CW2 Vending Machine Report

By Gareth Madoc-Jones ID:25646648

Module Leader: Dan Campbell

Edge Hill University

CIS 2701 Fundamentals of Web Coding

Contents

[**1.0 Introduction** 2](#_Toc187139908)

[**2.0 Chosen Technology** 2](#_Toc187139909)

[**2.1 Feature Designs and Implementation Discussions** 2](#_Toc187139910)

[**2.2 Adding Credits** 3](#_Toc187139911)

[**2.3 Product Selection** 4](#_Toc187139912)

[**2.4 Purchasing Items** 4](#_Toc187139913)

[**2.5 Refunding Credits** 6](#_Toc187139914)

[**3.0 Testing** 7](#_Toc187139915)

[**3.1 Functionality Testing** 7](#_Toc187139916)

[**3.2 Manual Testing** 10](#_Toc187139917)

[**3.21 Testing Input Values** 11](#_Toc187139918)

[**3.22 Testing Credit Value** 12](#_Toc187139919)

[**3.23 Testing Menus** 13](#_Toc187139920)

[**4.0 Reflection on Practices** 20](#_Toc187139921)

[**4.1 Accomplishments** 20](#_Toc187139922)

[**4.2 Challenges** 20](#_Toc187139923)

[**4.3 Short and Long-Term Programming Goals** 21](#_Toc187139924)

[**4.4 Conclusion** 22](#_Toc187139925)

[**5.0 References** 22](#_Toc187139926)

[**6.0 Appendix** 23](#_Toc187139927)

# **1.0 Introduction**

For this coursework, a vending machine application needs to be developed that can allow users to add credits. The credits act like the currency. The application should also allow users to buy a product or many products. The application will show the total cost and then the user should be able to checkout if they have sufficient credits. Also, at the end of the checkout the user should be able to refund any credits they have left. This application will be built using the vanilla JavaScript language without the use of unnecessary packages.

# **2.0 Chosen Technology**

For this project, the JavaScript language has been chosen. This is due to the popularity of JavaScript compared to PHP. In accordance with (Siddhpara, 2024) 98.9% of websites use JavaScript as the main client-side programming language and the Stack Overflow Report 2023 shows that JavaScript is among the top three programming languages in terms of popularity among professional developers. Using these statistics there is clear evidence that suggests that JavaScript has a popularity among the software developing community. This would mean that for this project if any issues were run into it would be much easier to troubleshoot and/or find possible solutions that can be adapted for the project.

In the project one module must be implemented into the code. This module is the readline module. This must be implemented due to their being no way in vanilla java script to implement user input, as JavaScript was not built to run console-based applications initially, without either using the prompt module or the readline module. The readline module was chosen over the prompt module due to the increased documentation and resources linking to the readline module and its uses. Also, the prompt module would more suit a web application rather than a terminal. Attached below is the code required to run the module.

A screen shot of a computer code

Description automatically generated

# **2.1 Feature Designs and Implementation Discussions**

For this project three Features where desired. The three features, which are desired is the adding of credits to the vending machine, the order selection, and the checkout. For this to be completed a flowchart was designed to help visualise the way that the system should work. The flowchart also acts as a visual representation of how the vending machine application runs. Before coding and completing this application having a well thought out visual diagram in the form of a flowchart can significantly increase the chances of the application running smoothly and having all the intended features such as adding credits and making purchases. Also, a flowchart allows for the overall inner workings of the vending machine application to be thought out visually and wrote down to give a more linear understanding of the application. Attached below is that Flowchart.

A split version of the flowchart is placed in the Appendix under images 3.0 and 4.0 to improve readability.

A black screen with white text

Description automatically generated

This flowchart shows an idea of how the application will work. Each feature implemented will be discussed more in depth in subsequent sections below.

# **2.2 Adding Credits**

For this feature to be implemented the addcredits function was added to the code. This function is a very simple function. It firstly asks the user to input how many credits they would like to add to the vending machine. The function handles an error check to firstly make sure that the credit value added is positive and secondly that it is a number. If one of these parameters is broken then the user will be prompted by the console that they have broken the input rules, and the function itself will reset. If the user enters a positive number, then they will be told by the console how many credits they have added and redirected to the main menu. There is a two second buffer for the redirection to give the user a chance to read the terminal to ensure they understand what they have done during this process on the vending machine application.

