Maintenance Manual

Project name: TrashClassifier

Team name: Big Data Energy

Team Members

Travis Hammond

Kenta Miyahara

Julian Hernandez

Jeffrey de Jesus

Bryan Burch

Daniel Smagly

Santiago Bermudez

Christopher Allen

Date: 05/12/2023

TABLE OF CONTENTS

1. Introduction

The TrashClassifier (TC) is a deep neural network-based program designed to assist users in the proper disposal of waste by accurately identifying and classifying various types of trash. The TC aims to improve waste management practices and minimize the environmental impact of improper waste disposal by promoting source separation of recyclables and organic materials. Leveraging cutting-edge deep learning techniques, the TrashClassifier processes images captured by smartphone cameras or other imaging devices and classifies the waste items according to established waste disposal guidelines. This document provides an overview of the TC features, functionality, and user interface, ultimately outlining a method for maintaining this project.

2. Implementation Tools and Technologies

2.1 Programming Languages

Python: A versatile, high-level programming language known for its readability and ease of use, often used in web development, data analysis, and artificial intelligence. Used for the Backend and the Terminal Front-End App/Script.

HTML: Hypertext Markup Language, a standard markup language used to create and structure content on the web. Used for the Front-End Website.

JS: JavaScript, a high-level, interpreted programming language primarily used for adding interactivity and dynamic content to web pages. Used for the Front-End Website.

CSS: Cascading Style Sheets, a stylesheet language used for describing the look and formatting of a document written in HTML or XML. Used for the Front-End Website.

2.2 Libraries and Frameworks

Flask: A lightweight Python web framework used for building web applications. Used for the Backend Server.

PyTorch: An open-source machine learning library for Python, used for deep learning and AI applications. Used for Model Inference on Trash Images.

OpenClip: A deep-learning model for image recognition and manipulation tasks, originally developed by OpenAI. Used for Model Inference on Trash Images.

ReactJS: A popular JavaScript library for building user interfaces, particularly for single-page applications. Used for the Front-End Website.

PyTest: A testing framework for Python that simplifies the process of writing and running tests. Used for testing the Front-End and Backend python code.

TablerReact: A ReactJS-based framework for building responsive, modern web applications with a clean design. Used for the Front-End Website.

2.3 IDE Technologies
VS Code: A lightweight, extensible, and customizable source-code editor developed by Microsoft, supporting multiple programming languages. Used for all code editing.

Jupyter Notebook: An open-source web application that allows the creation and sharing of live code, equations, visualizations, and narrative text. Used for interactive editing, manual testing, and work with the modals.

Powershell/Terminal: Command-line interface tools for Windows (Powershell) and Unix-based systems (Terminal) that enable users to interact with the operating system and perform tasks. Used for many things, Git, running scripts/tests, and more.

2.3 Web Services
HTTP: Hypertext Transfer Protocol. Used for communicating with the front-end and backend.

REST API: Representational State Transfer API, a set of architectural principles for designing networked applications. Used for the backend interactions for Model Inferences and the Annotation Database.

2.4 Databases/Data storages
SQLite: A lightweight, serverless, self-contained SQL database engine often used in mobile and desktop applications. Used for the Annotation Database.

2.5 Version Control System
GitHub: A web-based platform for version control and collaboration using Git, allowing developers to work together on projects from anywhere. Used for storing the code and handling updates to the repo.

Git: A distributed version control system designed to handle everything from small to large projects with speed and efficiency. Used for working with GitHub to update the repo.

2.6 Server
Data Science Club Server: A server provided by the Data Science Club. Used for running the larger models for inference.

3. Runbook

3.1 Server goes down
The server may go down if a critical error occurs, the VRAM on the GPU gets too large, or if an error is not caught.

To get the server running again:

1. Make sure the running consolidated_server.py is shutdown. Use Ctrl+C to force close it.

2. Re-run the consolidated_server.py from (Trash-Sorting\project\backend\consolidated_server.py)

3.2 Database becomes corrupt

The only database used in TrashClassifier is the annotation database. Luckily, the annotation database very likely will not become corrupt as per the SQLite documentation. However, we recommend making manual backups if this is a concern, so that in such a failure case, a manual backup of the database can be inserted back in. Restart the consolidated server to reload the database (follow the steps in section 3.1)

3.3 Third party services go down

There are many third party packages used in the TrashClassifier programs but there are mainly only two services used: Waste Wizard and Model Downloading.

Waste Wizard (Dictionary for trash classification)
The program utilizes waste wizard data which is from a third party, however the entire waste wizard dictionary is stored in the program itself. If it gets corrupted, then re-downloading is necessary. This dictionary is stored here project\backend\model_inference\models\efficientnet_v2_l\conversion_dict.json

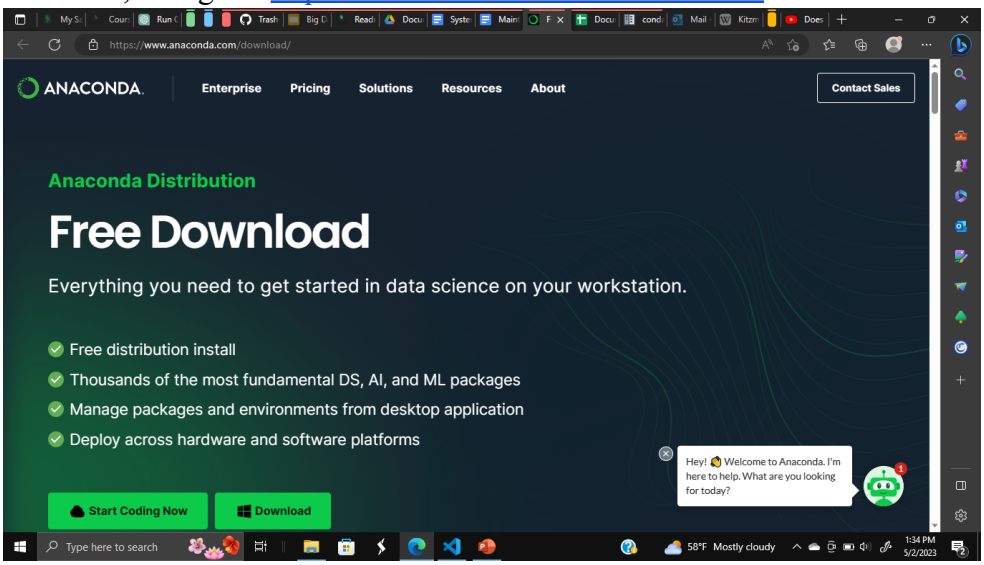
Model Downloading
The OpenClip and EfficentNet models are downloaded from third party sites. It is unlikely such sites would be down for long, if ever, however, we recommend either utilizing the models already downloaded in the meantime or going to the OpenClip GitHub repo or the PyTorch TorchVision GitHub repo and put up an issue.

3.4 Environment Setup

Python Environment

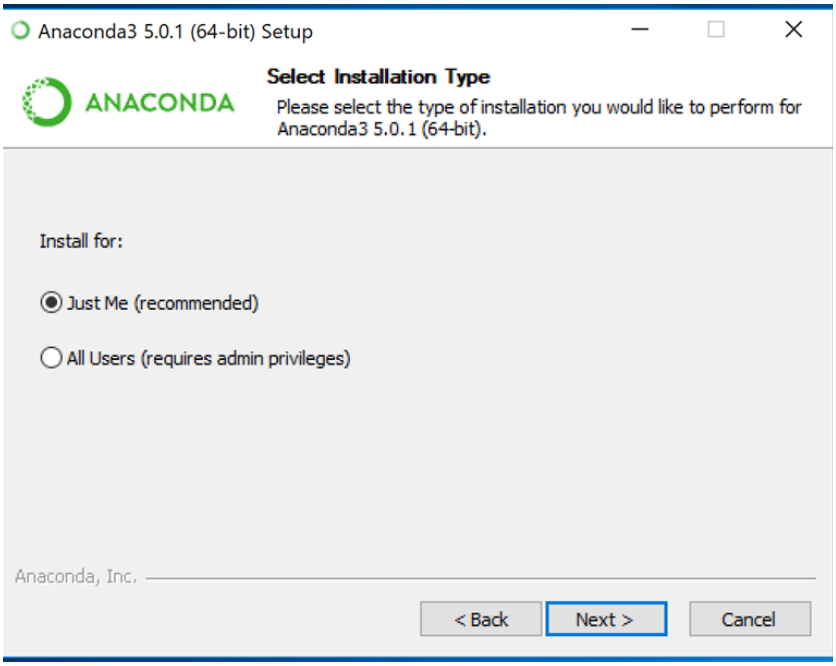
We recommend using Anaconda to create a python environment with all the requirements outlined in requirements.txt. For help with Anaconda setup, see the Anaconda site or the cheat sheet with the proper commands.

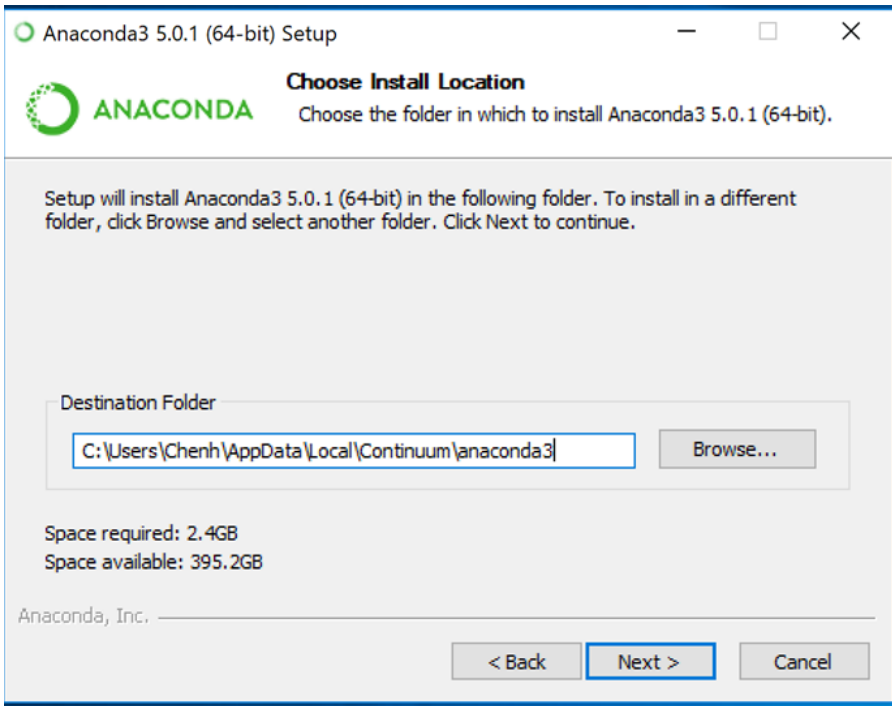
To set up Anaconda, first go to https://www.anaconda.com/download/ and download from there.

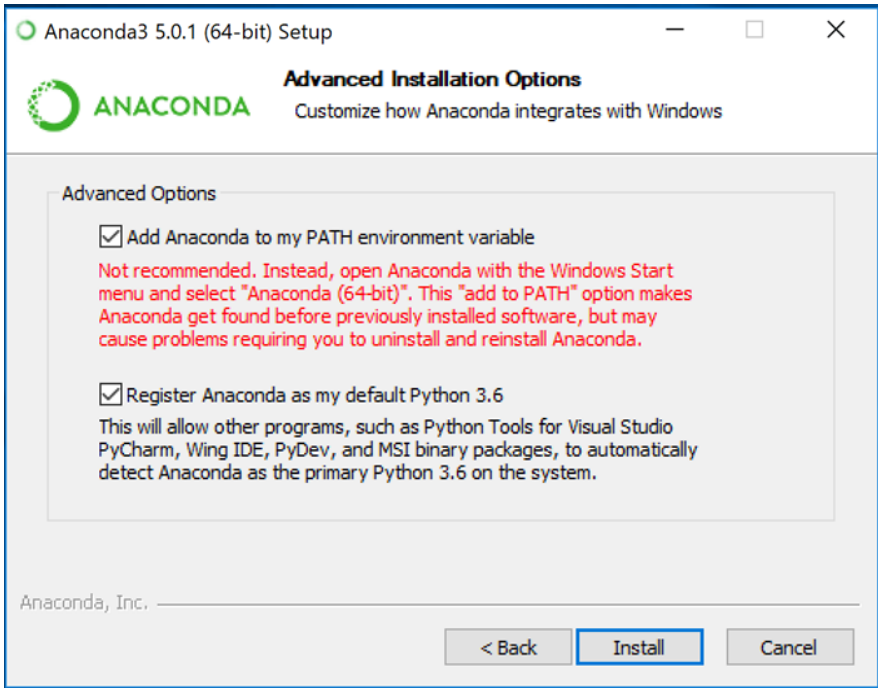


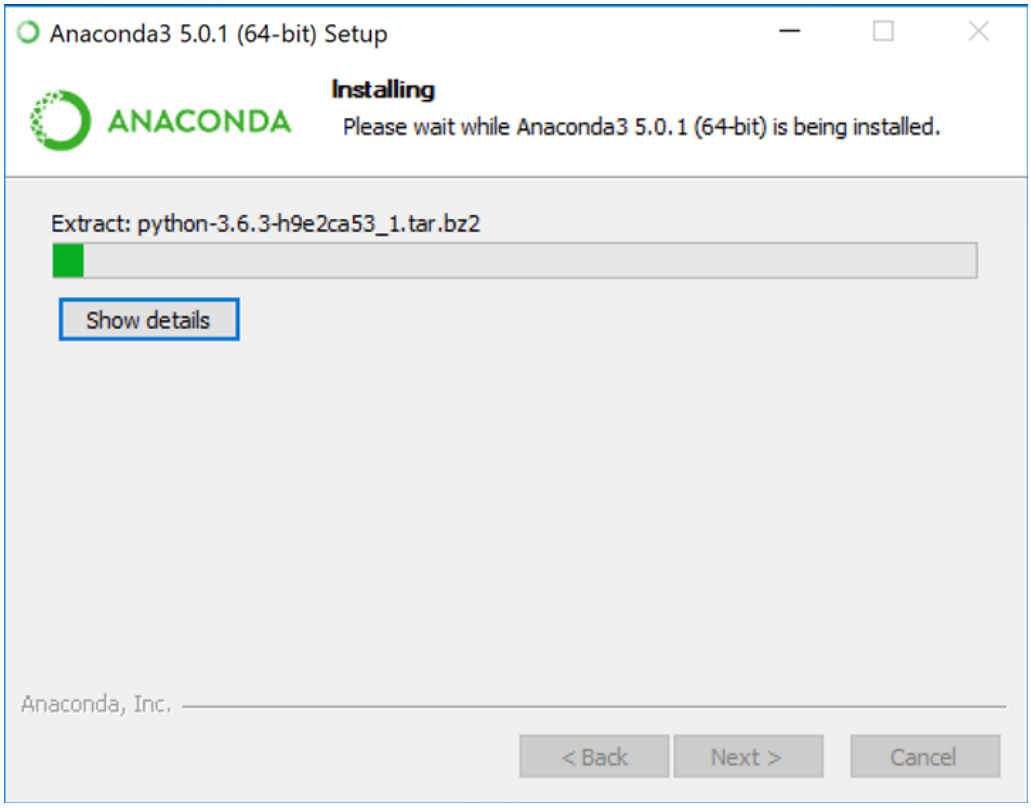
After installing, you should open up the file and see the following.

Select the following settings and click next.

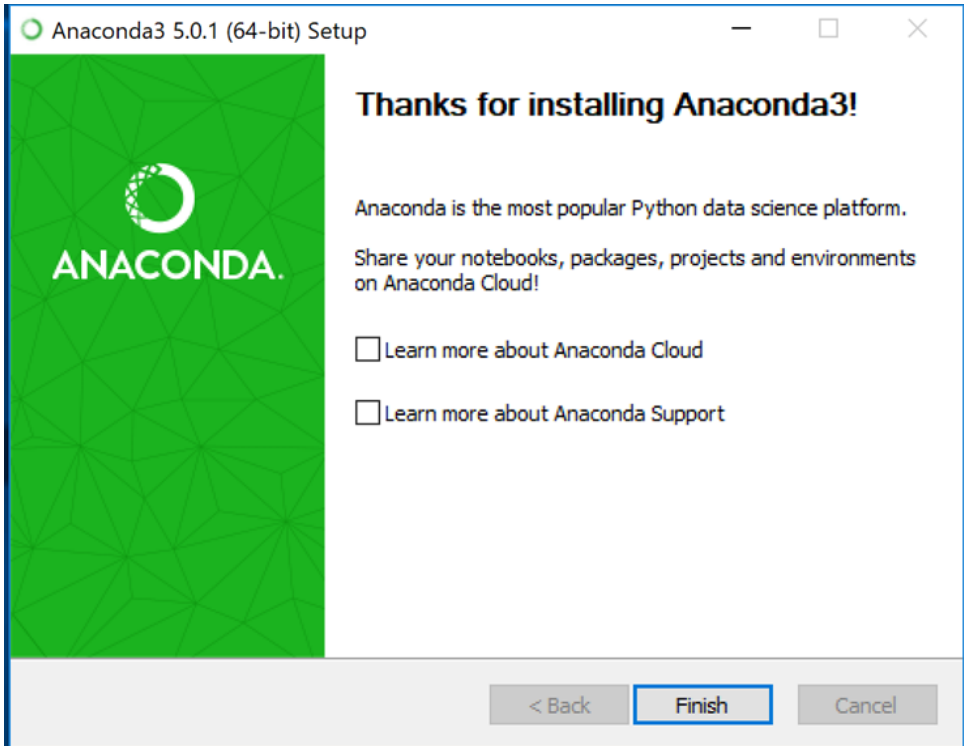

Anaconda3 5.0.1 (64-bit) Setup
ANACONDA
Select Installation Type
Please select the type of installation you would like to perform for Anaconda3 5.0.1 (64-bit).
Install for:
Just Me (recommended)
All Users (requires admin privileges)
Anaconda, Inc.
< Back
Next >
Cancel


Anaconda3 5.0.1 (64-bit) Setup
ANACONDA
Choose Install Location
Choose the folder in which to install Anaconda3 5.0.1 (64-bit).
Setup will install Anaconda3 5.0.1 (64-bit) in the following folder. To install in a different folder, click Browse and select another folder. Click Next to continue.
Destination Folder
C:\Users\Chenh\AppData\Local\Continuum\anaconda3
Browse...
Space required: 2.4GB
Space available: 395.2GB
Anaconda, Inc.
< Back
Next >
Cancel

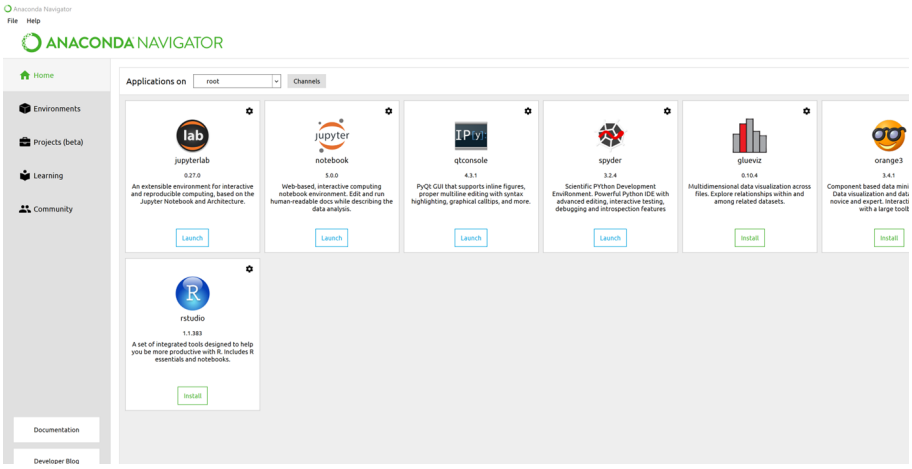




Now click finish.



