بسم الله الرحمن الرحيم

object oriented programming

DR. Mohammed Elramly

Faisal Ahmed Awad	20210804
Omer Sayedna Ali	20210703
Ahmed Dafalla Mohamed	20210912

Black And White Image Algorithm:

1. Step1: Built a function with type void and name it as(Black _ White).

2. Step2: Do nested loop both of them with size 256 which is equal to the variable(SIZE).

3. Step3: Check:

• if image[i][j] > 127:

• do => image [i][j] =255.

• else:

• do => image[i][j] = 0 .

Invert Image Algorithm:

1. Step1: Built a function with type void and name it as (invert).

2. Step2: Do nested loop both of them with size 256 which is equal to the variable(SIZE).

3. Step3: Check:

• If (image[i][j] == 255):

• do => image[i][j] = 0 ;

• else if (image[i][j] == 0):

• do => image[i][j] =255 ;

• else:

• do => image[i][j]=255 – image[i][j] .

Merge Image Algorithm:

1. Step1: Built a function with type void and name it as (Merge).

2. Step2: Do nested loop both of them with size 256 which is equal to the variable(SIZE).

3. Step3: Do => image = (image1[i][j] + image2[i][j]) /2.

Blur Image Algorithm:

1. Step1: Built a function with type void and name it as (Blur).

2. Step2: Do nested loop both of them with size 256 which is equal to the variable(SIZE).

3. Step3: Do => image[i][j] = the sum of pixels around them / the num of pixels.

Shrink Image Algorithm:

1. Step1: Declare int variable and name it as (skip row).

2. Step2: Declare int variable and name it as (skip column).

3. Step3: Declare int variable and name it as (shrink value).

4. Step4: Declare int variable and name it as (new size).

5. Step5: Ask the user about shrink value.

6. Step6: input shrink value.

7. Step7: (new size) = SIZE / (shrink value).

4. Step8: Do nested loop both of them with size 256 which is equal to (new size).

8. Step9: At the second loop Do:

• Image[i][j] = image[skip row][skip column].

• (Skip column) += (shrink value).

9. Step10: At the first loop DO :

• (skip column) = 0

• Skip row += shrink value.

Light and black Image Algorithm:

1. Start the lightorblack function.
2. Display a message to the user: "1 for Light 2 for Black."
3. Read an integer value from the user and store it in the variable a.
4. Check the value of a to determine the user's choice.
5. If a is equal to 1:
    - Enter a loop that iterates over rows and columns of the 2D image (ImageG) with two nested for loops.
    - For each pixel in the image, check if its value is greater than or equal to 206.
    - If the pixel value is greater than or equal to 206, set the pixel value to 200.
    - Otherwise, increase the pixel value by 50.
    - Repeat this process for all pixels in the image.
6. If a is equal to 2:
    - Enter a loop that iterates over rows and columns of the 2D image (ImageG) with two nested for loops.
    - For each pixel in the image, check if its value is less than or equal to 50.
    - If the pixel value is less than or equal to 50, set the pixel value to 0.
    - Otherwise, decrease the pixel value by 50.
    - Repeat this process for all pixels in the image.
7. If the value of a is neither 1 nor 2, display an error message: "Please Enter 1 or 2."
8. Re-prompt the user to enter a valid choice and store it in the variable a.
9. End the lightorblack function.

Detect Image Edges Algorithm :

1. Start the detect function.
2. Enter a nested loop with two counters, i and j, to iterate over the rows and columns of the image. The outer loop iterates from 0 to 254, and the inner loop iterates from 0 to 255.
3. For each pixel at position (i, j) in the image, calculate the absolute difference between the pixel's value ImageG[i][j] and the value of the pixel immediately to its right ImageG[i][j+1].
4. Check if this absolute difference is greater than 24. If it is:

- Set the value of the current pixel ImageG[i][j] to 0. This effectively turns the pixel black.

5. If the absolute difference is not greater than 24 (i.e., the else condition):

- Set the value of the current pixel ImageG[i][j] to 255. This makes the pixel white.

6. Repeat steps 3 to 5 for all pixels in the image, iterating over both rows and columns.
7. The result of this operation is that the function modifies the image in a way that pixels with an absolute difference in intensity greater than 24 with their immediate right neighbor become black (0), while the rest become white (255).
8. End the detect function.

