README

# Introduction

This folder will contain all the scripts in their respective folders. They all follow the implementation written in the report. In each folder there will be the scripts as well as extra functions, the results they’ve outputted and the unprocessed OCT Images.

This document will outline the requirements and a few steps to help the user use the code freely. However, more detailed explanations are written as comments in each function/script.

# Code A: Denoising and MAsking

**DISCLAMER: Training the CNN needs heavy GPU and/or CPU attention thus if you’re working on a laptop, it is suggested you use a desktop.**

## Requirments:

1. MATLAB R2019b (R2017b & R2018b also work, however R2019b is preferred).
2. It is preferred to have “Image processing Toolbox” downloaded in MATLAB. It is found in the link below: <https://www.mathworks.com/products/image.html>
3. BM3D which can be found in the ‘/Code A/BM3D’ folder or re-downloaded through the link below:

<http://www.cs.tut.fi/~foi/GCF-BM3D/>

Download BM3D MATLAB v3.0.5 released 16 Feb 2020

**NOTE: After extracting it. add the BM3D folder to Code A folder. Also change the FilePath of BM3D in the codes to fit the new FilePath. Many of the codes will not work without changing the FilePath!**

## CodeADenoising.m:

* This script is the main script it asks the user to input the file path of their image.
* The user can uncomment certain areas of the code to suit the type of image they’ve entered.
* Onwards, it trains TCNN (the already saved CNN) with the new training data produced by the functions above using the function TOCNN. **THIS IS VERY CPU and/or GPU heavy!**

### CreateTDATA.m

* This can be uncommented in CodeADenoising,m when the user will input a Y-stack image. This is a function which takes in the file path of an image.
* It creates new training data using thresholding, BM3D, BM3DDeb, BM3DSharp, FLLP and imsharpen from inputted image. Which will all be saved in folder ‘Code A/TrainingDataCreated’.
* It also creates new masking training data, using edge() function. Which will be saved in the folder ‘MaskingTD’

### CreateTDatabrackets.m

* This can be uncommented in CodeADenoising.m when the user will input an image of tooth containing brackets. This is a function which takes in the file path of an image.
* It creates new training data using thresholding, BM3D, BM3DDeb, BM3DSharp, FLLP, and imsharpen from inputted image. Which will all be saved in folder ‘Code A/TrainingDataCreatedBrackets’.
* It also creates new masking training data, using edge() function. Which will be saved in the folder ‘MaskingTDBrackets’

### CreateTDataHumanteeth.m

* This can be uncommented in CodeADenoising.m when the user will input a human tooth A-scan. This is a function which takes in the file path of an image.
* It creates new training data using thresholding, BM3DSharp, FLLP, and imsharpen from inputted image. Which will all be saved in folder ‘Code A/TrainingDataCreatedHT’.
* It also creates new masking training data, using edge() function. Which will be saved in the folder ‘MaskingTDHT’

### CreateTdatareg.m

* This can be uncommented in CodeADenoising.m when the user will input a registered B-scan. This is a function which takes in the file path of an image.
* It creates new training data using thresholding, BM3DSharp, FLLP, and imsharpen from inputted image. Which will all be saved in folder ‘Code A/TrainingDataCreatedReg’.
* It also creates new masking training data, using edge() function. Which will be saved in the folder ‘MaskingTDReg’

### TOCNN.m

* This function will train the already saved CNN (TCNN) using ADAM optimizer, using the training images in ‘Code A/TrainingDataCreated’ folder. Then saves the newly trained CNN and tests it. **THIS IS VERY CPU and/or GPU heavy!**

## CodeAMasking.m

* This script trains MCNN (the already saved CNN) with new training images already generated in CreateTData.m depending on the type of image the user inputted. Make sure to change the add path of the correct folder containing the training data. **THIS IS VERY CPU and/or GPU heavy!**

### MOCNN.m

* This function will train the already saved CNN (MCNN) using ADAM optimizer, using the training images in ‘Code A/MaskingTD’ folder. Then saves the newly trained CNN and tests it. **THIS IS VERY CPU and/or GPU heavy!**

# Code B: registration

This script of code B will implement the registration of OCT and XMT files.

### Codeb.m

* It starts with asking the user to enter the FilePath of moving and fixed image in the command window.
* Then it sets the optimizer to OnePlusEvolutionary and the metric to MattesMutualInformation, for Multimodal registration. It will output the figures after the registration is done.
* If the user would like to see the Monomodal registration, they can uncomment the sectioned mentioned in the code.

