



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.....	2
2. Implementation tools and technologies.....	2
2.1 Programming languages.....	2
2.2 Libraries and Frameworks.....	2
2.3 IDE Technologies.....	3
2.3 Web Services.....	3
2.4 Databases/Data storages.....	3
2.5 Version Control System.....	3
2.6 Server.....	3
3. Runbook.....	3
3.1 Server goes down.....	3
3.2 Database becomes corrupt.....	4
3.3 Third party services go down.....	4
3.4 Environment Setup.....	4
3.5 Making Code Changes.....	11
3.5.1 Cloning the Repository.....	11
3.5.2 Pushing Changes to a Repository.....	17
3.5.3 Checking for Conflicts before Pushing.....	21
3.6 How to deploy releases.....	23
3.7 How to run automated tests.....	24
3.8 How to do “sanity” check to ensure code changes didn’t break anything.....	29
4. Deployment.....	30
5. User interaction.....	30
6. Database.....	35
Appendices.....	35

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

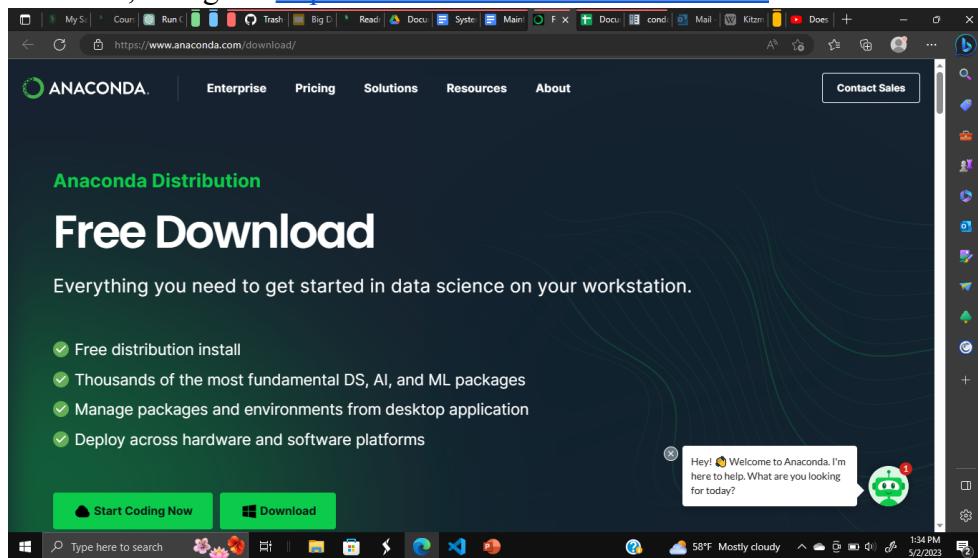
The OpenClip and EfficientNet models are downloaded from third party sites. It is unlikely such sites would be down for long, if ever, however, we recommend either utilizing the models already downloaded in the meantime or going to the [OpenClip GitHub repo](#) or the [PyTorch TorchVision GitHub repo](#)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

3.4 Environment Setup

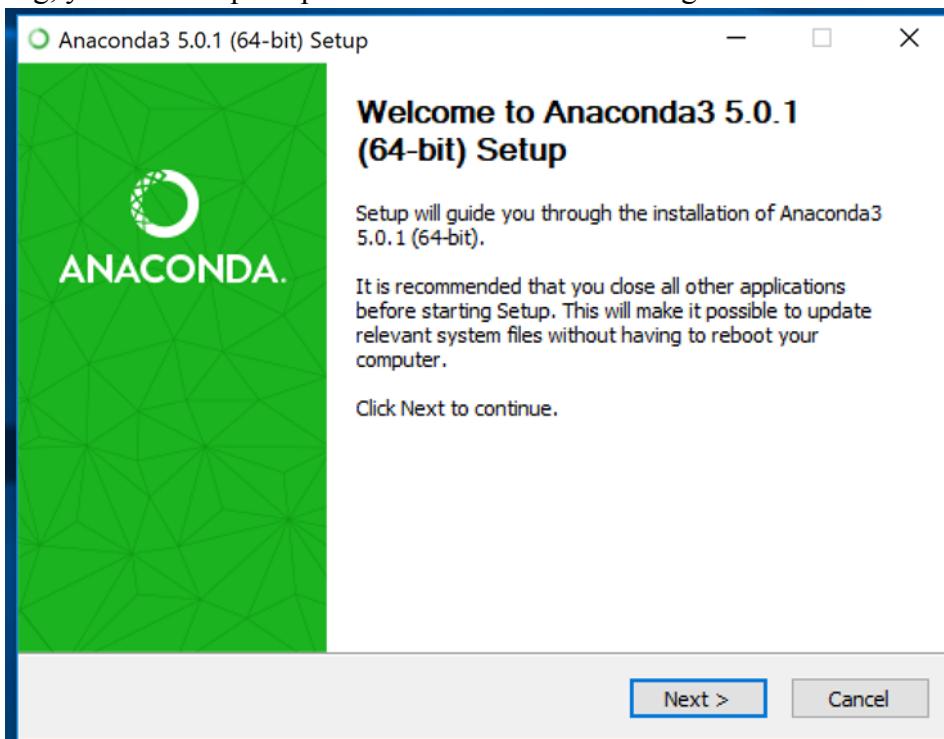
Python Environment

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.

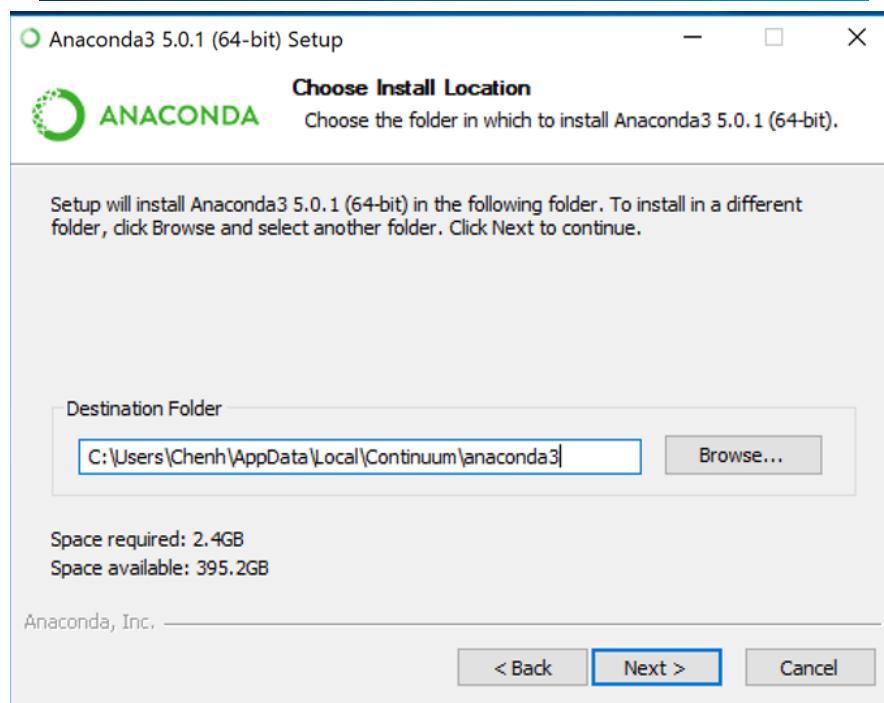
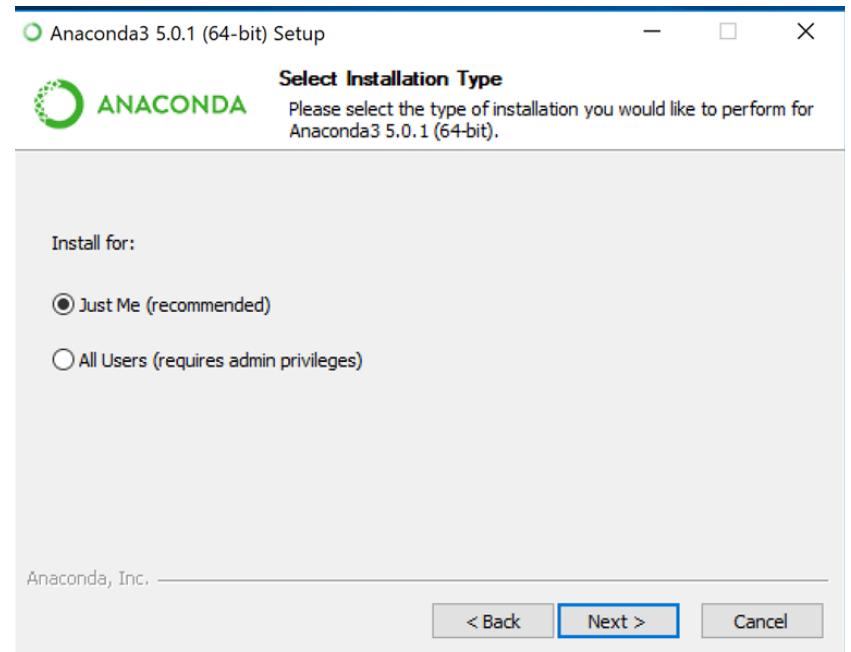
To set up Anaconda, first go to <https://www.anaconda.com/download/> and download from there.

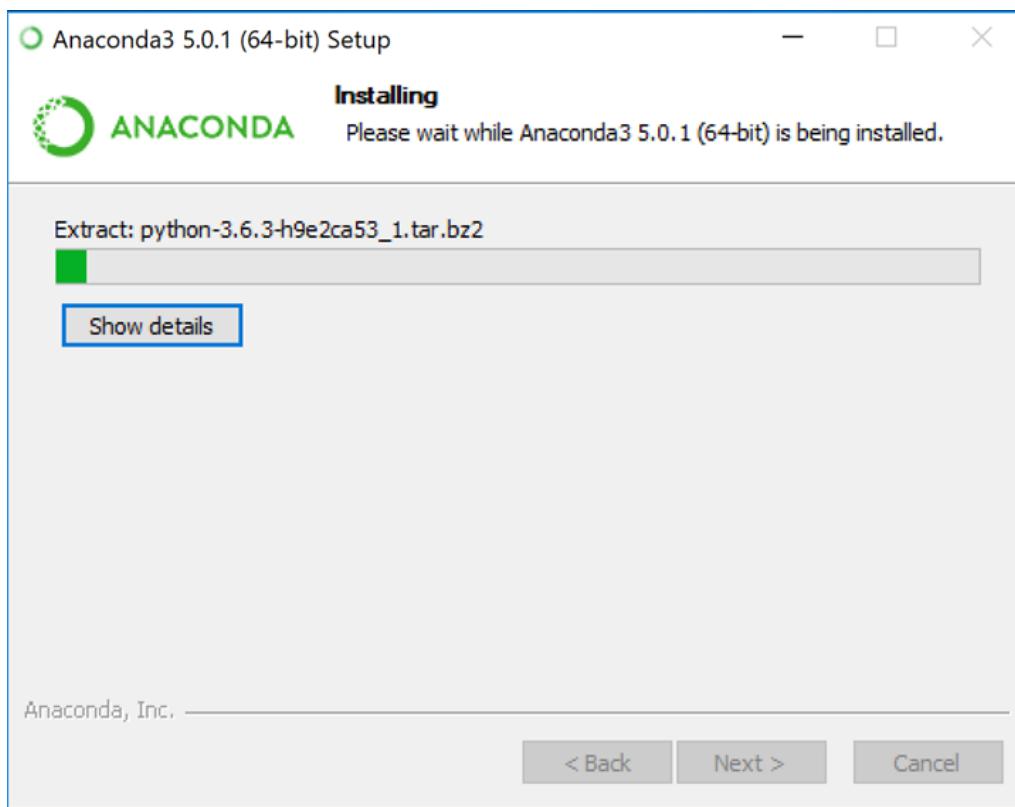
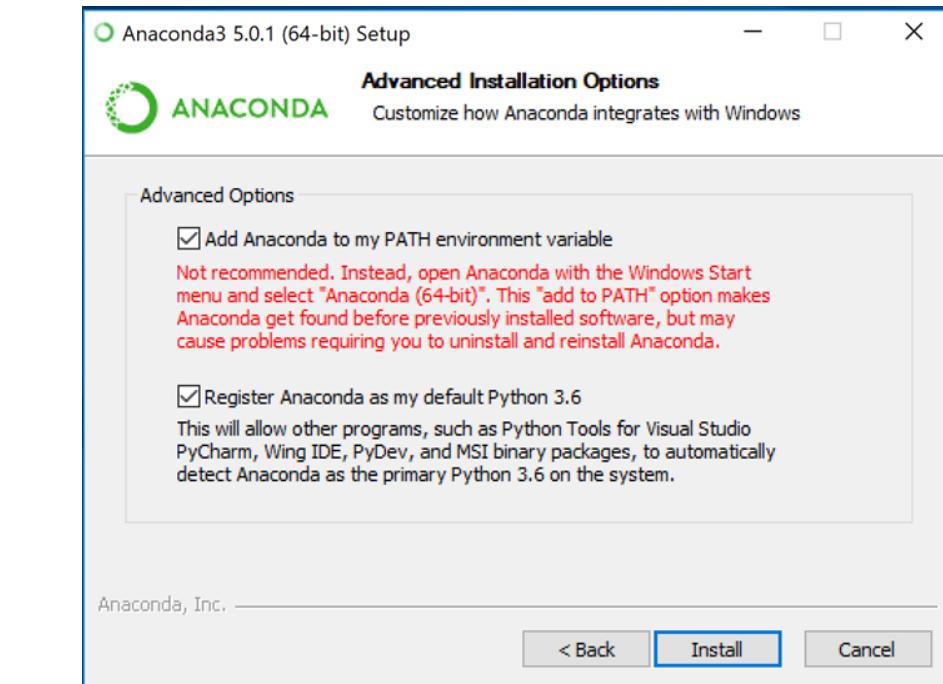


After installing, you should open up the file and see the following.

