**Sweets Shop**

*Second Year Project*

Developed By Group 6

Lamar Fadil – X00195342

Daryna Tertychna – X0020483

Sofia Kabanets – X00217101

Supervised By

James Doody

**Table of Contents**

1. Declaration of Academic Integrity pg 3-4
2. Member Responsibility and Participation pg 5
3. Project Introduction pg 6
4. User Requirements pg 6
5. Iteration 1 pg 14
6. Iteration 2 pg 16
7. Iteration 3 pg 19
8. References pg 26

A close-up of a logo

Description automatically generatedStudent Compliance with Academic Integrity

All students are expected to complete their courses in compliance with University regulations and standards. No student shall engage in any activity that involves attempting to receive a grade by means other than honest effort, for example:

1. No student shall complete, in part or in total, any examination or assignment for another person.
2. No student shall knowingly allow any examination or assignment to be completed, in part or in total, for himself or herself by another person.
3. No student shall plagiarize or copy the work of another and submit it as his or her own work.
4. No student shall employ aids excluded by the instructor in undertaking course work.
5. No student shall knowingly procure, provide, or accept any materials that contain questions or answers to any examination or assignment to be given at a subsequent time.
6. No student shall procure or accept assignments from any other student from current or prior classes of this course.
7. No student shall provide their assignments, in part or in total, to any other student in current or future classes of this course.
8. No student shall submit substantially the same material in more than one course without prior authorization.
9. No student shall alter graded assignments or examinations and then resubmit them for regrading.
10. All programming code and documentation submitted for evaluation or existing inside the students computer accounts must be the students original work or material specifically authorized by the instructor.
11. Collaborating with other students to develop, complete or correct course work is limited to activities explicitly authorized by the instructor.
12. For all group assignments, each member of the group is responsible for the academic integrity of the entire submission.

**N.B to be filled out by each member of the team.**

By including my name in the form below, I declare that I understand and will abide by the University Regulations and Policies covering Academic Integrity. I accept that each member of the group is responsible for the academic integrity of the entire submission. I will retain a copy of this agreement for future reference.

|  |  |  |  |
| --- | --- | --- | --- |
| **Student Name (1):** | **Lamar Fadil** | **Student No.:** | **X00195342** |
| **Student Name (2):** | **Daryna Tertychna** | **Student No.:** | **X00204831** |
| **Student Name (3):** |  | **Student No.:** |  |
| **Module title:** | **Project** | | |
| **Programme Title:** | **Computing with MLAI / Software Development** | | |
| **Date: DD/MM/YYYY** | **22/04/2024** |

**Member Responsibility and Participation**

*Lamar Fadil – X00195342*

* Use Case and Workflow Diagrams for Search Product, Add to Wishlist, View Order History
* Reports for User Requirements, Iteration 1, Iteration 2 and 3
* Shop App models and views
* Wishlist app (Including functionalities within Wishlist)
* Search app (Including functionalities within Search)
* Wishlist app testing and CSS
* Search app testing and CSS

*Daryna Tertychna – X0020483*

* Use Case and Workflow Diagrams for Update Stock, View Transaction, Update Profile
* Creating a shop project, a bit of CSS
* Shop App templates (including blender model for the “thank you” donut)
* Order app
* Reviews app
* Vouchers app
* Stripe and payment
* Tests for my apps

*Sofia Kabanets – X00217101*

*Project Introduction*

For our second-year project, us members set out to make an E-Commerce website for a sweets shop that aimed to have basic functionalities found in a website and additional features that would enhance the average user’s experience. Our team strategy was to have frequent meetings, especially during the earlier stages of the prototype when our tasks were reliant on each other. Further details will be discussed in the following paragraphs.

*User Requirements*

**Project goals:**

For our project, we plan on creating an ecommerce website for a sweets shop. We aim to make this website a place where customers can order products from the comfort of their home.

Our goal is to get the following features on the website; Register user, add to cart, checkout cart, search for product, add to wish list, view order history, update stock, view transaction and update profile. Thanks to the fulfillment of all these goals, we want to create a store that will be convenient to use and exceed the expectations of our customers.