When you run the navigator you should see something like the following.



Code Editor
We recommend using VS Code with the Python Language and Jupyter Notebook extensions

OS/System
The front-end and backend work on Windows, Mac, and Linux.

System Requirements
There are system requirements for the backend to have at least 8GB of RAM and for fast response a Nvidia GPU with at least 8GB of VRAM.

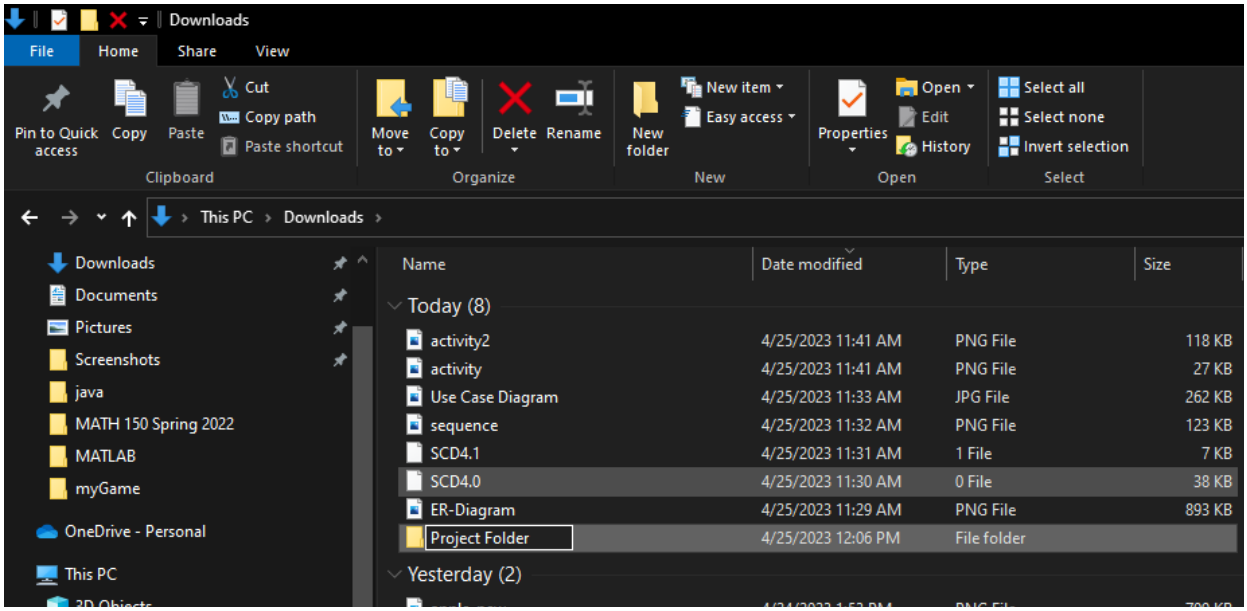
3.5 Making Code Changes

We recommend cloning the repo locally to make any changes and once cloned pulling and pushing can be done to keep up to date and make changes. Use Git Bash or the VS Code Git extension for cloning, pulling, and pushing to the repo.

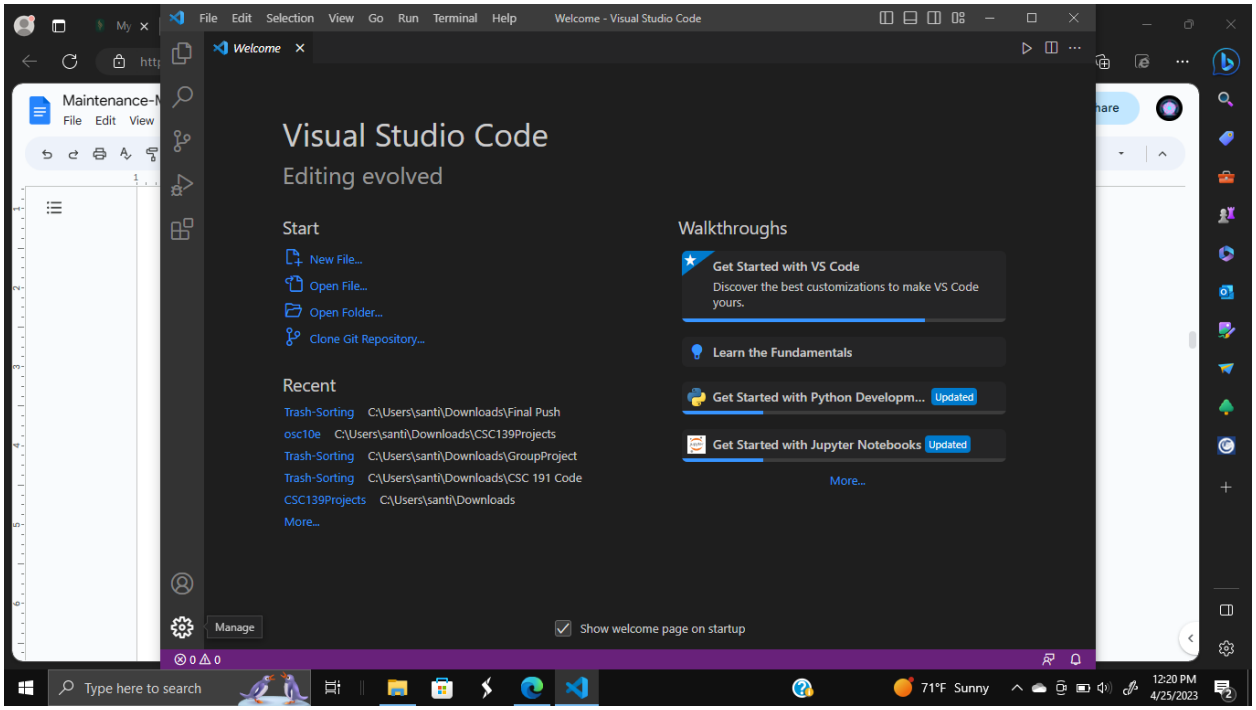
Our team followed the method of creating branches when doing multiple complete changes, so that the completed change can be merged into the main branch in one go. If a change did not have many parts or was complete, we would allow merging directly to main. However, we recommend code reviews and testing for merging directly to main for integrity in the main branch. Note to always pull before making changes and pull before pushing to handle merge conflicts if any.

3.5.1 Cloning the Repository

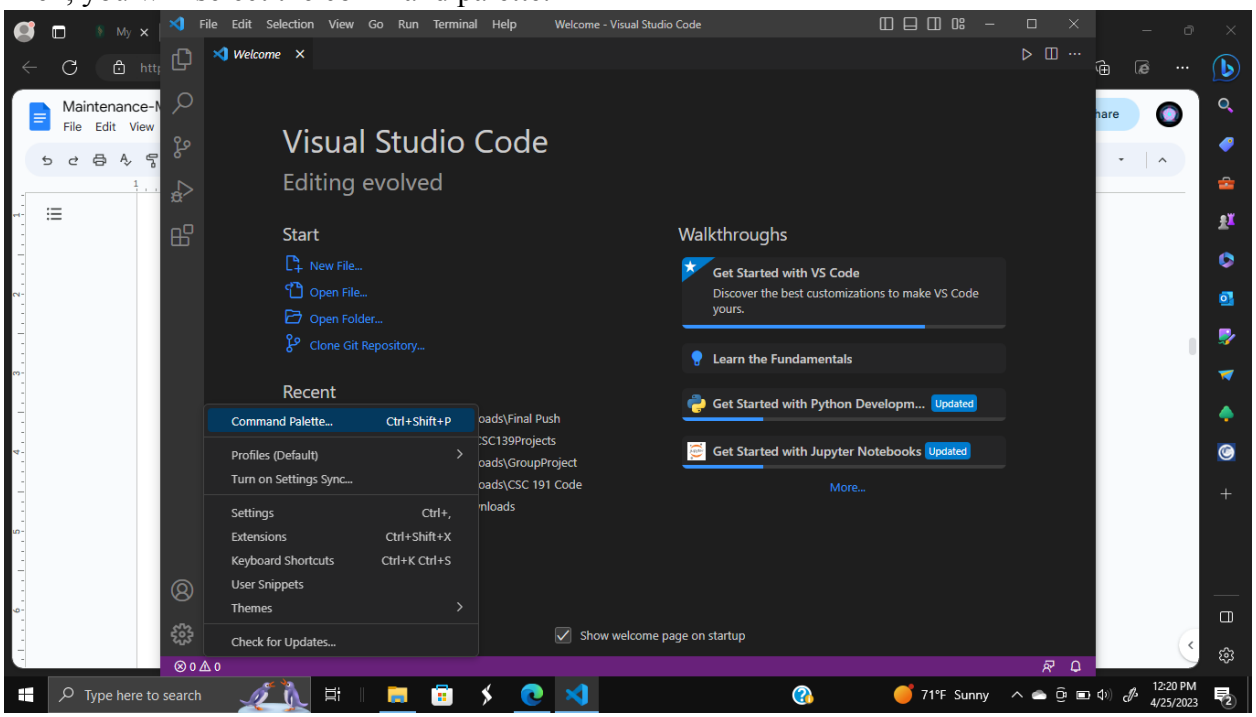
We will have to create a folder to hold the project files. You can create it anywhere and call it whatever you want. Just make sure it is in an easy to access location so that you can easily find it to get its file path.



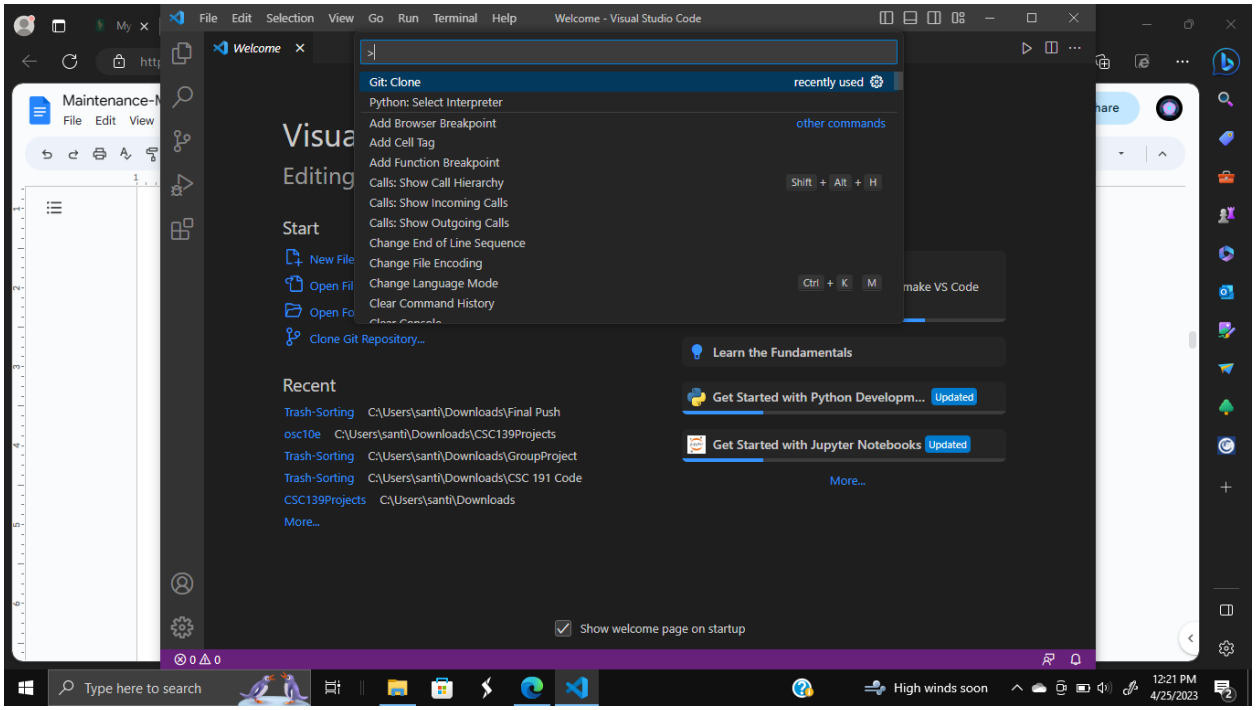
Then, you will want to open up VS Code. From there you will want to click on the settings icon on the lower left.



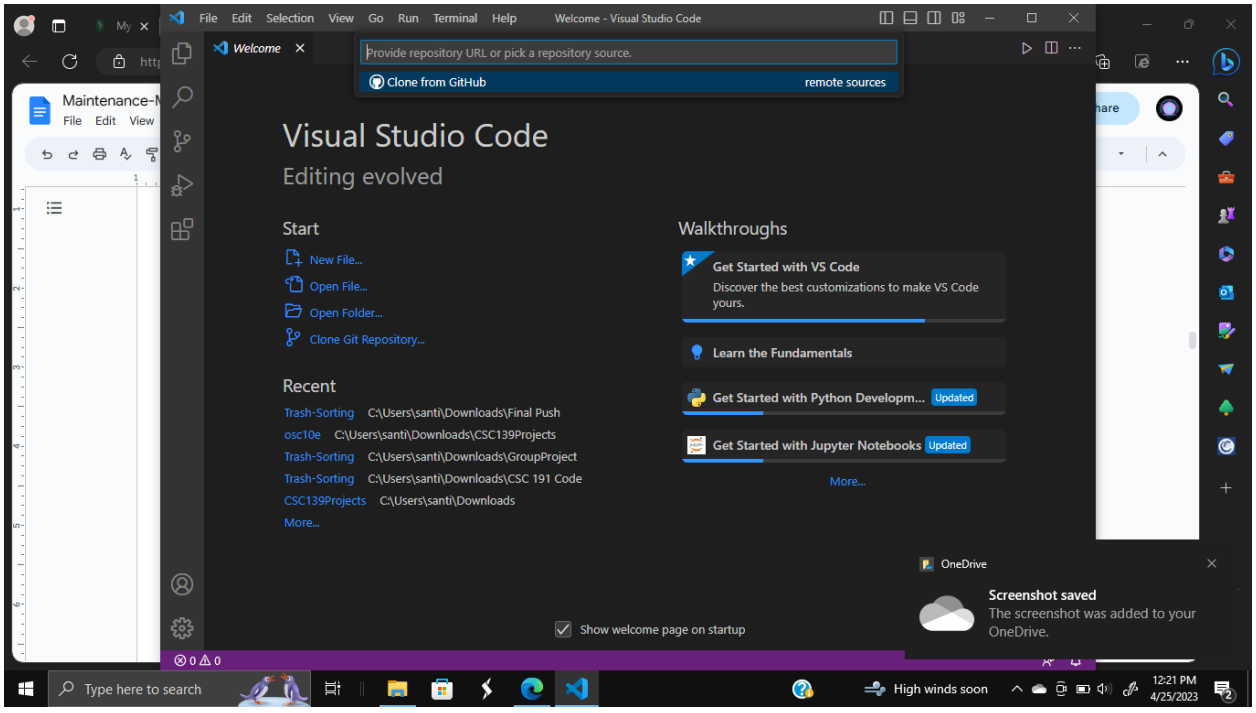
Then, you will select the command palette.



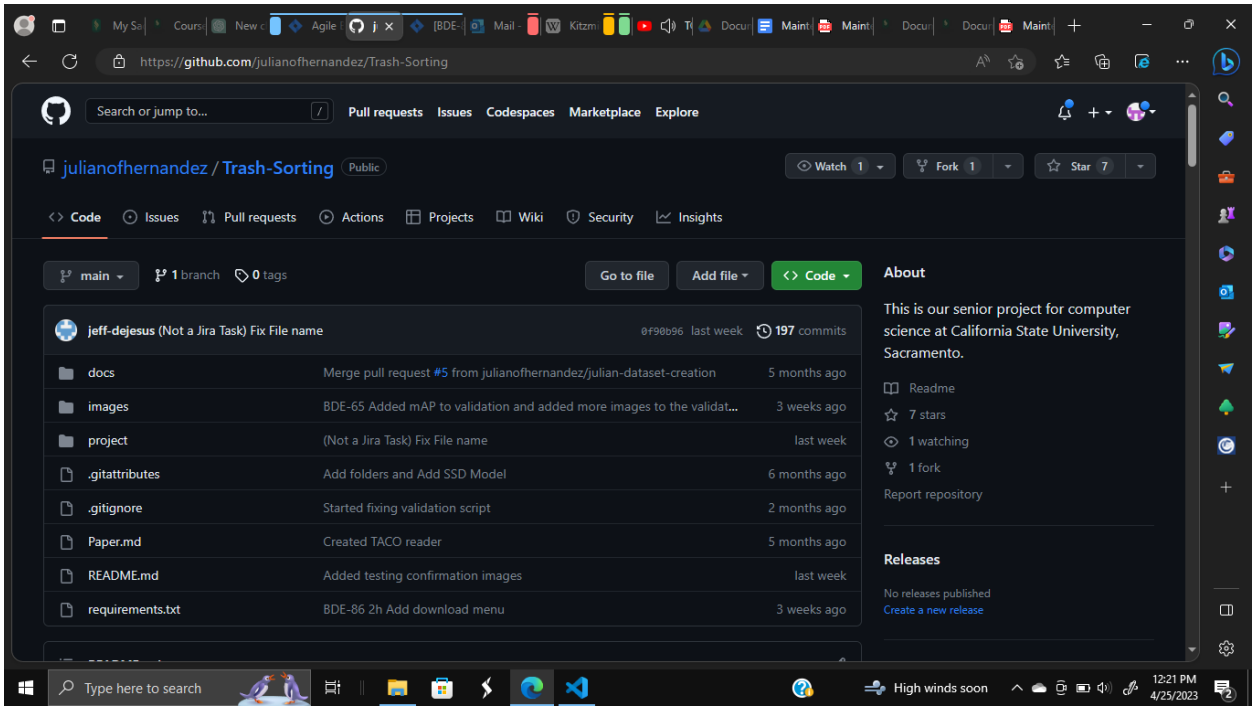
From there, you will select the Git: Clone option in the drop-down menu that appears from the search bar above.