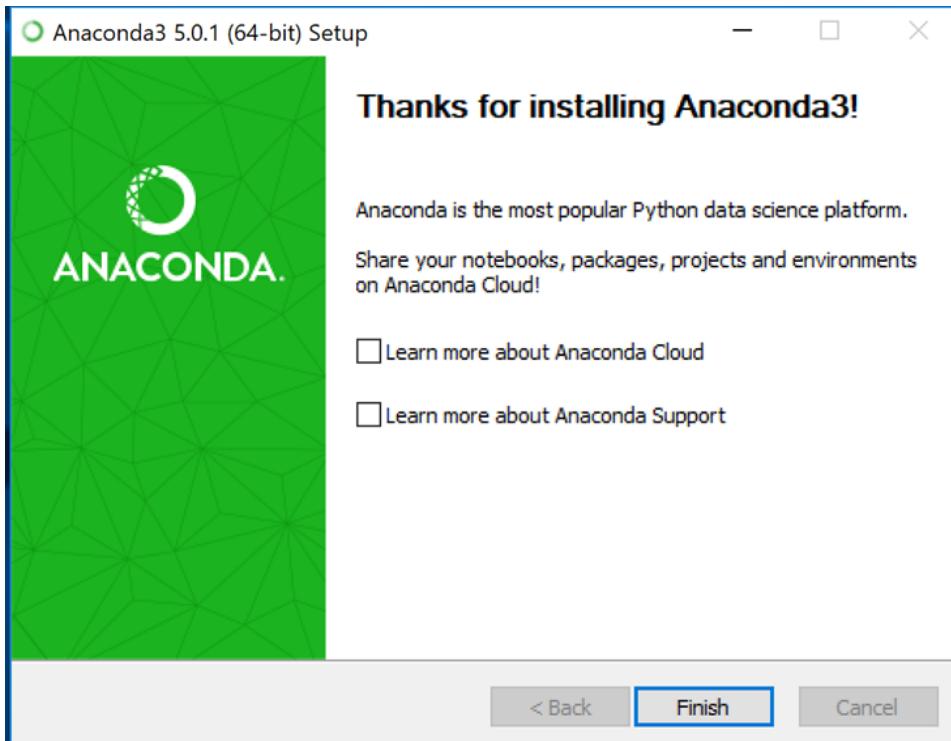


Select the following settings and click next.

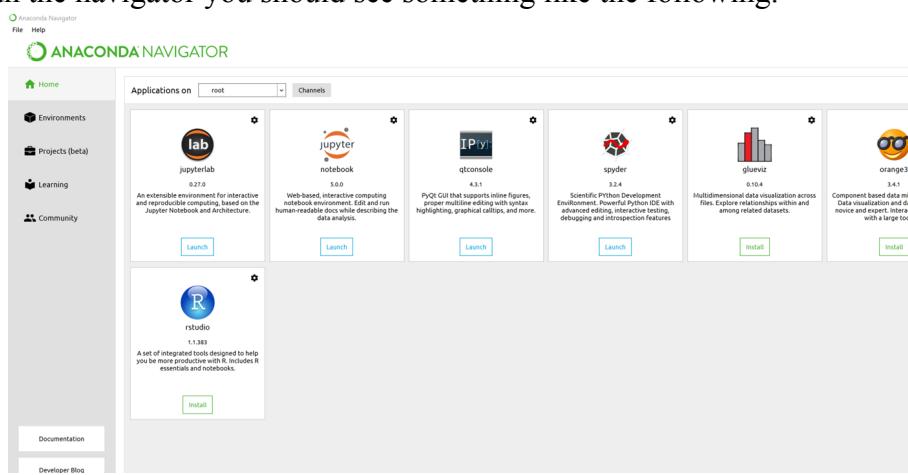




Now click finish.



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.

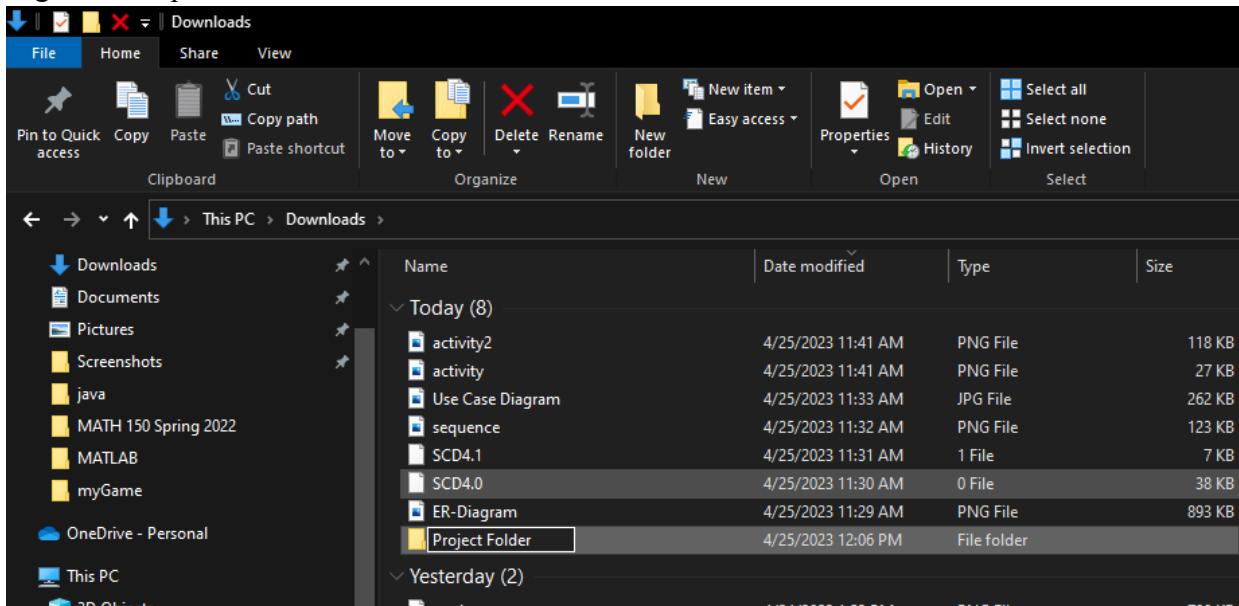
3.5 Making Code Changes

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.

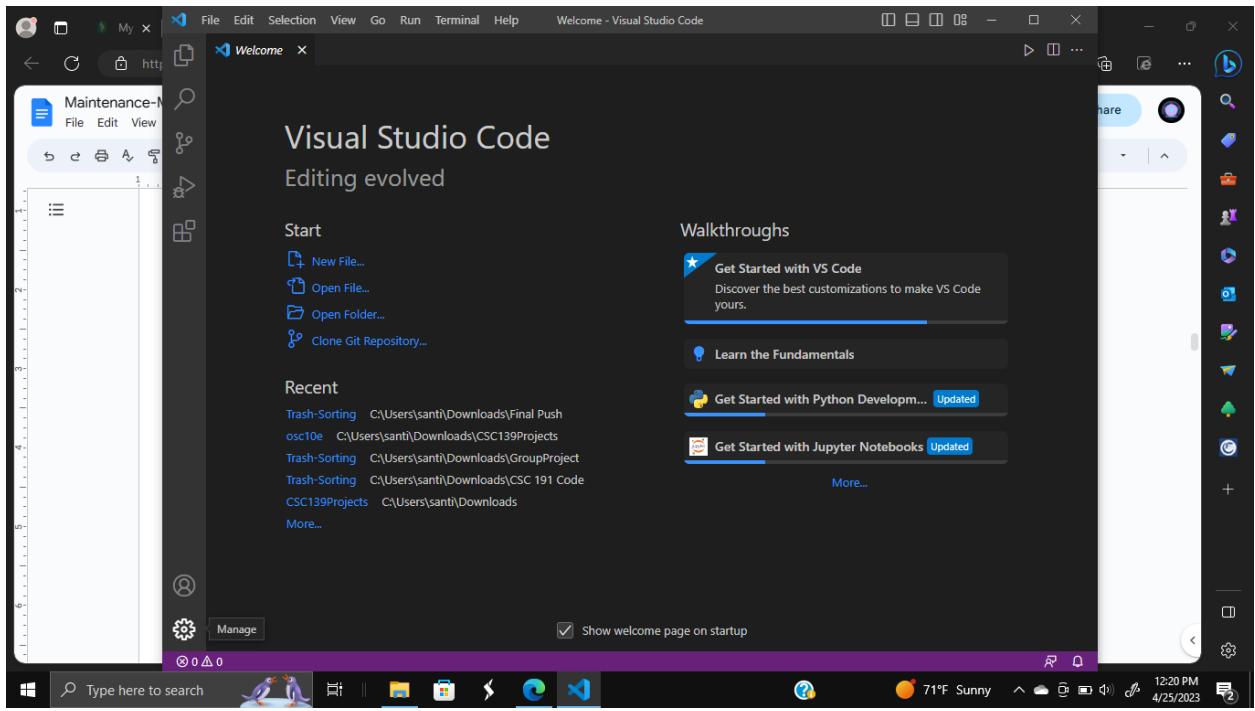
Our team followed the method of creating branches when doing multiple complete changes, so that the completed change can be merged into the main branch in one go. If a change did not have many parts or was complete, we would allow merging directly to main. However, we recommend code reviews and testing 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

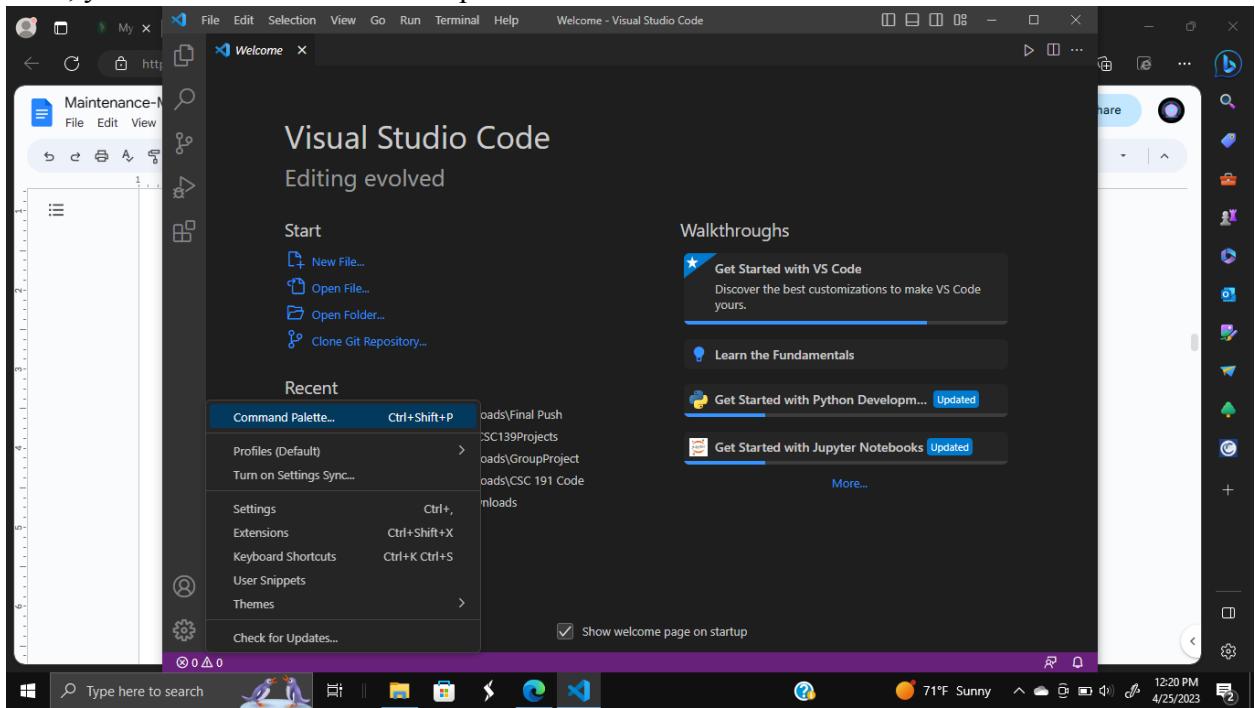
We will have to create a folder to hold the project files. You can create it anywhere and call it whatever you want. Just make sure it is in an easy to access location so that you can easily find it to get its file path.