# code c: REconstruction

This script of code will implement the reconstruction of OCT and XMT files.

### CodeC.m

* User will adjust the paths of the folders Nok1\_XMT\_Reg and OCTReg-denoised at the start, which are the XMT registered and OCT registered denoised images respectively that are provided. Then it outputs the two 3D models.

# Code D: Suggested decay

This script of code will attempt to implement the detection of tooth decay in OCT images. This code is run in python.

## Requirements:

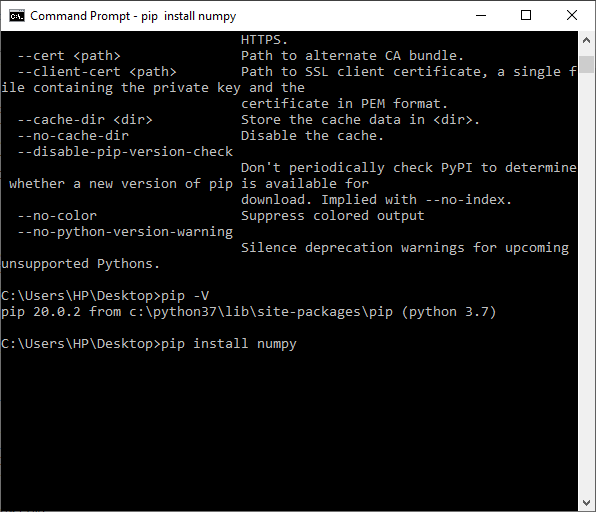
1. Python version 3.7, can be found using the link below:

<https://www.python.org/downloads/>

1. Pip version 20.1, can be found using the link below:

<https://pip.pypa.io/en/stable/installing/>

### Suggesteddecay.py

* Make sure you have installed the libraries mentioned in the code. For example: numpy needs to be installed, thus the user will go to their command line:

This will install numpy, and will notify the user.

* The user will then run the script. In the command line it will ask the user to input the filepath of the image. Then the output will be saved in ‘/Code D’ folder.

# Main script:

This script combines Code A – D, to create a menu for the user. This was made to help the user not to work with the code directly. It is also run in python with an extra requirement.

## Requirements:

1. MATLAB engine API for python, instructions are found below:

<https://uk.mathworks.com/help/matlab/matlab_external/install-the-matlab-engine-for-python.html>

**NOTE: command line has to be in administrator mode for this to work.**

1. Copy and paste the functions (.m files) in ‘/MATLABfunctions’ folder to the user’s MATLAB common file. **NOTE: Change the FilePath of BM3D in the codes to fit the new FilePath. Many of the codes will not work without changing the FilePath!**

### Mainscript.py

* User will run the code through Python IDLE. They will be asked what type of image they will input, as well as which type of function (i.e. denoising, registering, masking) they want done.
* The user will be asked to input the file path/s of their image/s. After the functions are done the output images will be saved into their respective folder or ‘/Main script’ folder.

### Braces.py

* This is a python script where it asks the user to input the file path of their image. Then it will output the length of differences between the edges in the image.
* The user can select this function in MainScript.py, under the selection of Length.

# Software:

This script combines Code A – D, to create a menu for the user in terms of a software. This makes it very user-friendly. It is also run in python with an extra requirement.