**Project feasibility:**

Currently, the project seems within reach for the given time frame as it follows the same structure as previous projects we’ve worked on with some extra added features. During our planning, we took external factors such as other modules, clubs, personal circumstances, etc. into consideration as to avoid any risks regarding the completion of this project within the time frame. We also made sure that all our desired features are within each of our capabilities. This way we can help each other more efficiently.

**Requirements:**

**A diagram of a diagram

Description automatically generated**

|  |  |
| --- | --- |
| **View Order History Use Case Description** | |
| **Use Case:** | view order history |
| **Actor (s):** | Registered User, Admin |
| **Goal:** | View past orders |
| **Overview:** | The registered user clicks on the “view order history” button and finds all their past orders listed |
| **Pre-Condition:** | User is registered |
| **Post Condition:** | All past orders are listed on the page |
| **Successful Scenario:** | 1-User logs in or signs up  2-User clicks on “view order history” button  3-All past orders are presented on the page  4-The user is able to view details of all past orders |
| **Alternative Scenario (s)** | 4-“The user is able to view details of all past orders”, the user is presented with an error message describing the reason of the error. |

|  |  |
| --- | --- |
| **Search Use Case Description** | |
| **Use Case:** | Search for product |
| **Actor (s):** | User, Registered user |
| **Goal:** | User can search for products |
| **Overview:** | a user uses the search interface to look for a product using the name of the product or a keyword |
| **Pre-Condition:** | Product must be stored in database and available for purchase |
| **Post Condition:** | Product appears on the page after the customer hits the search button |
| **Successful Scenario:** | 1- User clicks on the search box  2- User enters a name or keyword for the product  3- The desired product is shown the screen  4- User can look at the product’s details |
| **Alternative Scenario (s)** | 3 – “The desired product is shown on screen”, if the product doesn’t show on screen, spelling errors could be fixed or other keywords can be used to search for the product. |

|  |  |
| --- | --- |
| **Add to Wishlist Use Case Description** | |
| **Use Case:** | add the to wish list |
| **Actor (s):** | Registered user |
| **Goal:** | The user can add desired products to wish list |
| **Overview:** | If the user finds products they are interested in, They could be added to a wish list, then moved to cart when the user is ready to buy them |
| **Pre-Condition:** | The user must be registered |
| **Post Condition:** | The product stays in wish list for a limited amount of time |
| **Successful Scenario:** | 1-User logs in or signs up for an account  2-User finds a product they are interested in  3-User clicks the “add to wish list button”  4-Product is added to the user’s wish list  5-User can view the product in wish list |
| **Alternative Scenario (s)** | 4-“User can view the product in wish list”, if the product is not visible, then the wish list can be reloaded, if that doesn’t work then the user is given the option to add the item again |

|  |  |
| --- | --- |
| **View Transaction Use Case Description** | |
| **Use Case:** | View Transaction |
| **Actor (s):** | Payment, Admin |
| **Goal:** | Retrieve and review transaction details. |
| **Overview:** | Payment and Admin actors have the ability to view transactions. |
| **Pre-Condition:** | The transaction must be stored in the system. Payment and Admin actors must have the permissions to view transactions. |
| **Post Condition:** | Transaction details are displayed. |
| **Successful Scenario:** | * 1. Payment:   1.Open and log in to the transaction payments page.  2.See the list of transactions. Filter and select the needed.  3.Views transaction details (date, name, amount, and so on).  2. Admin:  1. Open and log in to the transaction payments page (in the admin page).  2. See the list of transactions. Filter and select the needed.  3. Views transaction details (date, name, amount, and so on). |
| **Alternative Scenario (s)** | 1.” See the list of transactions. Filter and select the needed.” - If there are no transactions in the system yet, the actor will see appropriate message.  2. “Filter and select the needed.” – if the wrong date selected (00/00/0000) the list will all transactions appears. |

