**Diagram

Description automatically generatedLogo

Description automatically generated**

**CS112-Structured Programming**

Second semester 2021-2022

Group: A

S5

TA: Hagar Ali

Assignment #3

Nour El-Din Ahmed Hussein Ismail, 20210430, Group A, [nour.sehs.3@gmail.com](mailto:nour.sehs.3@gmail.com), 01067935799.

Mohannad Hisham Abdelmoniem, 20210413, Group A, [muhannadhisham92@gmail.com](mailto:muhannadhisham92@gmail.com), 01004380128.

**Task 1, 2:**

**Algorithms used to make the filters:**

For black and white: We calculate the average of the pixels from the original image, then check every pixel if it is darker or lighter, if it is darker change it to black, if it is lighter change it to white.

For merge: We iterate over each pixel form the the two images and calculate the average between them, then store it to the new image.

For rotate: If we made a simple table and started changing the cells position according to how we want to rotate the table we will notice that in the 90 and 270 the cell’s row and column swaps, in the 90 rotation also the column shift by the size of the table, and in the 270 rotation the row shift by the size of the table. In the 180 rotation the row and column does not swap by they both shift by the size of the table. So we apply that to the image (and we store the original data in another array so we don’t lose any data while rotating).

For invert function: First by making two for loops which goes through a 2D array , this 2D array consists of a row and a column , those are the pixels in a row or a column. Second, by subtracting 255 from image pixels, it’ll change it to the corresponding value of it, that means if it’s dark it’ll change to light.

For flipping image: initializing a temporary integer and asking the user to enter the flip the he wants. If he chose Vertically, by making a 2D array for rows and columns , the column is divided to two because it switches the first and last column together in one step, so if it’s not divided it’ll return as it is. Saving the image in the temporary value “to avoid losing the value of the first value” then changing the columns with each other then returning back the second variable to the temporary. The same algorithm goes if he chose Horizontally but instead it changes rows not columns.

For darken and lighten: making a 2D array for rows and columns, if the user chose to Lighten the image , 64 pixels is added to each pixel in the image. But if it’ll exceed 255 , that means the whole image will turn to a white photo, in this case the pixel turns to 255 “white”. If he chose to Darken the image , 64 pixel is subtracted from each pixel in the image. If it’ll be less than zero , the pixel changes to 0 “black”.

For the edge detect filter: First we start by changing the image to black and white, then we iterate over every and detect first black pixel in the row, we keep first pixel in black and then iterate over each black pixel after this pixel we can demonstrate that by saying we are creating a line of black pixels, we change all the middle pixels to white and keep first and last pixels in the line as black.

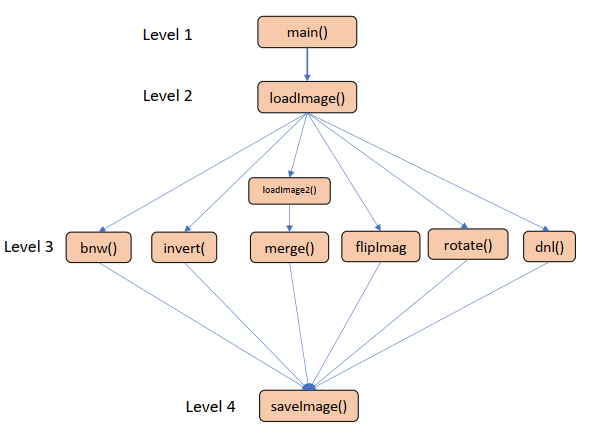
We make the same process over columns, but we have to use the original image as the reference not the modified one, and we can’t change any pixels in the new image to white so we just make first and last pixels in the black line from the original image as black in the new image.

For the shrink filter: In the case of half scale we take only one pixel from every two pixels and put them in the first quad of the image, and the rest of the image we make it white.

In the case of one third and quarter scale we use the same algorithm but with one pixel from every three pixels, and one pixel from every four pixels.