## Requriements:

1. MATLAB engine API for python, instructions are found below:

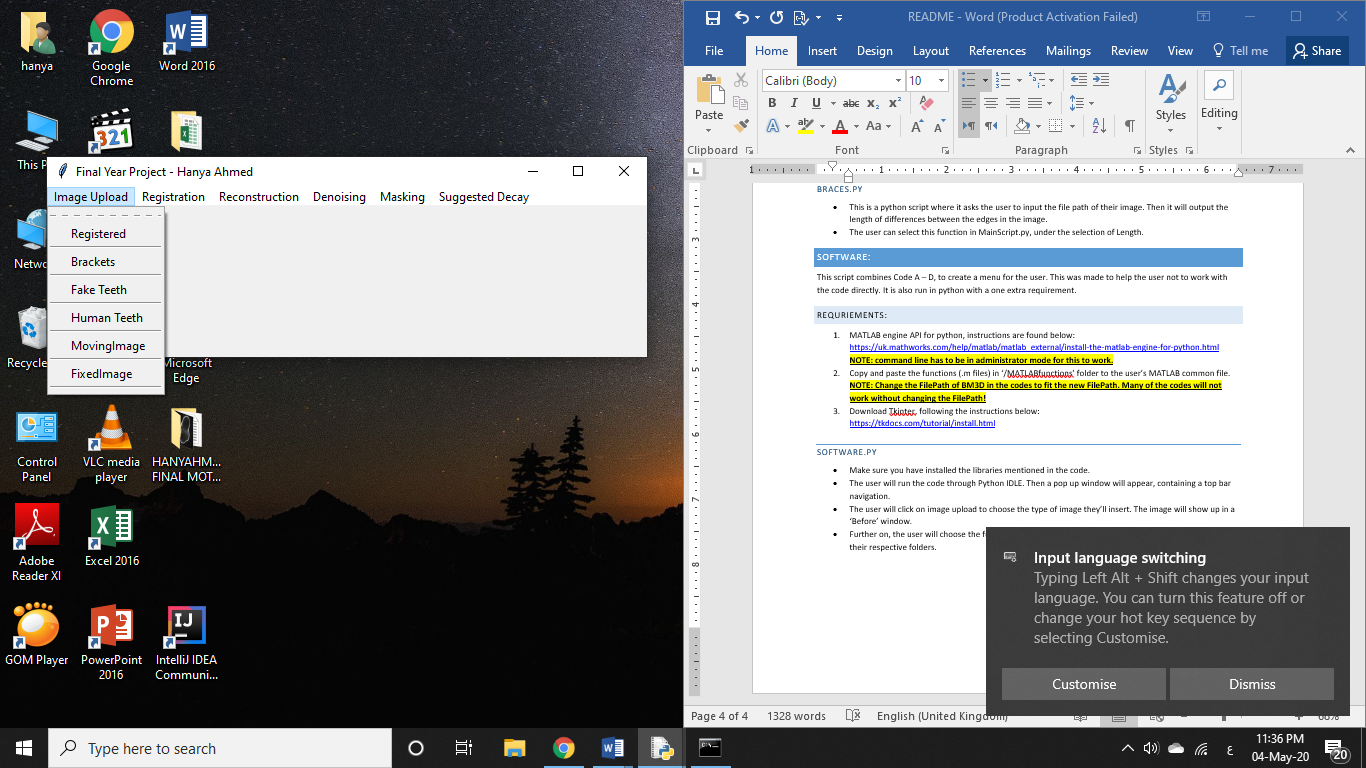
<https://uk.mathworks.com/help/matlab/matlab_external/install-the-matlab-engine-for-python.html>

**NOTE: command line has to be in administrator mode for this to work.**

1. Copy and paste the functions (.m files) in ‘/MATLABfunctions’ folder to the user’s MATLAB common file. **NOTE: Change the FilePath of BM3D in the codes to fit the new FilePath. Many of the codes will not work without changing the FilePath!**
2. Download Tkinter, following the instructions below:

<https://tkdocs.com/tutorial/install.html>

### Software.py

* Make sure you have installed the libraries mentioned in the code.
* The user will run the code through Python IDLE. Then a pop up window will appear, containing a top bar navigation.
* The user will click on ‘image upload’ to choose the type of image they’ll insert. The image will show up in a ‘Before’ window.
* Further on, the user will choose the function from the top navigation bar, they want to conduct. The output images will be saved in their respective folders.

# WEbsite:

This script combines Code A – D, to create a menu for the user. This was made to help the user not to work with the code directly. It is also run in python with an extra requirement.