|  |  |
| --- | --- |
| **Update Profile Use Case Description** | |
| **Use Case:** | Update Profile |
| **Actor (s):** | Admin, Registered User |
| **Goal:** | Allow registered users and admins to update profiles. |
| **Overview:** | Admin and registered users can modify user profiles. |
| **Pre-Condition:** | The user or admin must be in the system and have the permission to update info. |
| **Post Condition:** | Information in the system is successfully updated. |
| **Successful Scenario:** | 1. Registered User:   1.Log in to the store and select “change profile” option.  2.Edit any information like a name, contact details, or password and so on.  3. Save the changes.  2. Admin:  1. Log in to the store and select “change profile” option.  2. Select a user for updates.  3.Change information in the profile.  4.Save the changes. |
| **Alternative Scenario (s)** | 2 – “Edit any information like a name, contact details, or password and so on.”, if user tries to update someone’s else page, the page with error will display.  3 – “Save the changes” if user or admin put invalid information (like numbers for name) the message that data are invalid appears. |

|  |  |
| --- | --- |
| **Register User Use Case Description** | |
| **Use Case:** | Register User |
| **Actor (s):** | Guest User |
| **Goal:** | Create an account for new user. |
| **Overview:** | A person who is a not registered on the website is creating an account. |
| **Pre-Condition:** | The website is accessible and operational.  The user has a device with internet access.  The user has not registered on the website before. |
| **Post Condition:** | The user's account is created and stored in the website's database. The user can now log in using his email and password. |
| **Successful Scenario:** | • The user navigates to the website's registration page.  • They input essential details, including their name, email address, and password.  • The system verifies the c accuracy of the entered information.  • A new user account is created, linking the information to it. |
| **Alternative Scenario (s)** | • If the provided information fails validation, the system notifies the user and prompts them to correct errors.  • If the entered email address already exists in the system, the user is notified, and they are prompted to use a different email address.  • If the entered password does not meet security requirements, the system prompts the user to create a stronger password. |

|  |  |
| --- | --- |
| **Add to Cart Use Case Description** | |
| **Use Case:** | Add to Cart |
| **Actor (s):** | Registered or Guest User |
| **Goal:** | Enable users to conveniently select and add desired items to their shopping cart, facilitating a seamless and intuitive shopping experience. |
| **Overview:** | A person who is a not registered or registered on the website is adding products to cart. |
| **Pre-Condition:** | The website is accessible and operational.  The user has a device with internet access. |
| **Post Condition:** | After successfully executing the 'Add to Cart' functionality, the selected item is added to the user's shopping cart. User can see added items in the cart, the amount and price. |
| **Successful Scenario:** | • The user navigates through the website to explore available products.  • The user selects a product they wish to purchase.  • The user clicks on the "Add to Cart" button associated with the selected product.  • The system updates the user's shopping cart summary, displaying the added product, quantity, and total price  • The user can view the contents of their cart at any time, seeing a detailed list of selected products and the total cost. |
| **Alternative Scenario (s)** | • The user can adjust the quantity of selected products in the cart before proceeding to checkout.  • Users can remove items from the cart or clear the entire cart. |

|  |  |
| --- | --- |
| **Checkout Cart Use Case Description** | |
| **Use Case:** | Checkout Cart |
| **Actor (s):** | Registered User / Guest User |
| **Goal:** | Successful completion of the purchase ensuring accurate item selection, payment processing, and order confirmation. |
| **Overview:** | A person who is registered on the website is checking out cart. |
| **Pre-Condition:** | The website is accessible and operational.  The user has a device with internet access.  The user has items in their shopping cart.  The user is logged in. |
| **Post Condition:** | The used is shown the order confirmation page. The database is updated with new order. |
| **Successful Scenario:** | • The user accesses their shopping cart page.  • They inspect the contents of their cart, ensuring accuracy in quantities and making any necessary changes.  • Proceeding, the user clicks on the "Proceed to Checkout" option.  • The system prompts the user to input shipping information.  • Next, the user inputs payment details - card number, expiration date, and security code.  • The summary of the order, including items, shipping details, and total cost are displayed.  • The user confirms the order.  • The payment is processed.  • An order confirmation is displayed to the user. |
| **Alternative Scenario (s)** | • If the shipping address is incomplete or invalid, the user is prompted to correct it.  • If the payment cannot be processed successfully, the system notifies the user. |