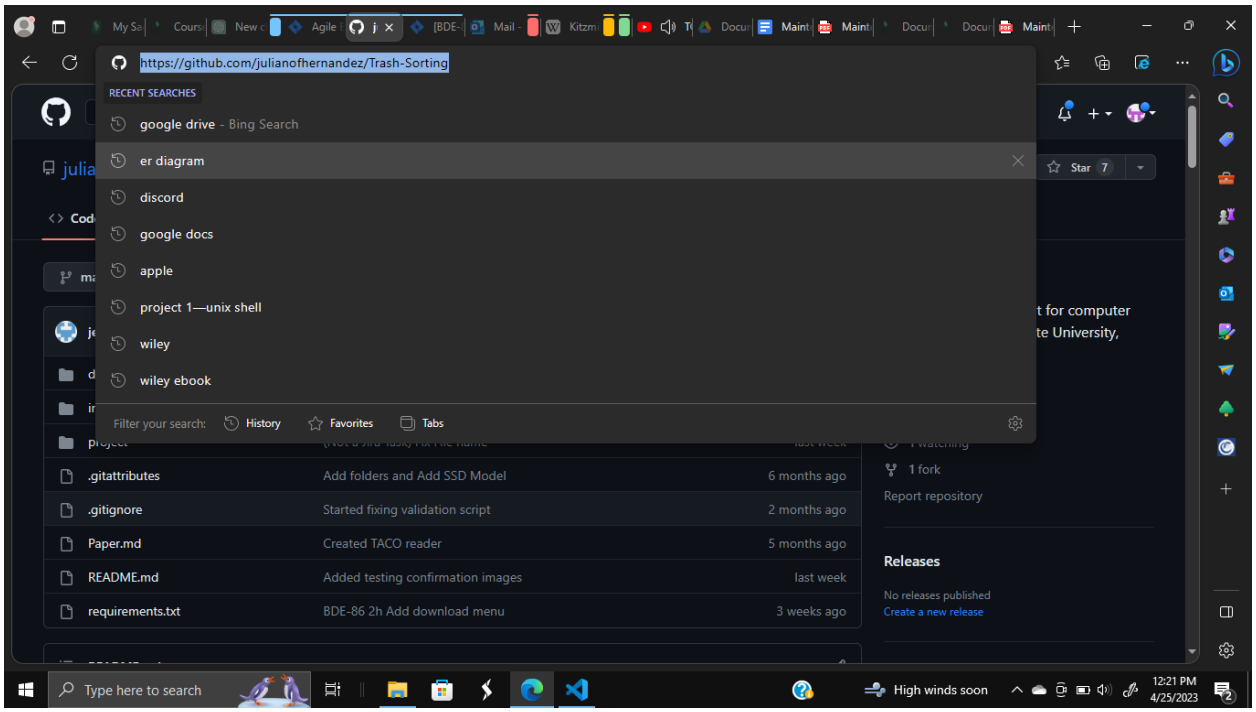


You will then select Clone from GitHub.

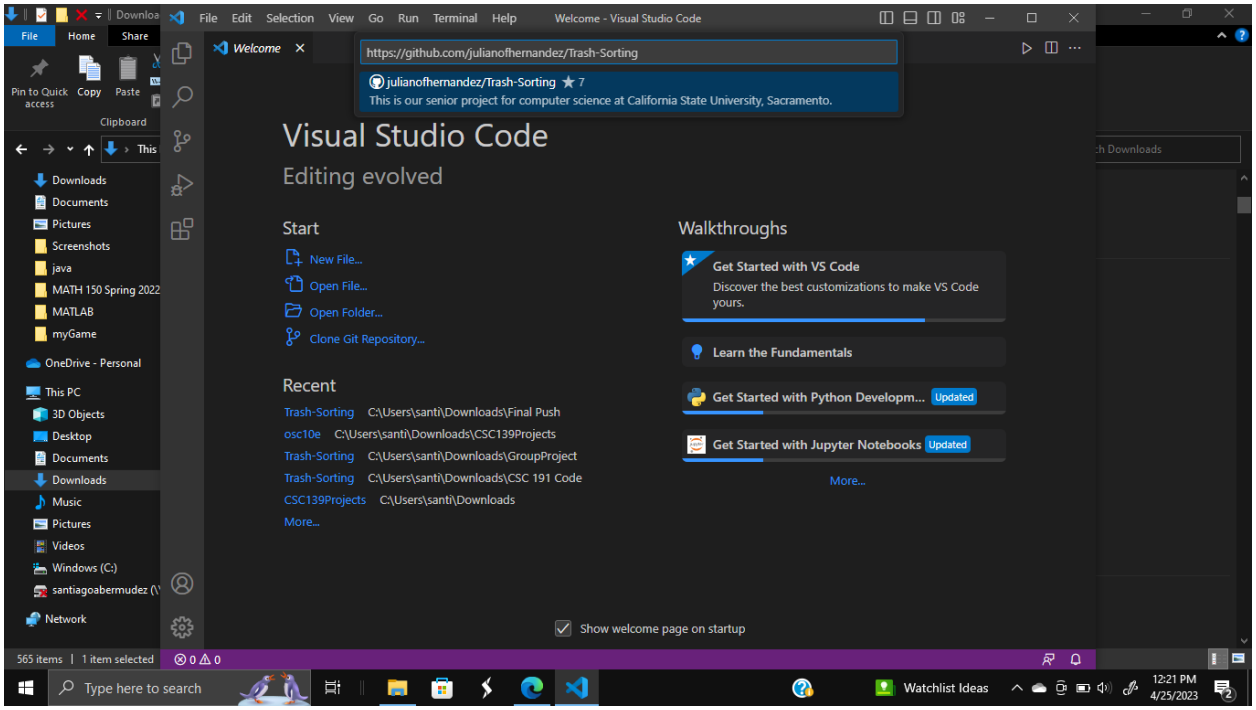


Then you will navigate to the GitHub repository and copy the repository URL.

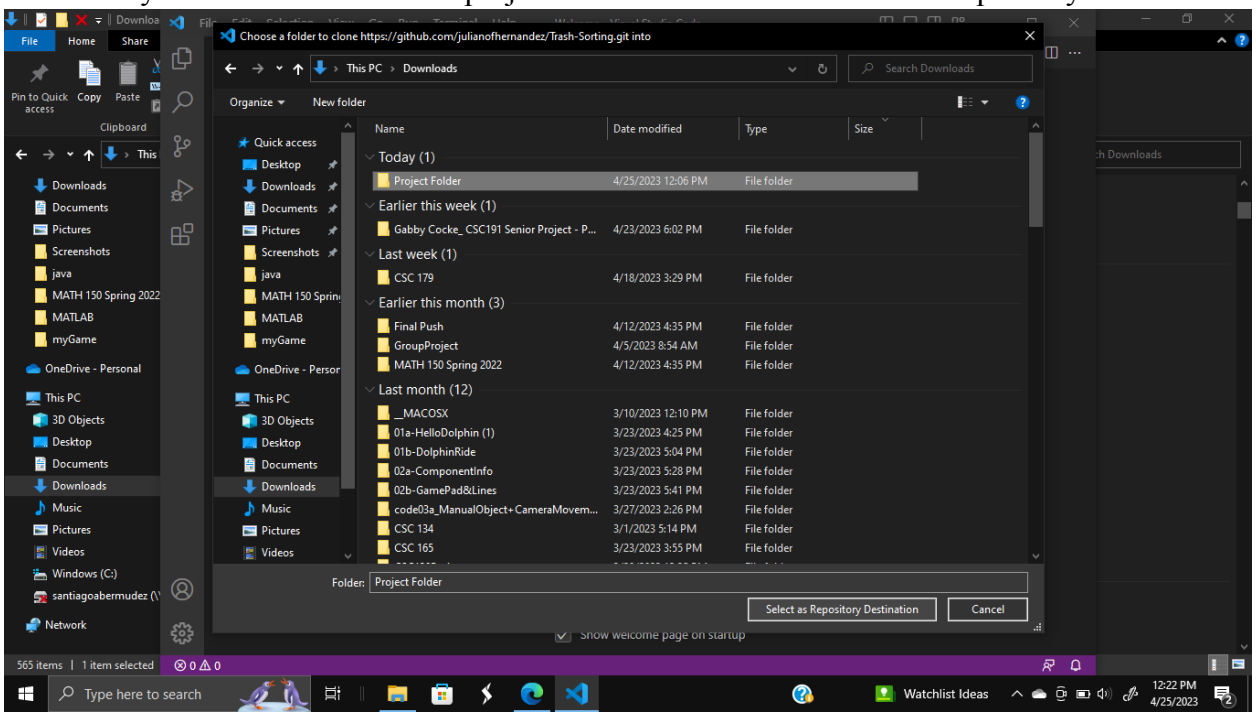




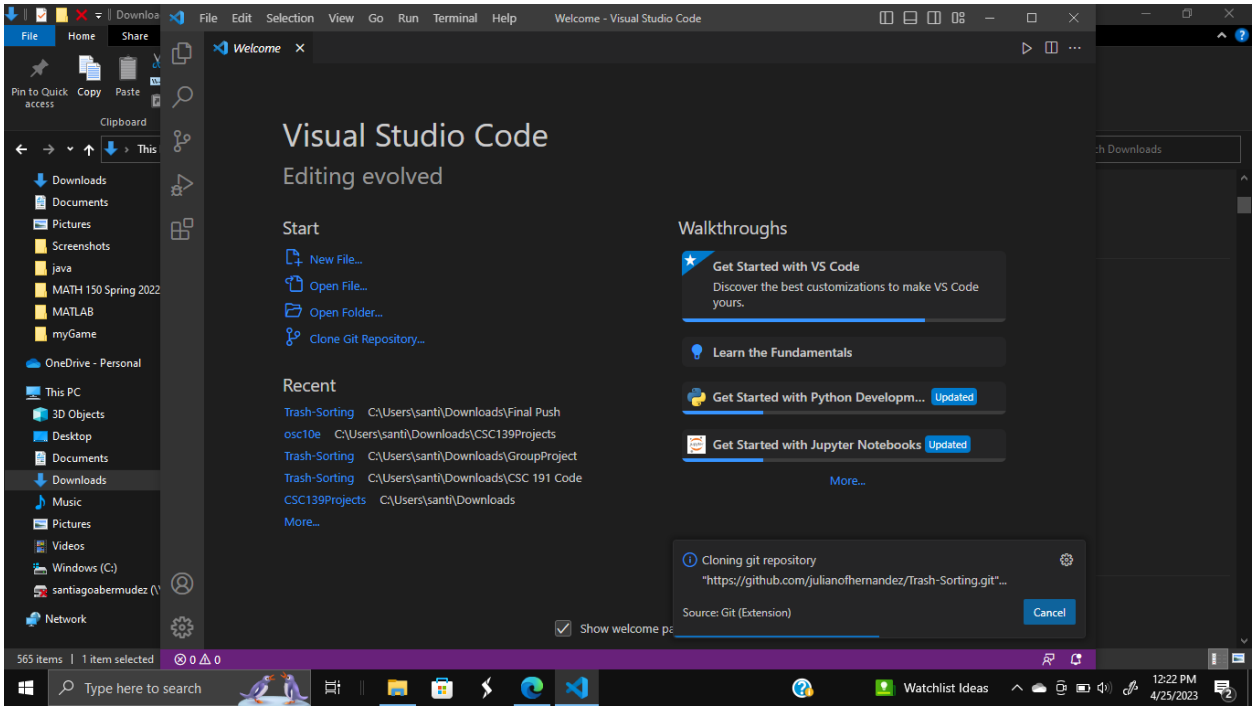
You will then paste the URL into the search bar and select the repository that gets listed below.



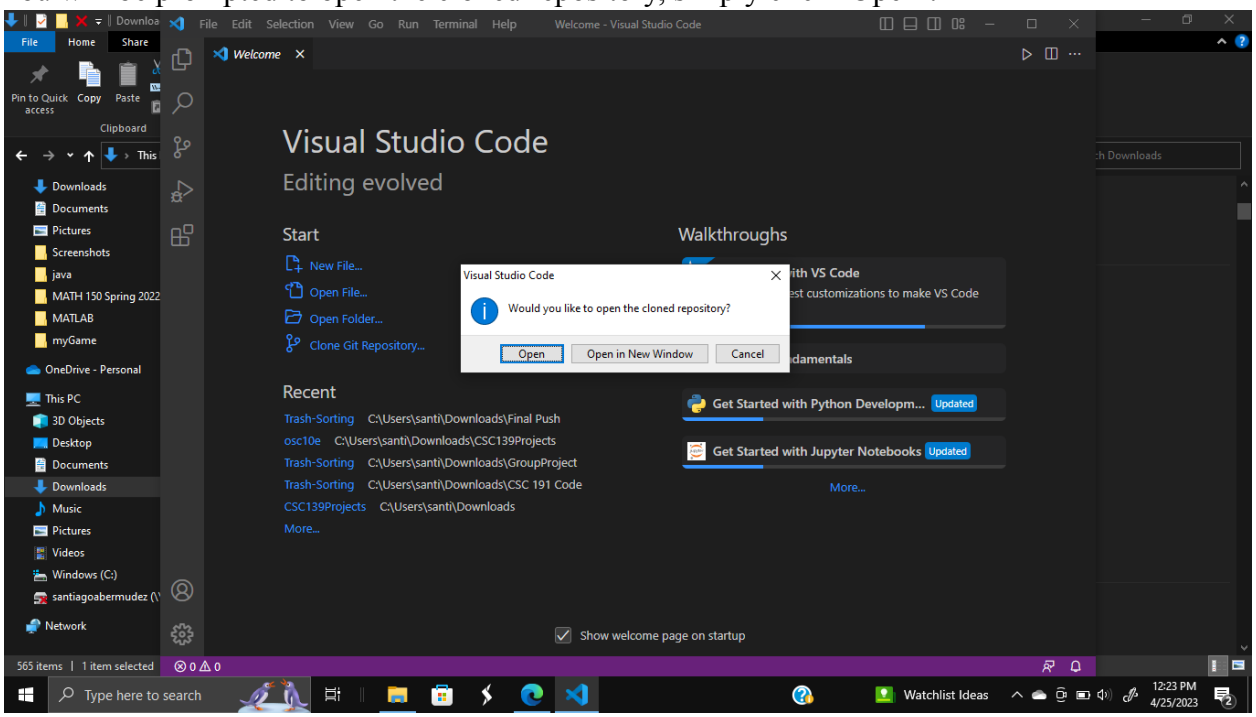
You will then be prompted to select a folder to store the project in. Navigate to and choose the folder that you have created for this project. You will then click Select as Repository Destination.



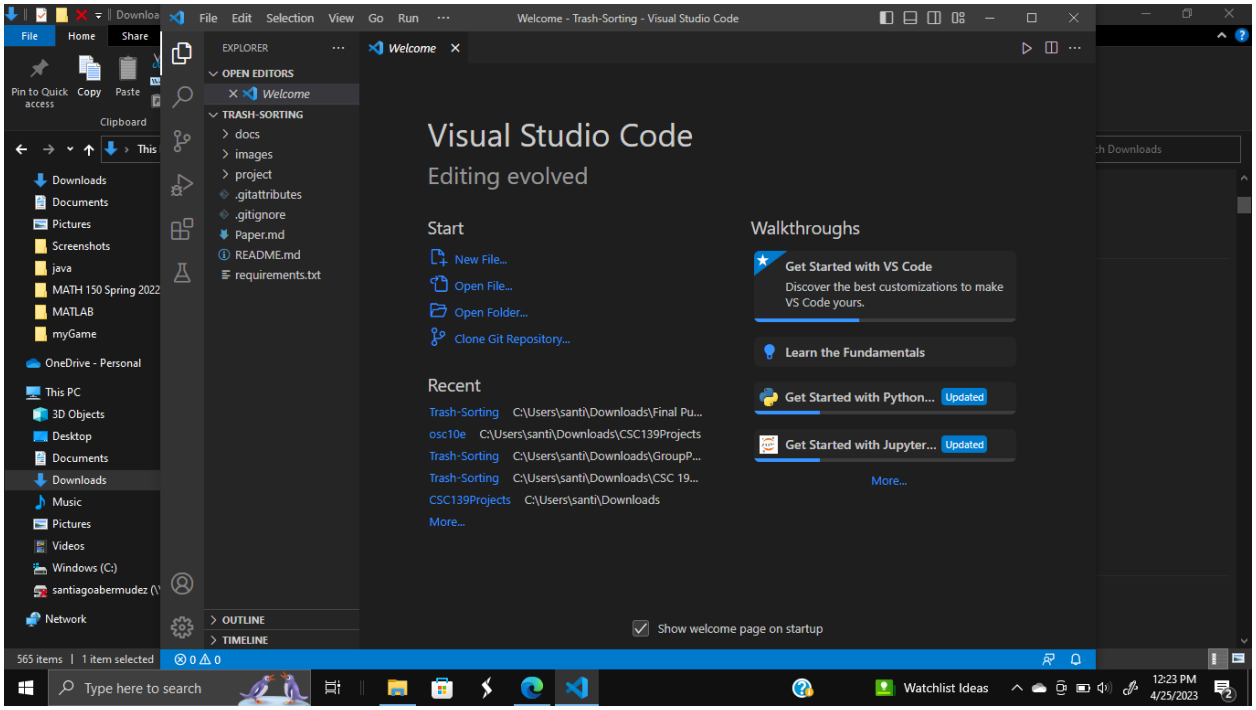
You will then see progress on the repository being cloned in the lower right.