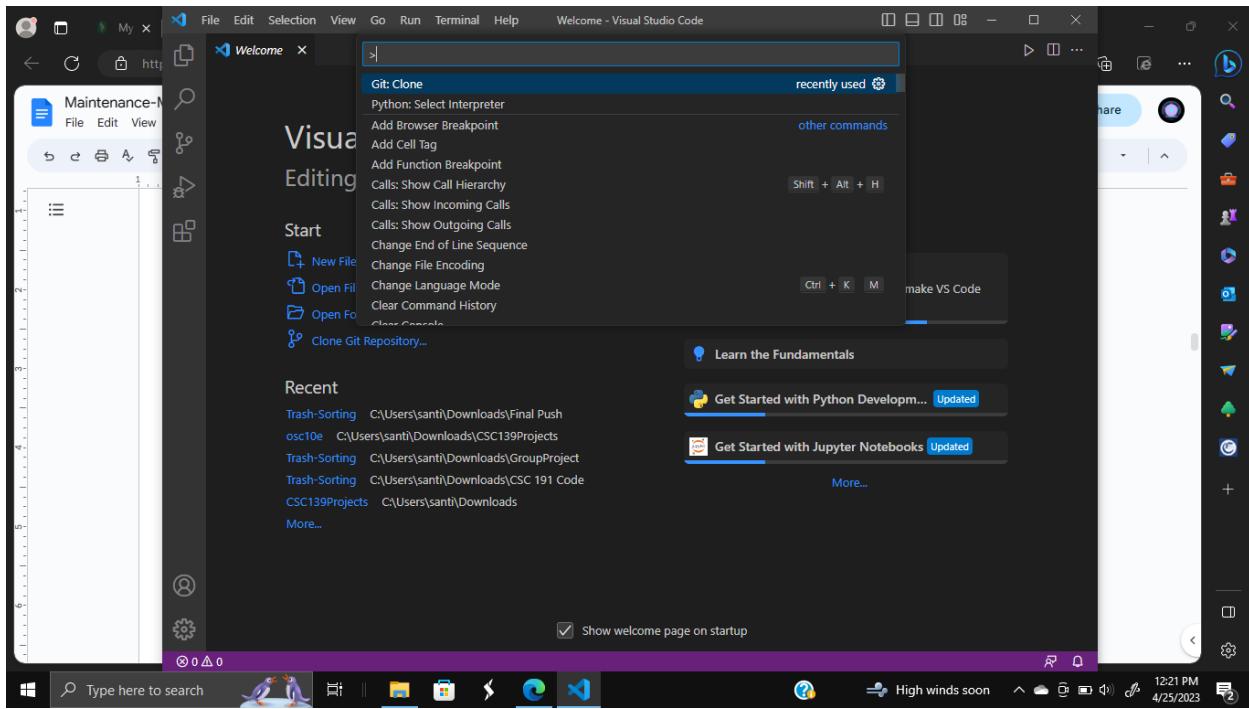
Then, you will want to open up VS Code. From there you will want to click on the settings icon on the lower left.



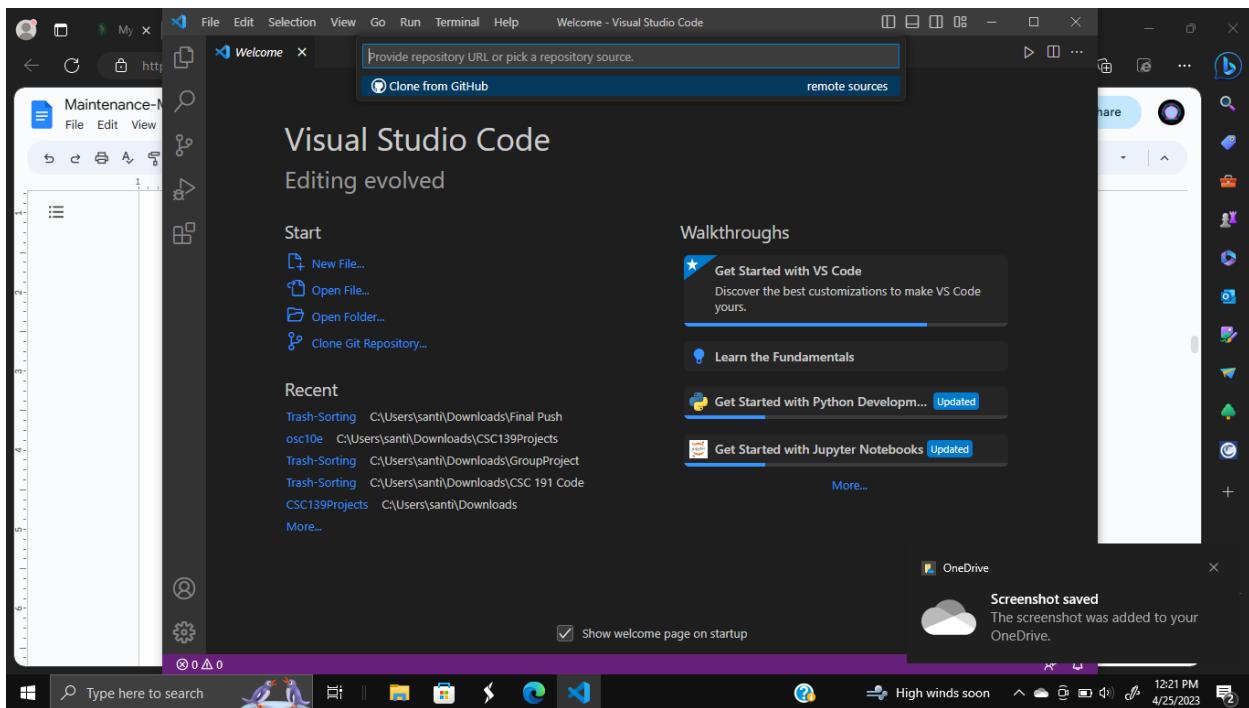
Then, you will select the command palette.



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.

<https://github.com/julianofhernandez/Trash-Sorting>

This is our senior project for computer science at California State University, Sacramento.

About

jeff-dejesus (Not a Jira Task) Fix File name

Merge pull request #5 from julianofhernandez/julian-dataset-creation

docs Merge pull request #5 from julianofhernandez/julian-dataset-creation

images BDE-65 Added mAP to validation and added more images to the validat...

project (Not a Jira Task) Fix File name

.gitattributes Add folders and Add SSD Model

.gitignore Started fixing validation script

Paper.md Created TACO reader

README.md Added testing confirmation images

requirements.txt BDE-86 2h Add download menu

Releases

No releases published

Create a new release

<https://github.com/julianofhernandez/Trash-Sorting>

This is our senior project for computer science at California State University, Sacramento.

About

jeff-dejesus (Not a Jira Task) Fix File name

Merge pull request #5 from julianofhernandez/julian-dataset-creation

docs Merge pull request #5 from julianofhernandez/julian-dataset-creation

images BDE-65 Added mAP to validation and added more images to the validat...

project (Not a Jira Task) Fix File name

.gitattributes Add folders and Add SSD Model

.gitignore Started fixing validation script

Paper.md Created TACO reader

README.md Added testing confirmation images

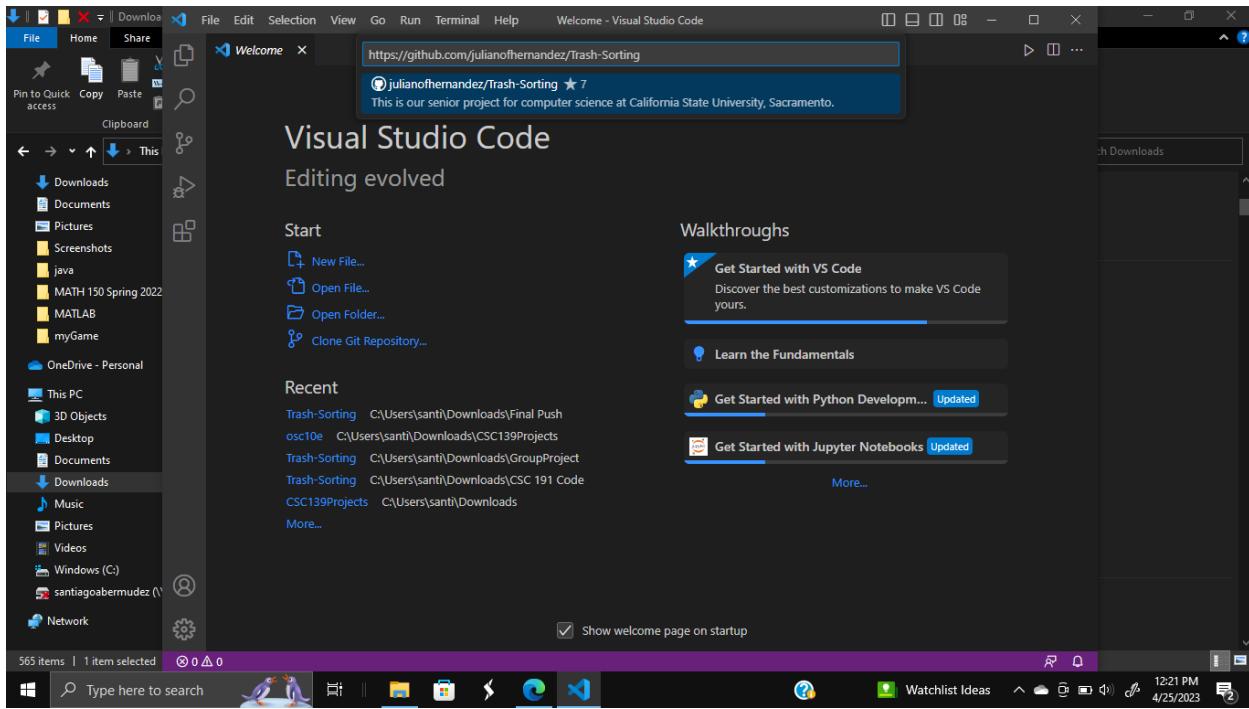
requirements.txt BDE-86 2h Add download menu

Releases

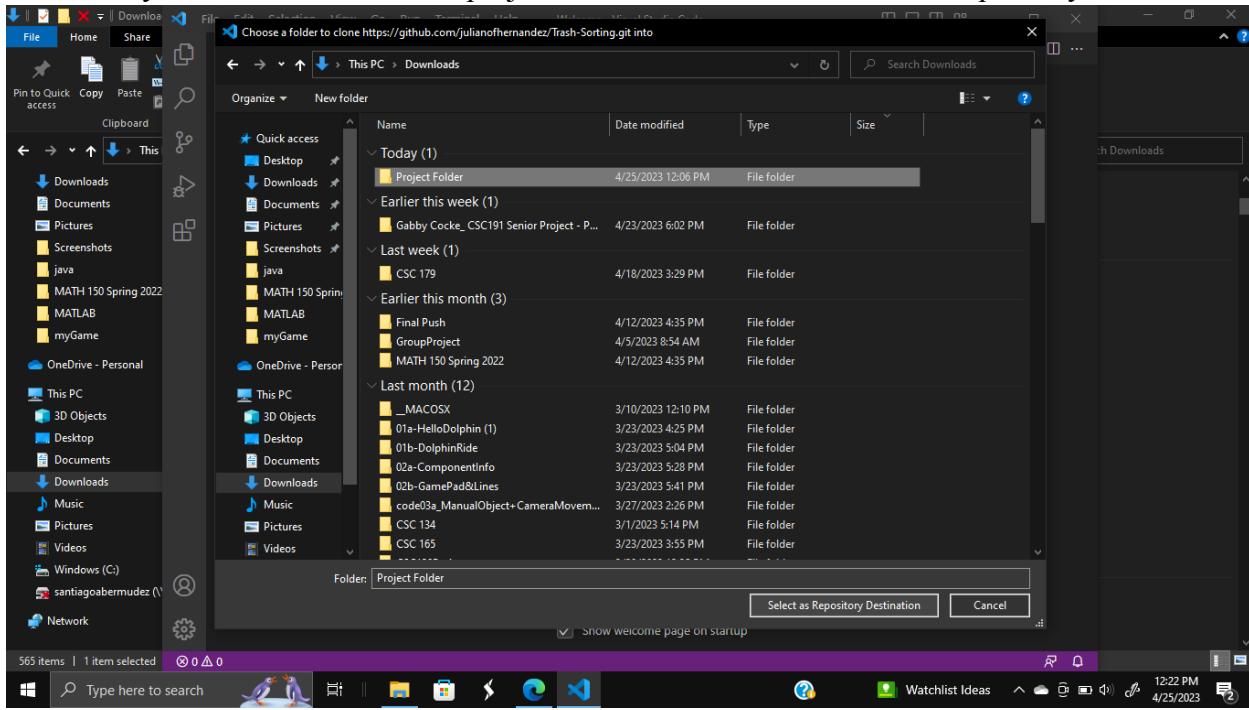
No releases published

Create a new release

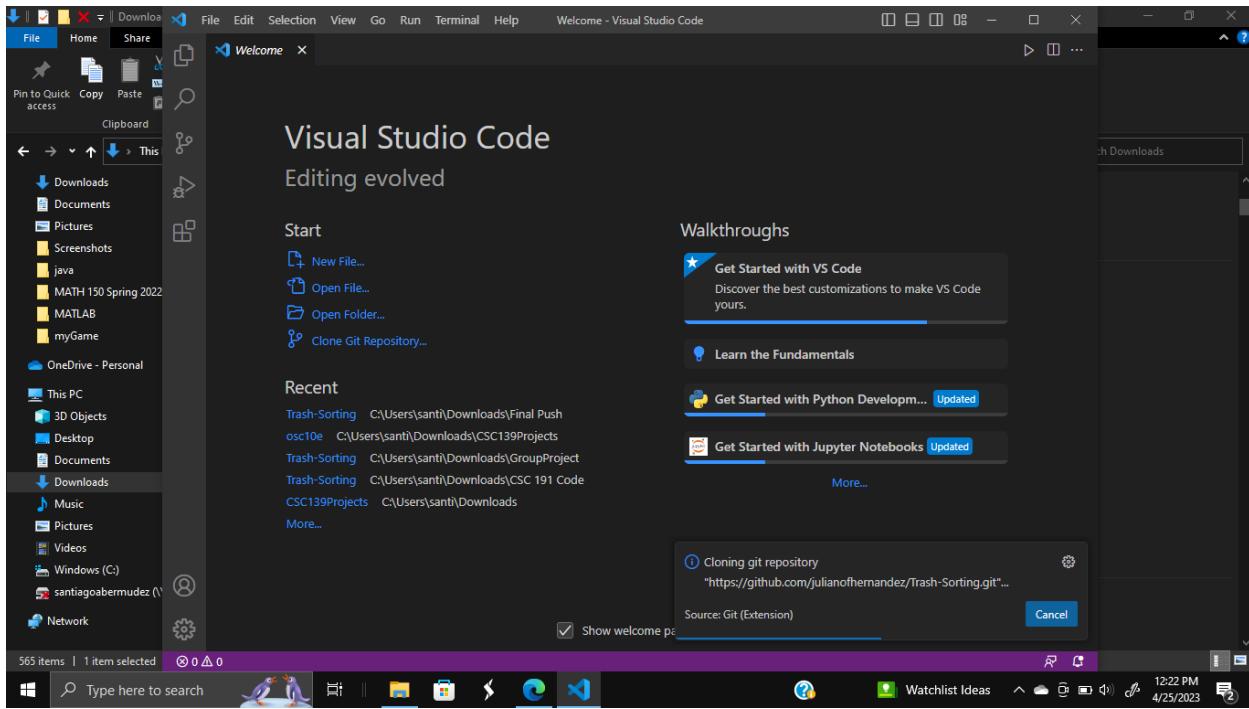
You will then paste the URL into the search bar and select the repository that gets listed below.