A black screen with colorful text

Description automatically generated

Below is an example of the addcredits function in use.

A screenshot of a computer program

Description automatically generated

# **2.3 Product Selection**

The next feature to be implemented is the product selection feature. This is where the user will be able to choose which product they would like to purchase. The user will be greeted with a product selection menu that firstly tells them how many credits they have and then there will be a drop-down list showing all the items available in the vending machine and their credit values as well as an option to return to the main menu. The user will be asked to input the number of what products they would like to purchase. If the user enters an incorrect value, then the Vending Machine will run an error to catch this value prompting the user to try again. It will then reload the Product Selection Page. Here is the code for the product selection function.

A screen shot of a computer program

Description automatically generated

Below is an example of this code in use. As soon as the user selects the product, they want the next feature is implemented.

A screenshot of a computer screen

Description automatically generated

# **2.4 Purchasing Items**

The purchasing of items in this vending machine application is managed by the complete\_purchase (itemname, itemprice) function. This function takes the input parameters itemname and itemprice that are assigned in the productselection menu and uses them to complete a calculation to determine the cost of an item. There are checks in place to determine if the user has more credits than the value of the item they are purchasing. If not, the user will be redirected to buy more credits. If they do, then the purchasing process will continue. The user will be asked how many of that product they would like to buy. If the amount they put in is negative and or a letter, then function will throw an error prompting the user to try again and it will re-run itself. If the amount is a positive number, then the function will continue and calculate the cost of that item. If the credit value for that is smaller than the total cost, then the user will be prompted again to buy more credits. If the cost is under or equal to the credit value, then the purchase will continue, and the credit value will be stored as a subtotal in case the user would like to make another purchase.

A screen shot of a computer program

Description automatically generated

Below is an example of this code running in the vending machine application.

A screenshot of a computer screen

Description automatically generated

The complete\_purchase function is a very complex function that after handling the initial purchase asks the user if they would like to make another purchase. The complete\_purchase function will manage the rebuying of items, this is when the user would like to buy more than one product at the same time, such as two bags of crisps and a chocolate bar. The complete\_purchase function will calculate the overall cost of both and take that from the credit value if the credit value is sufficient. If there are insufficient credits, then the user will be prompt to purchase more, and the purchase will be wiped from the system. For this to work the user will be asked if they would like to make another purchase if they do the above outcome will occur if not then the purchase will be completed, and the user will be returned to the main menu. There is an error catchment so if the user leaves the product selection page before completing a purchase in the case of buying more than one product, then the products chosen will be wiped from the system, and the purchase itself will vanish.

A black screen with colorful text

Description automatically generated

Once a purchase is complete the function then manages the storing of those items in an array. At the start of the function the chosen items and the amount of them were pushed to a temporary storing array called items []. Once the user has decided to check out with there items these will be pushed to a more permanent storing array called purchased\_items []. This is where the complete\_purchase function now depends on the outcome of another function.

A screen shot of a computer program

Description automatically generated

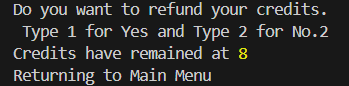
## **2.5 Refunding Credits**

The final feature needing implementation is the refunding of credits. The user will be asked if they would like to refund their credits, there will be two simple inputs one for yes and two for no. if the user inputs a wrong value, then the function will restart. If the user wants to refund their credits the credit value will be reset back down to zero and the user will be returned to the main menu. If the user does not want to refund their credits, then the credit value will stay the same and the user will be returned to the main menu.

A screen shot of a computer program

Description automatically generated

Below is an example of the code in use.



# **3.0 Testing**

For this vending machine application testing must be conducted to ensure that the application can complete everything up to the users' needs and the applications requirements. In accordance with Trio (2022) there are several ways to test a JavaScript application to ensure that everything works and runs as intended such as Unit Testing, Integration Testing and Functional Testing. For this vending machine application multiple test methods can be implemented such as Manual Testing and Functionality Testing. The manual testing will have three sub menus testing the overall usability of the application while the functionality testing will test if the application does what it is intended to do.