You will be prompted to open the cloned repository, simply click ‘Open’.

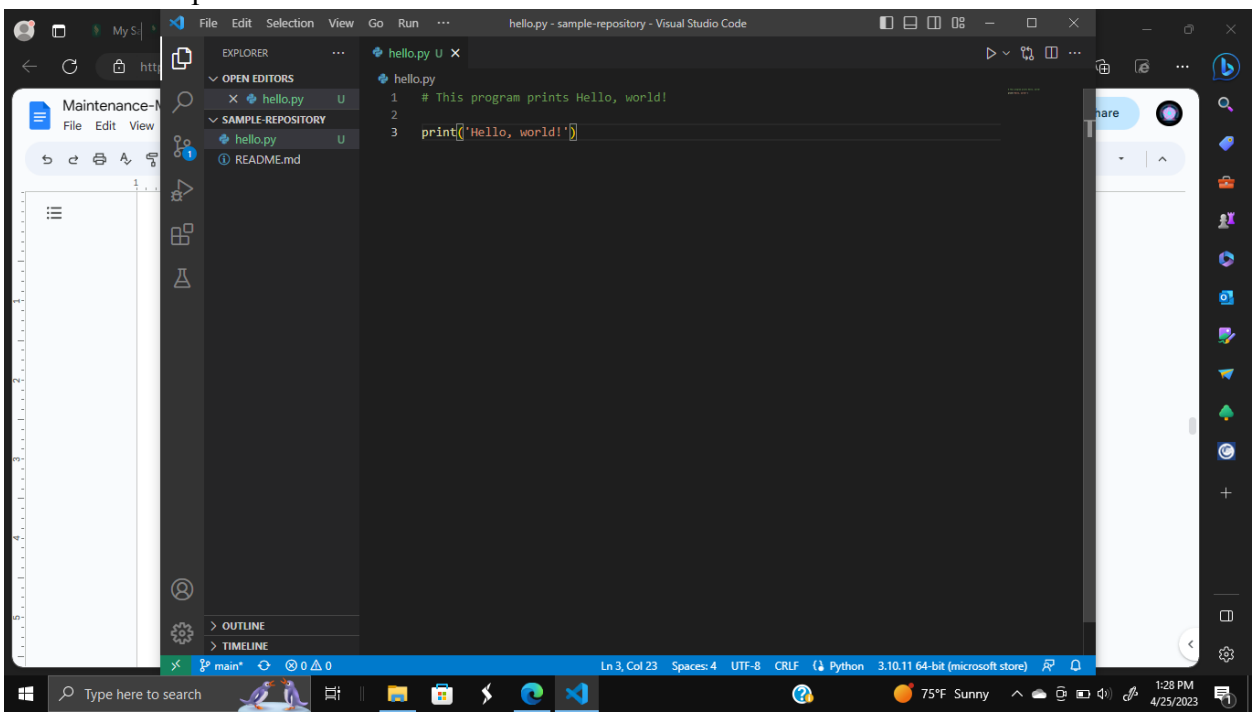


On the left, you should see some file links open up in the explorer navigation bar on the left. From there, you can navigate anywhere in the project.

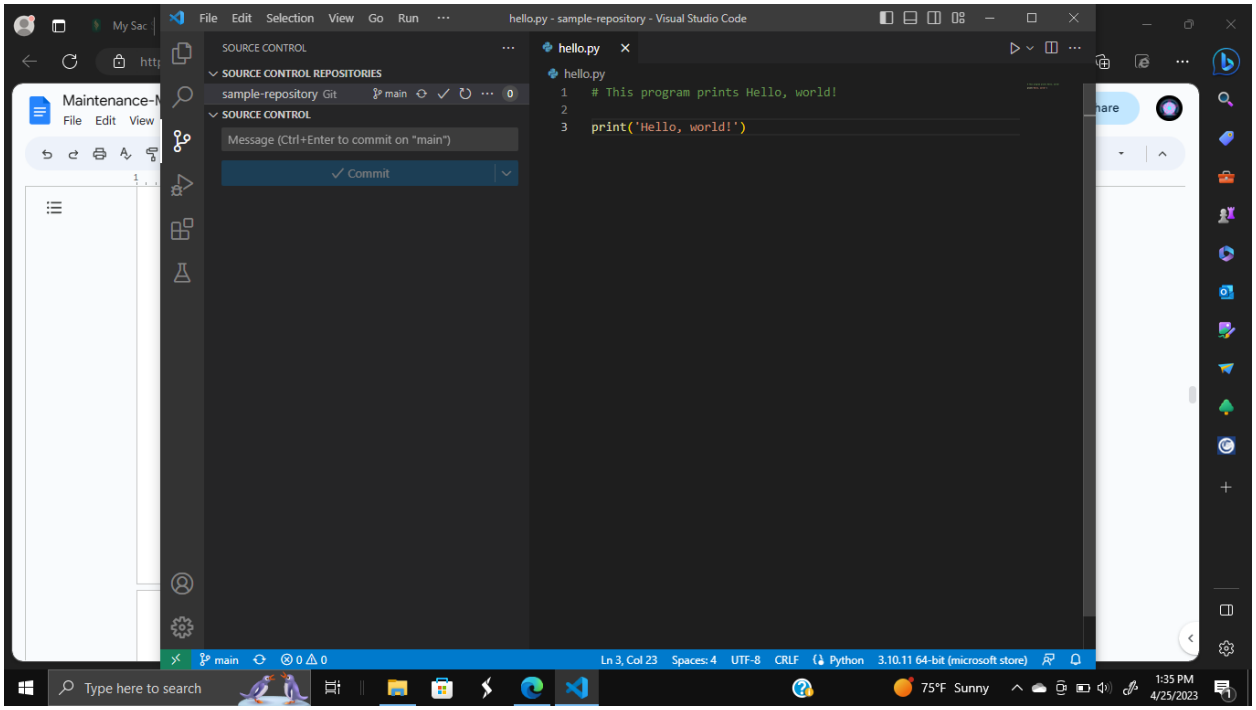


3.5.2 Pushing Changes to a Repository

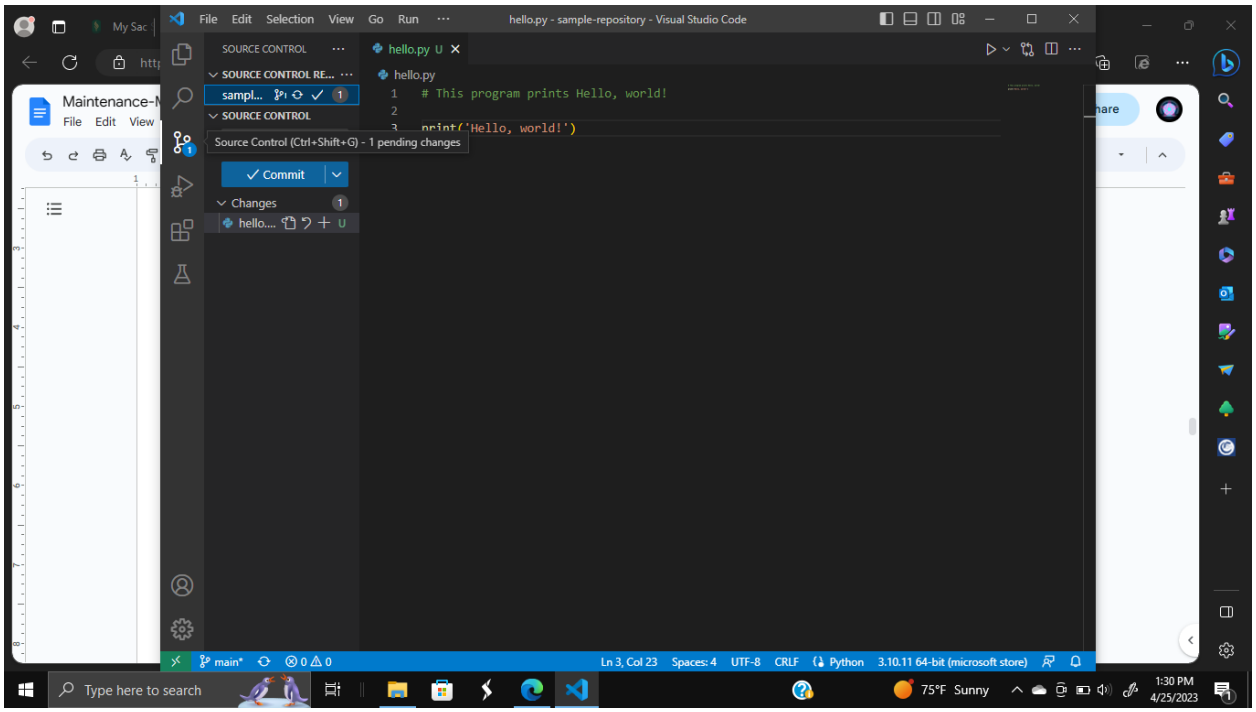
We have a sample code that we have created for use below.



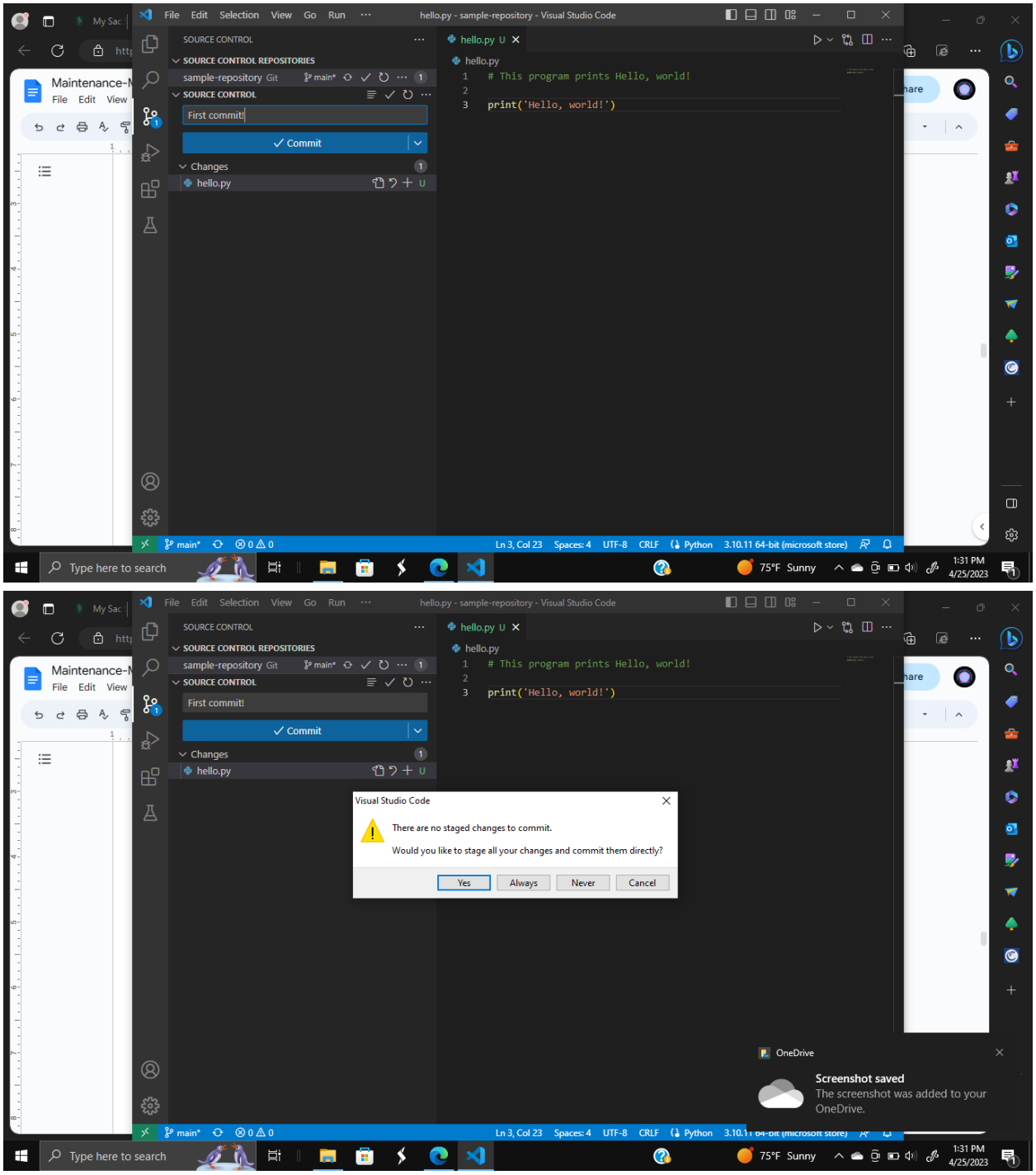
After selecting source control from the left navigational bar, one has the option to create a custom message for commits.



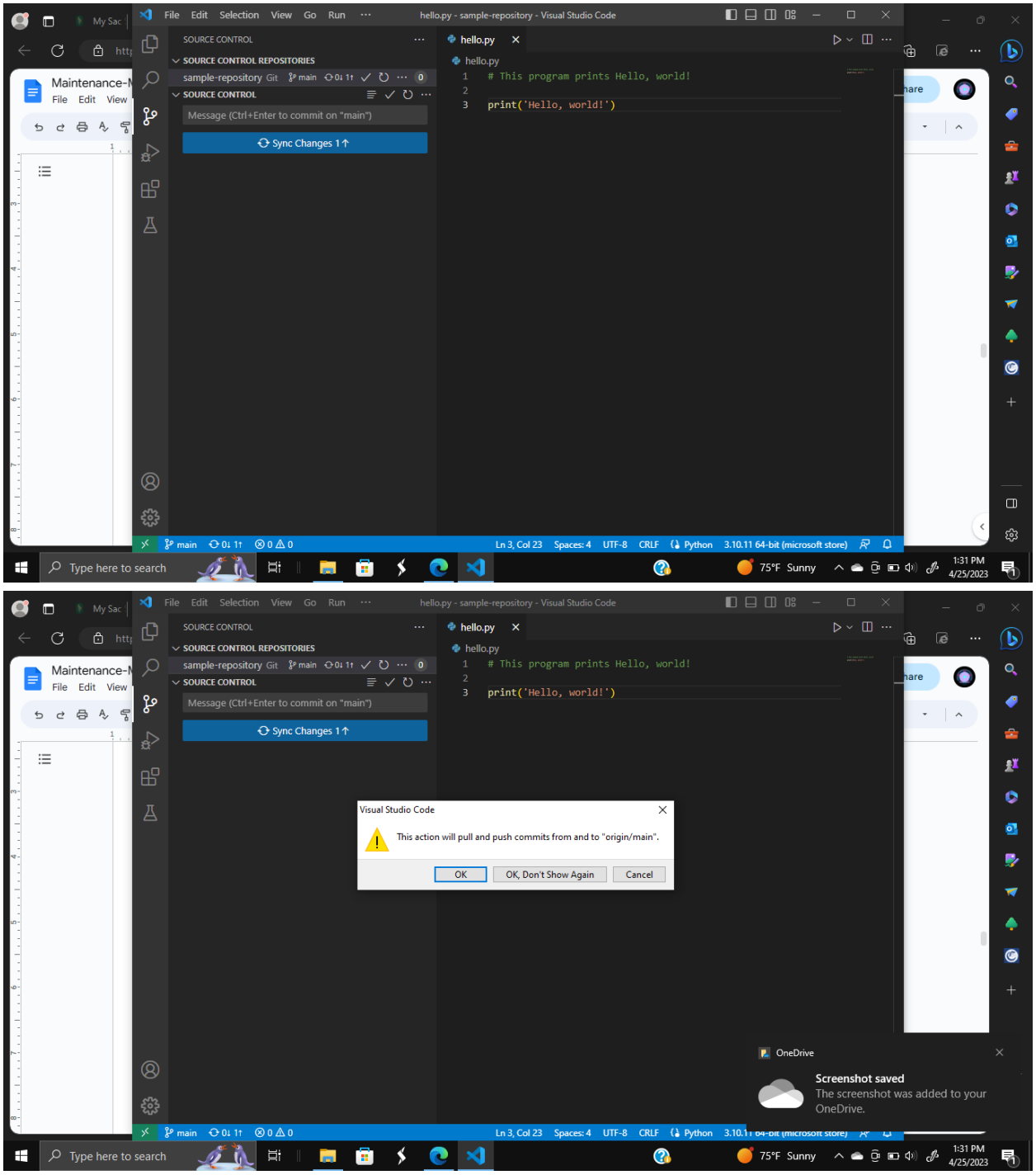
One can also see the pending changes that one has by hovering their mouse by the source control icon.