## REquirements:

1. MATLAB engine API for python, instructions are found below:

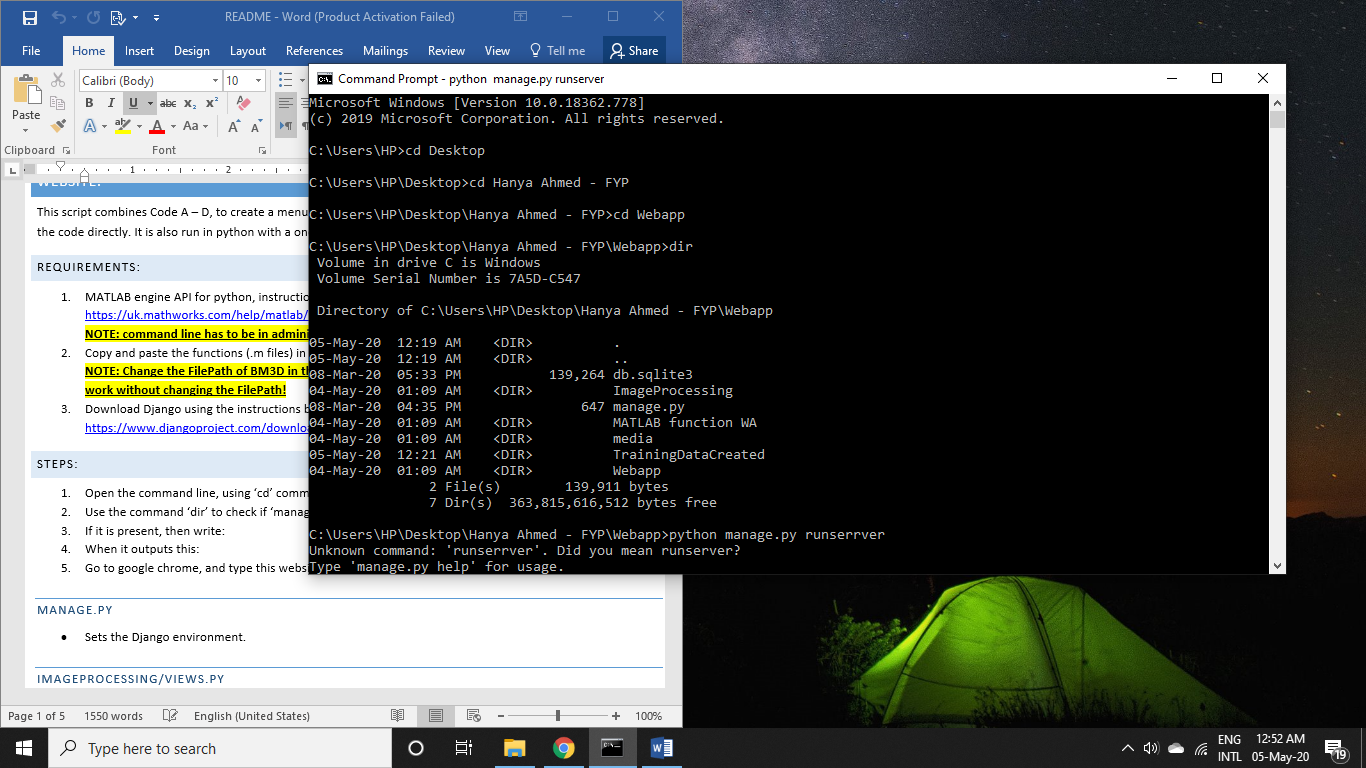
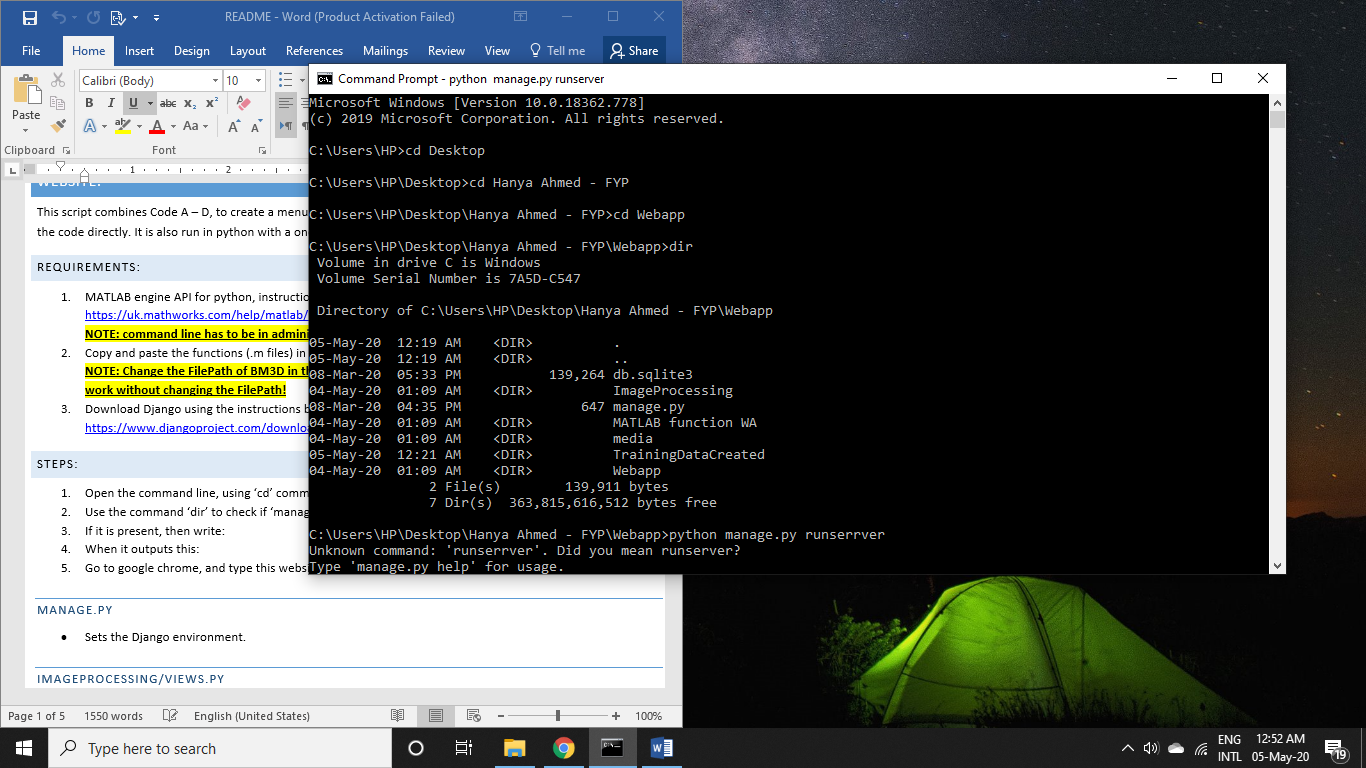