Enlarge Image Algorithm:

1. Start the enlarge function.
2. Define the number of choices as 4 and initialize the choice variable to store the user's choice. Create an array chosenQuarter to store the enlarged image quarter.
3. Enter a while loop that continues until a valid choice is made by the user.
4. Display a menu to the user, asking which quarter of the image they would like to enlarge and prompt them to enter their choice (1, 2, 3, or 4).
5. Read the user's input as a string into the choices variable.
6. Check if the input is valid using a function called valid, which is assumed to validate whether the input is one of the valid choices (1, 2, 3, or 4). If the input is valid, convert it to an integer and store it in the choice variable.
7. If the input is not valid, display an error message and allow the user to input their choice again.
8. Once a valid choice is made, proceed to the image enlargement process based on the selected quarter.
9. Use two nested loops to iterate through each pixel in the image (with i and j).
10. Depending on the choice made by the user:

- If choice is 1, enlarge the first quarter of the image into the chosenQuarter array.
- If choice is 2, enlarge the second quarter of the image into the chosenQuarter array.
- If choice is 3, enlarge the third quarter of the image into the chosenQuarter array.
- If choice is 4, enlarge the fourth quarter of the image into the chosenQuarter array.

<u>merge Function:</u>

1. Start the merge function.
2. Use two nested loops to iterate through each pixel in the image (with i and j).
3. Calculate the new pixel value for Image by averaging the values of the corresponding pixels in Image and another image called image1. Set the pixel value in Image to be the average of the two images.
4. Repeat this process for all pixels in the image, effectively merging the two images together.
5. End the merge function.

<u>Mirror Image Algorithm:</u>

1. Display a menu to the user, asking which direction they want to mirror the image.

- Option 1: Mirror left
- Option 2: Mirror right
- Option 3: Mirror upper
- Option 4: Mirror down

2. Read the user's choice and store it in the choice variable.
3. Check the value of choice to determine the chosen mirroring direction:

- If choice is 1, mirror the image to the left.
- If choice is 2, mirror the image to the right.
- If choice is 3, mirror the image upwards.
- If choice is 4, mirror the image downwards.

4. Depending on the chosen mirroring direction, use nested loops to iterate through the image pixels with i and j.
5. For each direction:

- If mirroring left:
- Copy the left half of each row in the original image to the left half of each row in image1.
- Copy the right half of each row in the original image to the right half of each row in image1 while flipping horizontally.
- If mirroring right:
- Copy the right half of each row in the original image to the right half of each row in image1.
- Copy the left half of each row in the original image to the left half of each row in image1 while flipping horizontally.
- If mirroring up:
- Copy the upper half of each column in the original image to the upper half of each column in image1.
- Copy the lower half of each column in the original image to the lower half of each column in image1 while flipping vertically.
- If mirroring down:
- Copy the lower half of each column in the original image to the lower half of each column in image1.

- Copy the upper half of each column in the original image to the upper half of each column in image1 while flipping vertically.

6. The result is a mirrored image stored in image1 based on the chosen mirroring direction.
7. The original ImageG is not modified in place, and the mirrored image is stored in image1.
8. The code completes the mirroring operation based on the user's choice.

<u>Skew Horizontally / Vertically Algorithm:</u>

1. Start the skew function.
2. Display a message to the user, asking them to enter 'h' to skew horizontally or 'v' to skew vertically.
3. Read the user's input and store it in the direction variable.
4. Prompt the user to enter the degree of skew.
5. Read the degree value and convert it from degrees to radians using the formula (degree * 22) / (180 * 7). This conversion is done to use the tan function in the following calculations.
6. Initialize two temporary arrays, shrink and skew, to store intermediate results. The shrink array will hold the image after the initial skew, and the skew array will store the final skewed image.
7. Calculate the variable x as (tan(rad) * 256) / (tan(rad) + 1). This value is used to determine the amount of skew.
8. Calculate the variables step and move for further skew calculations.
9. If the user chose to skew horizontally (direction is 'h'):