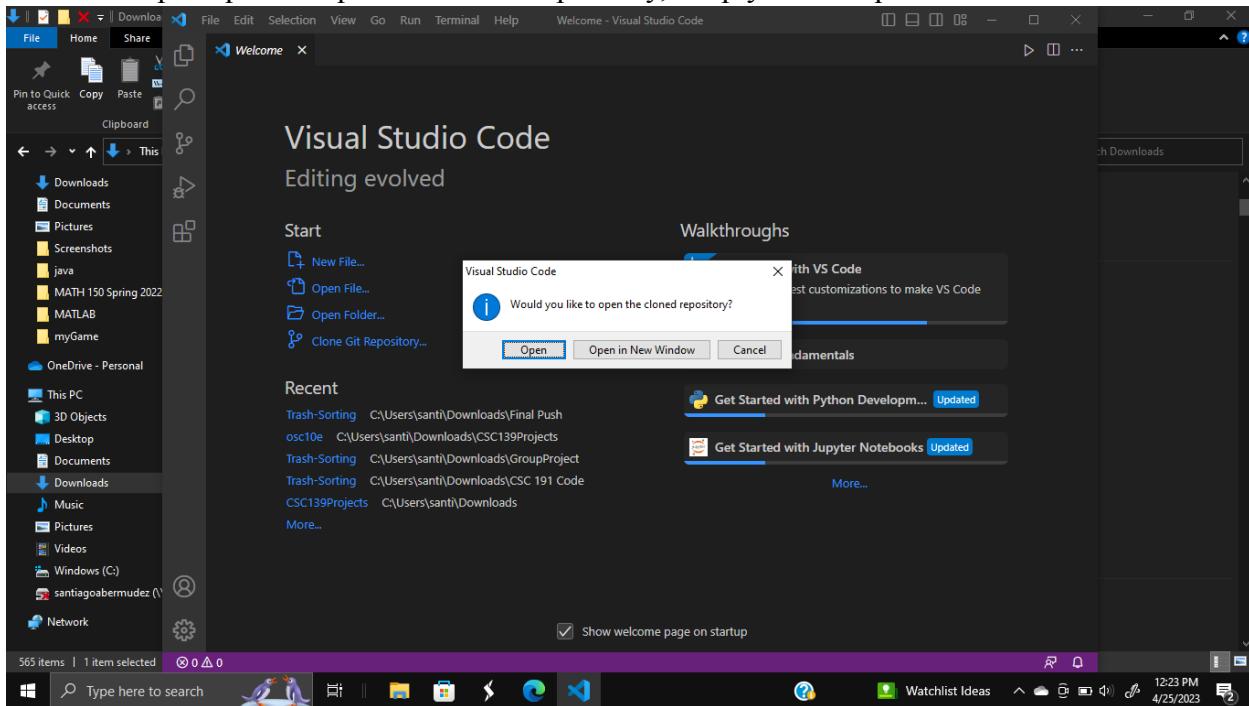
You will then be prompted to select a folder to store the project in. Navigate to and choose the folder that you have created for this project. You will then click Select as Repository Destination.



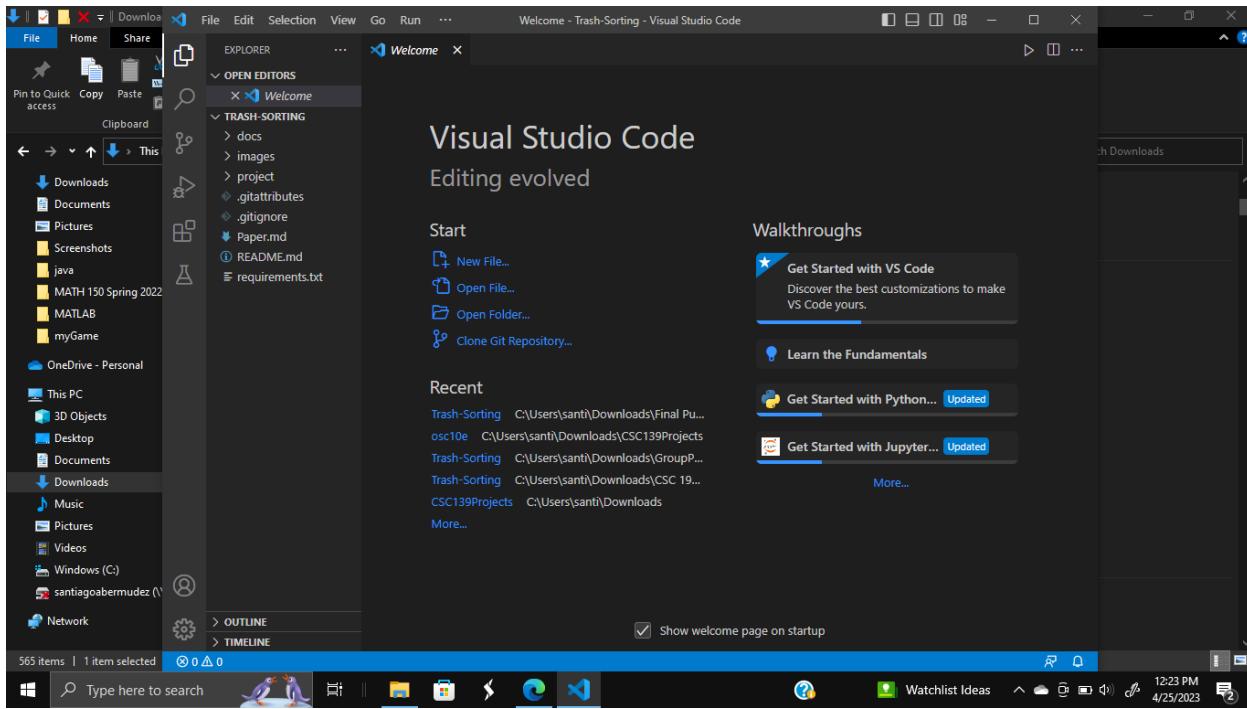
You will then see progress on the repository being cloned in the lower right.



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

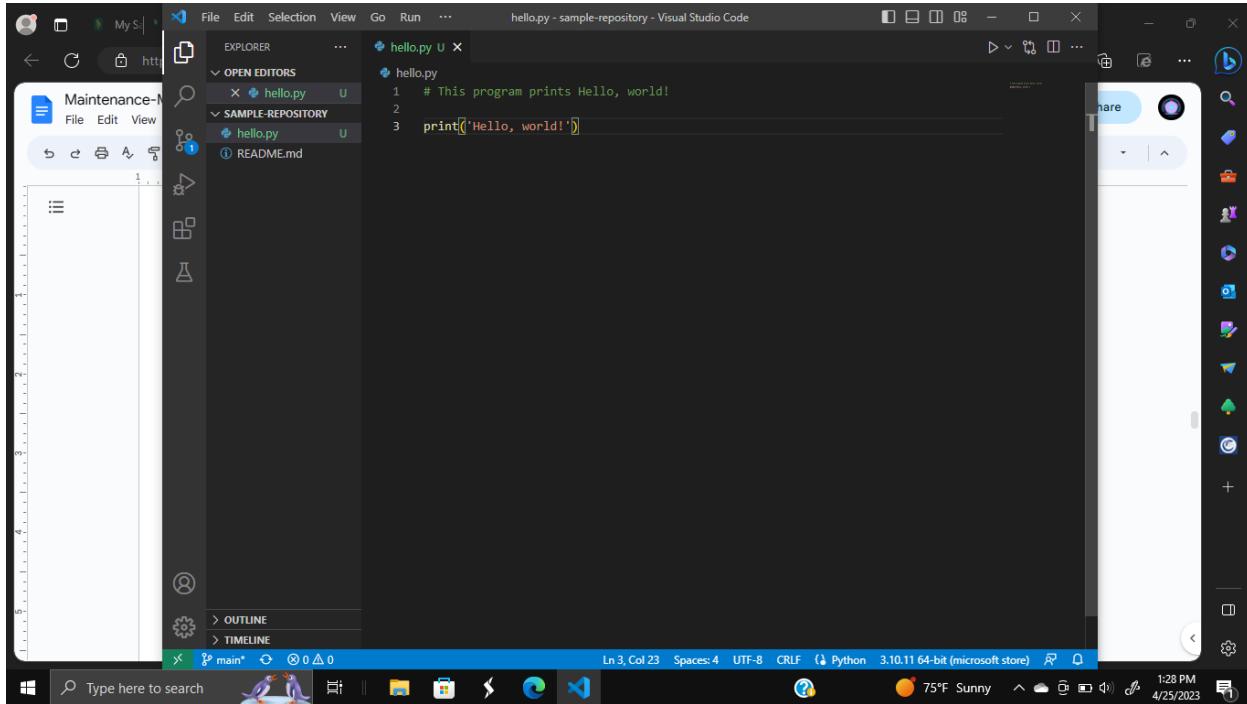


On the left, you should see some file links open up in the explorer navigation bar on the left. From there, you can navigate anywhere in the project.