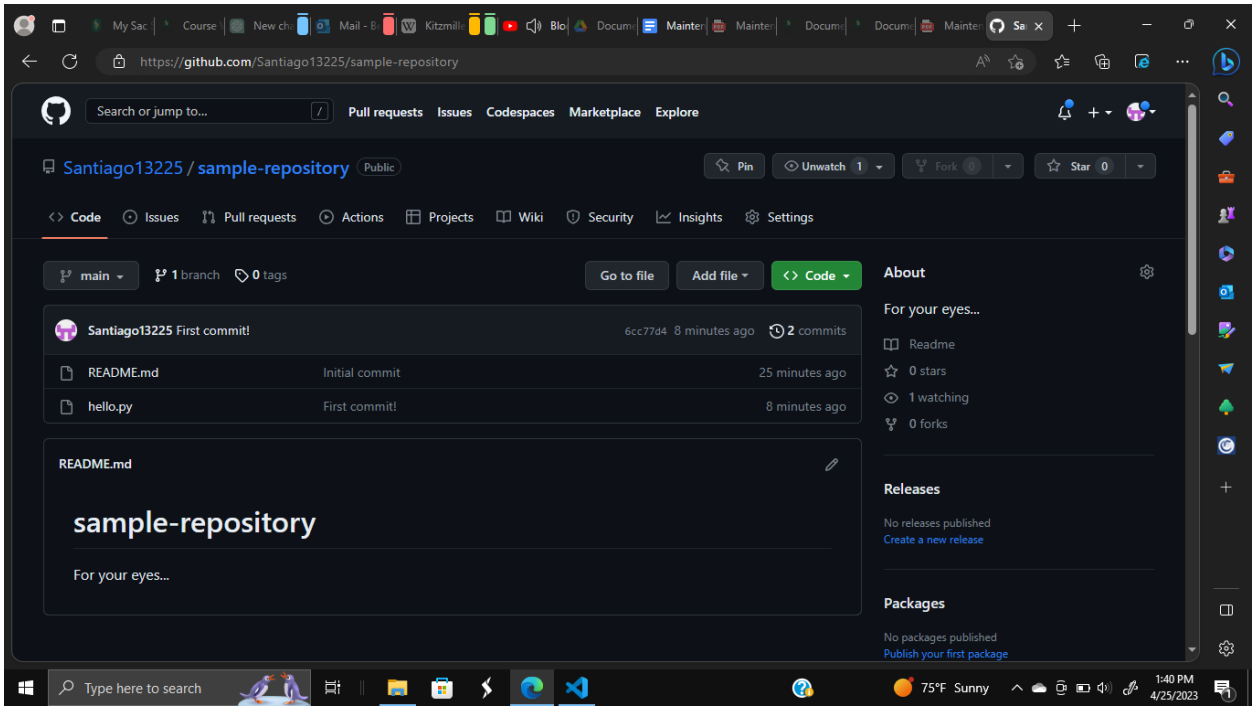
After creating a custom message and committing, one is prompted to stage their changes. You can simply click yes in this regard.

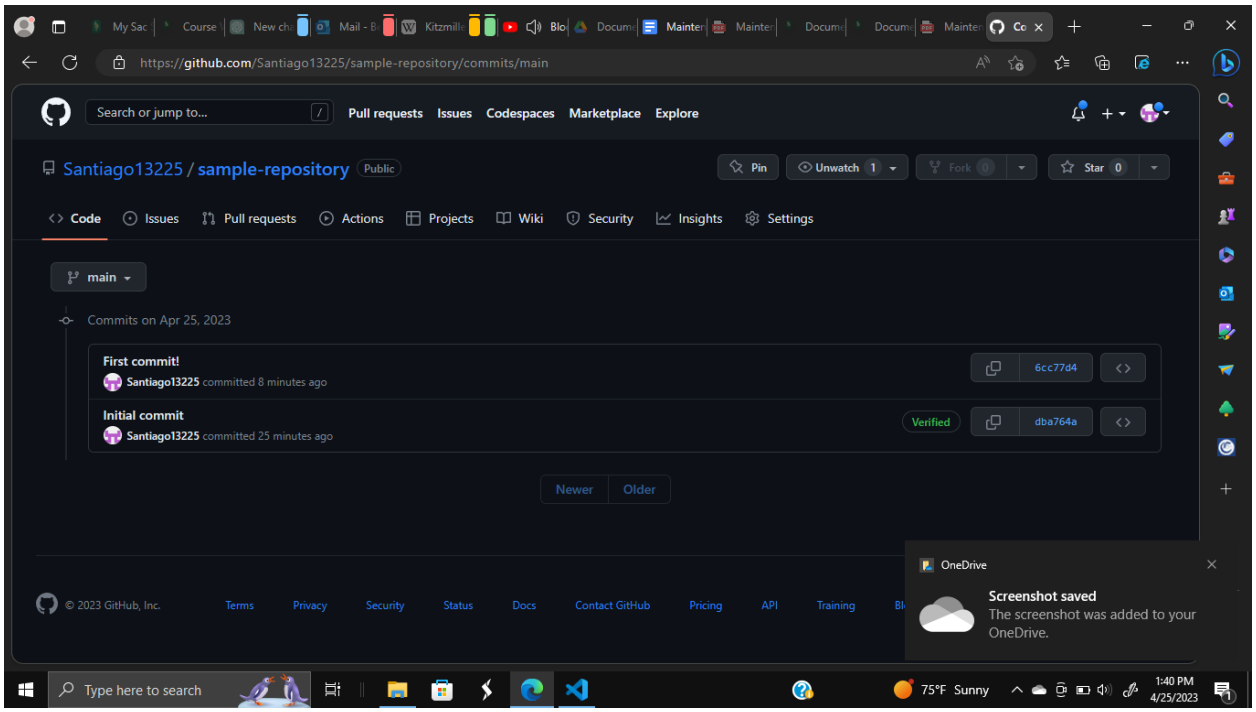


After, one has to sync their changes. Click on the Sync Changes button and click ok on the prompt that shows up.



After, one can see their changes by refreshing the GitHub repository and by checking the commit history.

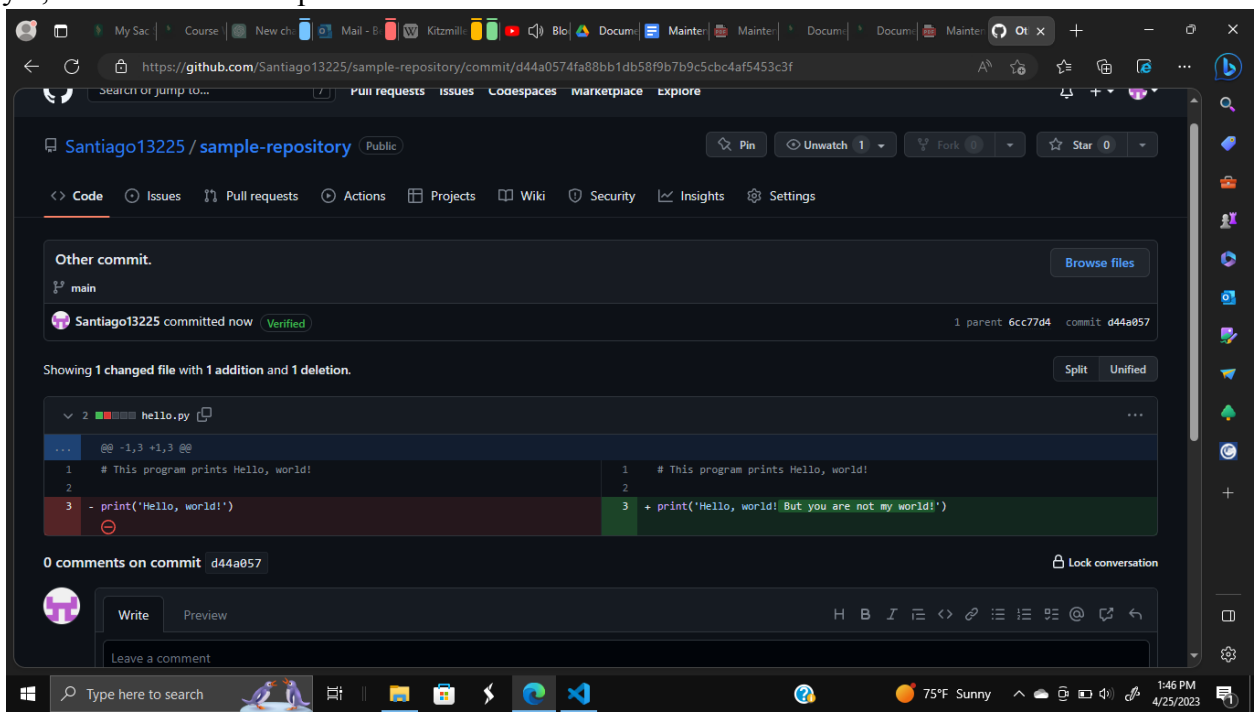




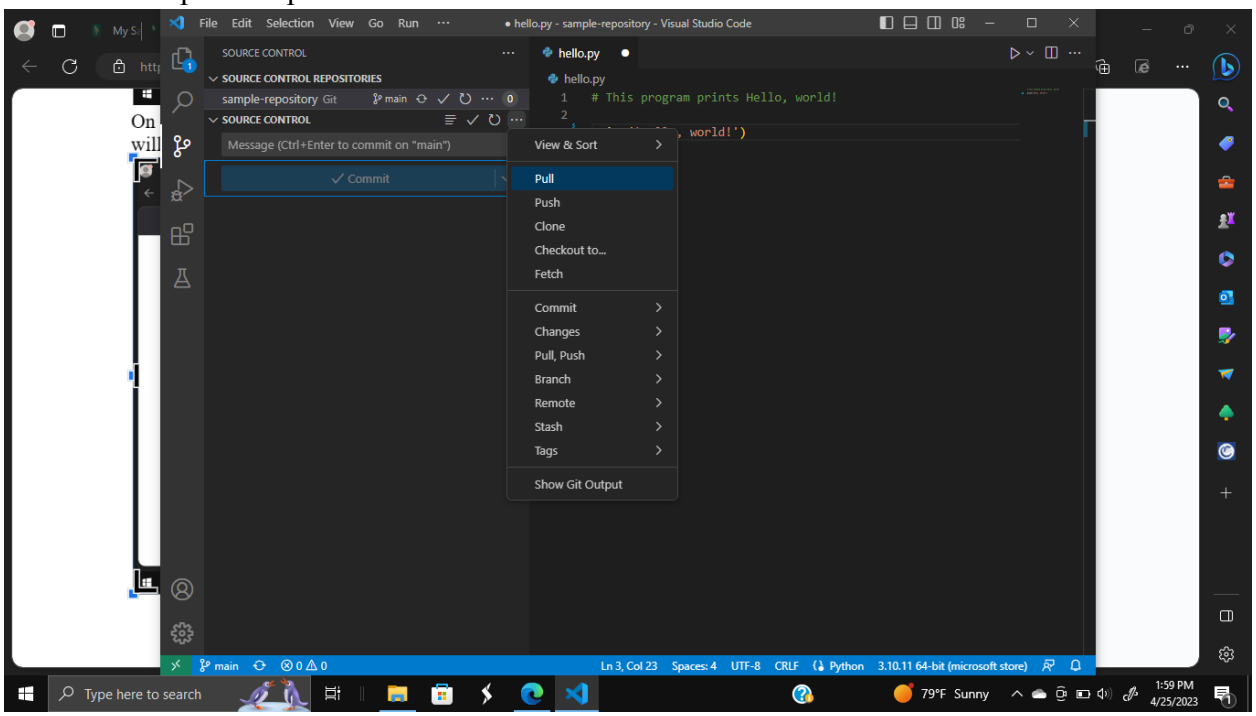
3.5.3 Checking for Conflicts before Pushing

Using the steps provided above, one can upload new code and changes to the current code by using commits. It is important to always ensure that one has the latest version of the repository before committing their changes, otherwise there will be conflicts that prevent one from committing their changes properly.

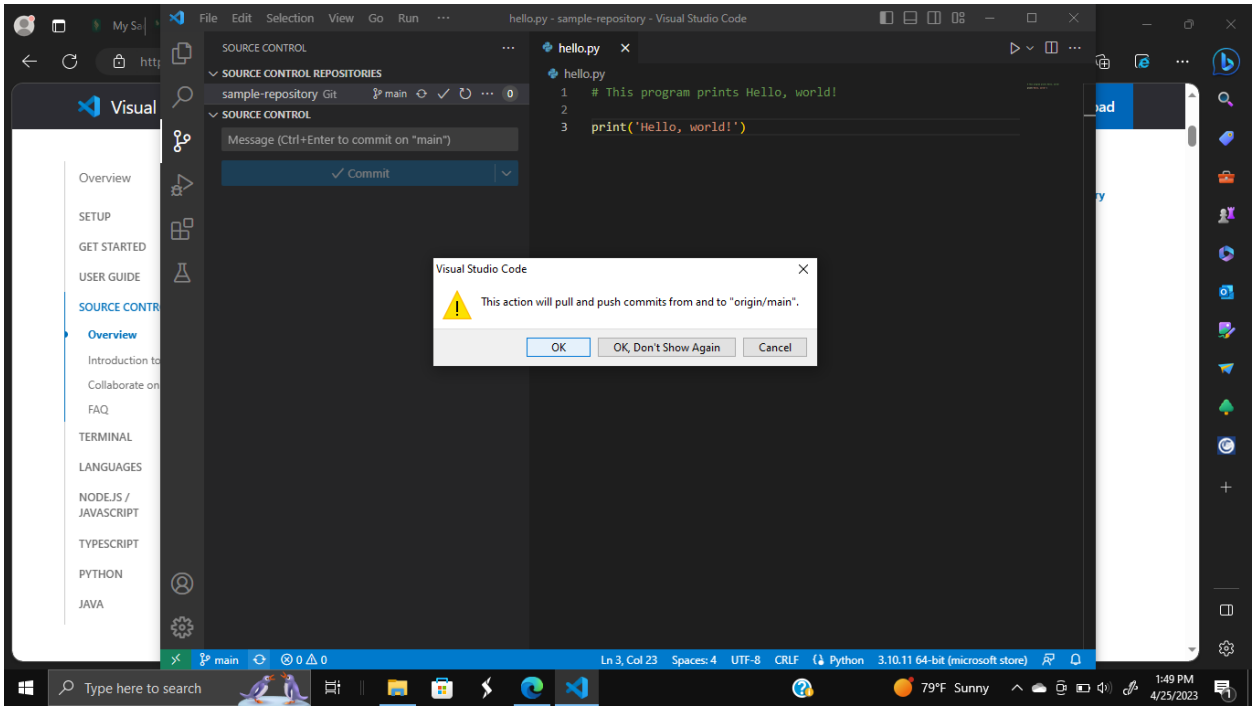
Below is a commit made by someone else. We do not have those changes in our VS Code files yet, so we will have to pull them.



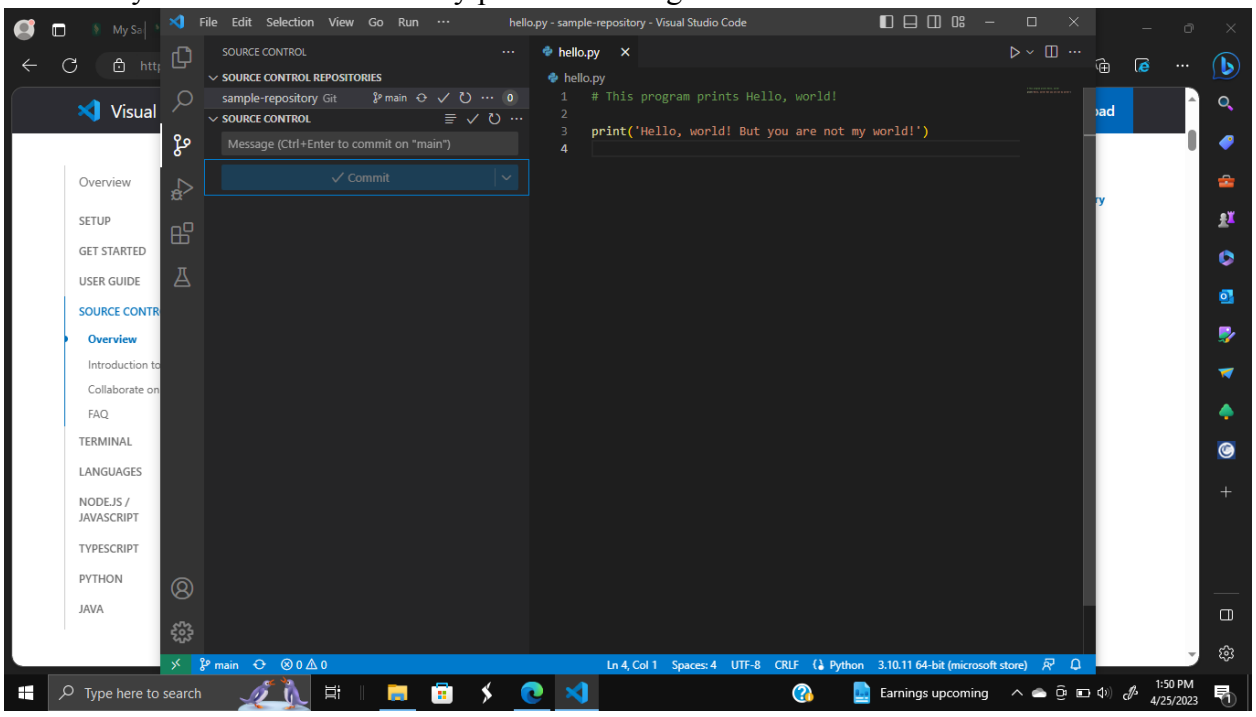
On VS Code in Source Control, there is a three-dot button on the right. After clicking that, you will have an option to pull.



Once you click pull, you will be prompted. Simply click ok.



After that you will have successfully pulled all changes.



You can now safely commit changes if you have made any. To be safe, you may want to save your modified code elsewhere before pulling changes.