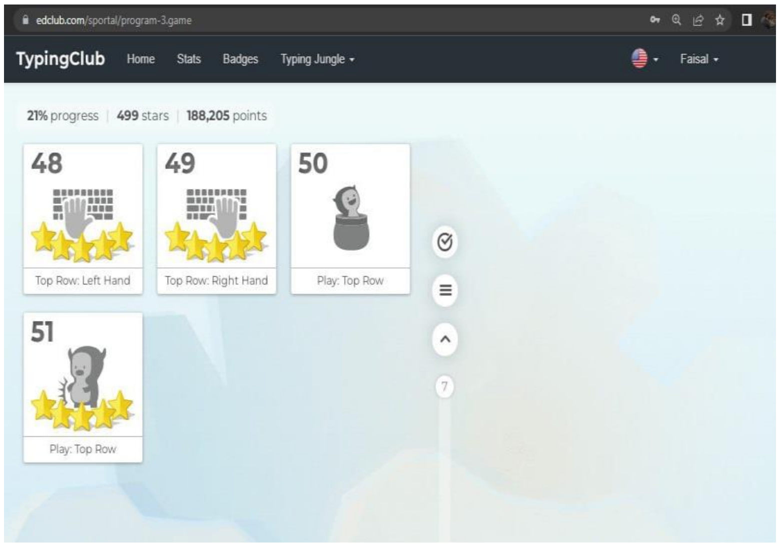
- Iterate through each pixel in the original image and copy it to the shrink array, applying the horizontal skew.
- Copy the resulting shrink array back to the original ImageG array.
- Initialize the skew array to all white (pixel value 255).
- Apply horizontal skew to the shrink array and store the result in the skew array.
- Copy the skew array back to the original ImageG array.

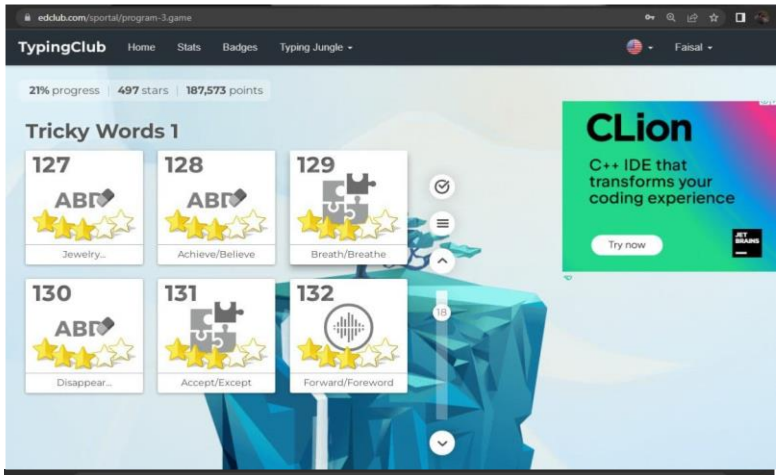
10. If the user chose to skew vertically (direction is 'v'):
    - Iterate through each pixel in the original image and copy it to the shrink array, applying the vertical skew.
    - Copy the resulting shrink array back to the original ImageG array.
    - Initialize the skew array to all white (pixel value 255).
    - Apply vertical skew to the shrink array and store the result in the skew array.
    - Copy the skew array back to the original ImageG array.

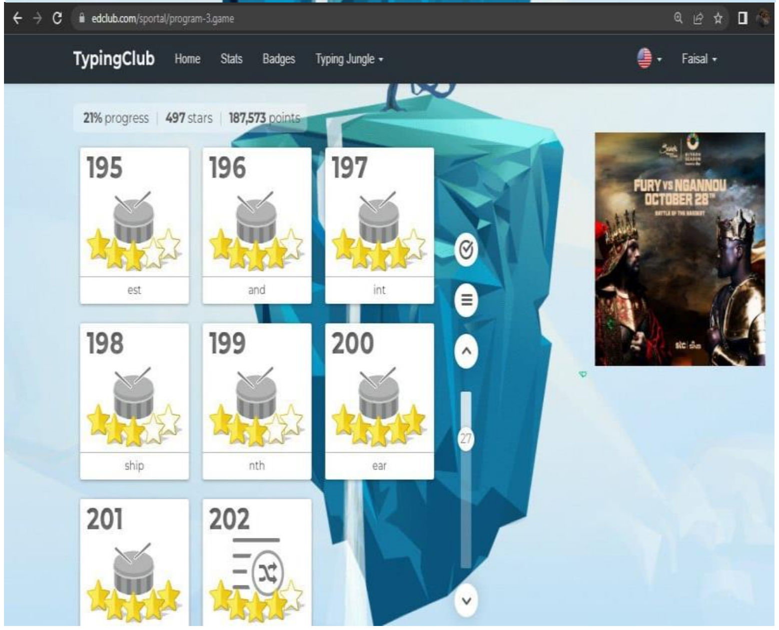
1. The result is a skewed image based on the user's choice and degree of skew.
2. The ImageG array is modified to reflect the skewed image.
3. End the skew function.