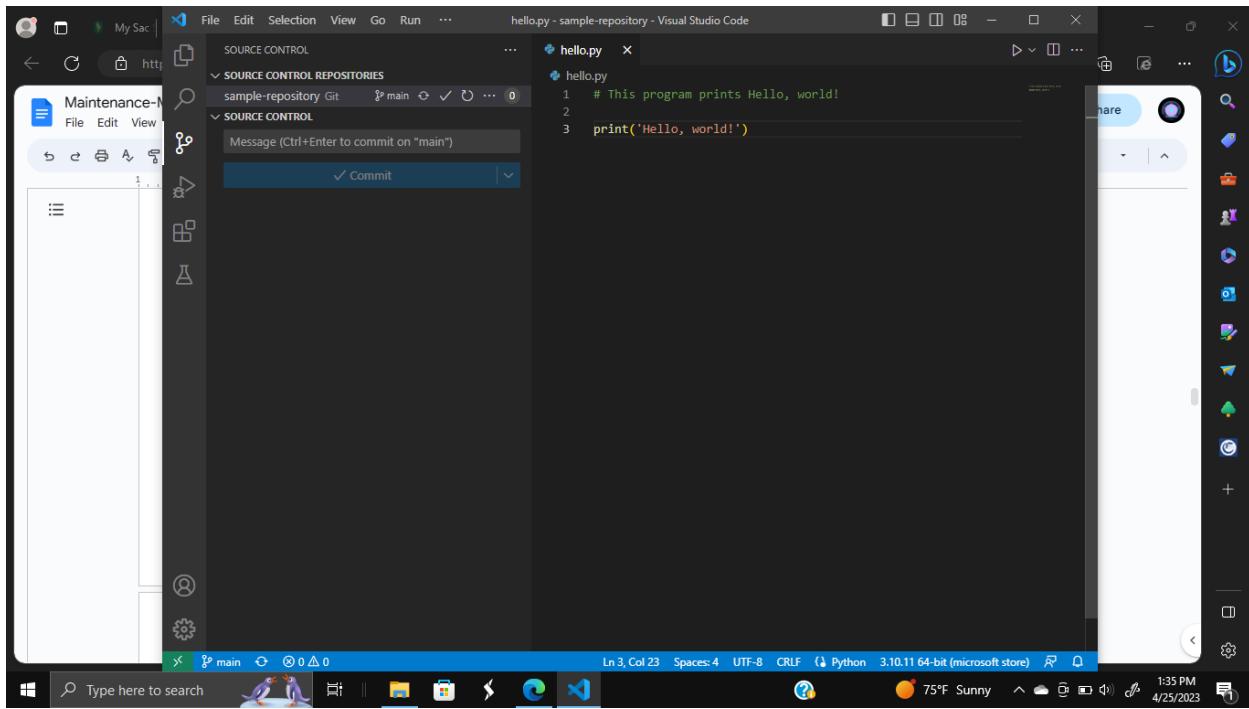


3.5.2 Pushing Changes to a Repository

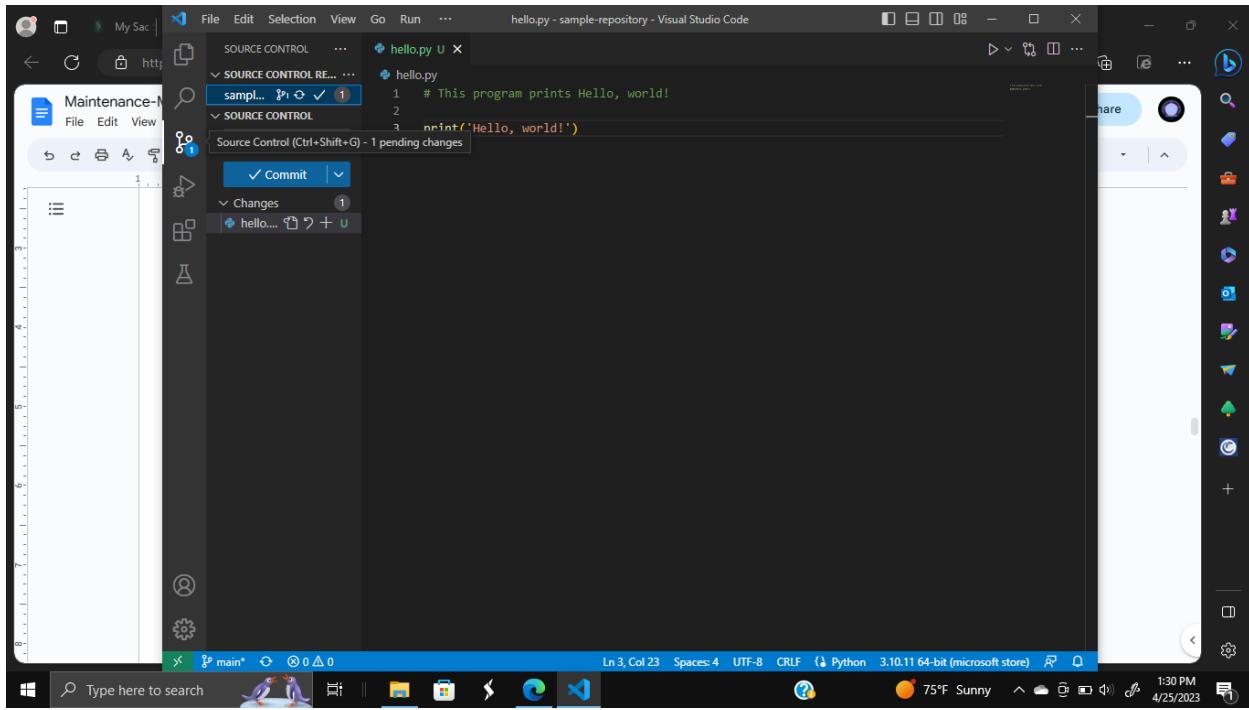
We have a sample code that we have created for use below.



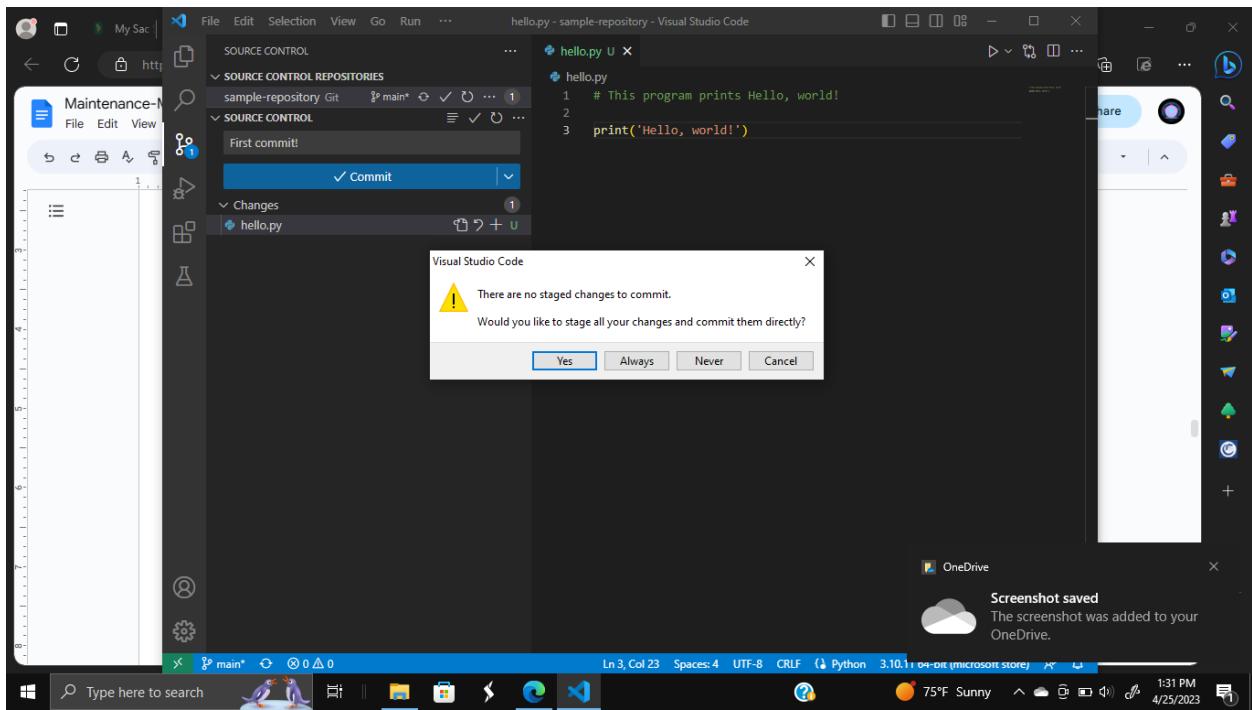
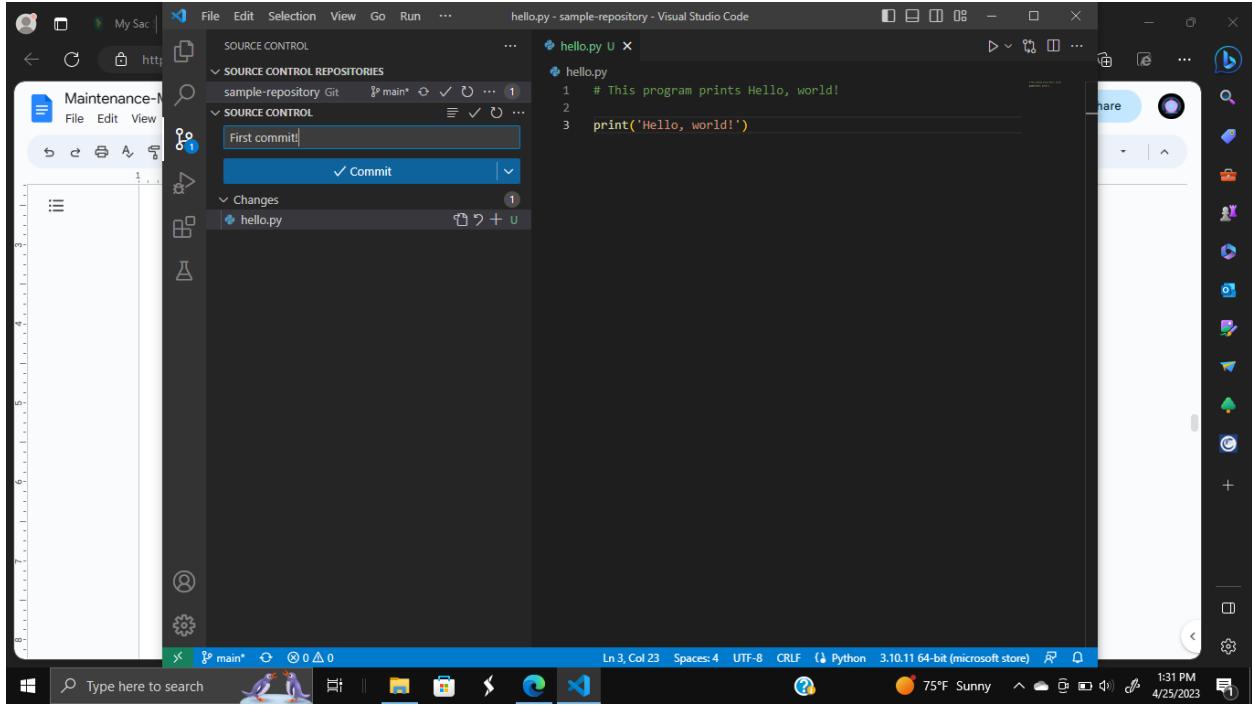
After selecting source control from the left navigational bar, one has the option to create a custom message for commits.



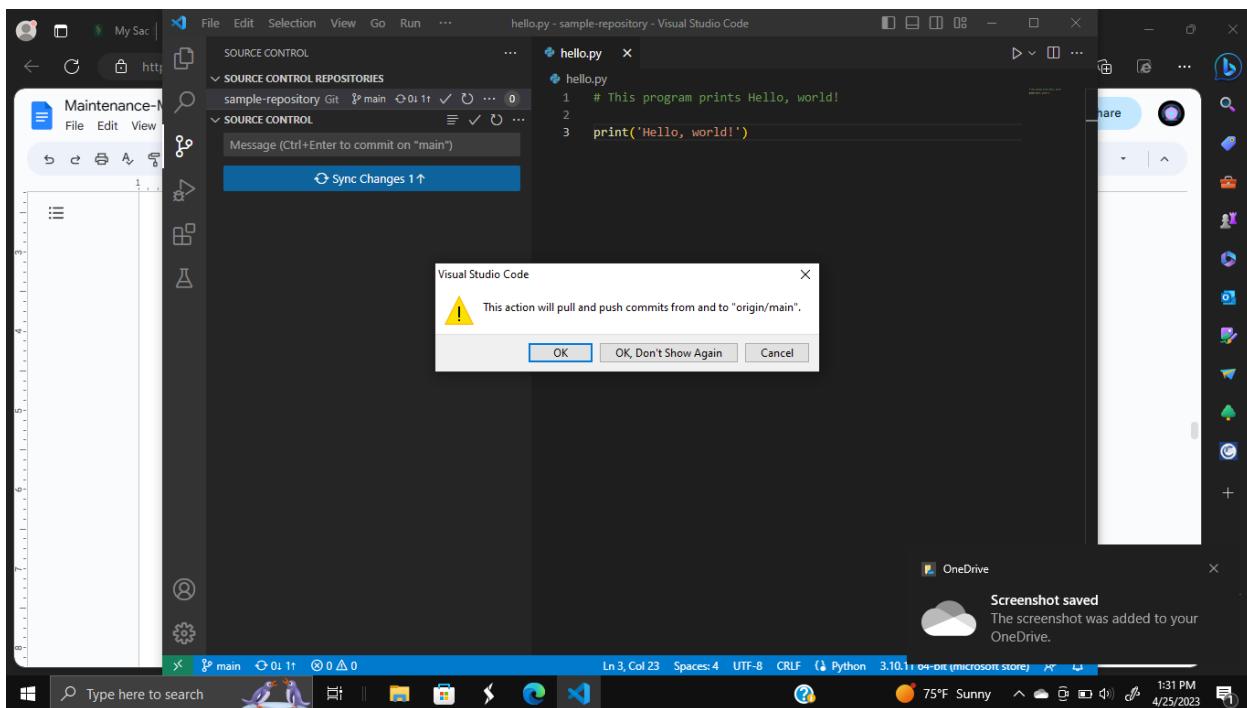
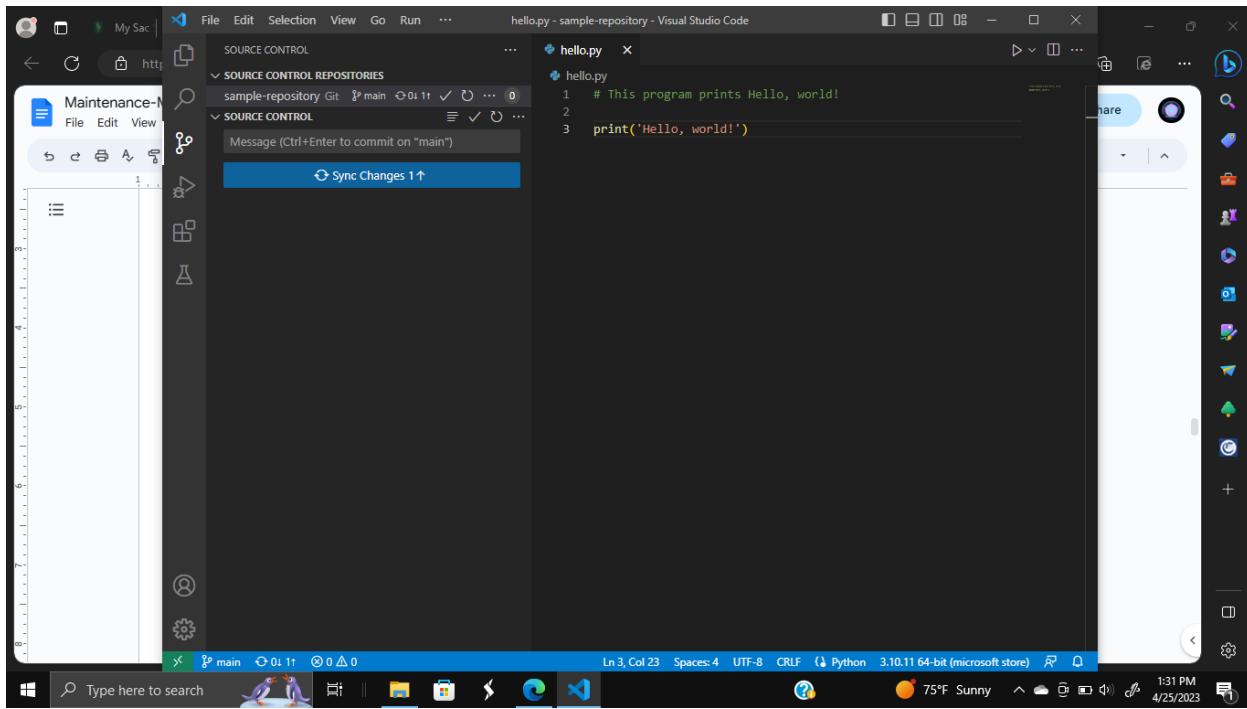
One can also see the pending changes that one has by hovering their mouse by the source control icon.



