trade are consistent with the predicted values, the test is successful. Multiple tests were performed using both valid and invalid inputs. The overview of the trading page can be seen in figure 1.

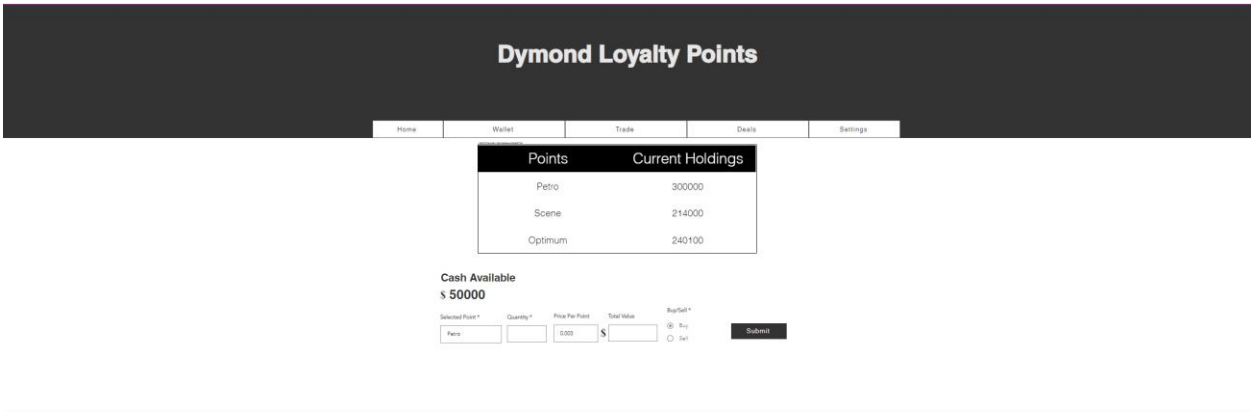

Figure 1. Trading Page for Loyalty Points Platform

a. Buying and Selling interface

A zoomed-in view of the buying interface can be seen in figure 2. The user would use the buy/sell radio buttons to choose the type of transaction they would like to perform. Depending on the selection, a conditional statement was used to add or subtract the corresponding total value and quantity from the profile wallet. Users select the point they would like the purchase by clicking on the table. From the input, the value in the points field will update to the respective loyalty point. The selected points field will also prompt the value for the price per point to update to the corresponding value. Users can choose the number of points they would like to purchase by inputting a value in the quantity box. From the quantity and the price per point, the program will automatically calculate the total value of the transaction and display it in the total value box. When users want to complete their transaction, a submit button is used to send the input, and the available cash and points holdings will update.

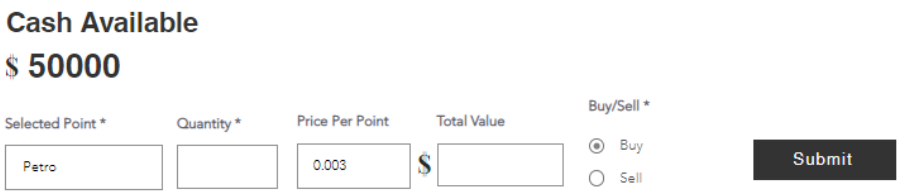

Figure 2. Buying and Selling interface for prototype II

b. Test Case 1: Buying Optimum points

An example test case for the prototype is shown in figure 3. This test case simulated a user purchasing Optimum points. The corresponding point was picked from the table, which updated the select points field to Optimum. The value per optimum point is $0.001, and this value is automatically updated in the price per points field. In this test case, the user will purchase 20000 Optimum points, which is equivalent to the value of $20. The buy option is selected from the radio buttons to let the software know the user wants to purchase points. Prior to the transaction the cash balance is $50000, and the number of Optimum points is 240100.

Points	Current Holdings
Petro	300000
Scene	214000
Optimum	240100

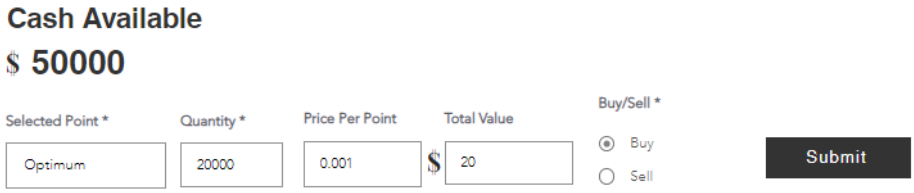

Figure 3. Test Case 1 Purchasing Optimum Points

After clicking the submit button, the cash balance and points holdings for Optimum were updated, as seen in figure 4. The cash available decreased by $20 from $50000 to $49980, which means the correct amount was subtracted. For the number of Optimum points, the value was updated from 240100 to 260100, showing an additional 20000 points were added.