# **3.1 Functionality Testing**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| # | Test | Expected | Actual | Outcome | Evidence of Change |
| 1 | Testing if User Input can be accepted into the Terminal. | User Input should be accepted to allow Users to Interact with the Terminal Application. | The Terminal accepts User Input through using reader.question to gather inputs. | No changes needed. | No change |
| 2 | Testing if Menu Options Work | The Vending Machine application has three menu options. The options are to add credits, buy products and exit. | The first option runs when the input variable is one, the second option runs when the input is two and when the input is three the third option runs. | No changes needed. | No change |
| 3 | Credits Purchase Function | The first Option is the Credits Purchased. Users will be able to select the option to add credits. Users should be able to add positive credits only and only input numbers. | When one is input into the terminal the User is prompted to the Credits Insertion section. If a User inserts a negative value or a non-number value, then the process restarts. | No changes needed. | No change |
| 4 | Product Selection Function | The second option is the Product Selection option. Users will have to have a credit value inserted to access the product selection menu. Once there is a Credit Value, then users will be able to enter the Product Selection menu. | When two is input into the terminal the user is prompted to add credits if they do not have any and if they do have credits, they are taken to the Product Selection Menu. | No Changes Needed. | No change |
| 5 | Exit Feature | The third option is the Exit Option. This will shut down the terminal application | The application is shutdown. | No changes Needed. | No change |
| 6 | Product Selection – Individual Products. | Each product selected in the Product Selection Menu will take the user to the appropriate buying menu. And there will be an option to return to the main menu. | All the five menu choices and the back to main menu option work fully. Also, if an improper value is input the User will be prompted to try again. | No changes needed. | No change |
| 7 | Product Purchase Amount. | When a user selects a product, they will be prompted to select an amount of that product that they want. This amount should firstly be a number and secondly positive. | The user is prompted to input the amount of product they want. The amount is not allowed to be anything but a number but can be negative. | Since the test was failed that code segment was checked. The code was tweaked from amount ==0 to amount <=0. Once this was changed the code worked as intended. | Before:  The code segment taken is related to the error management for product purchasing. This error management was designed to prevent any negative inputs being allowed for the number of products a user purchases. The code below allows negative quantities for purchases.    After:  By tweaking the code to <=0 instead of == 0 changes the meaning to less than or equal to 0. This prevents the user from inputting a negative value for a certain product they want. |
| 8 | Purchase Completion | When a user is purchasing a product, they will have two options. They will have the option to rebuy another product or to complete purchase. | When a user is completing a purchase, the function will check if they have enough credits to complete the purchase. Once this check is complete, the code will then allow the user between purchasing another product or completing purchase. The another purchase code will take the user back to the product selection code, to select another product storing the purchase value as a sub total. If the user goes back to the main menu that subtotal is wiped. If the complete purchase option is chosen the code will complete the purchase show the user an updated credit total and return them to the Main Menu. | No changes needed. | No change |
| 9 | Sufficient Credits for Purchase. | The user must have sufficient credits for purchasing items. If the user does not have sufficient credits, they will be prompted to add credits and then to try and make the purchase again. | If the User enters a purchase amount that is more than the credit value, they have the code breaks and does not redirect the user. This code needs to be implemented. | No change Needed | No change |
| 10 | Shopping Bag | Items when purchased will be added to a shopping bag, this shopping bag will be linked to the main menu so that the user can see what products they have bought. | Items are added to the shopping bag when bought, however if the user makes more than one of the same purchase the product will be duplicated into the shopping bag instead of merging to the original purchase. | Merging feature needs implementation. | Before:    The current items in the shopping bag do not merge even if they have the same item name. This should output four crisps instead of two separate transactions.  After:  This is sorted using the arraytohashtable function mentioned in the challenges section.  A screen shot of a computer code  Description automatically generated  Once this design had been successfully implemented the main menu screen that shows the user what they have purchased looked messy and cluttered. To change this a new option was added to the main menu called view purchases.  Before:  Image 1.0 In 6.0 Appendix Section  After:  Image 2.0 in 6.0 Appendix Section |
| 11 | Credit Refund | The user will be prompt after a completed purchase if they want to refund their credits or if they do not. | Users have the option to refund their credits. | No action needed. | No change. |

# **3.2 Manual Testing**

Another form of testing is manual testing. This is where a user brute forces every possibility to test the code and make sure the code functions accurately. For this code all input variables will be tested to ensure that they only accept the relevant inputs. The credit value will be tested to ensure that the credit value will always remain positive. Every menu option will be tested to ensure that it works.