|  |  |
| --- | --- |
| **Update Stock Use Case Description** | |
| **Use Case:** | Update Stock |
| **Actor (s):** | Admin |
| **Goal:** | Update the actual items in the store. |
| **Overview:** | Admin, through the interface for staff in the store, updates or creates new items, changes prices, and performs other relevant modifications. |
| **Pre-Condition:** | Admin must have the right to create, change, and modify stock. |
| **Post Condition:** | Product appears on the store page, or the price, quantity, or any other parameter of the product has been successfully changed. |
| **Successful Scenario:** | 1. Admin adds a new name and picture for a new product.  2. Admin enters price, ID, quantity, and description for the product.  3. The page is saved with all details.  4. Admin can check the final product on the main page as a "customer." |
| **Alternative Scenario (s)** | Admin can check the final product on the main page as a "customer." If the admin cannot see the product, the level of access to the view page for the product must be changed or permission granted to that admin. |

**Priorities:**

Currently, we are putting our main focus into planning the look and feel of the website in order to plan out where all the features will go to provide an efficient interface for the user. Our second priority is getting started on the CRUD functionalities along with our use case, “Register user”.

**Iteration Goals:**

Iteration 1: As mentioned above, we will put an effort to get the CRUD functionalities out of the way, along with our desired features such as; Register user, Search for product, Update stock and Update profile. This is also when we will start creating our unit tests.

Iteration 2: During this time period we hope to set up stripe along with our cart features and order history. This will also be the time we start working on the design of our website using CSS.

Final Iteration: By this time we plan to have our website fully functioning. We will use this time to deploy our website, brush up on features that need extra work, and add new simple features if the time allows.

**Class Diagram:**

**A diagram of a program

Description automatically generated**

*Iteration 1 Report*

For iteration one we were asked to produce a prototype for our ecommerce website’s core functionalities. Our group’s collective objective is to fully develop the create, read, update and delete functionalities for our website by the second to last week of this iteration.

To initiate these next three weeks, us members held a short meeting regarding how the work would be split amongst the three of us. It was then decided to organise the workload by different apps and .py files such as shop app and urls.py. We also took the time to roughly plan out how much time each task was going to consume and decided to take the advice of “doubling” the initial estimated amount of time needed per task.

Before starting to write the code for our prototype, we took note of any questions raised during the first few days and had a very brief meeting with our advisor to discuss them and ask about the best way to go about creating the prototype as well as the test cases.

In order to establish the foundation of our website, we applied principles and constructs that were taught to us this previous semester, which served as a valuable guide for us and helped refresh our memory on how to initiate the project.

After the project structure was in place, our main focus shifted to determining the necessary apps and their corresponding models. For the prototype, we came to the conclusion that initially creating the main app, “shop app”, would suffice along with our “templates” folder. Our reasoning behind this was that the primary models (category and products) were all we needed to fit the requirements for this iteration, and the templates were then of course used as a medium to assess how the CRUD functionalities were working.

Our attention is now turned to writing the code required for the models, views, urls and admin py files. Afterwards, a superuser will be created for the purpose of building up the prototype of our database with some tester data. To develop the CRUD functionalities, each member of the group was responsible for a specific file/folder (similar to how the workload was split up in the first paragraph).

After creating the prototype’s models, admin, and views, we encountered our first mistake. While pushing our different codes to the repository, we failed to recognize that our codes were being pushed without the use of branching. However, the problem only became apparent after changes were made to the projects urls file, and the addition of a new templates folder and a new shop app urls file. While trying to pull the code in order to add a new view to the shop app’s views file, all pull requests were failing due to lack of branching. Luckily, the issue was easily subsided after the creation of a new branch. Some difficulty was found at first with branching, but after some research, it was found that the use of the command “git stash” before entering a pull request seemed to fix this issue. The updated views.py file was then pushed to the repository with ease.

When it came to creating the unit tests for our CRUD functionalities, we felt unconfident at first as we left it for last. There were some misunderstandings about the task at hand, however after a short meeting between the three of us, we were able to design the Test Case Matrixes for our existing (and future) functionalities.