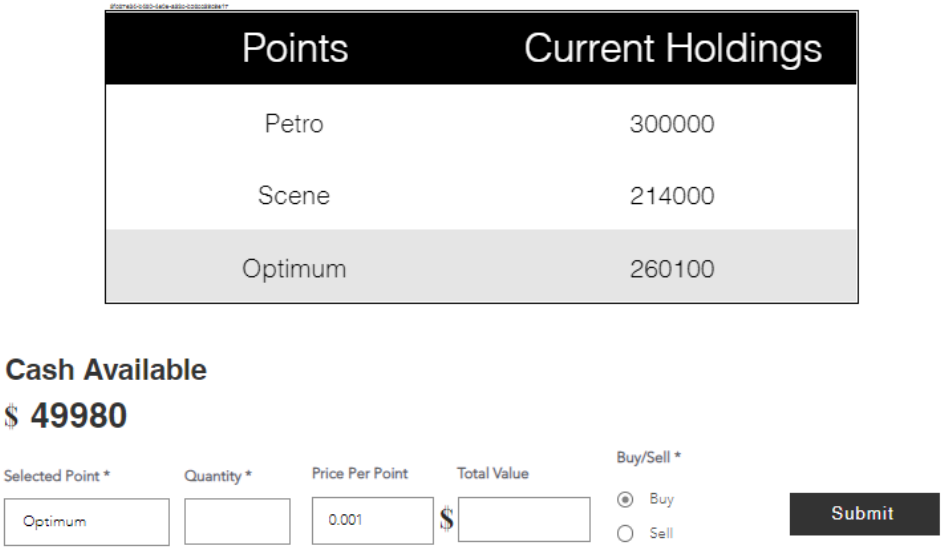

Figure 4. Test Case 1- Post transaction

Results from the first test case show a simple purchasing transaction can be performed if a valid quantity of loyalty points is entered and sufficient funds are on the profile.

c. Test case 2: Selling Scene points

In this second test case example, we will demonstrate a selling transaction for Scene points. Similarly, to test case 1, the user will select the loyalty point from the table, which updates the selected points parameter. When the user picks Scene, the price per point will update to $0.002. After the user specifies a quantity, in this case, the user will sell all their Scene points, the total value of the loyalty points, $428, is displayed in the total value box. The transaction before submission can be seen in figure 5.

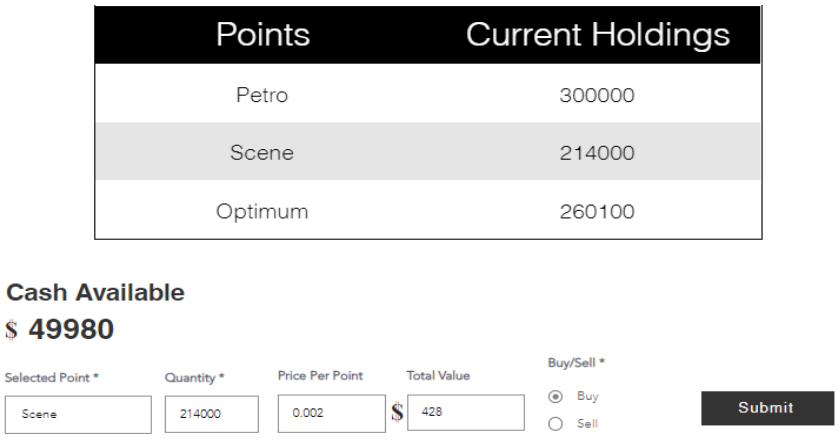

Figure 5. Test Case 2. Selling Scene points prior to transaction

The balances for the loyalty points and cash available can be seen post-transaction in figure 6. After submitting the transaction, the current holdings for the Scene points were updated to zero as expected, and the cash balance was increased to $50408, which was also the expected value. The results of test case 2 show that the prototype can simulate a selling function when valid values are input into the parameters.