Typing club level SC :

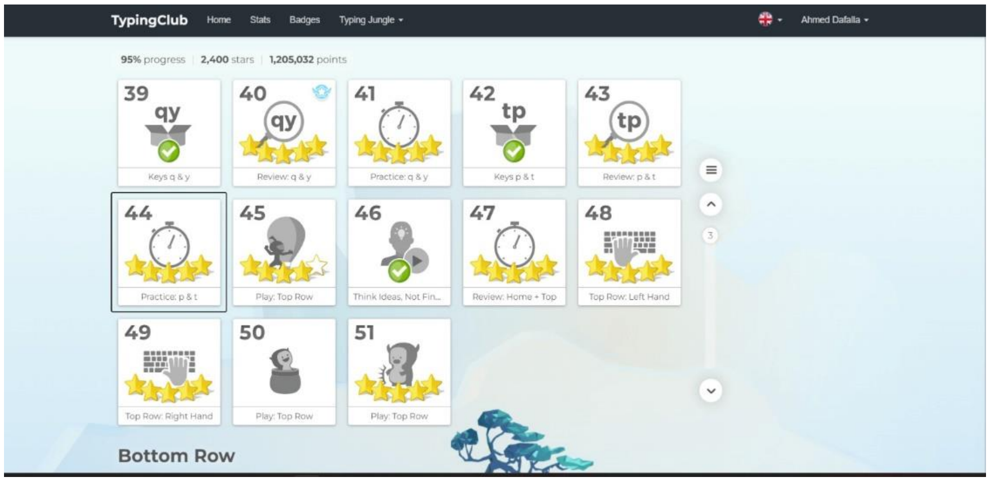
- Faisal Ahmed Awad 20210804

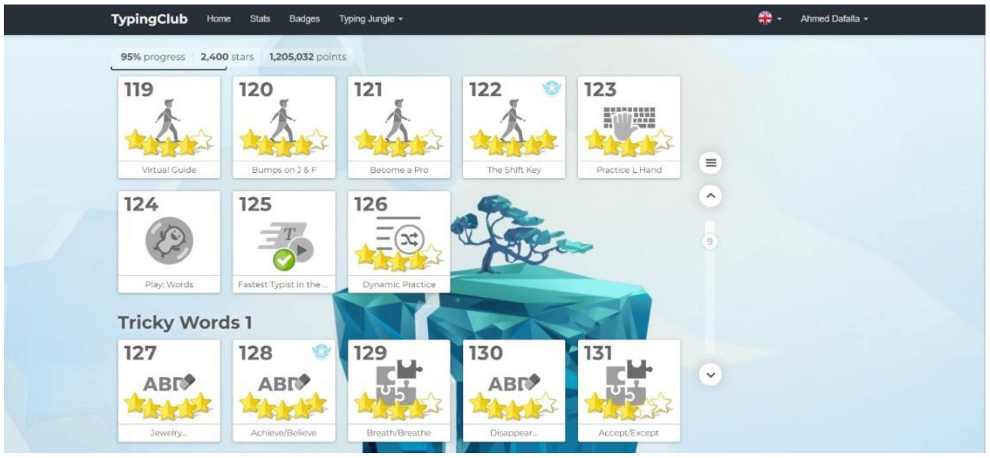


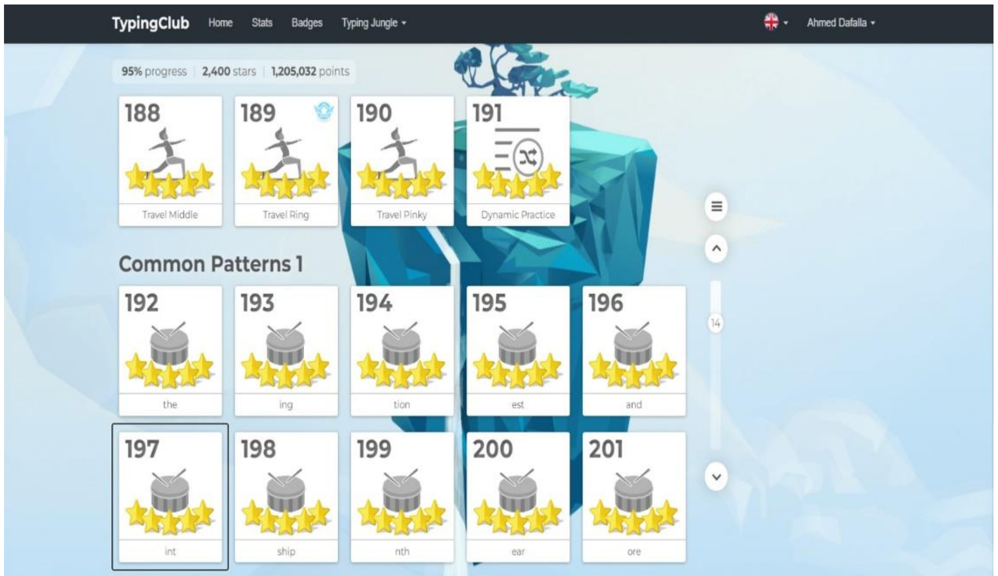




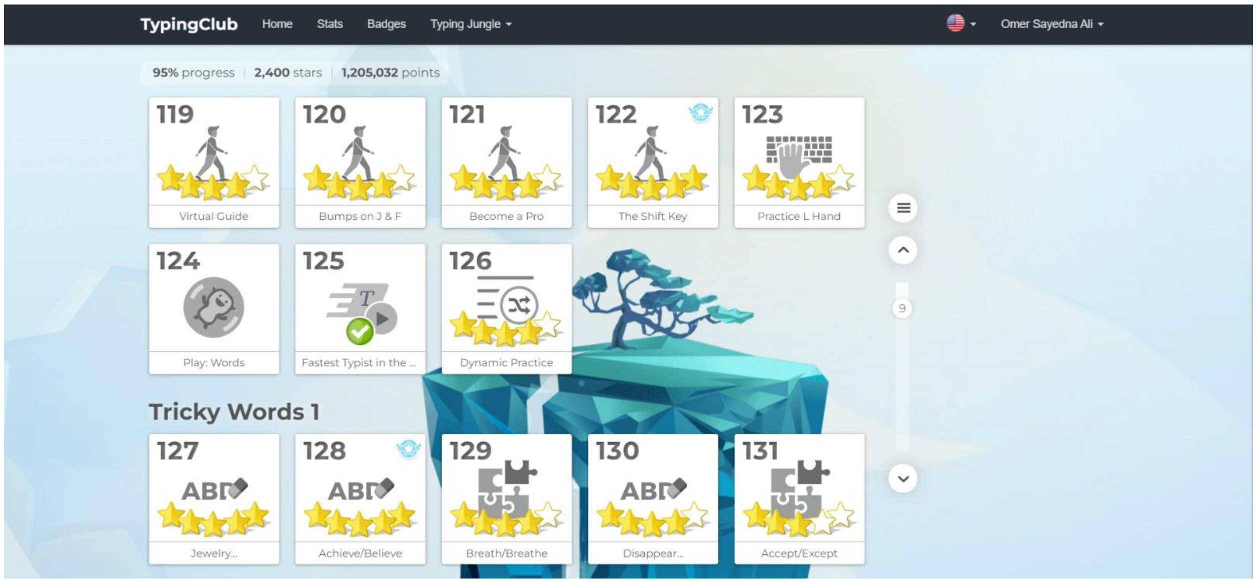
Ahmed Dafalla Mohamed 20210912







Omer Sayedna Ali 20210703



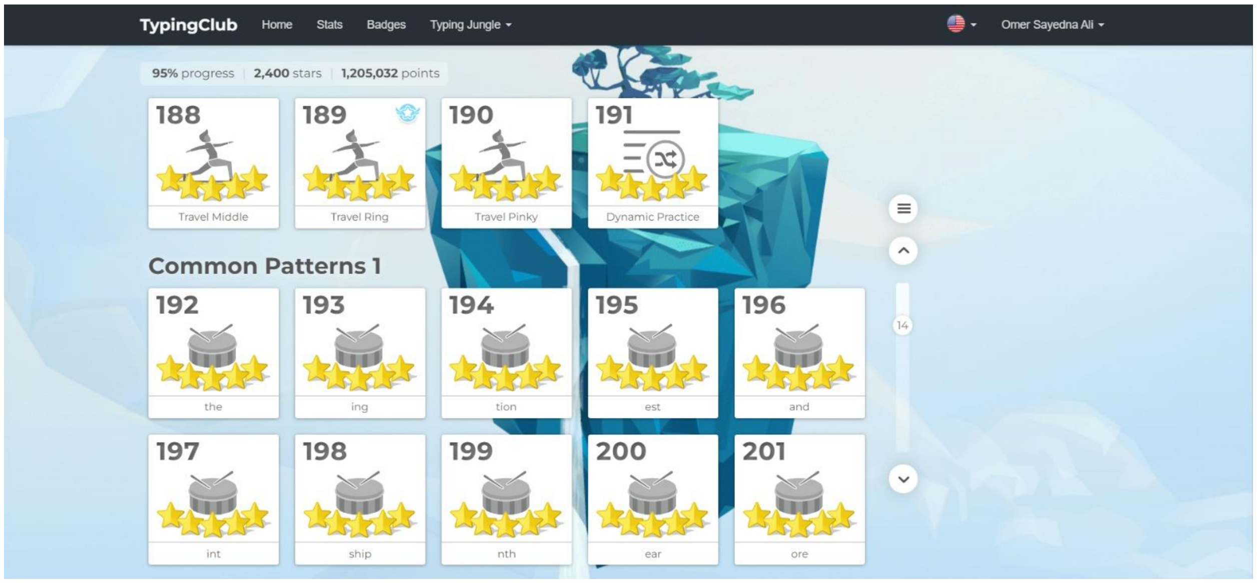
TypingClub Home Stats Badges Typing Jungle Omer Sayedna Ali

95% progress | 2,400 stars | 1,205,032 points

119 Virtual Guide
120 Bumps on J & F
121 Become a Pro
122 The Shift Key
123 Practice L Hand
124 Play: Words
125 Fastest Typist in the ...
126 Dynamic Practice

Tricky Words 1

127 Jewelry...
128 Achieve/Believe
129 Breath/Breathe
130 Disappear...
131 Accept/Except



TypingClub Home Stats Badges Typing Jungle Omer Sayedna Ali

95% progress | 2,400 stars | 1,205,032 points

188 Travel Middle
189 Travel Ring
190 Travel Pinky
191 Dynamic Practice

Common Patterns 1

192 the
193 ing
194 tion
195 est
196 and
197 int
198 ship
199 nth
200 ear
201 ore