# **3.21 Testing Input Values**

For this test every opportunity in the code for a user to input a choice into the terminal will be tested to ensure that firstly the input variables can only be the related data type, e.g. a string or number, and secondly that the input values match and perform the necessary tasks. After testing all input variables, they all conform to the required input value. If there is an accidental letter typed the code error catches and gives the user another chance to re-enter the specified value.

A screen shot of a computer

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

# **3.22 Testing Credit Value**

For this test, the credit value will be tested to ensure that it always remains positive. Firstly, this is tested on the actual insertion of the credits. After this testing, the get Credits function the minimum value the credits can take is zero. The second test will be during the purchase of items. The credit value after a purchase must always be greater than or equal to zero. This will be tested in two ways. Firstly, if a user goes to buy a product that is worth more than their credit value. After testing this, the user is not allowed to complete the purchase. Instead, the user is taken to the credits insertion menu to add more credits. The second test is if a purchase of multiple items is worth more than the credit value then the purchase should not be completed. This test was successful. If the credit value is less than the purchase value of more than one item, then the purchase will be wiped from the system and the user will be prompt to add more credits.

Below is the outcome of the test.

As can be seen in the first screenshot the addcredits function is robust. The function will not allow letters to be input since the credits used in this application are numerical (eg 0.80). Also, the function will not allow for any negative numbers to be accepted as credits. In both cases the user is shown an error message that tells them what went wrong and what they must input to continue through the process.

A screenshot of a computer screen

Description automatically generated

A screenshot of a computer program

Description automatically generated

In the second screenshot, if there are any chances that the credit value will become negative, then the application itself will run an error and ask the user to add more credits. During the building of the application the fact that the credit value always had to be above zero was one core aspect considered for all functions. This prevents the user from completing a purchase that they do not have the specific credit value needed to do so. Instead, this takes the user back to the add credit menu to insert more credits, as well as telling the user that their purchase has been wiped from the system.

# **3.23 Testing Menus**

This test focuses on the usability of the menus on the vending machine application. For this test to be a success the user must be able to travel through the menus of the vending machine application as if they were using the application themselves. There must be no errors that prevent the user from enjoying the overall ease of use of the system.

When testing the Main Menu function there are no errors that stop the user from progressing. The three main options for the user can use. These are the add credits option the product selection option and the view purchases option. These three options are functions in the application whilst the last option the exit option shuts down the application. In testing all four of these work with their intended output.

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

After testing all inputs for the product selection menu, they all work appropriately. Each input needed to purchase a product worked sufficiently as well as the error management for any accidental inputs. If the user entered a value that was not in the appropriate range, the console would throw an error that would explain that there was an invalid input, and the user needs to try again.

However, another issue came up. When a user wants to buy multiple products there was no way for them to see what they already bought. This could mean that users can accidentally buy multiple of the same product by mistake. This was rectified by adding to the console.log, used for the PRODUCT SELECTION [Current Credits] = line, [“Current Items] =,” items. This allows the user at all stages in the purchasing process to see what they have bought so far.

Before:



As seen in the above testing images there is no clear way of seeing what items are in the shopping bag.

After:



Below is an example of the products being shown in the shopping bag. This does not display items that have featured in a completed purchase, which is shown in the main menu.



**Cancel Purchase Function**

In the code during testing there was no way to cancel a purchase without returning to the main menu midway through the purchase, this function could have its own separate menu to make sure the user knows what it can be used for. This also allows users to return to the main menu midway through purchases to add more credits if needed.