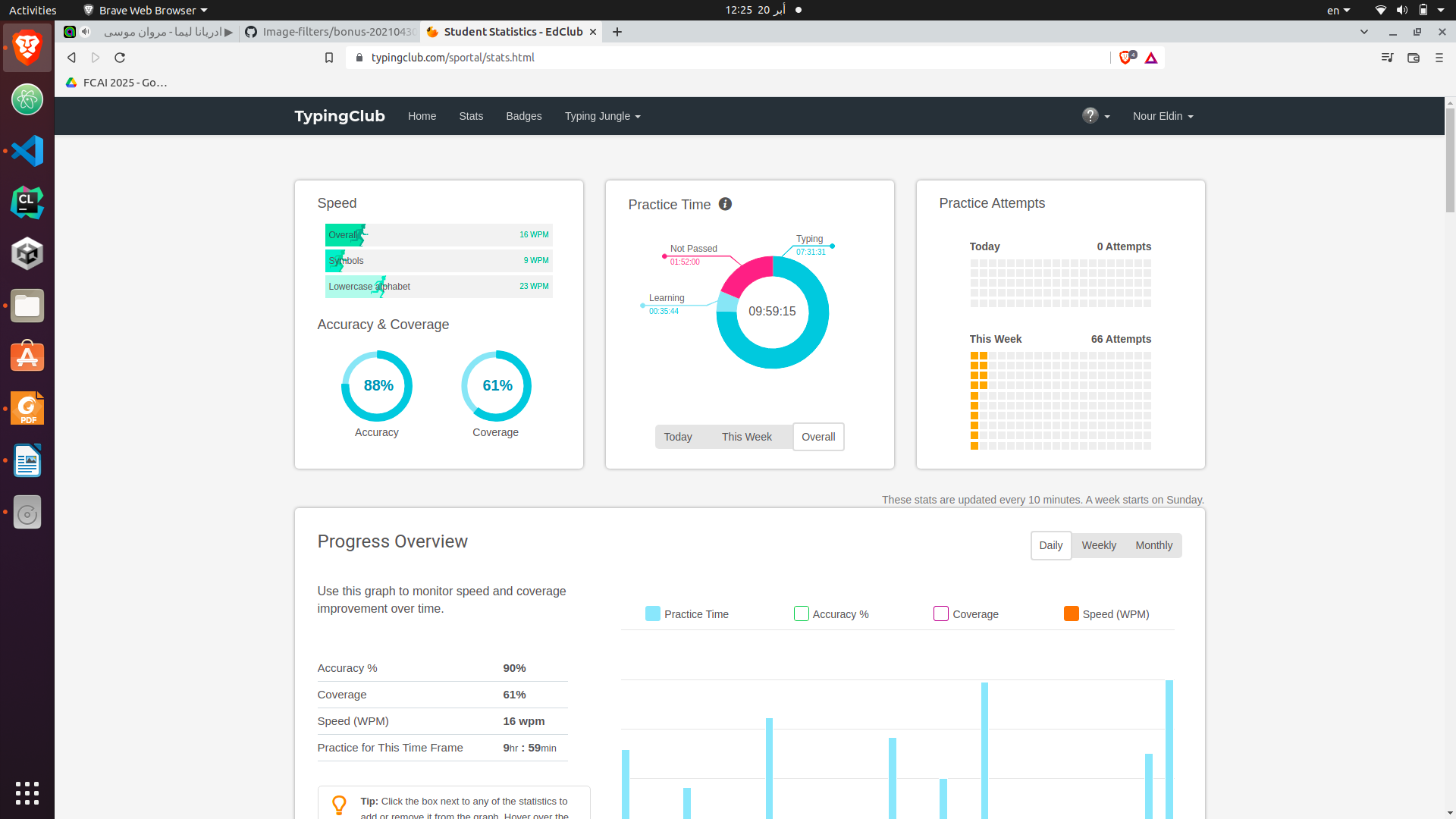
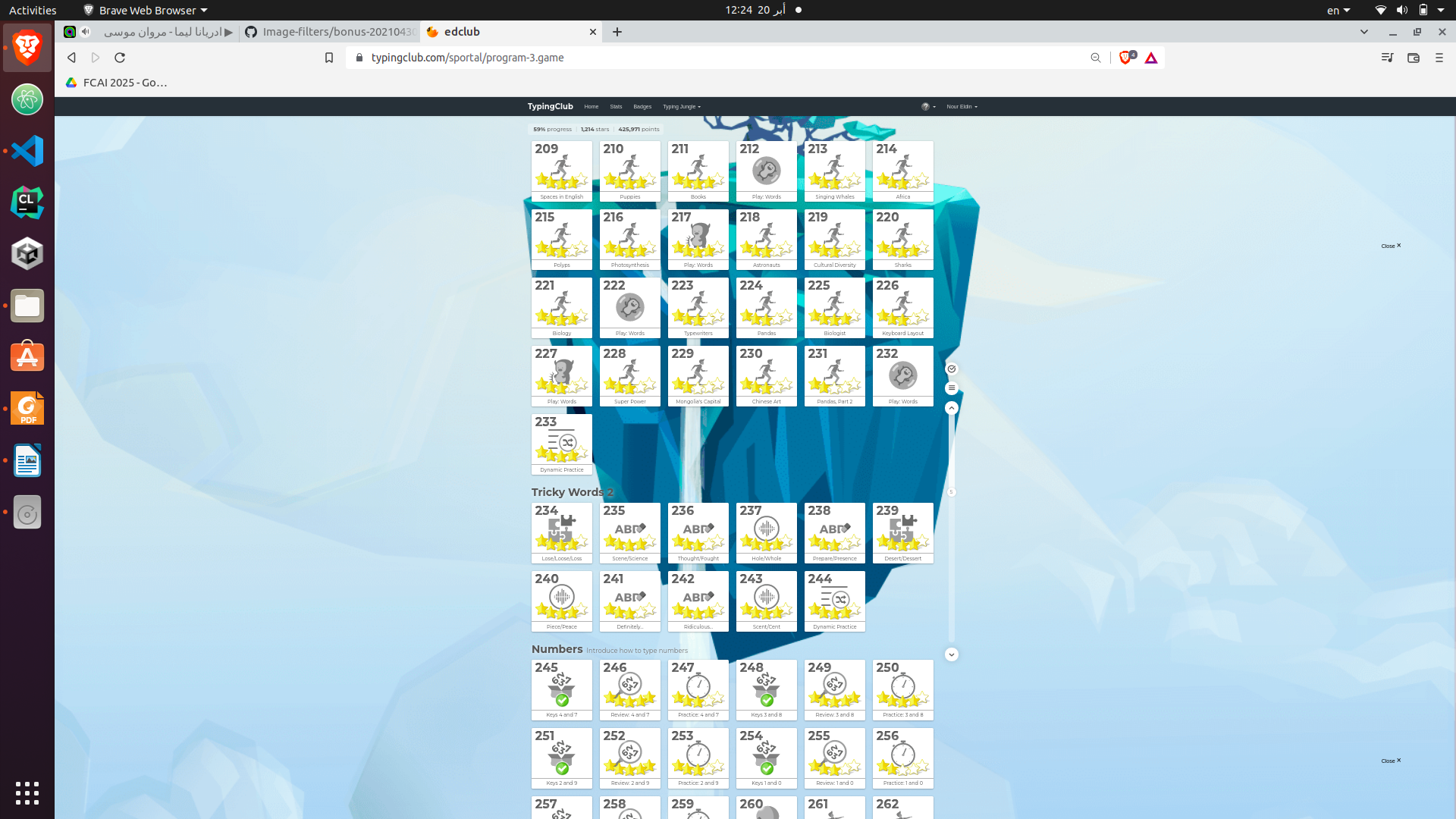
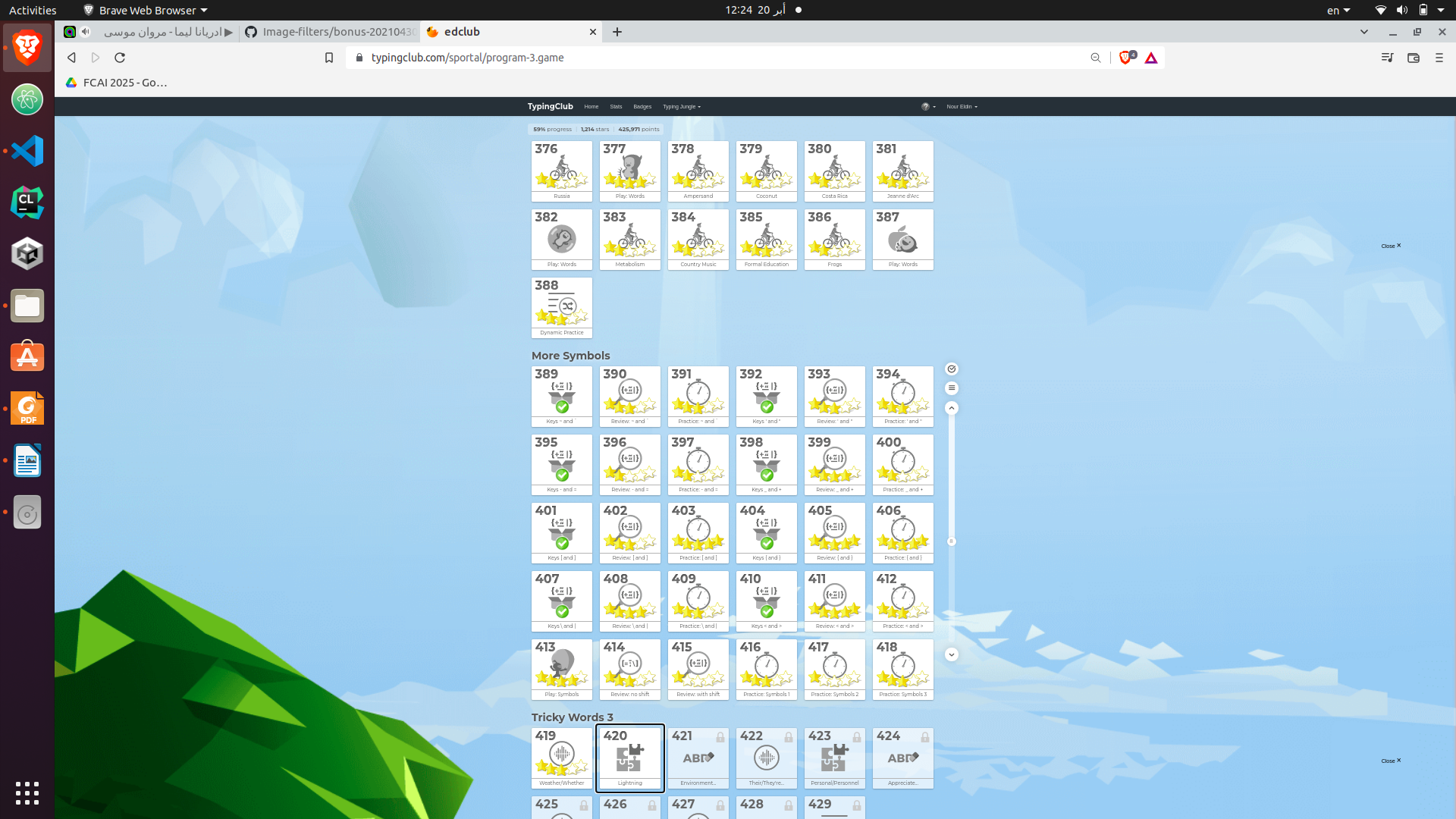
For the blur filter: We start by making a copy from the original image so we use it as a reference while calculating average of pixels, we iterate over each pixel and calculate the average of the pixels in the same vertical and horizontal lines and diagonals to a specified level, store this average in the pixel. (that will cause inefficiency as we increase the level of blur as there will be more pixels not included in the average calculated).