Github.com repo

https://github.com/nbow2/Photo-Editing-Application/tree/master

Ahmed Dafalla Mohamed 20210912

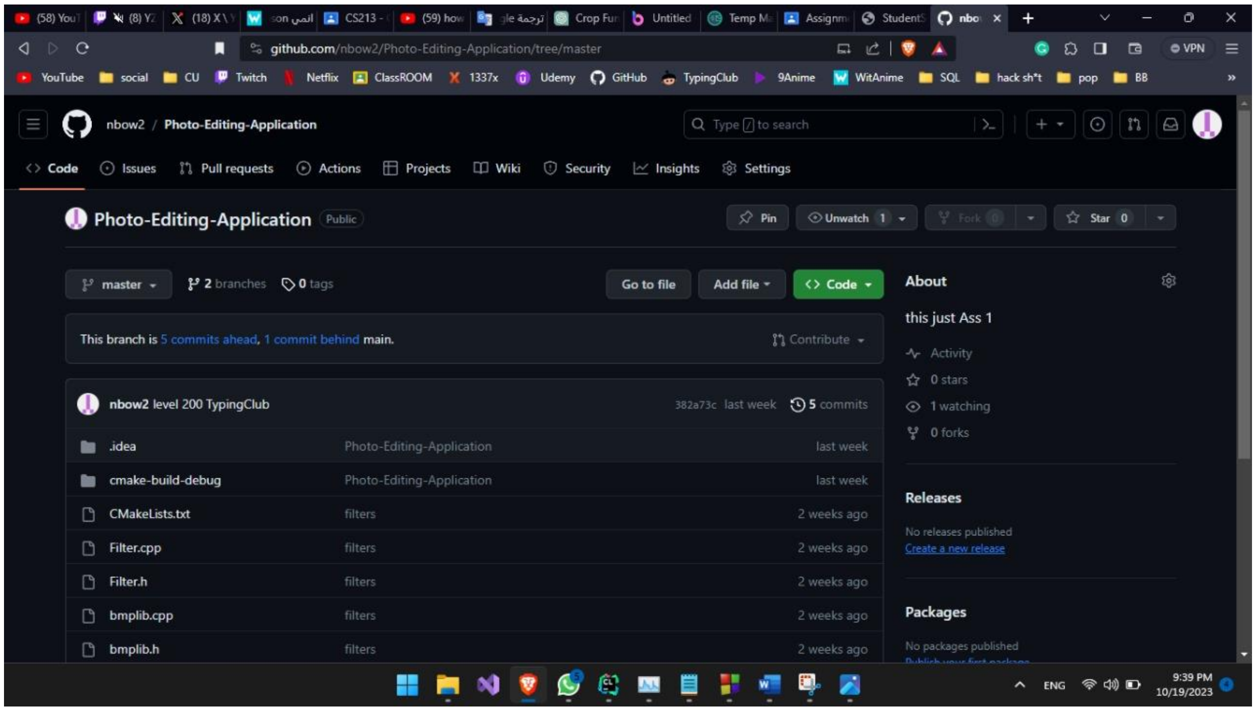


Photo-Editing-Application Public

master 2 branches 0 tags

This branch is 5 commits ahead, 1 commit behind main.

nbow2 level 200 TypingClub 382a73c last week 5 commits

.idea	Photo-Editing-Application	last week
cmake-build-debug	Photo-Editing-Application	last week
CMakeLists.txt	filters	2 weeks ago
Filter.cpp	filters	2 weeks ago
Filter.h	filters	2 weeks ago
bmplib.cpp	filters	2 weeks ago
bmplib.h	filters	2 weeks ago