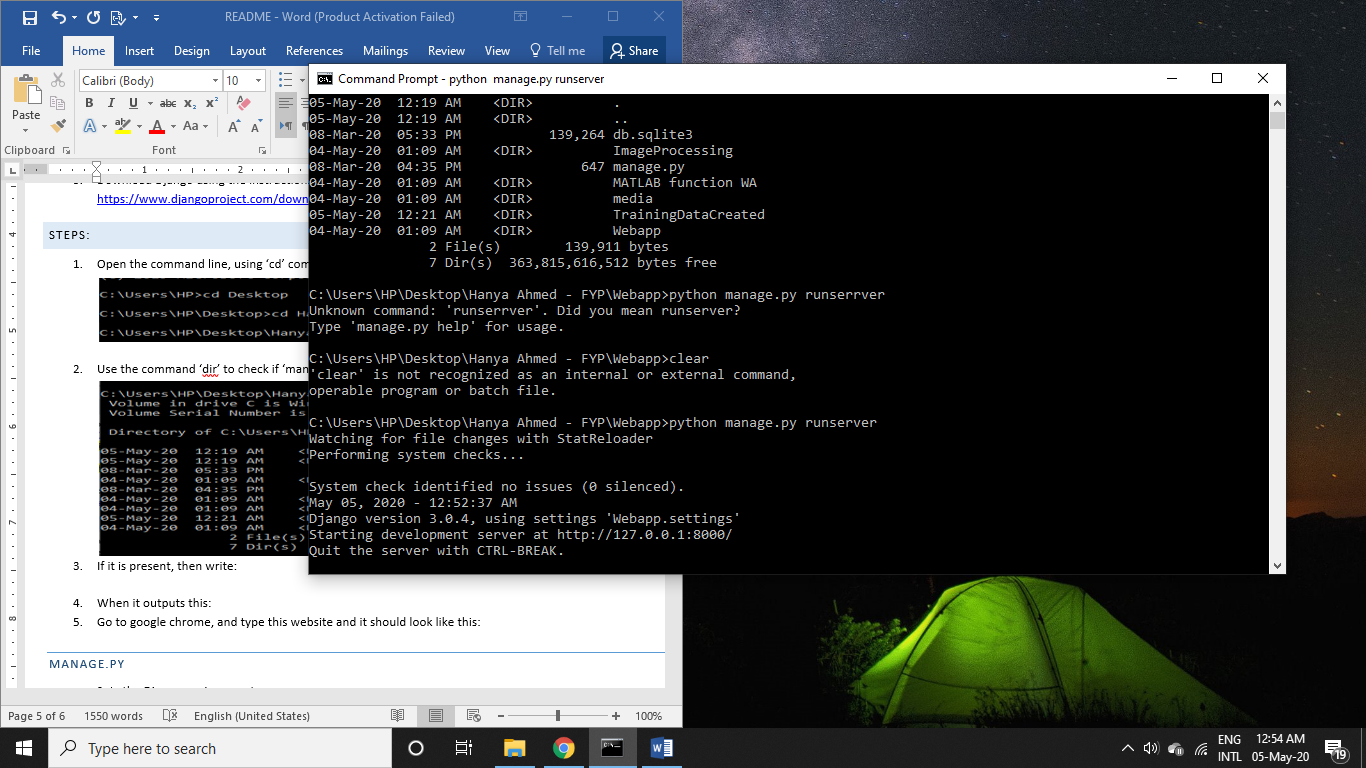
<https://uk.mathworks.com/help/matlab/matlab_external/install-the-matlab-engine-for-python.html>