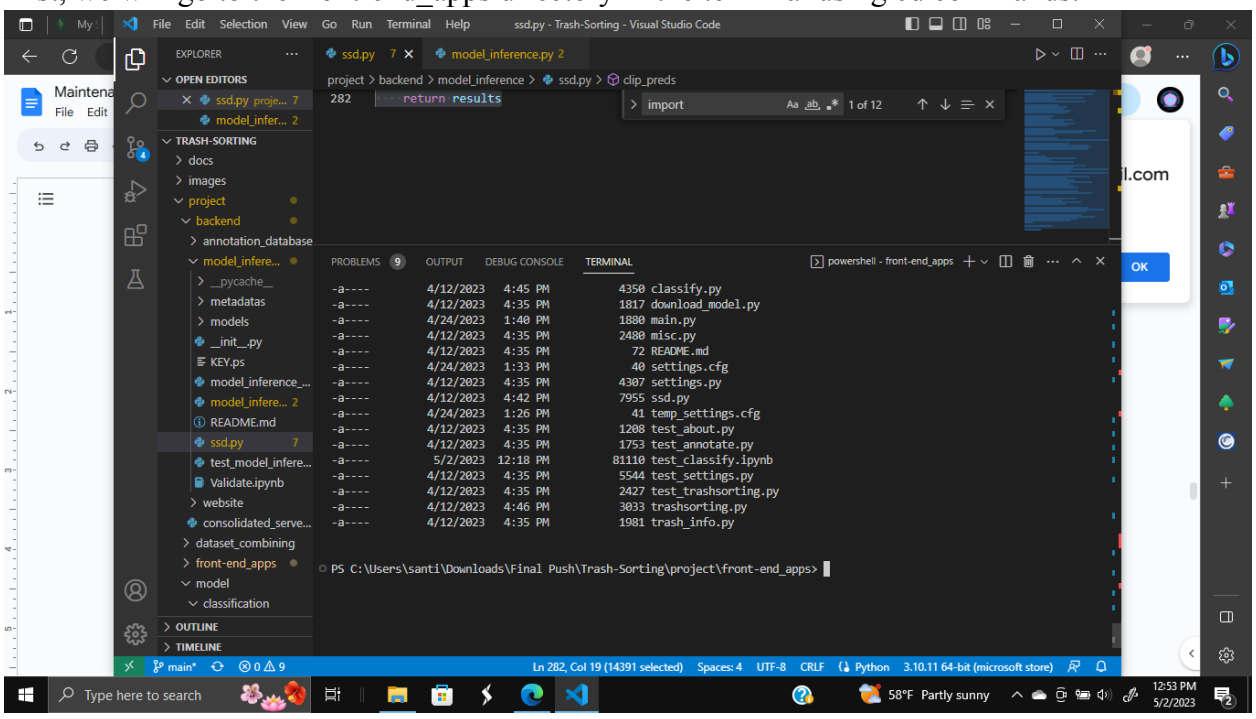
3.6 How to deploy releases

Deploying releases is subjective to the Client and future engineers. Nonetheless, when deploying releases we recommend making sure the connection between the backend, and the three front-ends is still working. Also, run the tests to verify the release before actually deploying it.

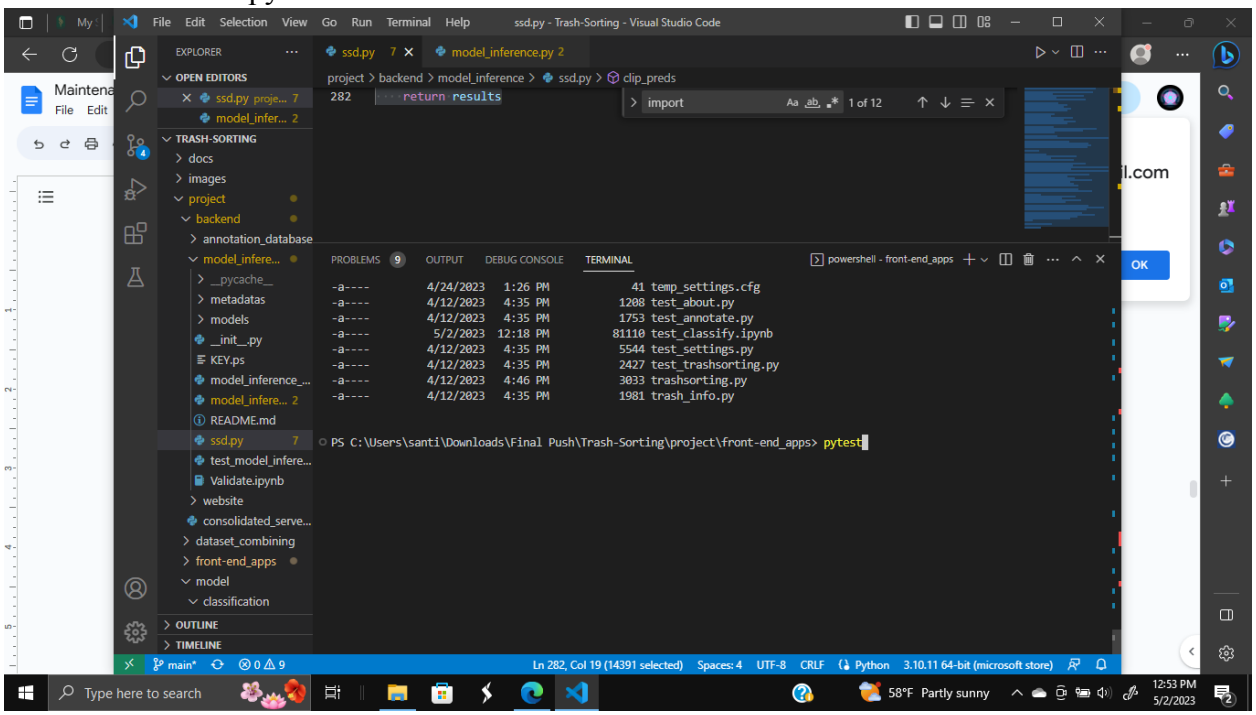
3.7 How to run automated tests

Running the pytest command in the front-end_apps folder will automatically run tests on all terminal based front end features.

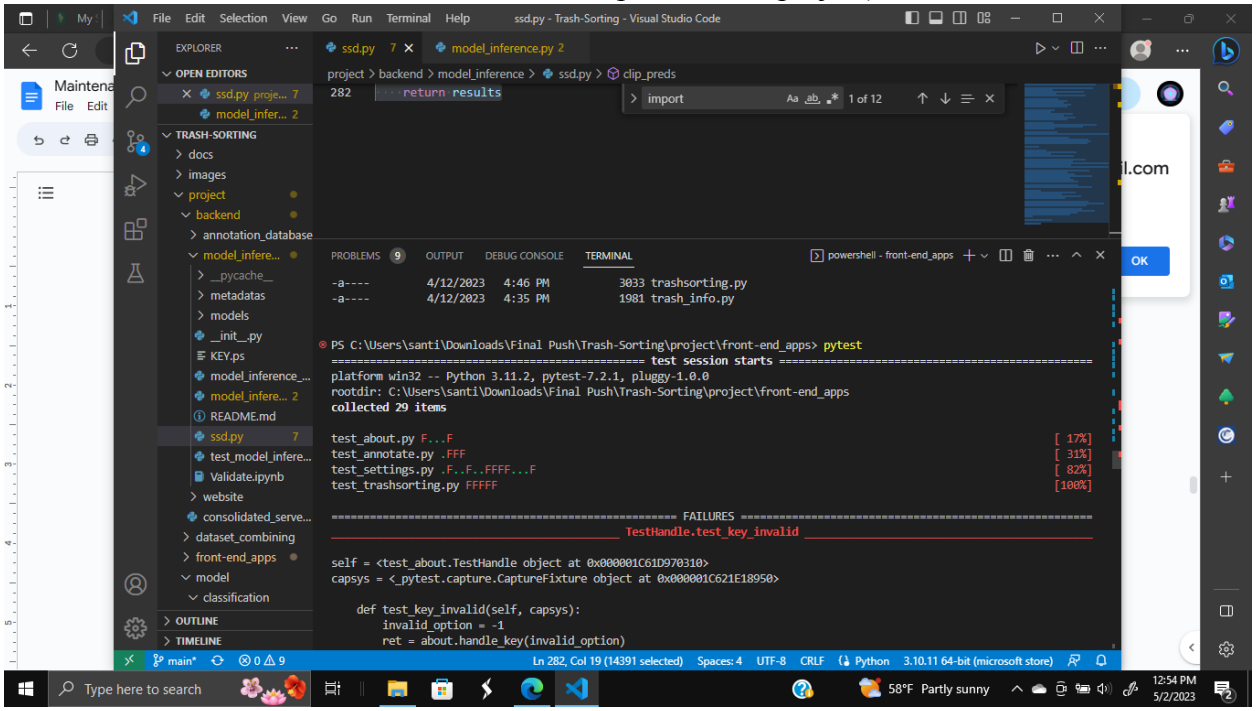
First, we will go to the front-end_apps directory in the terminal using cd commands.

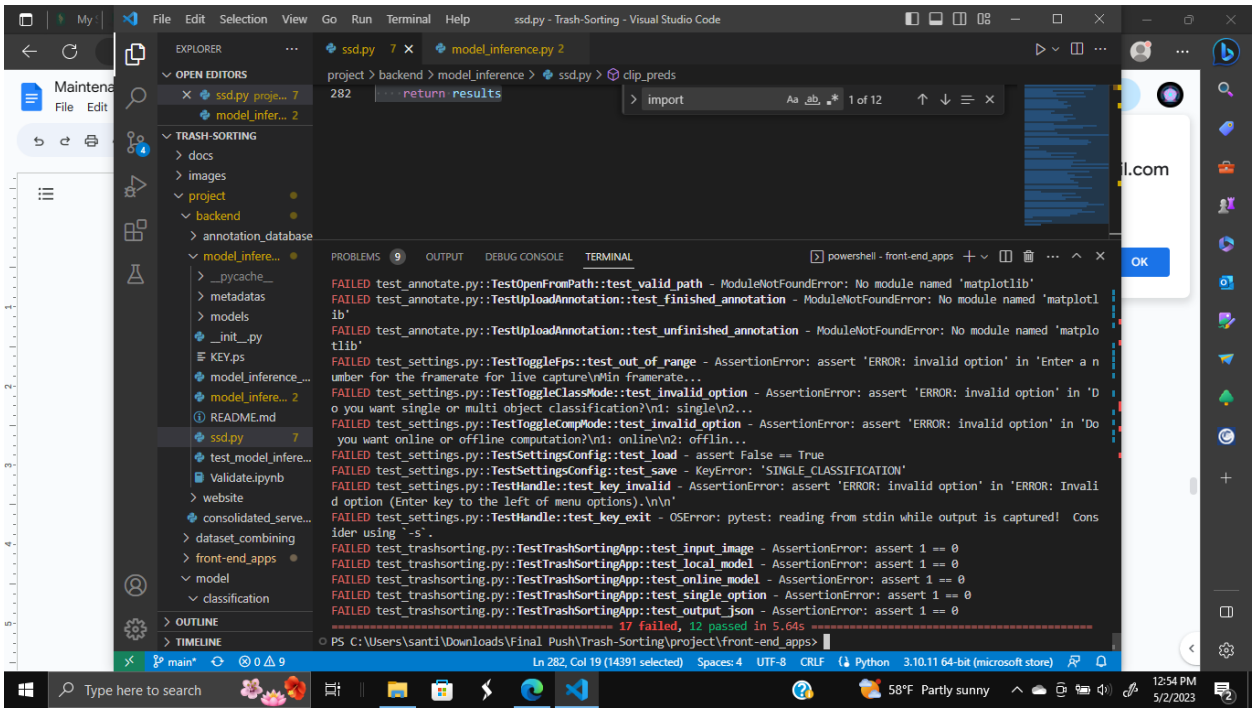


Then we enter the 'pytest' command to run the tests.



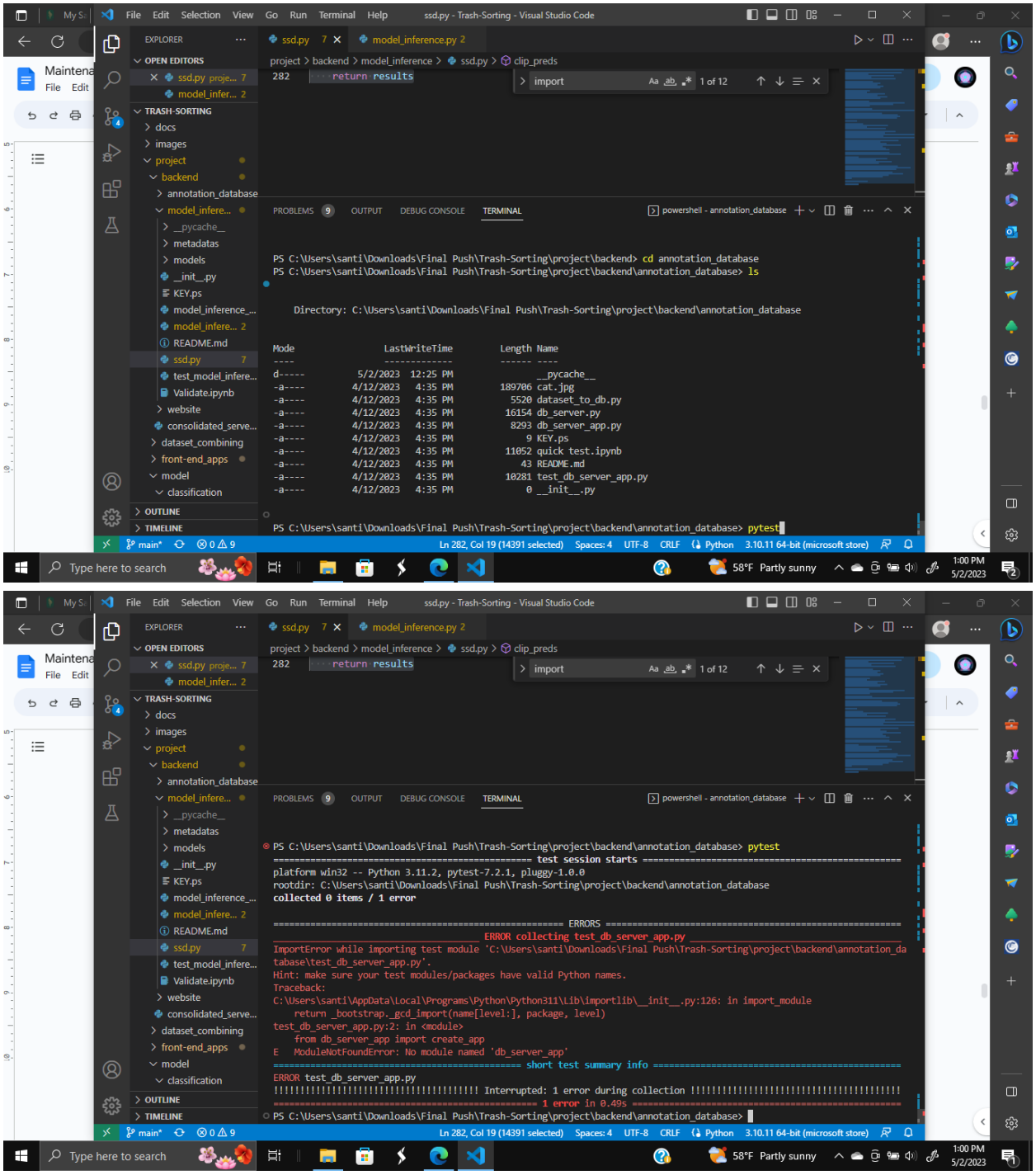
After running ‘pytest’ you should expect to see something like the results below (*please note that the results shown below are from an earlier phase in the project).



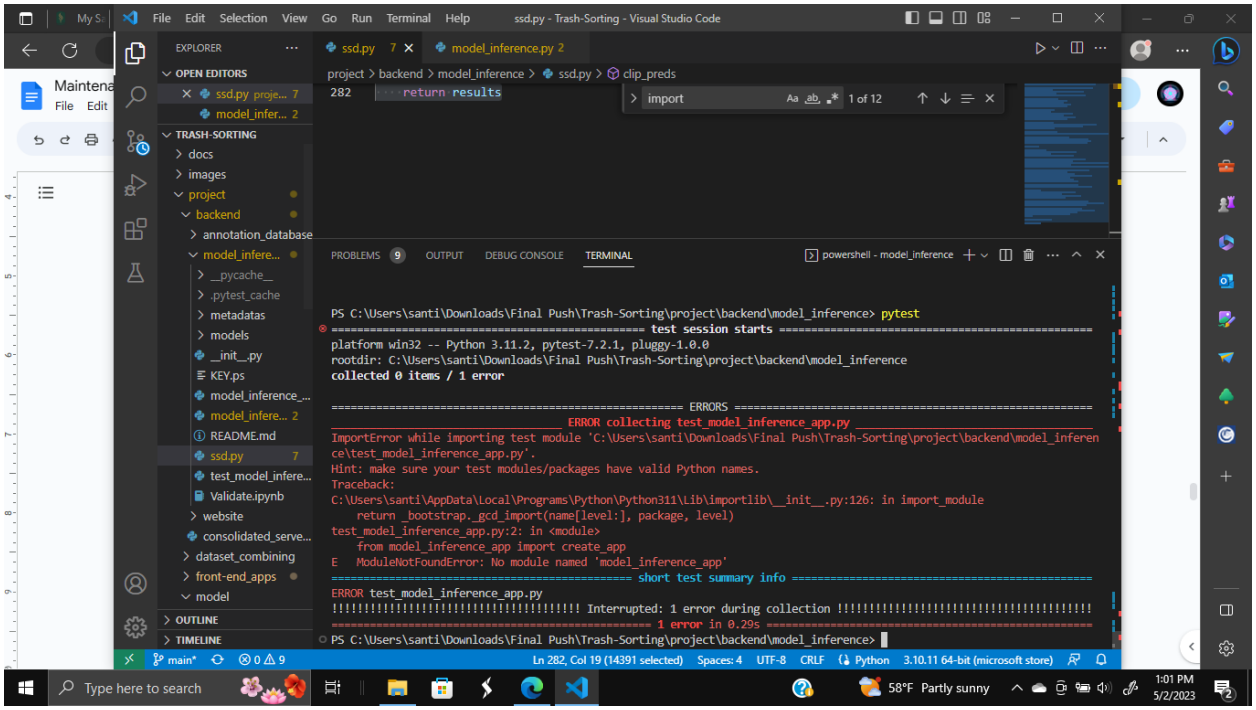


a. Running the pytest command in the backend model_inference and annotation_database folders will test all of the backend server functionality.

Again, from the steps shown in part a, you can navigate to the annotation_database directory and run the ‘pytest’ command to test things in that directory.