For the shuffle filter: We create an array of four small images to store quads of the image, we iterate over each pixel in the image and store it to its corresponding quad, then we take the order from the user, we store the pixels form the quads in the new order.

**Diagram showing the order of calling the functions:**

**Task 3:**

Nour El-Din:

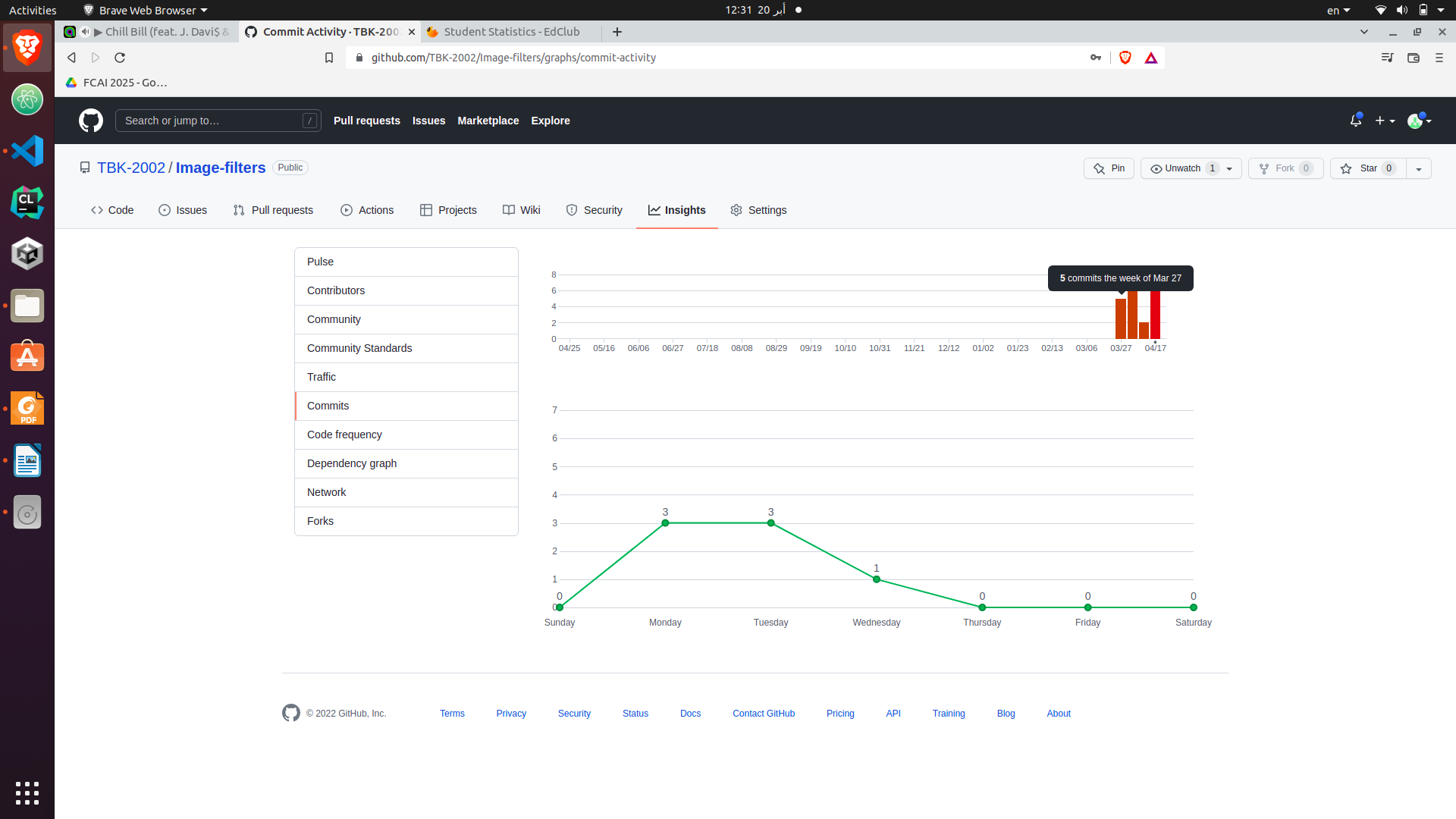