Now that we have come to the end of this iteration, it became clear that despite previous planning during the earlier phase of the iteration, we were not effective enough with organizing our time and tasks. Initially, we hoped to have all of our core functionalities fully coded, however in these three weeks, we were only able to accomplish 3 out of 4 functionalities (update, read and delete), and we didn’t make room in our schedule for possible errors that needed to be fixed collectively as a group.

In future iterations, we will carefully and thoughtfully plan out each task and create end of week goals to stay on track and hopefully finish all iteration requirements in good time.

*Iteration 2 Report*

A screenshot of a computer

Description automatically generated

A diagram of a product

Description automatically generated

Lamar Fadil (X00195342) – Search App, Wishlist App

Daryna Tertychna (X00204831) – Order App, Payments(Stripe)

Sofia Kabanets (X00217101) – Account App, Cart App

**Search App:**

I decided to start off this iteration with the search app. It was planned to create an advanced search using methods such as “Q Objects” to give us the ability of filtering search using price range, category, etc. This search bar is to be located on the navigation bar, where it would be easily accessible by the user, aiding in the creation of a more user friendly interface.

During the implementation stage, I encountered a flaw where after using the search bar on the main page, the user would get redirected to a different page with a different search bar dedicated for filtering items. I assumed that this was an issue with the urls, however, after playing around with the code and testing different methods of implementation, the problem still remained.

It was decided that a simpler search bar was to be implemented temporarily for this iteration. Ultimately, we would like to have the option of a filtered search by the end of this project, and I plan on testing more methods as well as bringing the issue up with our project advisor in attempt to put the desired search bar into effect.

**Wishlist App:**

In order to make the wishlist app, I thought it would be best to implement it in a way that is similar to a cart app. This method worked very well as the wishlist and cart app hold the same function for different purposes. Previous labs where a cart app was applied were very helpful as I used them as a guideline. Of course, there were some differences such as leaving out the subtotal and checkout sections that were unnecessary for a wish list. I also didn’t make the product availability effect whether the user can or can’t add a product to the list, as it’s not typically a characteristic for wishlists.

During implementation, I was reminded that this feature was to be accessed by registered users only. To make this work I imported “login\_required” to my app’s views.py.

**Order App and Stripe:**

The order app is created to show the customers their orders. First as list of orders, but if clicked on specific order the customer will see all information about this order. I created models and views for this. Before that, payment through Stripe was introduced. There were no problems with this, but after authorization (only a registered users can see their orders), the entire list of orders did not appear. It was redone, but the problem was not found. There is no problem with DB, so in the next step we have to ask the supervisor for advice and fix this bug. It will also be necessary to implement an additional form of order notification. For example, after placing an order, the client will see the details of the order on the main screen, and will also receive a letter to the e-mail.

**Accounts App:**

During the development of the "accounts" app for our Django project, I encountered several challenges that required careful consideration and problem-solving. One of the key aspects I focused on was user authentication and profile management. Setting up user registration, login, and logout functionalities was relatively straightforward, but ensuring seamless integration with user profile creation proved to be more complex. I faced issues such as foreign key constraint violations and integrity errors when attempting to create user profiles upon registration. Configuring login and logout functionalities, including specifying redirect URLs and implementing custom views, also posed challenges. Resolving conflicts between migrations, cache files, and Git branches was another hurdle I had to overcome.

**Cart App:**

Setting up the basic functionality of the shopping cart, including adding and removing items, was relatively straightforward. However, I faced challenges when implementing more advanced features, such as updating item quantities and handling out-of-stock items. I had to carefully design the data models and logic to ensure consistency and accuracy in the cart's behavior. I spent a lot of time trying to make session tokens work to create a cart for any user.

*Iteration 3 Report*

Now that we have entered the end of this semester, rounding off the website’s prototype proved to be difficult as we had many CAs and assignments along with fixing multiple errors within our code. This iteration was definitely more challenging in terms of time management. Here is how it went for us.