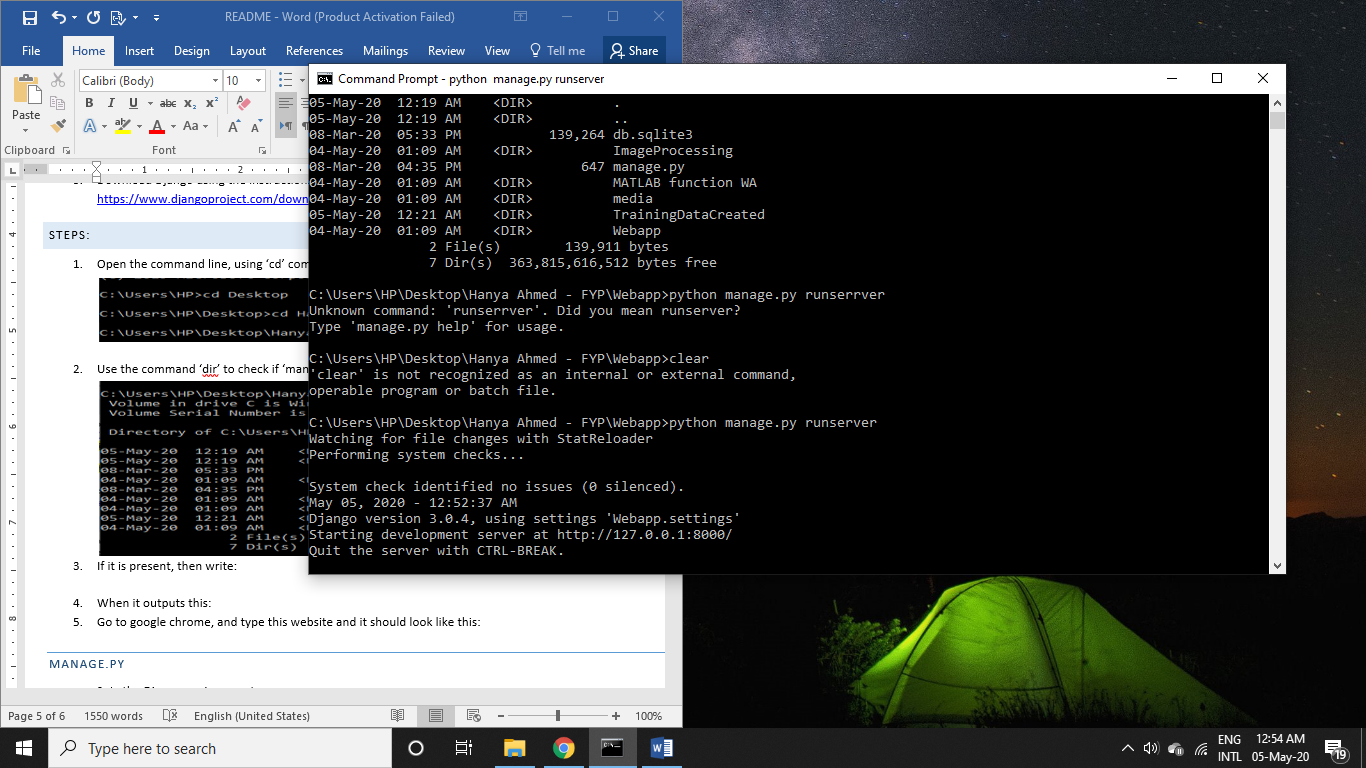
**NOTE: command line has to be in administrator mode for this to work.**