Before:

A screen shot of a computer program

Description automatically generated

A screen shot of a computer

Description automatically generated

After:

Product Selection page code

A computer screen with text

Description automatically generated

A screen shot of a computer program

Description automatically generated

Output:

A screen shot of a computer

Description automatically generated

A computer screen with white text

Description automatically generated

This allows users to now checkout after selecting an item instead of needing to buy another item to checkout. Also, this allows users to go back to the main menu to add credits for there purchase without there purchase being deleted. They also, can cancel their purchase if they have selected the wrong item.

**Checkout Function**

The checkout function affects the complete purchase function because code from that function being used in the checkout function. Therefore, the complete purchase function is trimmed down to allow this.

Before the changes, all code in the checkout function would have been used in the complete purchase function. There are some minor tweaks that allow a user to go back and complete a purchase once they add more credits instead of that purchase being wiped from the system. In section 2.4 Completing Purchases in this report there are screenshots showing the old complete\_purchase function.

After: A screen shot of a computer code

Description automatically generated

Now this code is used in the checkout function. This function allows users to checkout whenever they want to. This function will also run after any product is added to the shopping bag.

A screen shot of a computer program

Description automatically generated

Once these features were tweaked or added the same test was conducted again with the same results. The new features have added more usability to the application while not affecting anything that has already been developed.

# **4.0 Reflection on Practices**

# **4.1 Accomplishments**

Through out this project there have been several accomplishments achieved. For this project, some of the accomplishments gained where gained through the completion of the code itself. These accomplishments were related to my knowledge and understanding of the JavaScript language. A note worthy accomplishment would be the streamlining of code using functions.

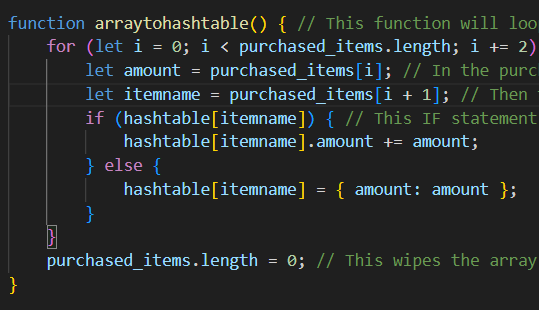
In my draft build of the vending machine that I used to design a simple vending machine application that could act as a basis for the main code, the purchasing of items was not streamlined at all. There was one product selection function with six sub options that resulted in an extremely long and unreadable code block. This code was streamlined by using a function and using two parameters that would be defined through another function. This meant that the code itself was less long and much more readable allowing users that read the source code, to have a more in-depth understanding of the code itself.

Another accomplishment would be the flexibility of my application. My application allows the user at multiple stages to cancel the purchase like in an actual vending machine. When researching for this application a lot of resources online would build the vending machine in an overall linear way without any flexibility. They would not allow users to purchase many of the same items in the same purchase instead having the purchases be stacked up one by one. Despite this being a coursework requirement, I felt it necessary to make the purchase of items allow many items to be purchased in the same transaction, due to the overall usability of my product.

# **4.2 Challenges**

The main challenge faced when building this application was the sorting and merging of the purchased items. This means that when a user buys the same product, in two separate transactions the result is merged. An example of this would be if the user buys two crisps in one transaction and then decides to buy another packet of crisps in a second transaction, then the outputted result, would be three crisps instead of the two separate values, two crisps and one crisp.

To overcome this challenge a function converting the array to a hashtable was used. This function would push the items in the array in pairs of two into the hashtable. The function then checks if the itemname of each pair shows up more than once. If this is the case the item amounts are merged and stored as one value. If this is not the case the items are simply pushed into the hashtable. At the end, the original array is wiped to ensure that no values are kept in the array so if another purchase is made then there will be no duplicate values.



Example output

A screen shot of a computer code

Description automatically generated

As mentioned in the Functionality Testing Section this challenge being overcome caused a new challenge to occur. As can be seen above, the desired outcome is there but the overall design looks messy and cluttered. To change this a new function called viewpurchases was designed. This function would be used by users to display their purchases away from any menus so even if they are displayed in a cluttered manner, there is no overall effect to usability.