**Lamar:**

I was in charge of polishing off the search and wishlist apps along with testing the apps’ functionalities.

Personally I had a lot of trouble with managing my time during this iteration as I had many errors that needed to be solved.

**Errors**

To start off, my filters within the search app were unusable as they returned “None” instead of the filtered products. This led to me receiving an error stating “Cannot use None as a query value”. After looking closely at the code, I found that my models weren’t set up properly which was causing me to receive the “None” error. There were was also trouble with filtering by category, but a solution was found with the aid of the “Advanced Search” lab from last semester.

There was an unfortunate delay to the start of my wishlist app. I had hoped to develop further functionalities In the earlier stages of this iteration / during the Spring holiday, however due to an error with login, I was unable to access the wishlist interface for some time which caused a step back. Though I was able to regain access to the app shortly after the start of this iteration, and had the capability of adding most of my desired features (add / reduce quantity, delete product, search through wishlist). I also planned on creating a feature where the user would be able to share their wishlist with friends, however I couldn’t implement it correctly leading to some errors that I couldn’t troubleshoot due to lack of time as I still had testing left over.

**Testing**

Testing was quite challenging as I had left for last.

Making unit and Selenium IDE tests for the search app was simple enough, however the real challenge was in creating unit tests for the wishlist app.

The unit tests I made for wishlist all failed and I’m sure that the cause was due to the test user not being able to gain access because of the set user permissions. I tried to play around with get\_user\_model, Custom user and client, but unfortunately my attempts did not lead to any solution. I decided to mainly rely on Selenium IDE for the wishlist app as I was running out of time and fortunately, all of my tests passed.

Daryna:  
  
 For the last iteration of this project, I was trying to fix order app and add some more features that I thought were quite important. I focused on the order app. My models and views were a bit of mess, so I started over.

**Errors:**

During the work on the order app, I faced an error related with database. My models were set up wrong and I decided to work more with notes from the previous year. After fixing the main functions of order app the problem with admin page was discovered. Unfortunately, I did wrong with the names in admin and models.py and got corrupted database. Checking names in those files and commands for migrations helped in this case.

For the review app, I was trying to achieve the list of reviews from the newest to the latest, but views didn’t work out.

Also, I’ve got some errors related to the env. I used python 11, but later reinstalled it to 12 version. Though in tests in vsCode my python couldn’t find the way to the app (order or voucher). I checked settings, names and installed apps, but nothing worked.  
  