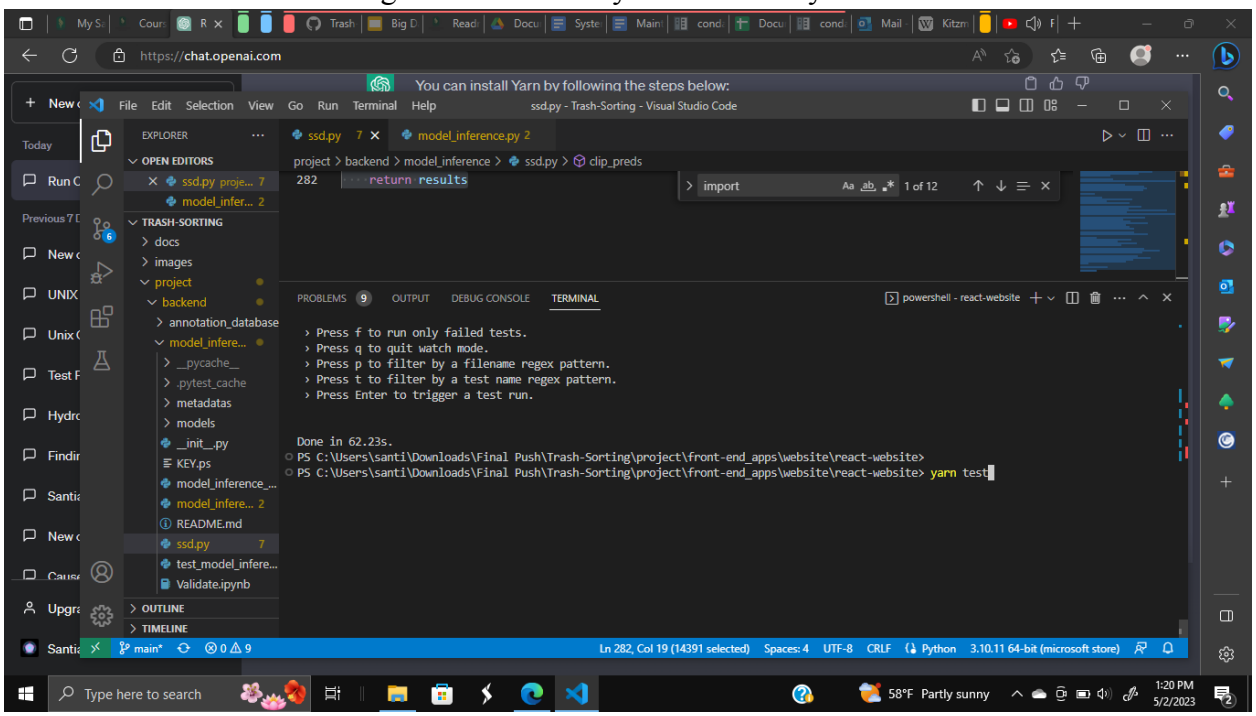


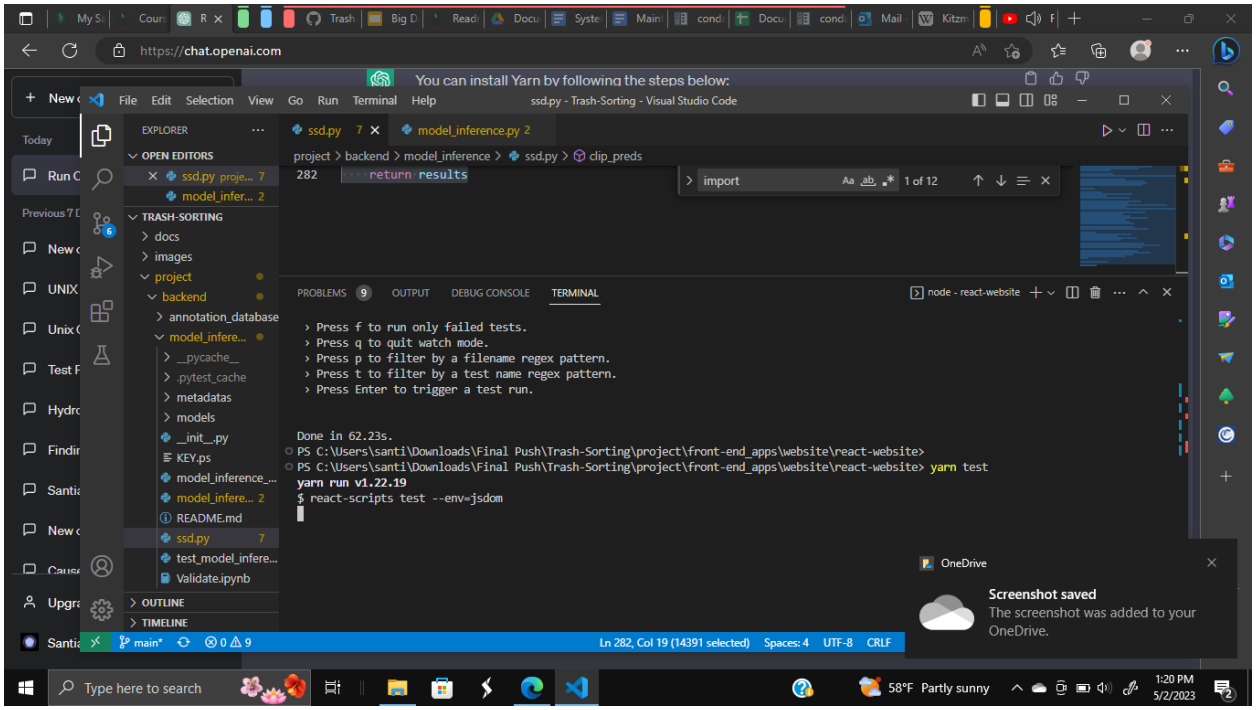
You can likewise do the same for the model_inference directory.

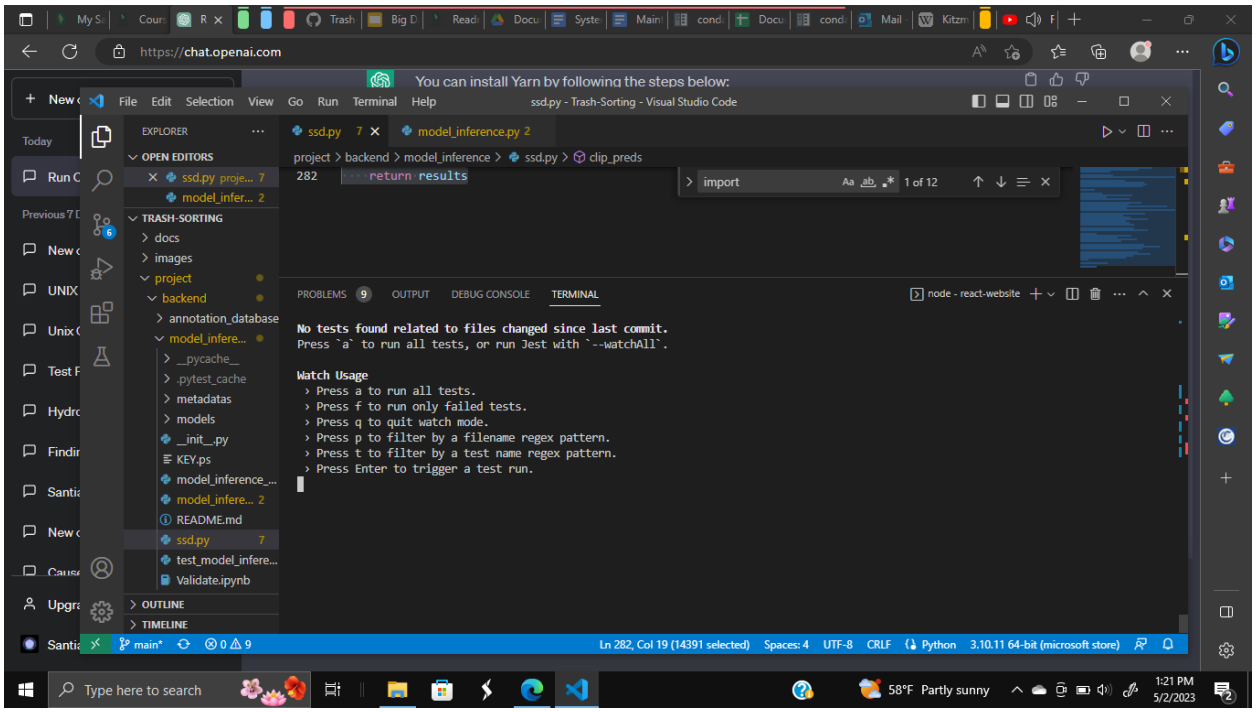


b. Go to project\front-end_apps\website\react-website and run `yarn test` then press `a` for all, this will test the website.

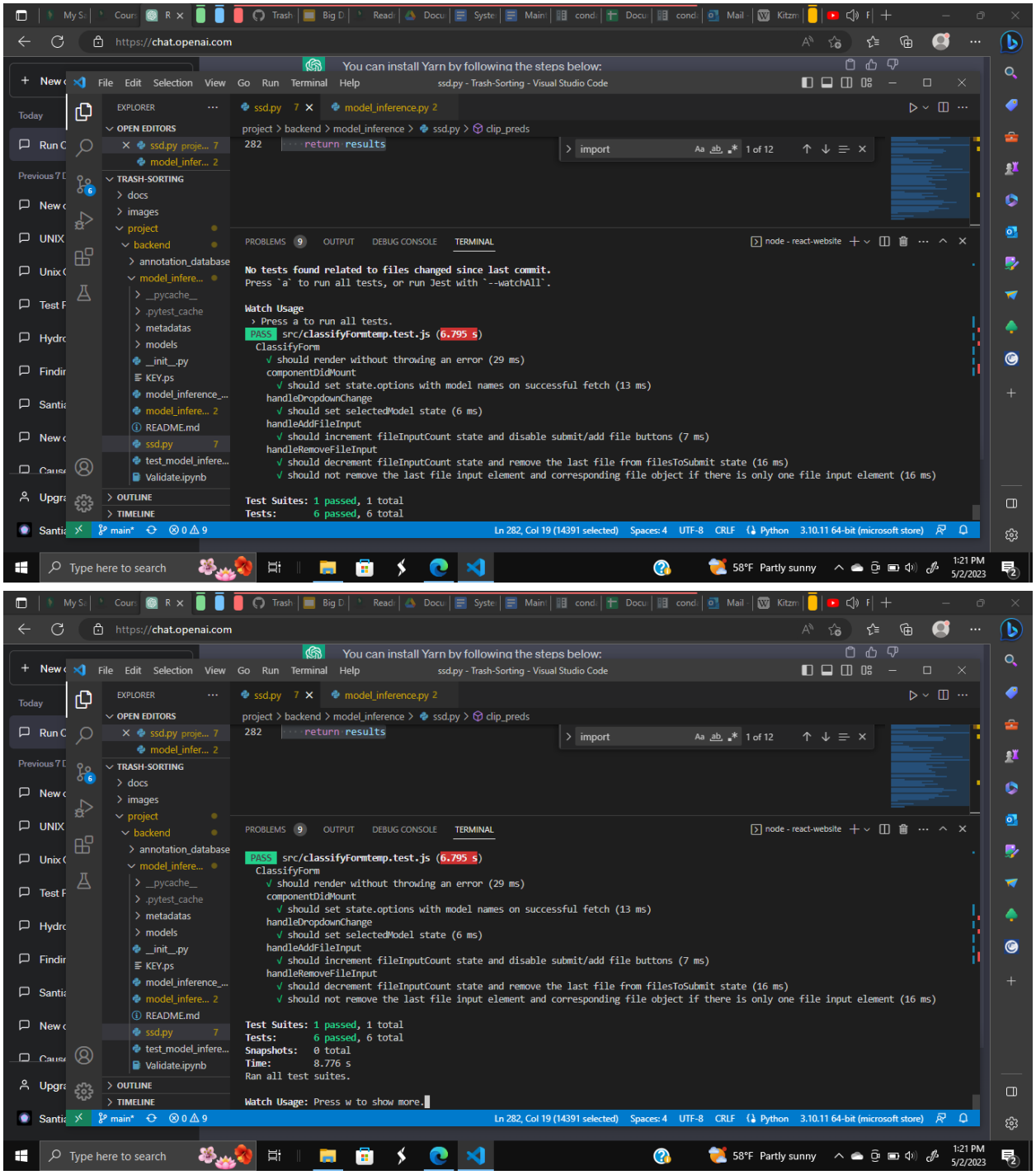
You can use cd commands to go to said directory then run the ‘yarn test’







Below are the results you can get from running all tests.



3.8 How to do “sanity” check to ensure code changes didn’t break anything

Running the respective tests will produce most possible errors and issues with their locations that may have occurred upon a code change. Furthermore, we recommend manually testing the main feature(s) changed.

4. Deployment

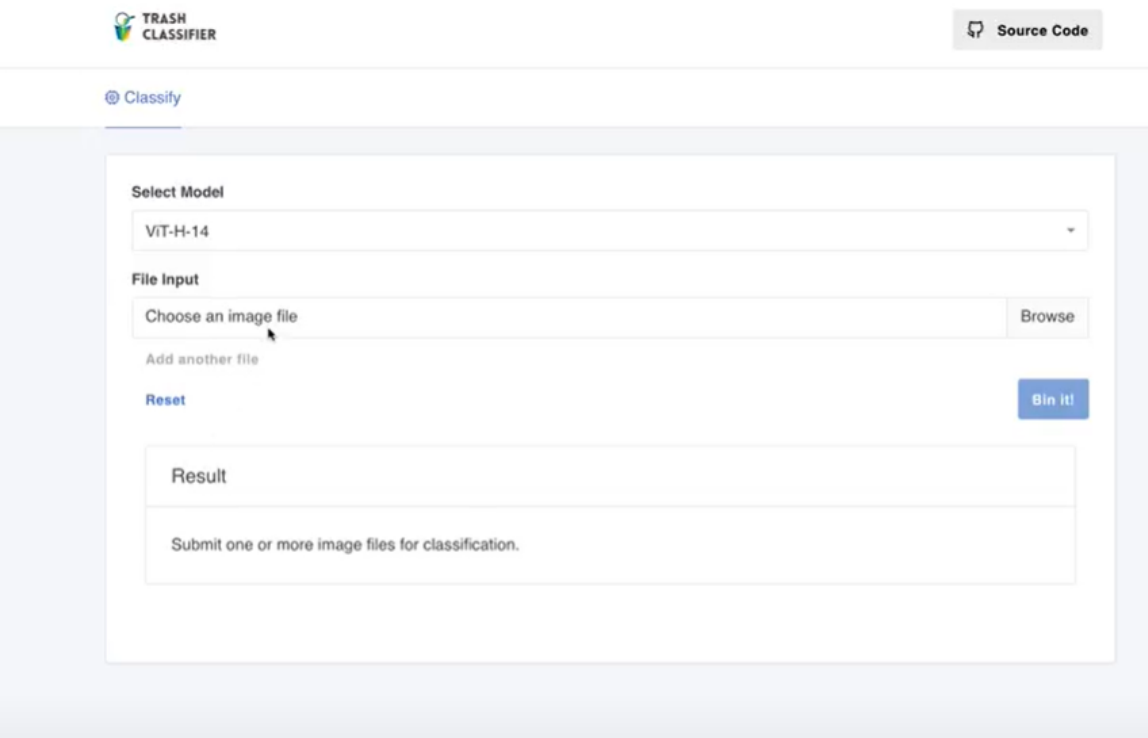
Deploying releases is subjective to the Client and future engineers. Nonetheless, when deploying releases we recommend making sure the connection between the backend, and the three front-ends is still working. Also, run the tests to verify the release before actually deploying it.

At the time of project completion, the scope consisted of getting the server to run on a local machine, such as the Data Science Club Server. This has both the website for model inference, as well as a REST API that can handle, inference, training, retrieving and downloading models, and annotating new images.

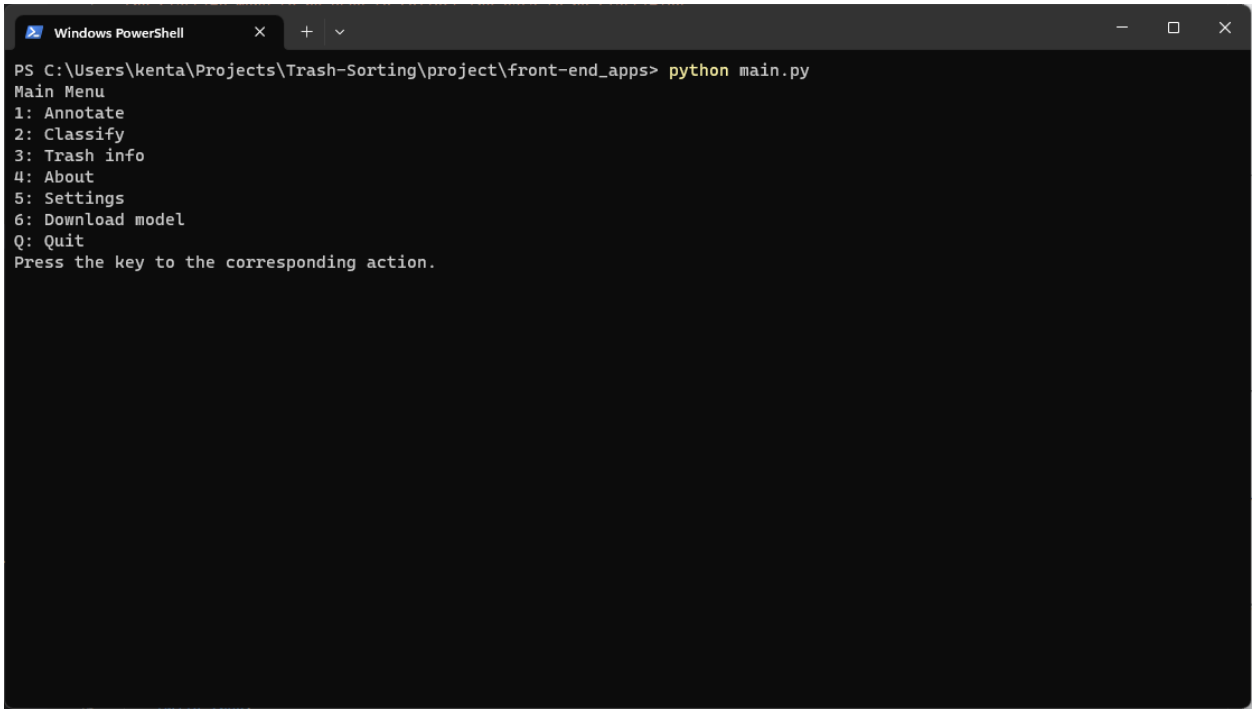
If choosing to deploy remotely, some popular hosting providers to look into are AWS and Heroku. Note that the root route serves the website which is simply a single page app.

