Task 0: Explain what you are doing/ going to accomplish

In version 0.6 I will create an application form to add new stock

Task 1: Sketch interface design

*Draft a rough design for the interface that allows the user to trigger functionality in task 1, while also annotating where the information in task 2 will be displayed. Create another sketch listing the interface widgets used to create the interface.*

N/A

Task 2: Identify any classes required

*Bro class – holding all information for each bro*

Task 3: Identify information to be displayed

*Return page – Users inputs*

*Return success page – Product name*

Task 4: Identify user inputs

*Name, last name*

Task 5: Identify any constants or existing data if required

*Test Data will be created as well as a list of month words with corresponding numbers*

Task 6: Identify indexed data structures

*bros – Holds all Bro objects*

*months – Holds all month words with number keys*

Task 7: Determine what calculations are necessary

None

Task 8: Develop a modular structure for your program

*Describe any functions that the computer program will have, identifying any sub-functions where required.*

The only functions being created are the page functions containing page instructions and routing, and Static File import functions giving file directions.

Task 9: Define the functions identified

*Describe the functions for both the main program and any classes in terms of input and/or output where required. You may choose to do this with flow charts or pseudo-code (not Python code!). Add in additional steps or explanations using sequential, conditional, iterative statements where required. Identify global and/or local variables.*

Import bottle functions

Import datetime functions

Create Bro class:

Create constructor method passing name, description, image link, cost, stock and booked details(set to “” by default):

*Set all self.variables to passed varaibles*

*Self.name = name*

*Self.description = description*

…

Create months dictionary with month names as the key and the month number as the data

*‘Jan”:1,*

*“Feb”:2,*

…

Create bros class holding all bro objects:

Fill with test data

*Bro("Tom","…”,” tom.jpg", 970, True),*

…

Create index page function and routing using (‘/’)

Create product page function and routing using (‘/products.html’)

Return bros array to page to be displayed

Create purchase page function and routing using (‘/purchase.html’) with “name” as a parameter passed by the link

Create variable found\_bro

Loop through every bro in bros array

Check if parameter name is the same as bro.name

Set found\_bro to this bro

Set current\_bro to found\_bro

Return current\_bro to page to be displayed

Create purchase\_success page function and routing using (‘/purchase\_success.html’) with method “post”

Collect all form data and store in apporopriate variables: Fname, LName and date\_

Set variable Curr\_date to current date using dateTime function

Set variable difference to the result of (date\_ - curr\_date)

Set variable total\_cost to the resilt of (difference \* current\_bro.cost)

Set found\_bro stock to False

Set current\_bro booked details to the Fname, Lname, current date, date\_ and total\_cost

Return current\_bro to page to be displayed

Create return\_product page function and routing using (‘/return\_product.html’)

Do nothing

Create return\_success page function and routing using (‘/return\_success.html’) with method post

Get form first name and store in Fname variable

Get form last name and store in Lname variable

Loop though every bro in bros array:

If current bros booked details matches fname and lname

Set found\_bro variable

Change stock

Return bro

Create application page function and routing using (‘/application.html’)

Do nothing

Create application\_success page function and routing using (‘/application\_success’) with method “POST”

Save the form data in appropriate variables

Append a new bro with data from forms to bros array. Add “empty.jpg” as the default image

Return the new bro

Route images from folder “./Images” using route “/img/<filename>”

Route Css files from folder “./Css” using route “/css/<filename>”

Route Script from folder “./Script” using route “/script/<filename>”

Call ‘run’ function passing port 399

Task 10: Address any relevant implications such as usability, functionality, legal/ethical requirements.

The user will be giving their name, so it needs to not be displayed to other users for privacy purposes. Controls must be obvious as page won’t have instructions. Buttons must be clearly labelled.

The input boxes highlight green when inputs are successfully inputed. They will also highlight when the user is editing in them to make sure the user easily understands what they are doing. If incorrect information is inputed or a required input is not filled it will highlight red.

The submit button is highlighted blue so that the user can easily understand the page and what to do.

The default image is a stock siluhett photo that is not copy righted. This is important so that there is no legal issues displaying the stock image.

Task 11: Document test cases for testing the program

*Document any testing that can be used to test your program. If any input is inputted using the keyboard, describe the expected input, plus any exceptional, boundary or invalid cases.*

Create a few different applications and check that each one can be purchased and returned

Use words for cost in application

Task 12: Refine the plan

*Note any modifications here when iterating through the development cycles.*

Surround all of the application success code with a try except, if except make the page display an error and ask them to try again.

Task 13: Document testing

*Show screenshots of your program working with descriptions of each image. These images should test the tests cases listed above.*

|  |  |  |  |
| --- | --- | --- | --- |
| Test case | Expected outcome | Actual outcome | Solution |
| Create multiple users | Users are created | Users are created | None |
| Try purchasing the new user | Purchase successful | Purchase successful | None |
| Try returning the new user | Return successful | Return successful | None |
| Use text for cost | Ask them to input only numbers, no decimal | Crashed | Surround all of the application success code with a try except, if except make the page display an error and ask them to try again. |

Task 14: Evaluation

*How did your version turn out*

As planned, just one small issue to fix. Through the methods above I was able to indetify the issues and change everything that is required. The program is simple and efficient and easy to navigate. Using simple colours I am able to navigate the page simply.