**Testing:**   
 In my vouchers app I have problem with voucher\_id, so I did simplified version of test that passed.

In my order tests I found out that NOT NULL constraint in the product field. It wasn’t my app, and I was limited in time, so I just rewrote test and simplified it as much as possible. “.Order.DoesNotExist: Order matching query does not exist.” - I’ve got this error. So, I checked my data setup, but in the end just deleted that part of the test.   
 In my last application reviews, I got errors related to the URLs. I put my review\_list.html in the templates folder, but the program couldn't reach it.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Case Matrix** | | | | | |
| **Test Case No** | **Test Case Name** | **Input** | **Expected Outcome** | **Actual Outcome** | **Result (Pass/Fail)** |
| 1 | Search Product | 1-Go to 127.0.0.1:8000  2-Click on “Search” field  3-Input “Cake”  4-Click “Search” button | All products with the word “Cake” in the product name or description will appear on screen | Products containing “cake” appeared | Unit test:  Pass  Selenium IDE:  Pass |
| 2 | Search Unavailable Product | 1-Go to 127.0.0.1:8000  2-Click on “Search” field  3-Input “zzz”  4-Click “Search” button | -No product appears on screen.  -Will get a “You have searched for: zzz” text on screen | Received text “You have searched for: zzz” | Unit test:  Pass  Selenium IDE:  Pass |
| 3 | Filter by Min Max | 1-Go to 127.0.0.1:8000/search  2-Click on “Min” field and enter “90”  3-Click on “Max” field and enter “120” | All products with prices ranging from 90-120 will appear on screen | Products with prices ranging from 90-120 will appear | Unit test:  Pass  Selenium IDE:  Pass |
| 4 | Filter by Min Max Category | 1-Go to 127.0.0.1:8000/search  2-Click on “Min” field and enter “90”  3-Click on “Max” field and enter “120”  4-Click on “Category” field and choose “Cake” option | All cakes within the price range 90-120 will appear on screen | Cakes within the price range 90-120 will appear | Unit test:  Pass  Selenium IDE:  Pass |
| 5 | Enter Wishlist User | 1-Go to 127.0.0.1:8000  2-Click on the heart icon between user and cart icon in the top right navbar | Will be asked to log in or create an account | Selenium IDE:  Asked to log in or create account | Selenium IDE:  Pass |
| 6 | Enter Wishlist Registered User | 1-Go to 127.0.0.1:8000  2- Click on the user icon in the top right navbar  3-Enter “testuser” in username field and “test12345678” in password field  4-Click on the heart icon between user and cart icon in the top right navbar | Will be able to view wishlist | Selenium IDE:  Redirected to wishlist | Selenium IDE:  Pass |
| 7 | Add item to Wishlist | 1-Go to 127.0.0.1:8000  2-Click on “View item” button for “Bespoke Cake”  3-Click on “Add to Wishlist” button | Item will be added to wishlist | Selenium IDE:  Item was added to wishlist | Selenium IDE:  Pass  Unit test:  Fail |
| 8 | Add Wishlist Item Qty | 1-Go to 127.0.0.1:8000/wishlist/  2- Press “+” icon for “Bespoke Cake” | Item quantity will increase by one | Selenium IDE:  Quantity increased by 1 | Selenium IDE:  Pass  Unit test:  Fail |
| 9 | Lessen Wishlist Item Qty | 1-Go to 127.0.0.1:8000/wishlist/  2- Press “-” icon for “Bespoke Cake” | Item quantity will decrease by one | Selenium IDE:  Quantity decreased by 1 | Selenium IDE:  Pass  Unit test:  Fail |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 10 | | Delete Wishlist Item | | 1-Go to 127.0.0.1:8000/wishlist/  2- Press bin icon for “Bespoke Cake” | | Item will be removed from wishlist | | Selenium IDE:  “Bespoke cake” removed | | Selenium IDE:  Pass  Unit test:  Fail |
| 11 | | Search Wishlist | | 1-Go to 127.0.0.1:8000/wishlist/  2-Click on “Search Wishlist” field  3-Input “Donut”  4-Click “Search” button | | Any item that has the word “Donut” in the title will appear on the right side of the screen | | Selenium IDE:  Donut appeared on right side of screen | | Selenium IDE:  Pass  Unit test:  Fail |
| 12 | | Search by Min | | 1-Go to 127.0.0.1:8000/search  2-Click on “Min” field and enter “90”  3-Click on “Apply filter” button | | All products with prices minimum price of 90 appear | | Products starting from 90 euro appear | | Unit test:  Pass  Selenium IDE:  Pass |
| 13 | | Search by Max | | 1-Go to 127.0.0.1:8000/search  2-Click on “Max” field and enter “120”  3-Click on “Apply filter” button | | All products with prices <= 120 appear | | Products that are maximum 120 euro appear | | Unit test:  Pass  Selenium IDE:  Pass |
| 14 | Making order | | 1-Go to 127.0.0.1:8000  2 – Click on the user icon in the top nav bar  3 – Enter “x” username and “123” as a password  4 – Click on the product, add to cart  5 – Go to the cart, click pay with a card  6 – Enter [x@gmail.com](mailto:x@gmail.com) email, “d” for others fields and Ireland as a country, click submit  7 – Enter card number – “4242 4242 4242 4242” and end date – “11/55” and cv2 - “111”, click pay  8 – Check the order number on the “thank you page” | | Order was made and page “thank you for an order is displayed” | | Order was made and page “thank you for an order is displayed” | | Pass | |
| 15 | Checking order history | | 1-Go to 127.0.0.1:8000  2 – Click on the user icon in the top nav bar  3 – Enter “x” username and “123” as a password  4 – Click on the user icon in the top navbar  5 – Click on the “Order History”  6 – Order is displayed | | Previous order and the status are displayed | | Previous order and the status are displayed | | Pass | |
| 16 | Cancelling order | | 1-Go to 127.0.0.1:8000  2 – Click on the user icon in the top nav bar  3 – Enter “x” username and “123” as a password  4 – Click on the user icon in the top navbar  5 – Click on the “Order History”  6 – Click on “view order” for a pending order  7 – Click “cancel”  8 – Go back to the order history | | The status of order is displayed as “Cancelled” | | The status of order is displayed as “Cancelled” | | Pass | |
| 17 | Successful payment | | 1-Go to 127.0.0.1:8000  2 – Click on the user icon in the top nav bar  3 – Enter “x” username and “123” as a password  4 – Click on the product, add to cart  5 – Go to the cart, click pay with a card  6 – Enter [x@gmail.com](mailto:x@gmail.com) email, “d” for others fields and Ireland as a country, click submit  7 – Enter card number – “4242 4242 4242 4242” and end date – “11/55” and cv2 - “111”, click pay  8 – Check the order number on the “thank you page” | | Order was made and page “thank you for an order is displayed” | | Order was made and page “thank you for an order is displayed” | | Pass | |
| 18 | Checking reviews | | 1-Go to 127.0.0.1:8000  2 – Click on any product  3 – Check the section “Reviews” | | The text “No reviews” or an actual review are displayed | | The text “No reviews” or an actual review are displayed | | Pass | |
| 19 | Adding review | | 1-Go to 127.0.0.1:8000  2 – Click on the user icon in the top nav bar  3 – Enter “x” username and “123” as a password  4 – Click on the any product  5 – Scroll to the review’s sections  6 – Click “Add review”  7 - Enter the “5”  8 - Set the number as 5  9 - Click “Submit”  10 – Go back to the product page | | The review is displayed | | The review is displayed | | Pass | |
| 20 | Apply voucher (666) | | 1-Go to 127.0.0.1:8000  2 – Click on the user icon in the top nav bar  3 – Enter “x” username and “123” as a password  4 – Click on the product, add to cart  5 – In the section code enter “666”  6 – Click “Apply”  7 – The total is changed (66% off) | | The total amount after an applied voucher is 66% less | | The total amount after an applied voucher is 66% less | | Pass | |
| 21 | Apply voucher (111) | | 1-Go to 127.0.0.1:8000  2 – Click on the user icon in the top nav bar  3 – Enter “x” username and “123” as a password  4 – Click on the product, add to cart  5 – In the section code enter “111”  6 – Click “Apply”  7 – The total is remaining the same | | The total is remaining the same | | The total is remaining the same | | Pass | |