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.

The screenshot shows a GitHub repository page for 'Santiago13225 / sample-repository'. The 'Code' tab is selected, showing the 'main' branch with one commit. The commit details are as follows:

File	Message	Time
README.md	Initial commit	25 minutes ago
hello.py	First commit!	8 minutes ago

The README.md file content is:

```
sample-repository
```

For your eyes...

On the right side, there are sections for 'About', 'For your eyes...', 'Releases', and 'Packages'.

The screenshot shows the 'Commits' page for the 'main' branch of the repository. It displays two commits:

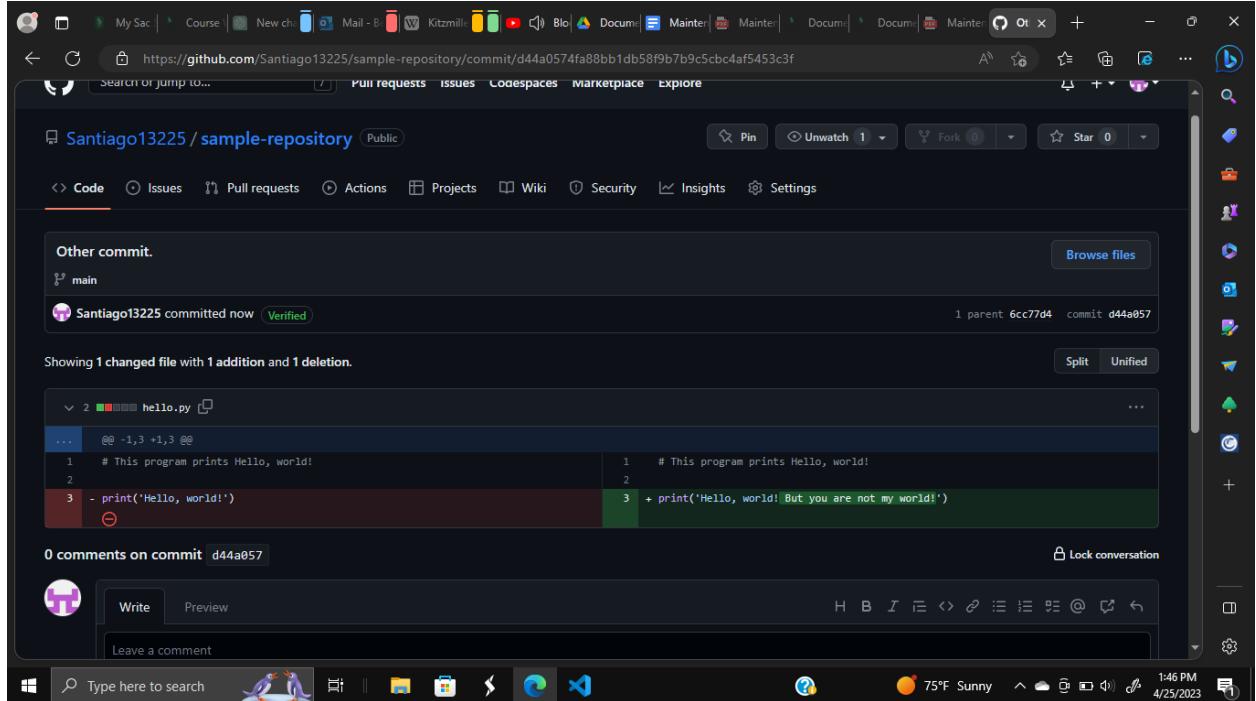
- First commit**: Santiago13225 committed 8 minutes ago (commit 6cc77d4)
- Initial commit**: Santiago13225 committed 25 minutes ago (commit dba764a)

A tooltip message 'Screenshot saved' is visible in the bottom right corner, indicating the screenshot was saved to OneDrive.

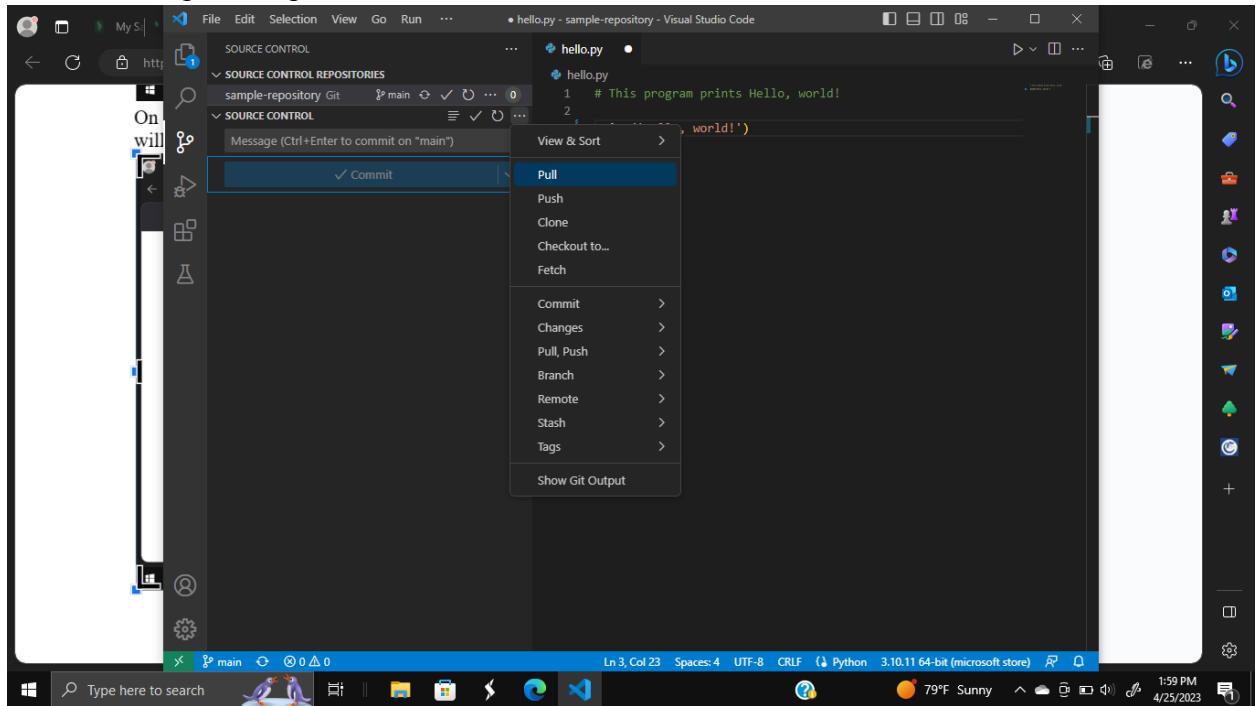
3.5.3 Checking for Conflicts before Pushing

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.

A screenshot of Visual Studio Code showing a Python file named `hello.py` with the following code:

```
1 # This program prints Hello, world!
2
3 print('Hello, world!')
```

The interface shows a "SOURCE CONTROL" sidebar with a "sample-repository Git" entry. A confirmation dialog box is displayed in the center, reading: "This action will pull and push commits from and to 'origin/main'." It has three buttons: "OK", "OK, Don't Show Again", and "Cancel". The status bar at the bottom shows "Ln 3, Col 23" and "3.10.11 64-bit (microsoft store)".

After that you will have successfully pulled all changes.

A screenshot of Visual Studio Code showing the same `hello.py` file after pulling changes. The code now includes an additional line:

```
1 # This program prints Hello, world!
2
3 print('Hello, world! But you are not my world!')
```

The "SOURCE CONTROL" sidebar shows the repository is up-to-date. The status bar at the bottom shows "Ln 4, Col 1" and "3.10.11 64-bit (microsoft store)".

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.

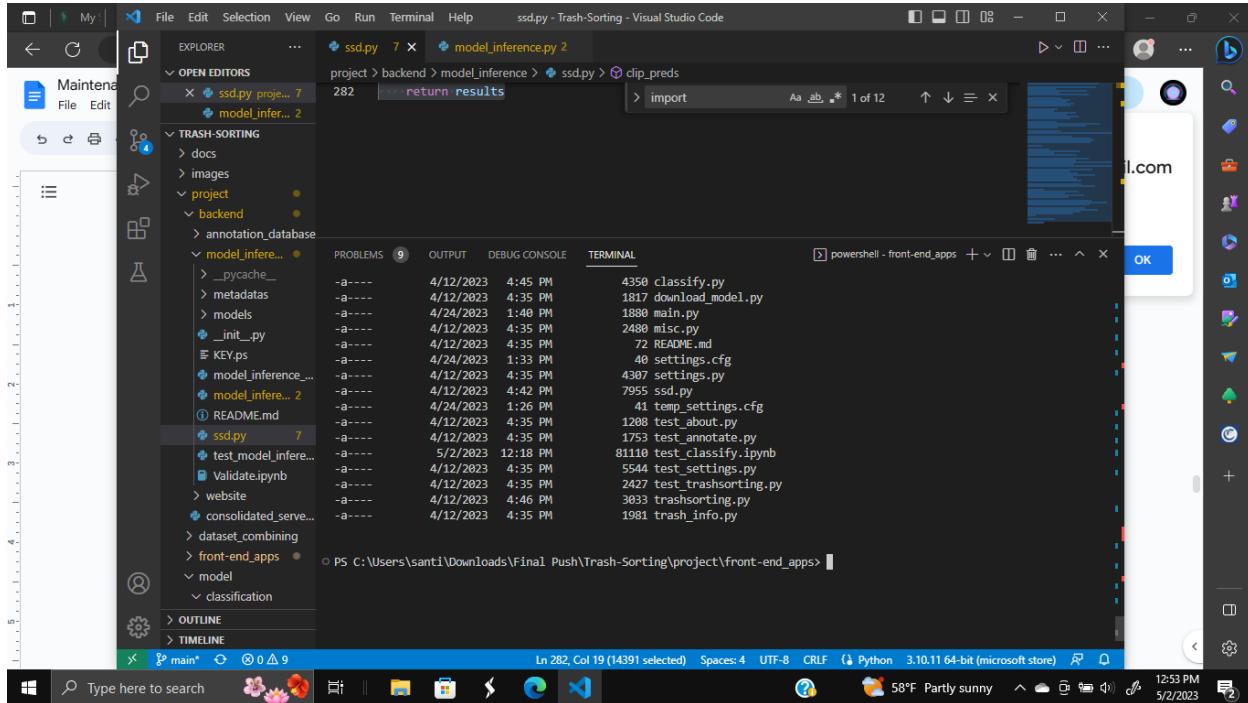
3.6 How to deploy releases

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

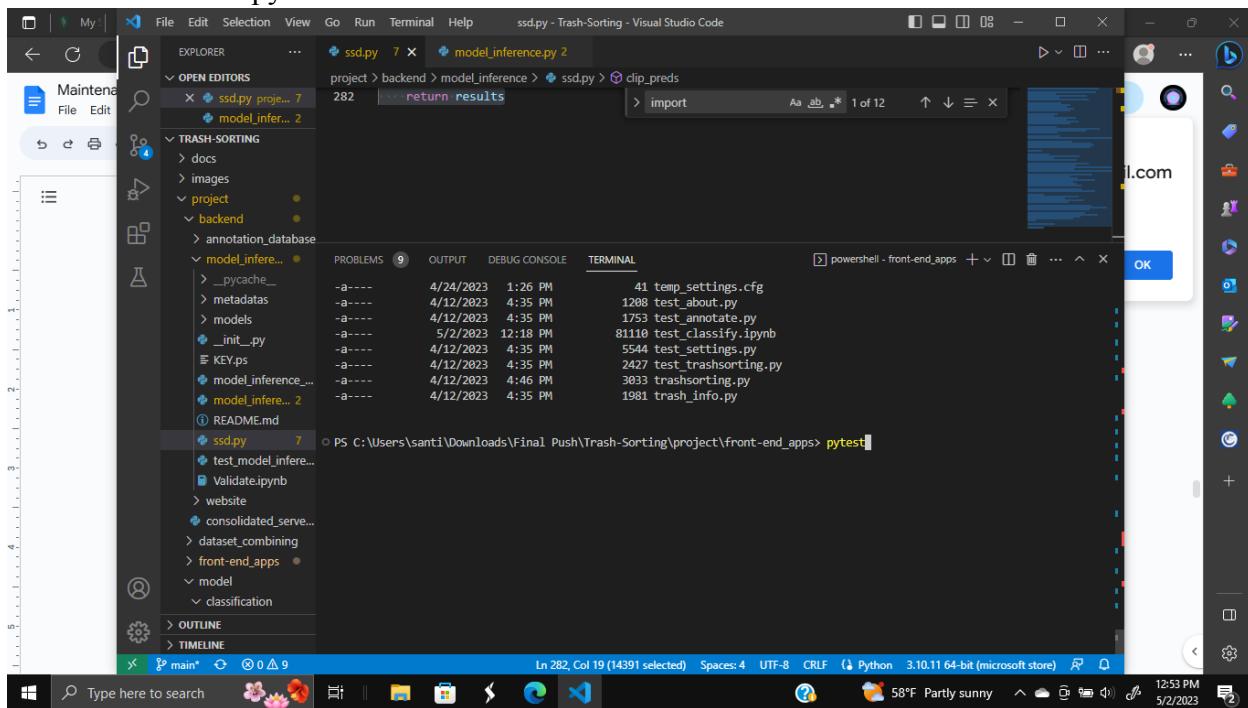
Running the pytest command in the front-end_apps folder will automatically run tests on all terminal based front end features.