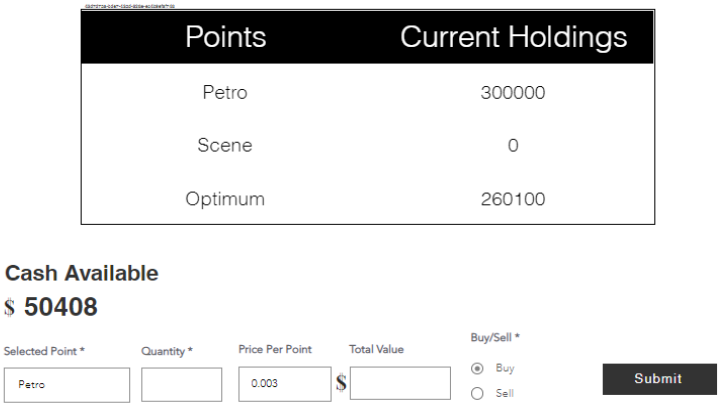

Figure 6. Test Case 2: Selling Scene Points post-transaction

d. Test Case 3: Invalid Inputs

The two previous test cases demonstrated the software's ability to process a transaction when valid inputs are entered into the program, but not invalid parameters. The ability to detect invalid inputs is necessary to prevent improper transactions from occurring. Examples of invalid inputs are users trying to sell more loyalty points than what they have available or users trying to buy loyalty points with insufficient funds. One method to prevent improper transactions is by using conditional statements to prevent the user from submitting their input when they don't have enough cash available to buy or not enough loyalty points to sell. The conditional statement can compare the total value, when buying, to the funds available, and if the value exceeds the purchasing power, the software will reject the submission. Currently, this feature needs to be developed into the software as invalid inputs can occur, as seen in figure7. This detection is the main limitation of our prototype that needs to be revised in further iterations.

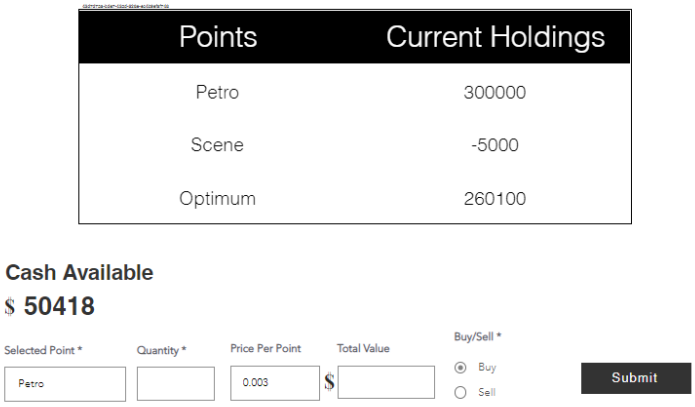

Figure 7. Example of an invalid transaction.

3. Prototype II Testing and Results

Our testing consisted of trying out our buying and selling features using various options and values. Overall, our buying and selling feature worked very well, and was able to effectively add and subtract the required amounts from both the user's point total as well as the user's funds. However, a few features of our testing did not work. First off, our code could not filter out invalid inputs in which the user tried to buy more than they could afford or sell more than they owned. Our code was also supposed to display a message reading "transaction complete" but this feature did not function in our testing.

4. Prototype II Future Additions and Troubleshooting

In the future we would like to improve upon our codes' limitations with our buying and selling function. To improve our code, we need to limit the user's ability to buy if the user does not have enough money as well as limit the user's ability to sell if the user does not have enough points. To fix this issue will create a conditional statement that checks what the user is trying to sell and checks it against how much the user owns. This would be implemented similarly for both buying and selling. Another issue is for people buying or selling points that have a value lower than one cent, this becomes a problem if the user decides to only buy or sell a few of these low valued points because the points end up coming to a total cost that is less than one cent. Our current code would still go through with the transaction and leaves the user with a money value that is more precise than what is allowed with Canadian currency. To fix this we will add code to force the user to buy and sell these low value stocks by factors that would make the transaction possible if it were to be done with real world money. In the future we would also like to add a feature that shows pending trades because not all trades happen instantly if this were to be a live website with active users. We would also like to add a changing value with the points based off the demand of the points as well as a graph to highlight the changing value. Future revisions will also include verifying the fidelity of the database. Data consistency is essential for our platform, so validating the dataset involved in account balances is necessary for the trading prototype. Furthermore, revisions in the overall usability of the site will also be explored for future iterations.

5. Prototype III Testing Plan

The focus for testing prototype III is to ensure users can properly access our engagement page and select surveys to complete for points. The inputs for the survey are to be stored in a database with input checks to ensure accurate submissions. Testers will be asked to access the page and fill out 2-3 surveys. In doing so, we will confirm that submissions are accurately filled out and properly stored in the database. As points are to be awarded to the user for completion of the test, a points check is needed to confirm that wallets are correctly updated after completion of the survey. If the final amounts match the records, the test is deemed successful.

Table 1. Prototype III Test Plans

Prototype Number	Test Objective (Why)	Description of Prototype used and of Basic Test Method (What)	Description of Results to be Recorded and how these results will be used (How)	Estimated Test duration and planned start date (When)
3.3	To create a functional engagement page for users to complete surveys for points	A page will be created to list available surveys for testers to complete. Points are distributed after successful completion of the survey(s).	Inputs to the database will be recorded to ensure surveys are functioning properly. Distribution of points will be recorded and checked for accuracy.	Creating a functioning prototype will require 3 days. Testing is to begin on Nov 15 and complete on Nov 18.

6. Conclusion

The results of our second prototype demonstrate that simple valid transactions for purchasing and selling can be performed using our platform. If the user inputs invalid parameters into the buying or selling software, transactions are still completed. In future interactions, conditional statements will be included to detect and prevent incorrect transactions. Furthermore, our group will also focus on creating the third prototype to test survey features to earn additional loyalty points.