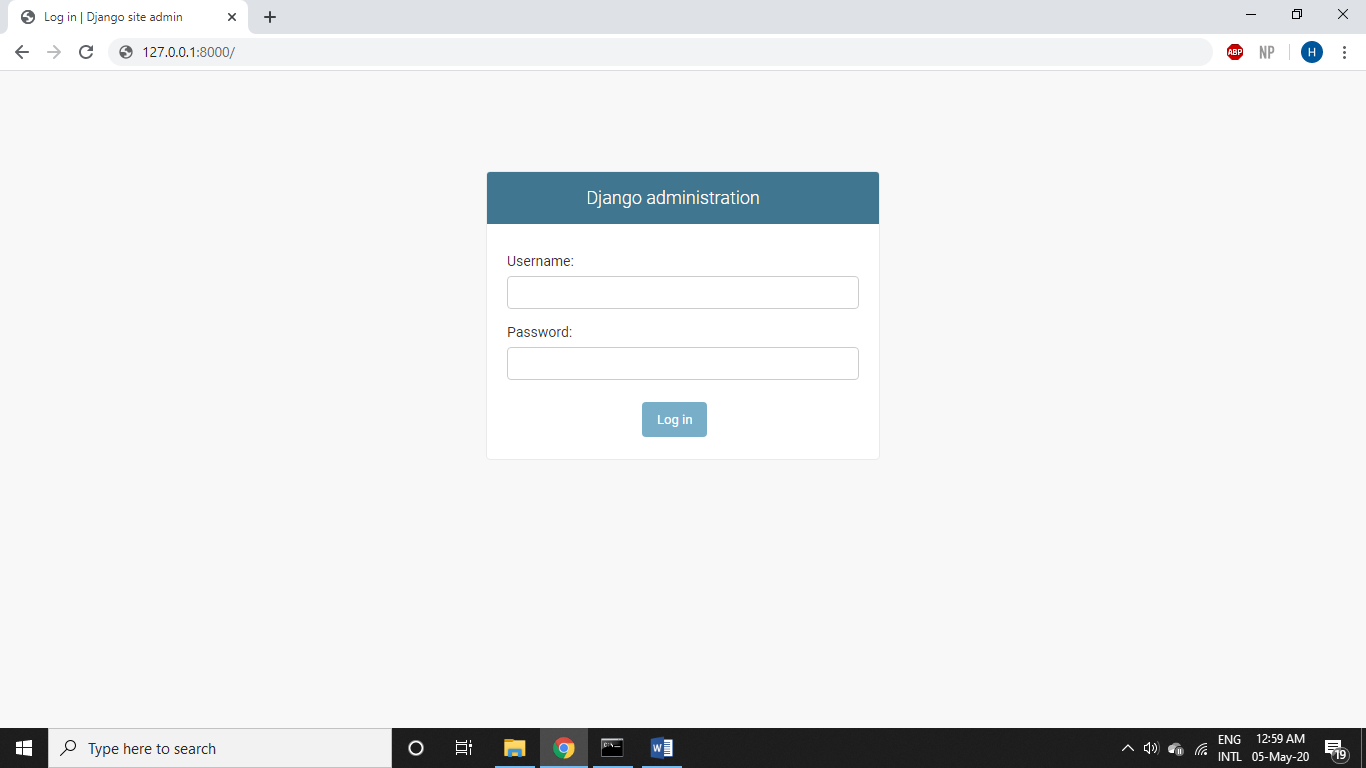
1. Copy and paste the functions (.m files) in ‘/MATLABfunctions’ folder to the user’s MATLAB common file. **NOTE: Change the FilePath of BM3D in the codes to fit the new FilePath. Many of the codes will not work without changing the FilePath!**
2. Download Django using the instructions below:

<https://www.djangoproject.com/download/>

## Steps:

1. Open the command line, using ‘cd’ command to reach the folder Webapp:
2. Use the command ‘dir’ to check if ‘manage.py’ is present.
3. If it is present, then type: (if it is not present, then the user is in the wrong folder)
4. The output should be the following:

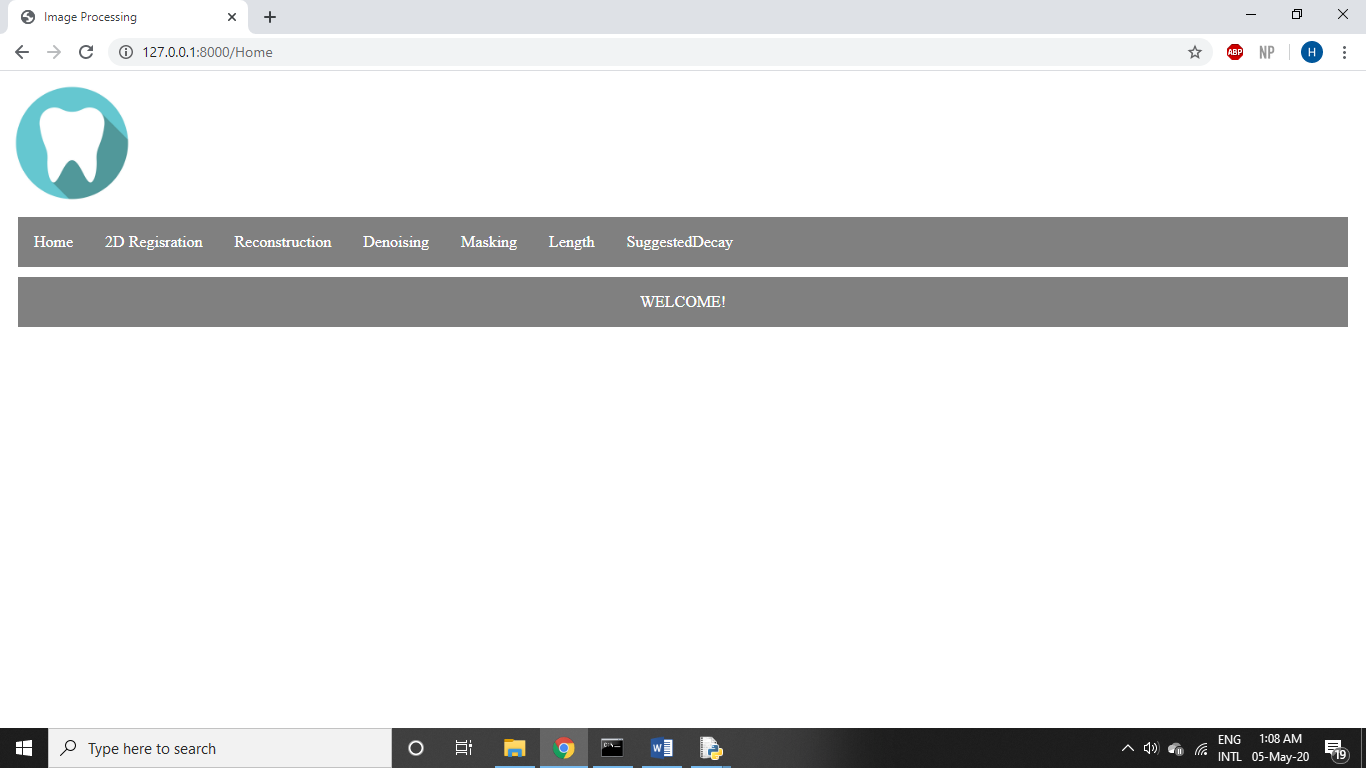


1. Open google chrome, and type this website ‘http://127.0.0.1:8000’ and it should look like this:
2. This is the Django administration home page, where you can add as many users as you want to be able to access this website. The main username and password are:

**User name:** hp

**Password:** QMUL1105

1. When signed in, you can go to the main website using ‘http://127.0.0.1:8000/Home’ which will take you to this page:



**MAnage.py:** Sets the Django environment.

**ImagePRocessing/views.py:** Contains all the functions and the receiving the information from the HTML.

**Imageprocessing/urls.py:** Connects the HTML to the function in the views.py script.

**ImageProcessing/templates/ImageProcessing:** Holds all the HTML links that creates the simple GUI for the users.

**WEbapp/urls:** Connects the Django home page to the ImageProcessing HTMLs (this project created).

# Images provided to test with:

|  |  |  |  |
| --- | --- | --- | --- |
| Image | Type of Image | Training Data Folder | Masking TD Folder |
| OCTImage.png | Brackets | TrainingDataCreatedBrackets | MaskingTDBrackets |
| OCTImage0100.png | Human teeth | TrainingDataCreatedHT | MaskingTDHT |
| OCTImage0183.png | Fake teeth (Y-stack) | TrainingDataCreated | MaskingTD |
| OCTRegistered0080.png | Registered (B-scan) | TrainingDataCreatedReg | MaskingTDReg |