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



```
ssd.py - Trash-Sorting - Visual Studio Code
File Edit Selection View Go Run Terminal Help ssd.py - Trash-Sorting - Visual Studio Code
OPEN EDITORS ssd.py 7 model_inference.py 2
project > backend > model_inference > ssd.py > clip_preds
282 ... return results import Aa ab _* 1 of 12
TRASH-SORTING
docs
images
project
backend
annotation_database
model_inference...
_init_.py
KEY.ps
model_inference...
model_inference...
README.md
ssd.py 7
test_model_infer...
Validate.ipynb
website
consolidated_serve...
dataset_combining
front-end_apps
model
classification
OUTLINE
TIMELINE
PROBLEMS 9 OUTPUT DEBUG CONSOLE TERMINAL
powershell - front-end_apps + - x
Ln 282, Col 19 (14391 selected) Spaces: 4 UTF-8 CRLF Python 3.10.11 64-bit (microsoft store)
12:53 PM 5/2/2023
```

Then we enter the ‘pytest’ command to run the tests.



```
ssd.py - Trash-Sorting - Visual Studio Code
File Edit Selection View Go Run Terminal Help ssd.py - Trash-Sorting - Visual Studio Code
OPEN EDITORS ssd.py 7 model_inference.py 2
project > backend > model_inference > ssd.py > clip_preds
282 ... return results import Aa ab _* 1 of 12
TRASH-SORTING
docs
images
project
backend
annotation_database
model_inference...
_init_.py
KEY.ps
model_inference...
model_inference...
README.md
ssd.py 7
test_model_infer...
Validate.ipynb
website
consolidated_serve...
dataset_combining
front-end_apps
model
classification
OUTLINE
TIMELINE
PROBLEMS 9 OUTPUT DEBUG CONSOLE TERMINAL
powershell - front-end_apps + - x
Ln 282, Col 19 (14391 selected) Spaces: 4 UTF-8 CRLF Python 3.10.11 64-bit (microsoft store)
12:53 PM 5/2/2023
```

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).

The image shows two side-by-side screenshots of the Visual Studio Code interface, each displaying a terminal window with the output of a pytest command.

Top Terminal Output:

```

PS C:\Users\santi\Downloads\Final Push\Trash-Sorting\project\front-end_apps> pytest
=====
test session starts =====
platform win32 -- Python 3.11.2, pytest-7.2.1, pluggy-1.0.0
rootdir: C:\Users\santi\Downloads\Final Push\Trash-Sorting\project\front-end_apps
collected 29 items

test_about.py F....F
test_annotate.py .FFF
test_settings.py .F.F.FFFF...F
test_trashsorting.py FFFFF

=====
FAILURES =====
=====
TestHandle.test_key_invalid
self = <test_about.TestHandle object at 0x000001C61D970310>
capsys = <_pytest.capture.CaptureFixture object at 0x000001C621E18950>

def test_key_invalid(self, capsyst):
    invalid_option = -1
    ret = about.handle.key(invalid_option)

=====

```

Bottom Terminal Output:

```

PS C:\Users\santi\Downloads\Final Push\Trash-Sorting\project\front-end_apps> pytest
=====
test session starts =====
platform win32 -- Python 3.11.2, pytest-7.2.1, pluggy-1.0.0
rootdir: C:\Users\santi\Downloads\Final Push\Trash-Sorting\project\front-end_apps
collected 29 items

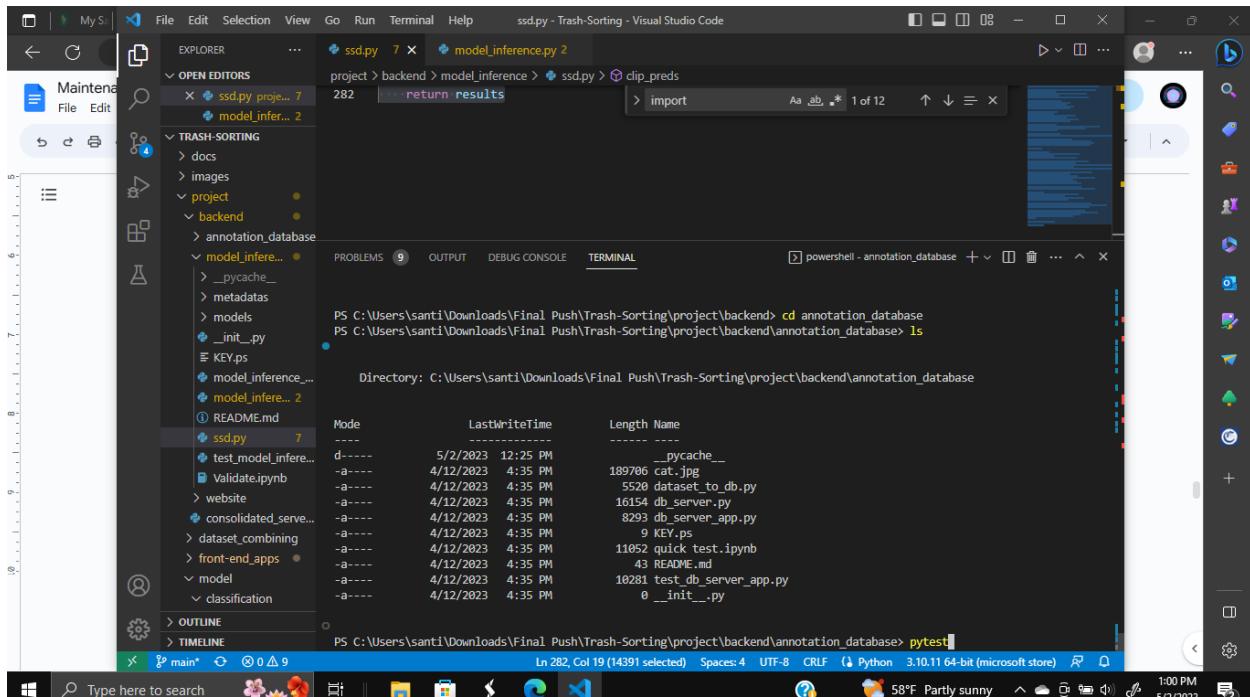
FAILED test_annotate.py::TestOpenFromPath::test_valid_path - ModuleNotFoundError: No module named 'matplotlib'
FAILED test_annotate.py::TestUploadAnnotation::test_finished_annotation - ModuleNotFoundError: No module named 'matplotlib'
FAILED test_annotate.py::TestUploadAnnotation::test_unfinished_annotation - ModuleNotFoundError: No module named 'matplotlib'
FAILED test_settings.py::TestToggleFps::test_out_of_range - AssertionError: assert 'ERROR: invalid option' in 'Enter a number for the framerate for live capture\Win framerate..'
FAILED test_settings.py::TestToggleClassMode::test_invalid_option - AssertionError: assert 'ERROR: invalid option' in 'Do you want single or multi object classification?\n1: single\n2...'
FAILED test_settings.py::TestToggleCompMode::test_invalid_option - AssertionError: assert 'ERROR: invalid option' in 'Do you want online or offline computation?\n1: online\n2: offline'
FAILED test_settings.py::TestSettingsConfig::test_load - assert False == True
FAILED test_settings.py::TestSettingsConfig::test_save - KeyError: 'SINGLE CLASSIFICATION'
FAILED test_settings.py::TestHandle::test_key_invalid - AssertionError: assert 'ERROR: invalid option' in 'ERROR: Invalid option (Enter key to the left of menu options).\n\n'
FAILED test_settings.py::TestHandle::test_key_exit - OSError: pytest: reading from stdin while output is captured! Consider using '-s'.
FAILED test_trashsorting.py::TestTrashSortingApp::test_input_image - AssertionError: assert 1 == 0
FAILED test_trashsorting.py::TestTrashSortingApp::test_local_model - AssertionError: assert 1 == 0
FAILED test_trashsorting.py::TestTrashSortingApp::test_online_model - AssertionError: assert 1 == 0
FAILED test_trashsorting.py::TestTrashSortingApp::test_single_option - AssertionError: assert 1 == 0
FAILED test_trashsorting.py::TestTrashSortingApp::test_output_json - AssertionError: assert 1 == 0

===== 17 failed, 12 passed in 5.64s =====

```

- 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.



The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows the project structure under the 'backend' folder, including 'annotation_database', 'model_inference', and 'ssd'.
- Terminal:** Displays the command `pytest` being run in the directory `C:\Users\santi\Downloads\Final Push\Trash-Sorting\project\backend\annotation_database`. The output shows a test session starting, collecting 0 items, and then listing errors related to importing the module `test_db_server_app.py` due to a missing module named `db_server_app`.
- Status Bar:** Shows the current file is `ssd.py` at line 282, column 19, with 14391 selected. It also shows Python 3.10.11 64-bit (microsoft store) and the date/time as 1:00 PM 5/2/2023.

You can likewise do the same for the model_inference directory.

```

PS C:\Users\santi\Downloads\Final Push\Trash-Sorting\project\backend\model_inference> pytest
----- test session starts -----
platform win32 -- Python 3.11.2, pytest-7.2.1, pluggy-1.0.0
rootdir: C:\Users\santi\Downloads\Final Push\Trash-Sorting\project\backend\model_inference
collected 0 items / 1 error

----- ERRORS -----
ERROR collecting test model_inference_app.py
ImportError while importing test module 'C:\Users\santi\Downloads\Final Push\Trash-Sorting\project\backend\model_inference\test_model_inference_app.py'.
Hint: make sure your test modules/packages have valid Python names.
Traceback:
C:\Users\santi\AppData\Local\Programs\Python\Python311\Lib\importlib\_init_.py:126: in import_module
    return _bootstrap._gcd_import(name[level:], package, level)
test_model_inference_app.py:2: in <module>
    from model_inference_app import create_app
E   ModuleNotFoundError: No module named 'model_inference_app'

----- short test summary info -----
ERROR test_model_inference_app.py
!!!!!!!!!! Interrupted: 1 error during collection !!!!!!!
1 error in 0.29s
PS C:\Users\santi\Downloads\Final Push\Trash-Sorting\project\backend\model_inference>

```

- 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'

```

https://chat.openai.com
You can install Yarn by following the steps below:
File Edit Selection View Go Run Terminal Help ssd.py - Trash-Sorting - Visual Studio Code
EXPLORER OPEN EDITORS ssd.py 7 model_inference.py 2
project > backend > model_inference > ssd.py > clip_preds
282 ... return results > import Aa ab 1 of 12 ↑ ↓ ⌂ x
----- PROBLEMS -----
----- OUTPUT -----
----- DEBUG CONSOLE -----
----- TERMINAL -----
powershell - react-website + x
PS C:\Users\santi\Downloads\Final Push\Trash-Sorting\project\front-end_apps\website\react-website> yarn test
Press f to run only failed tests.
Press q to quit watch mode.
Press p to filter by a filename regex pattern.
Press t to filter by a test name regex pattern.
Press Enter to trigger a test run.

Done in 62.23s.
PS C:\Users\santi\Downloads\Final Push\Trash-Sorting\project\front-end_apps\website\react-website>
PS C:\Users\santi\Downloads\Final Push\Trash-Sorting\project\front-end_apps\website\react-website> yarn test

```

A screenshot of a Windows desktop showing Visual Studio Code (VS Code) running in the foreground. The title bar of the VS Code window says "ssd.py - Trash-Sorting - Visual Studio Code". The terminal tab is active, displaying the command "yarn run v1.22.19" followed by "\$ react-scripts test --env=jsdom". The output shows a series of test cases being run, with some failing and some passing. A tooltip from OneDrive indicates that a screenshot was saved. The taskbar at the bottom shows other open applications like File Explorer, Edge, and FileZilla.

A second screenshot of the same Windows desktop and VS Code setup. This time, the terminal output shows "No tests found related to files changed since last commit. Press `a` to run all tests, or run Jest with `--watchAll`." The rest of the interface is identical to the first screenshot, including the OneDrive tooltip and the taskbar at the bottom.

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

A screenshot of the Visual Studio Code interface. The title bar shows "ssd.py - Trash-Sorting - Visual Studio Code". The left sidebar has a tree view of files and folders. The main area has two open editors: "ssd.py" and "model_inference.py". Below the editors is a "TERMINAL" tab showing test results:

```
No tests found related to files changed since last commit.  
Press 'a' to run all tests, or run Jest with '--watchAll'.  
Watch Usage  
> Press a to run all tests.  
PASS src/classifyFormtemp.test.js (6.795 s)  
ClassifyForm  
✓ should render without throwing an error (29 ms)  
componentDidMount  
✓ should set state.options with model names on successful fetch (13 ms)  
handleDropdownChange  
✓ should set selectedModel state (6 ms)  
handleAddfileInput  
✓ should increment fileInputCount state and disable submit/add file buttons (7 ms)  
handleRemovefileInput  
✓ should decrement fileInputCount state and remove the last file from fileToSubmit state (16 ms)  
✓ should not remove the last file input element and corresponding file object if there is only one file input element (16 ms)  
Test Suites: 1 passed, 1 total  
Tests: 6 passed, 6 total
```

A second screenshot of the Visual Studio Code interface, identical to the first one, showing the same terminal output and file structure.

