Iteration 1 – Initial Prototype

Group 6: X00195342, X00204831, X00217101

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.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Case Matrix** | | | | | |
| **Test Case No** | **Test Case Name** | **Input** | **Expected Outcome** | **Actual Outcome** | **Result (Pass/Fail)** |
| 1 | Search for Product | “Chocolate” | Displays either requested product or related products along with error message |  |  |
| 2 | Add to wish list | “Chocolate” is added to wish list via wish list button | The product will be displayed in a list along with other previously added products |  |  |
| 3 | View Order History | Click on “Order History” tab on the website | An organised list of the user’s previous orders is displayed alongside order details |  |  |
| 4 | Update Stock | Product id, price, qty | The product displayed in the store with required parameters |  |  |
| 5 | View Transaction | Order id | The detailed transaction information about the order is displayed |  |  |
| 6 | Update Profile | User’s password | The user’s data are changed and saved. Displayed the last data in the user’s page |  |  |
| 7 | Delete Product | Click on the “delete” link underneath description | Prompted with “are you sure you want to delete <product>” |  |  |
| 8 | Edit Product | Click on the “edit” link and make changes in the description/price or availability | New description/price or availability of the product will be saved and displayed in the shop |  |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 9 | Sign Up | User registration data (username, passwords matching) | A new user should be created, and the user should be redirected to the login page. |  |  |

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.