View Purchases code:

A black screen with text

Description automatically generated

Output:

A screenshot of a computer program

Description automatically generated

# **4.3 Short and Long-Term Programming Goals**

After the completion of this project a short-term goal would be to refine the code, adding some features that were not necessary in the coursework requirements such as a stock counter for items in the vending machine, and more of an item choice available for the machine.

By doing this the Vending Machine application will become more of a unique piece of work than just sticking to the same coursework requirements needed for my module. Therefore, I can then use the vending machine as a piece of work in an overall portfolio of pieces that can be used to help boost any future career prospects as well as showing potential employees my abilities to build applications.

In the long term I would like to use the abilities and understanding of JavaScript to build more projects to help build the portfolio up, adding more unique projects that closer to my overall interests. These may not necessarily be just JavaScript, I would like to build my use of JavaScript pairing with HTML to build web applications that can help display my overall understanding of the JavaScript language, as well as showing off my interests in a creative and useful way.

Another long-term goal would be to rebuild my vending machine application in the PHP language to help further develop my understanding of that language and broaden my skill set. By doing this I can develop further programming skills that would be needed in a future career as well as using it as an interesting challenge to see if I can do it.

# **4.4 Conclusion**

To conclude this project a vending machine application was built using the JavaScript programming language. The application allows users to add credits, select products, make purchases, and refund credits. These are the features required in the coursework document. Having all three of these features successfully implemented allows users to successfully use a vending machine application that allows users basic functionality. To build on the coursework as stated before new features could be implemented including a stock counter. To do this the application would have to be changed so it is less reliant on functions to do everything and instead objects can be used to successfully add in these features.

# **5.0 References**

**Report References**

Vishal Siddhpara, 2024, Top JavaScript Statistics You Must Know to Kickstart Your Business in 2024 [Online] Available from: <https://radixweb.com/blog/top-javascript-usage-statistics> [Accessed 12th November 2024]

Trio ,(2022), Top 8 JavaScript Testing Frameworks: Everything You Need to Know [Online] Available from: <https://trio.dev/top-8-javascript-testing-frameworks/> [Accessed 27h November 2024]

**Code References**

Code Academy,(2024),Getting User Input in Node.js [Online] Available from: <https://www.codecademy.com/article/getting-user-input-in-node-js> [Accessed 17th November 2024]

BetterStack,(2024), How to exit in Node.js [Online] Available from: <https://betterstack.com/community/questions/how-to-exit-in-node-js/> [Accessed 19th November 2024]

Mdn web docs, (2024) isNaN() [Online] Available from: <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/isNaN> [Accessed 20th November 2024]

Sentry, (2022), How do I empty an Array in Javascript Available from: <https://sentry.io/answers/how-do-i-empty-an-array-in-javascript/> [Accessed 22nd November 2024]

W3 Schools ,(2024), JavaScript Function Parameters [Online] Available from: <https://www.w3schools.com/js/js_function_parameters.asp> [Accessed 22nd November 2024]

Stack Overflow, (2024), What does ${} (dollar sign and curly brackets) mean in JavaScript? Available from: <https://stackoverflow.com/questions/35835362/what-does-dollar-sign-and-curly-braces-mean-in-a-string-in-javascript> [Accessed 23rd November 2024]

W3 Schools, (2024), JavaScript Array Push() Available from: <https://www.w3schools.com/jsref/jsref_push.asp> [Accessed 23rd November 2024]

FreeCodeCamp,(2023), How to Loop Through Arrays in JavaScript [Online] Available from: <https://www.freecodecamp.org/news/loop-through-arrays-javascript/> [Accessed 10th January 2025]

Any references related to the application development are also referenced in Harvard style in comments in the source code.

# **6.0 Appendix**

Image 1.0 Main Menu before changes.

A screenshot of a computer program

Description automatically generated

Image 2.0 Main Menu after changes. Purchased Products option added.

A black screen with white text

Description automatically generated

Images 3.0 and 4.0 Flowchart split into two to be readable. 3.0 first half and 4.0 second half

A black screen with white text

Description automatically generated

A black screen with white text

Description automatically generated