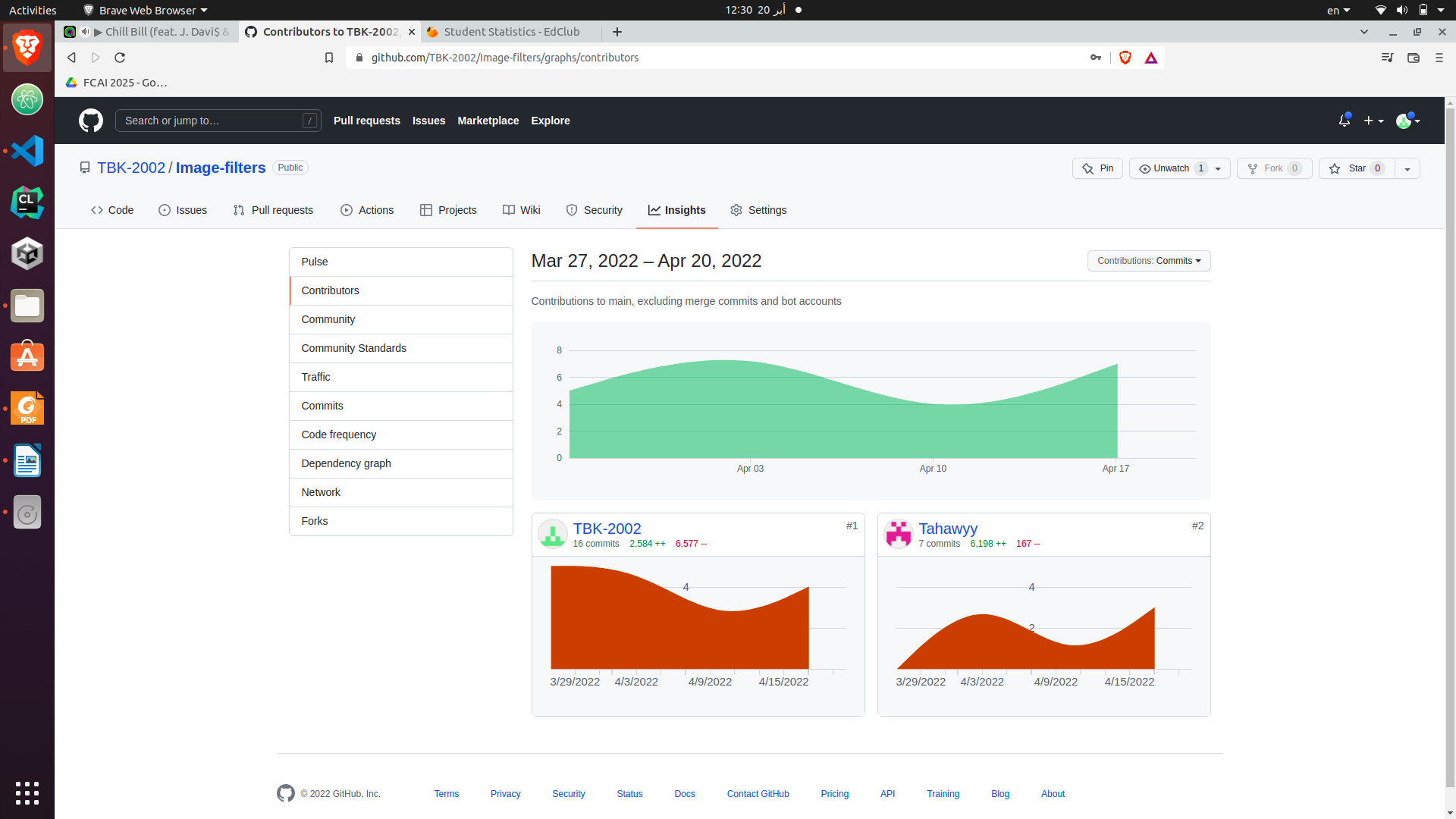
Mohanad Hisham:

**Task 4:**

Nour El-Din used Linux terminal to commit changes to the code.

Mohanad used git-bash terminal to commit changes to the code.





**Bonus Project:**

Done with same algorithms, made the required modifications to make it work on RGB images, and added one more colored image to the folder.