5. User interaction

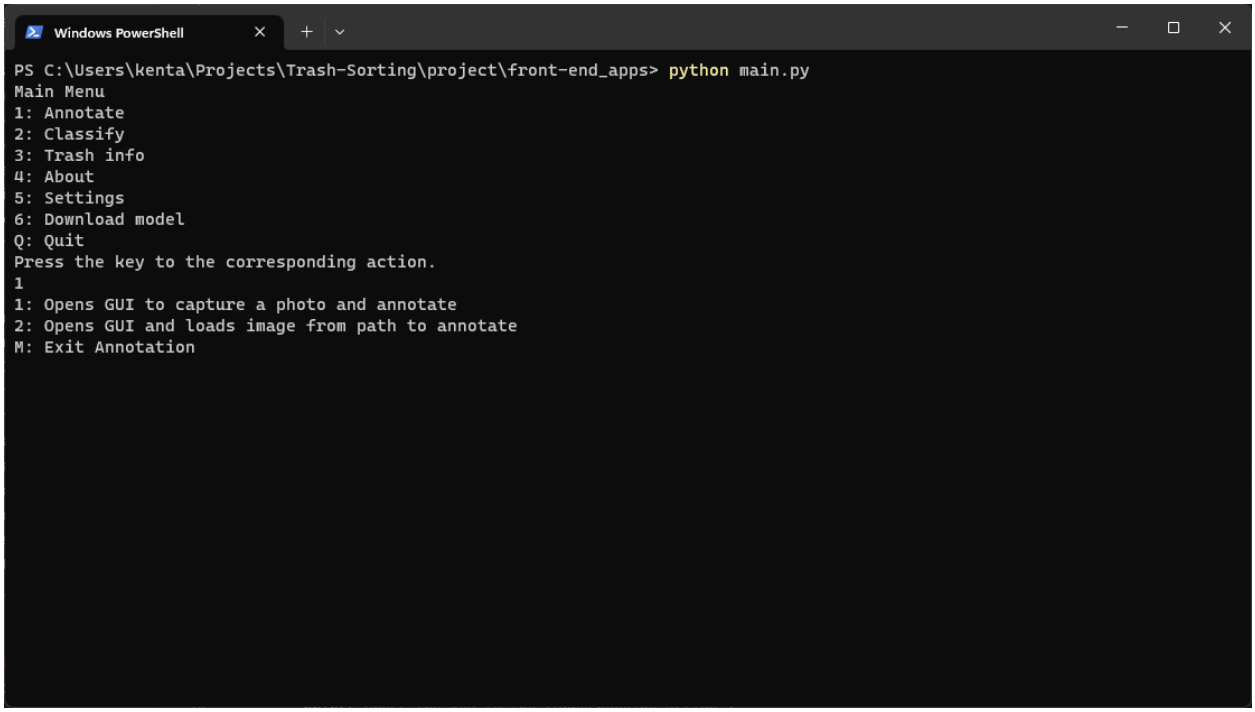
1. React Js Website (User view)
    a. Code files for the screenshot
        i. Trash-Sorting\project\front-end_apps\website\react-website\src**HomePage.react.jsclassifyForm.js**
        ii. Trash-Sorting\project\front-end_apps\website\react-website\src**SiteWrapper.react.js**
        iii. Trash-Sorting\project\front-end_apps\website\react-website\src**classifyForm.js**



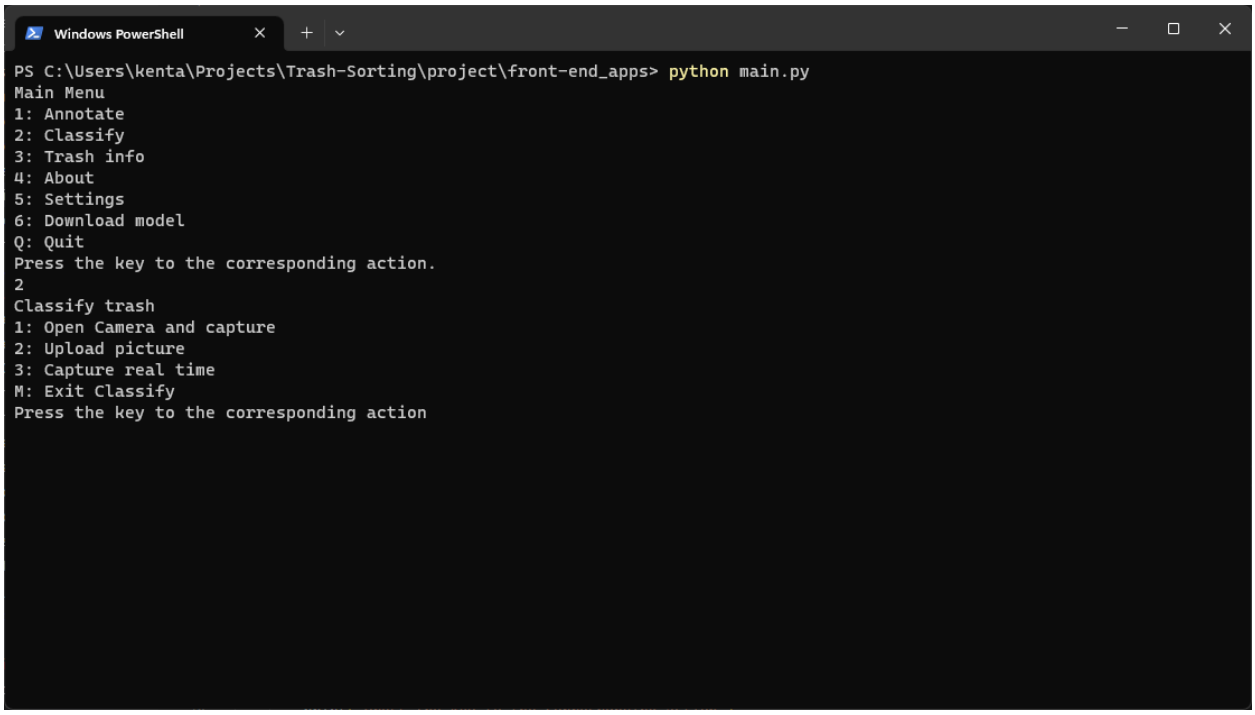
2. Command line interface (User view)
    a. Main Menu
        i. Trash-Sorting\project\front-end_apps**main.py**



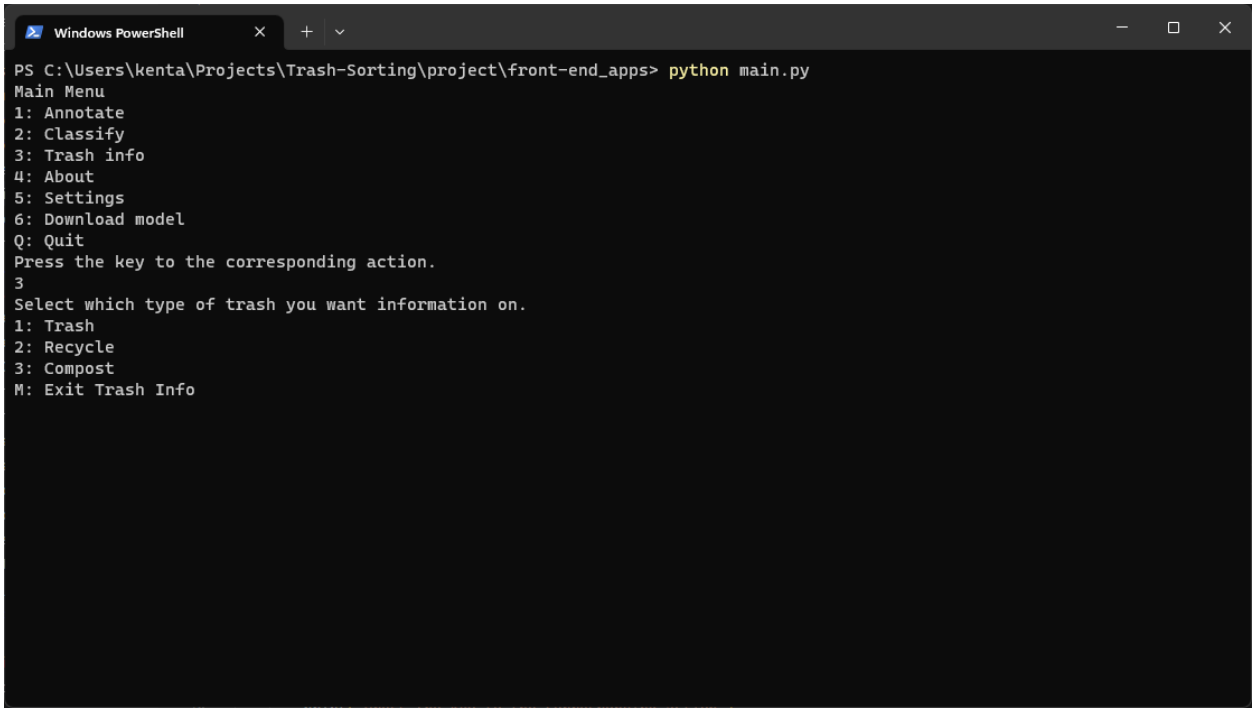
    b. Annotate
        i. Trash-Sorting\project\front-end_apps**annotate.py**



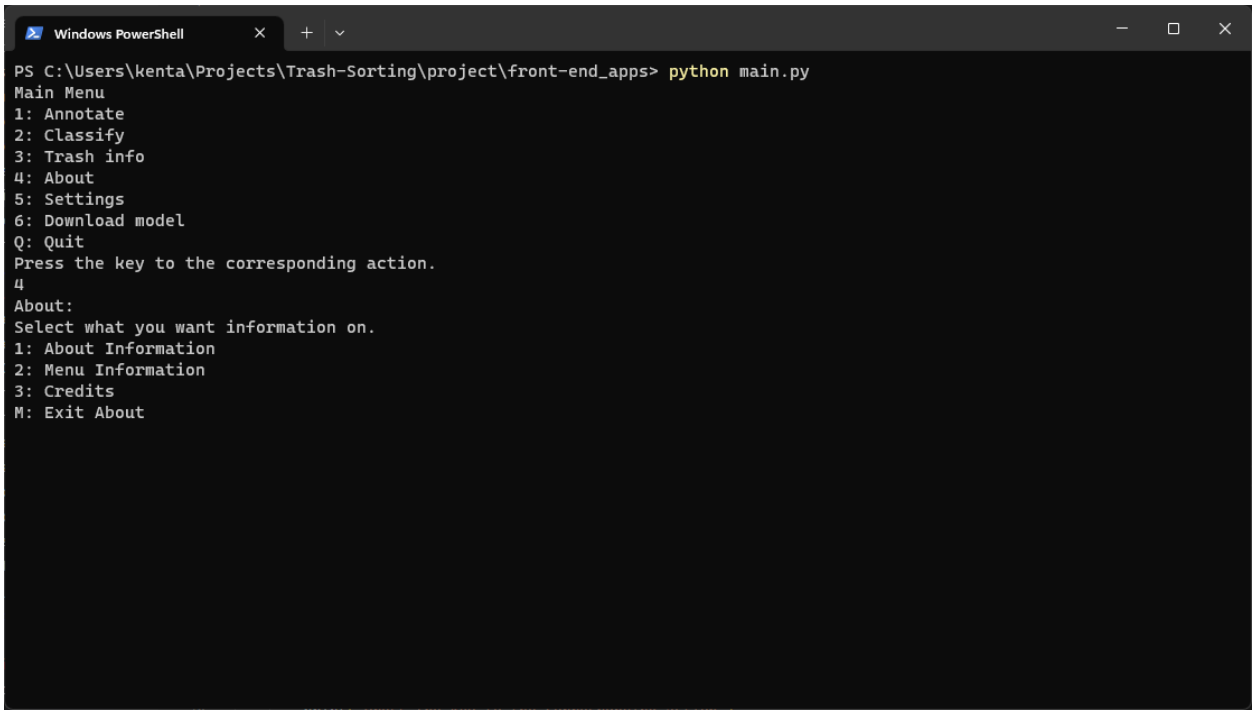
c. Classify
    i. Trash-Sorting\project\front-end_apps**classify.py**



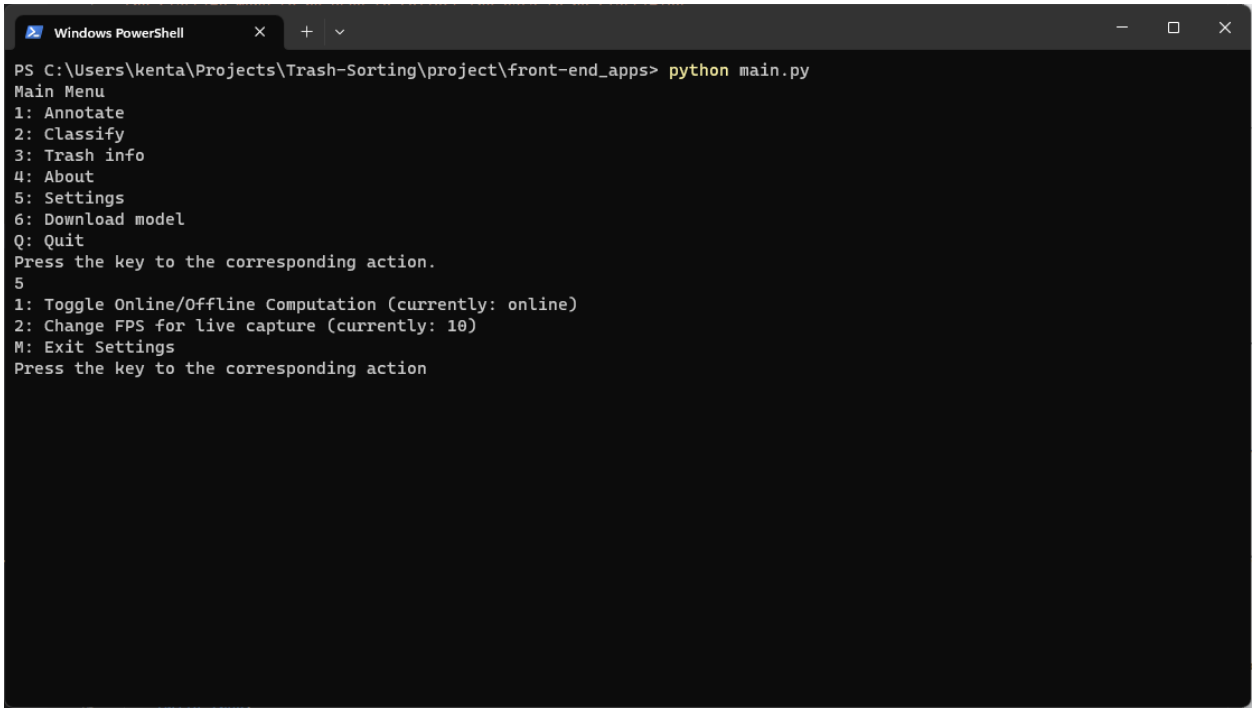
d. Trash info
    i. Trash-Sorting\project\front-end_apps**trash_info.py**



e. About
    i. Trash-Sorting\project\front-end_apps**about.py**

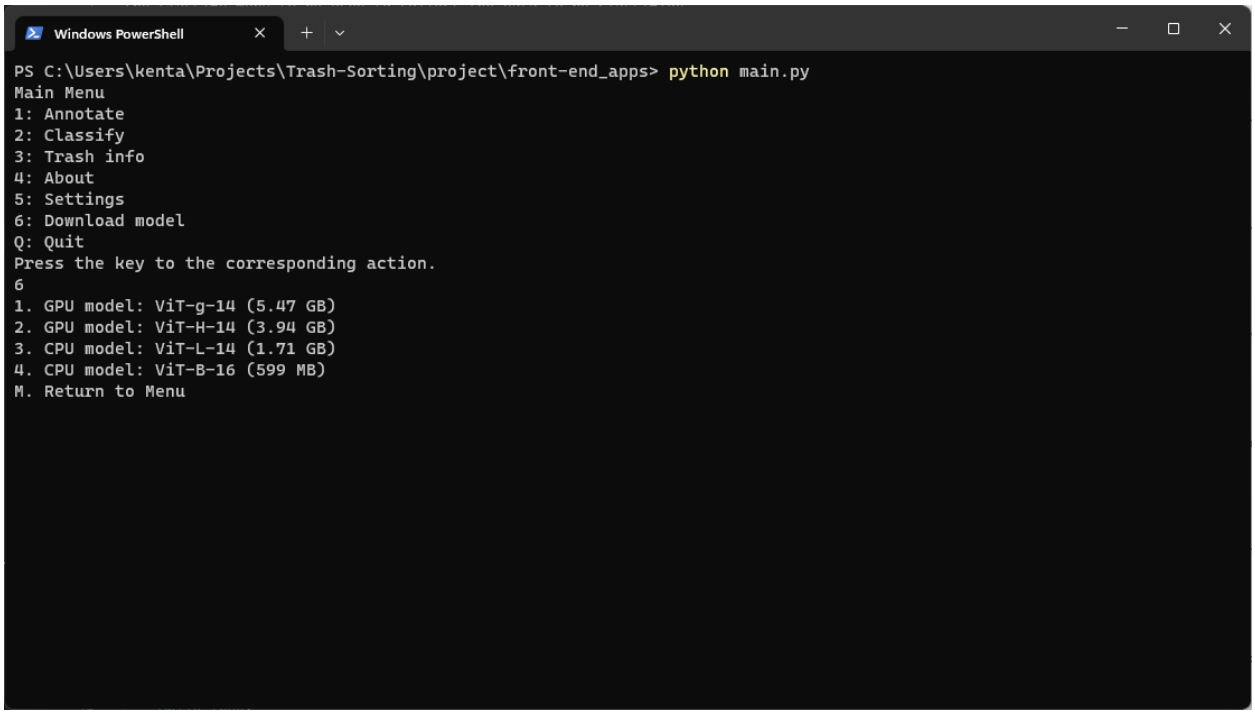


f. Settings
    i. Trash-Sorting\project\front-end_apps**settings.py**



g. Download Model

i. Trash-Sorting\project\front-end_apps**download_model.py**



6. Database

We only have one table which is for the images. This information is stored in a SQLite server that is accessed in db_server.py script, wrappers were created so that CRUD operations can be performed externally using the REST API.

ID	ID is for the unique id, starting at 1 and incrementing as we add new images
annotation	A list of objects outlined in COCO format bounding boxes
num_annotations	How many objects are in the image
dataset	Where we pulled the image from in case we need to find the source
metadata	EXIF if we store, or if there's an issue with an image