About

this just Ass 1

Activity
0 stars
1 watching
0 forks

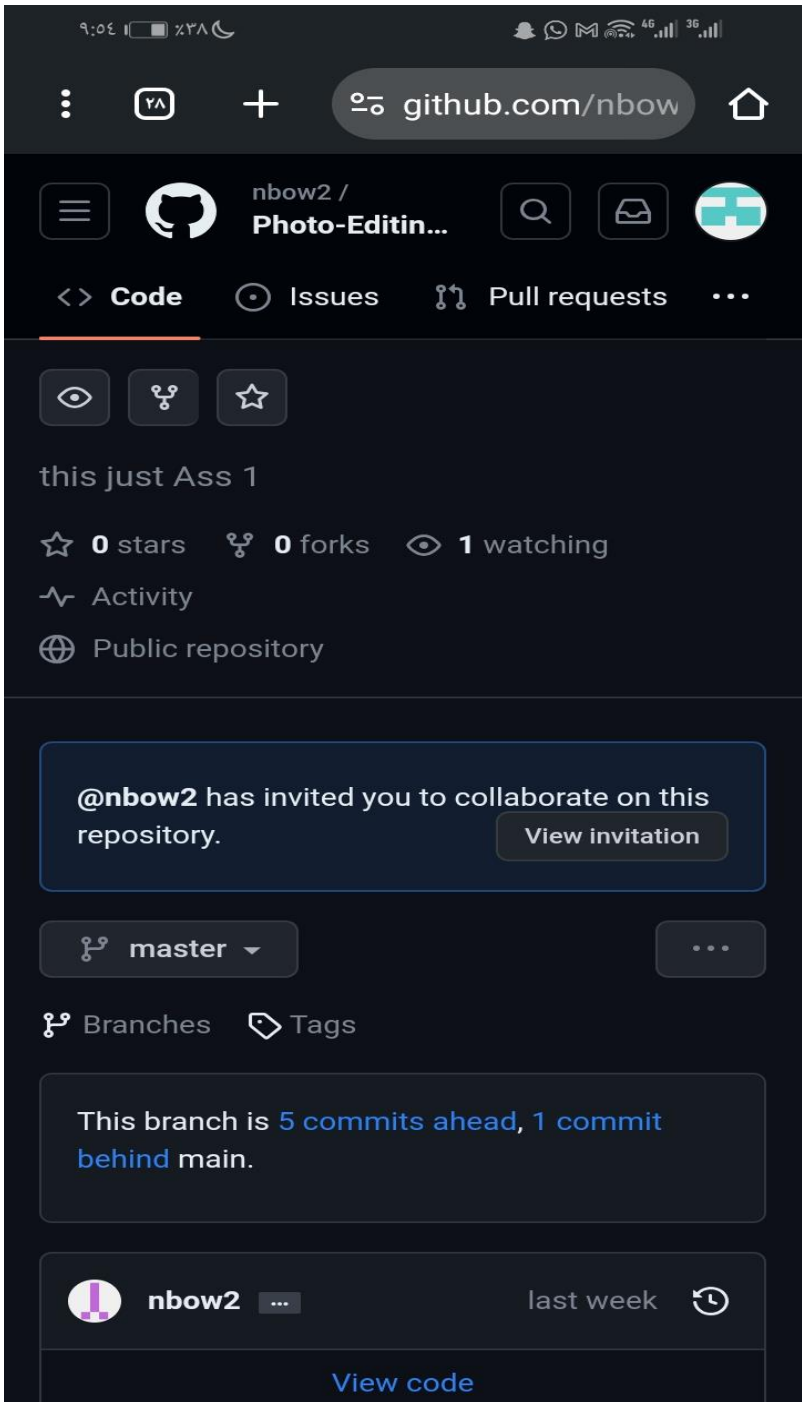
Releases

No releases published
Create a new release

Packages

No packages published

Faisal Ahmed Awad 20210804



github.com/nbow

nbow2 /
Photo-Editin...

Code Issues Pull requests

this just Ass 1

0 stars **0** forks **1** watching

Activity

Public repository

@nbow2 has invited you to collaborate on this repository.

View invitation

master

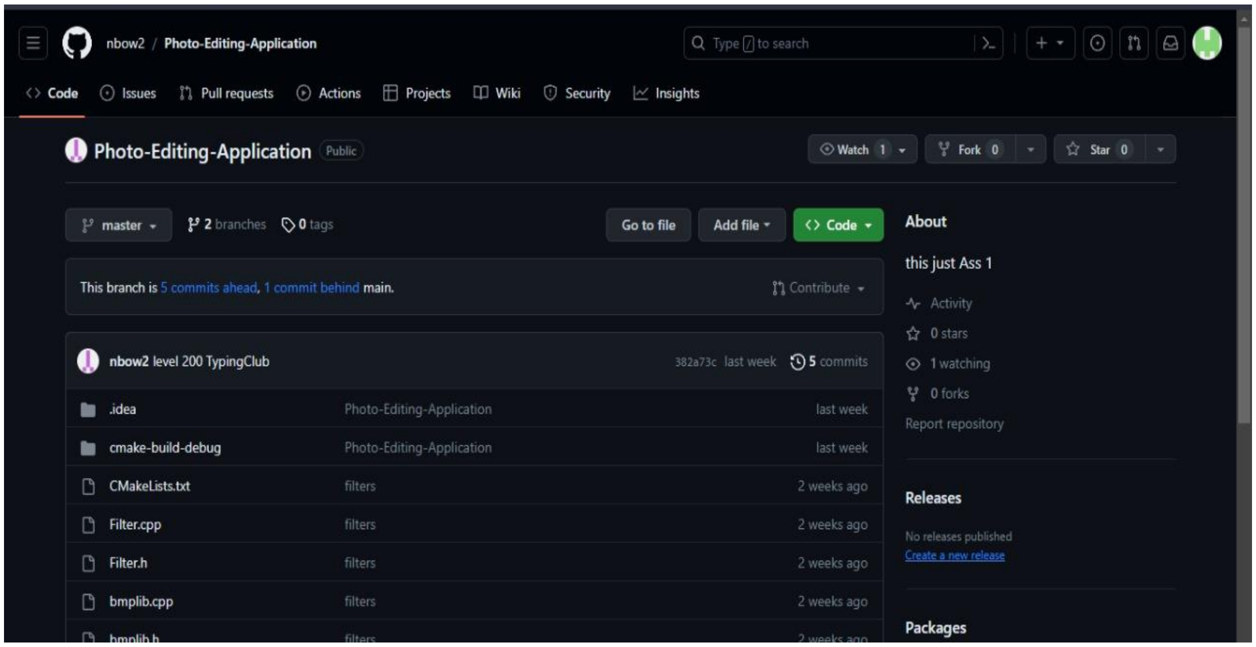
Branches Tags

This branch is 5 commits ahead, 1 commit behind main.

nbow2 last week

View code

Omer Sayedna Ali 20210703



nbow2 / Photo-Editing-Application

Photo-Editing-Application Public

master 2 branches 0 tags

This branch is 5 commits ahead, 1 commit behind main.

nbow2 level 200 TypingClub 382a73c last week 5 commits

.idea	Photo-Editing-Application	last week
cmake-build-debug	Photo-Editing-Application	last week
CMakeLists.txt	filters	2 weeks ago
Filter.cpp	filters	2 weeks ago
Filter.h	filters	2 weeks ago
bmplib.cpp	filters	2 weeks ago
bmplib.h	filters	2 weeks ago

About

this just Ass 1

Activity
0 stars
1 watching
0 forks
Report repository

Releases

No releases published
Create a new release

Packages