To conclude, this project has given us a taste of what projects would be like in the work field and has taught us of what to do and what not to do in the future. The main highlight of this project was learning how to communicate with others in a project setting and how to translate many ideas into one cohesive structure.

When we find ourselves in another project setting, I believe our main priority would be time management as there were factors that we didn’t make time for. Some of those factors would include merge conflicts, errors with reliant codes, extra functionalities, etc.

This module was a very important learning curve that increased our confidence in our capabilities and helped us find our weaknesses.

**References**

<https://stackoverflow.com/questions/53977734/is-it-possible-to-modify-the-search-bar-length-defaultvalue-doesnt-seem-to-w>

<https://docs.djangoproject.com/en/5.0/topics/email/#:~:text=In%20most%20cases%2C%20you%20can,and%20recipient_list%20parameters%20are%20required.>

<https://gist.github.com/LowerDeez/9a8b30428a96c4b965d059925a7bd659>

<https://www.studytonight.com/bootstrap/how-to-center-buttons-in-bootstrap#:~:text=The%20buttons%20in%20container%20class,buttons%20in%20the%20container%20class.>

<https://www.geeksforgeeks.org/setup-sending-email-in-django-project/>

<https://stackoverflow.com/questions/2619102/djangos-self-client-login-does-not-work-in-unit-tests>

<https://stackoverflow.com/questions/10868640/align-html-input-fields-by>

Notes from Software Development 3 and Software Quality Assurance and Testing