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.

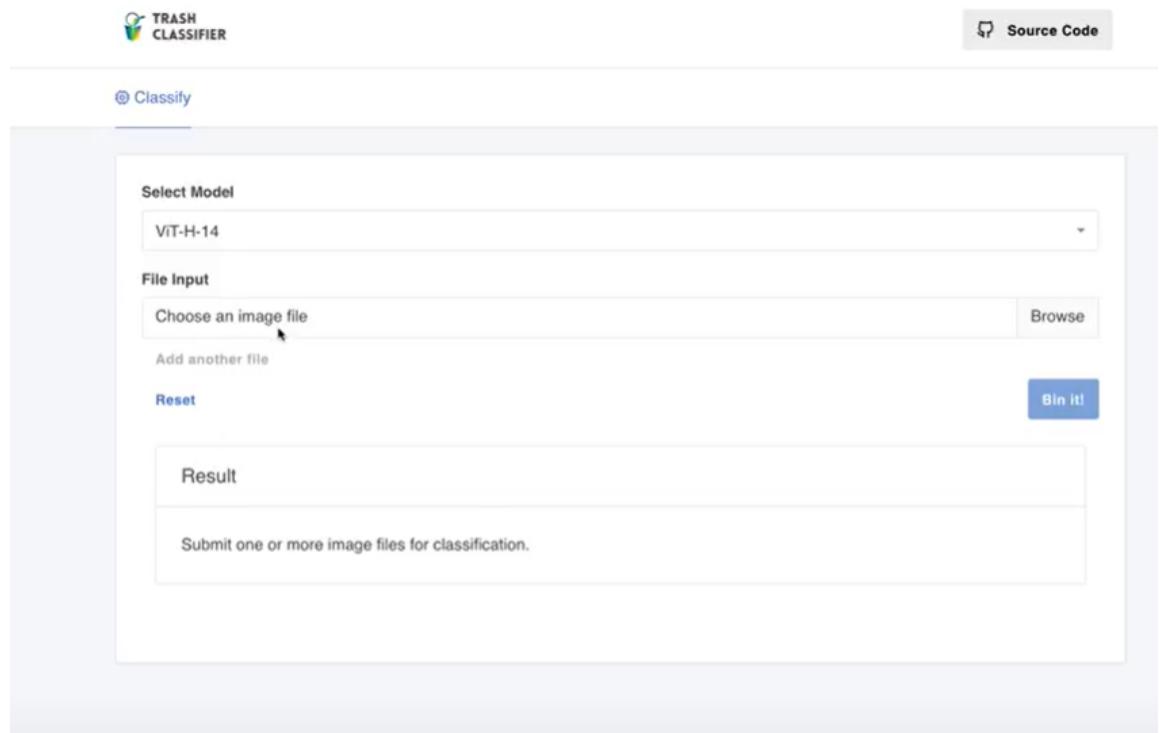
At the time of project completion, the scope consisted of getting the server to run on a local machine, such as the Data Science Club Server. This has both the website for model inference, as well as a REST API that can handle, inference, training, retrieving and downloading models, and annotating new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

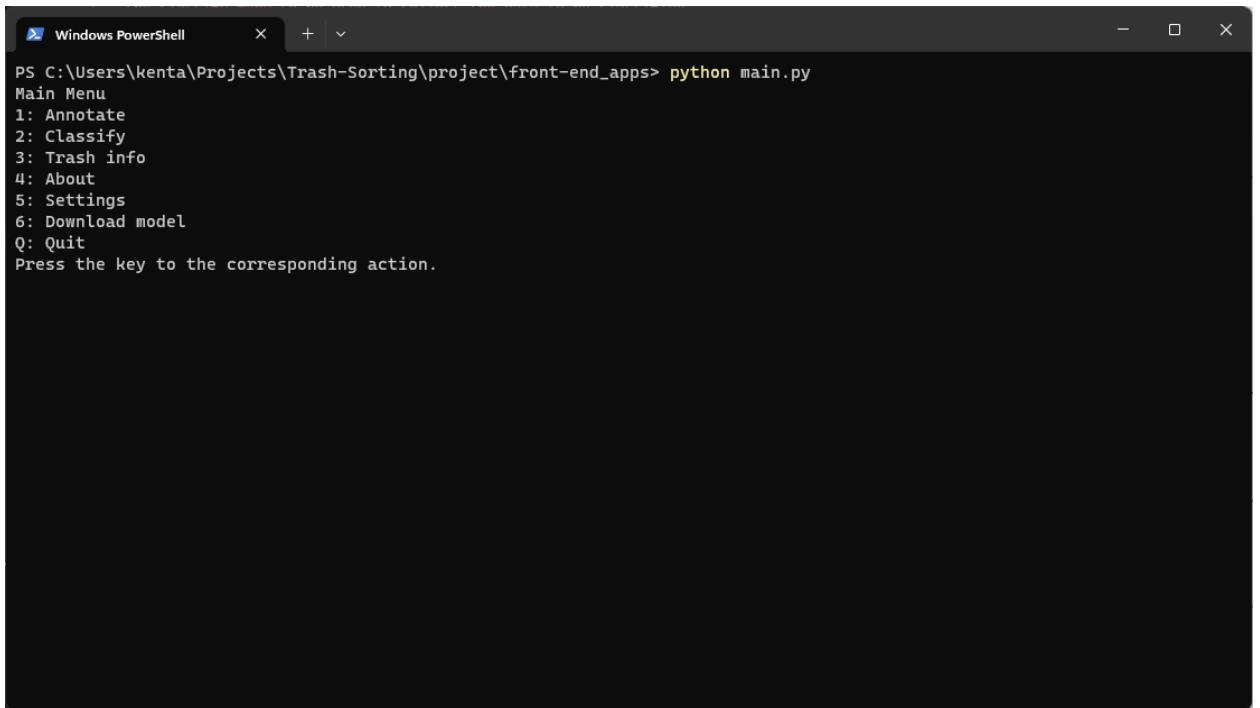
- a. Code files for the screenshot
 - i. Trash-Sorting\project\front-end_apps\website\react-website\src\HomePag e.react.jsclassifyForm.js
 - ii. Trash-Sorting\project\front-end_apps\website\react-website\src\SiteWrap per.react.js
 - iii. Trash-Sorting\project\front-end_apps\website\react-website\src\classifyFo rm.js



2. Command line interface (User view)

a. Main Menu

i. Trash-Sorting\project\front-end_apps\main.py



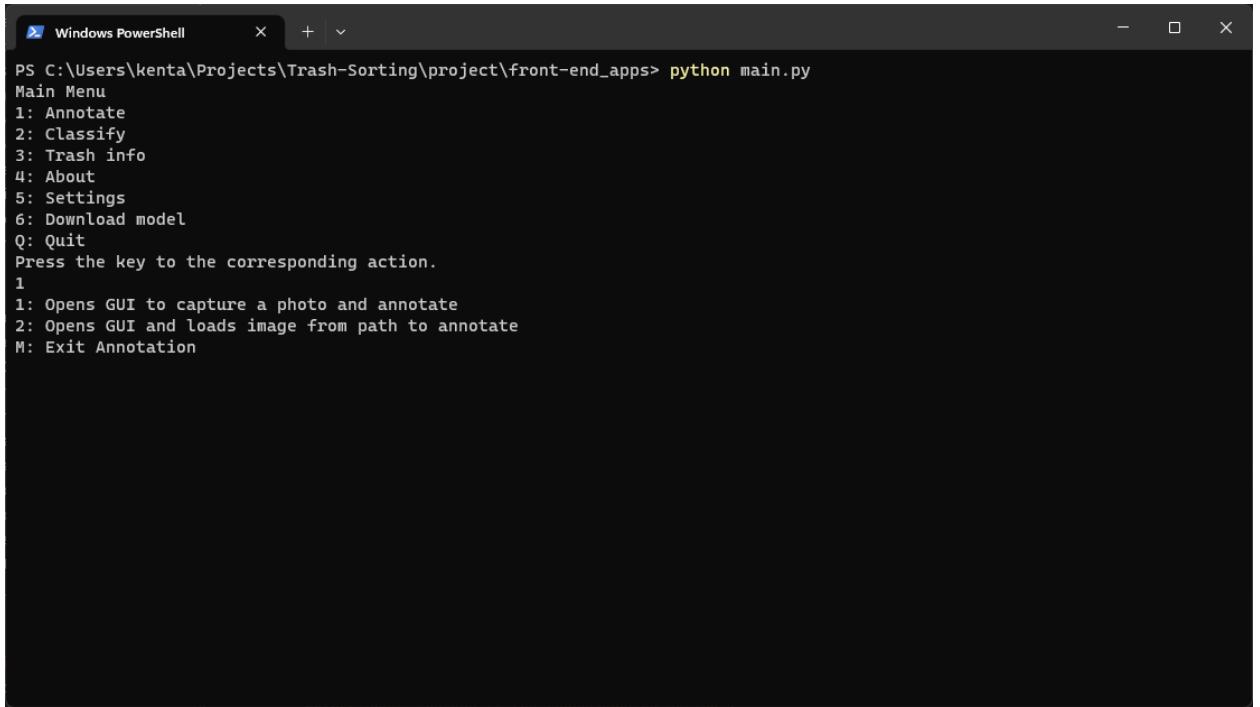
A screenshot of a Windows PowerShell window titled "Windows PowerShell". The window shows the command "PS C:\Users\kenta\Projects\Trash-Sorting\project\front-end_apps> python main.py" followed by the application's main menu. The menu items are:

- Main Menu
- 1: Annotate
- 2: Classify
- 3: Trash info
- 4: About
- 5: Settings
- 6: Download model
- Q: Quit

Below the menu, the text "Press the key to the corresponding action." is displayed.

b. Annotate

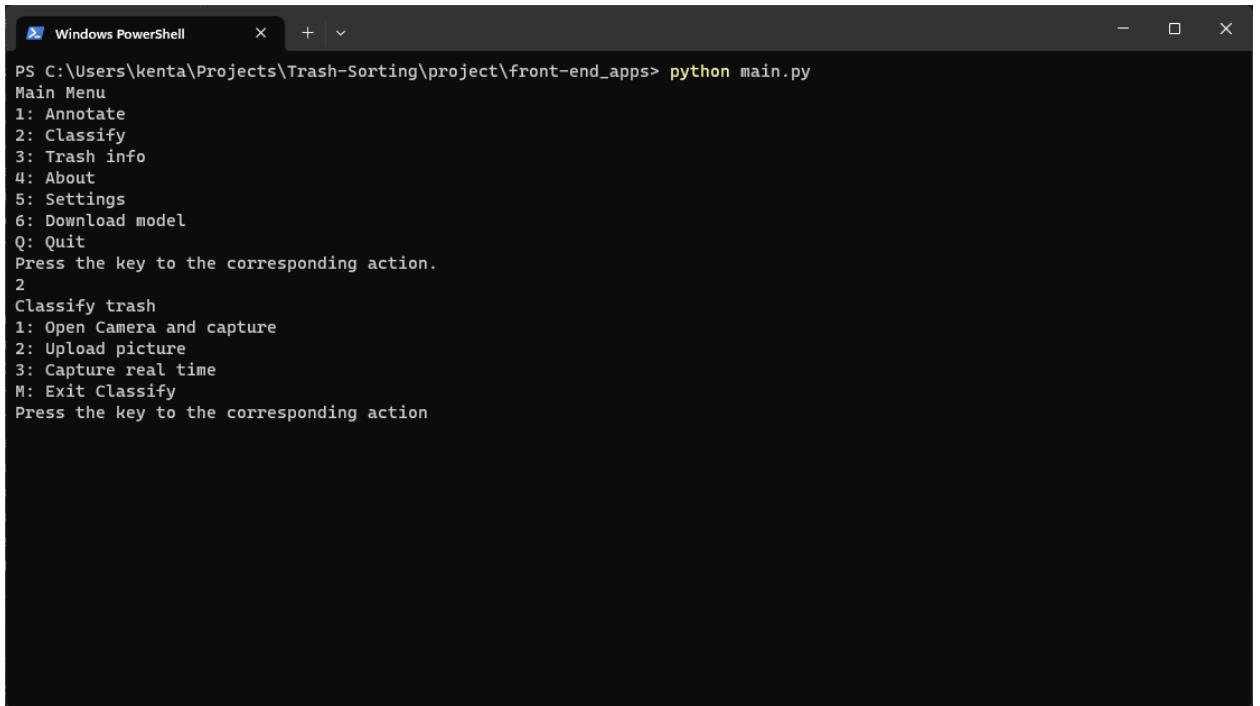
i. Trash-Sorting\project\front-end_apps\annotate.py



```
PS C:\Users\kenta\Projects\Trash-Sorting\project\front-end_apps> python main.py
Main Menu
1: Annotate
2: Classify
3: Trash info
4: About
5: Settings
6: Download model
Q: Quit
Press the key to the corresponding action.
1
1: Opens GUI to capture a photo and annotate
2: Opens GUI and loads image from path to annotate
M: Exit Annotation
```

c. Classify

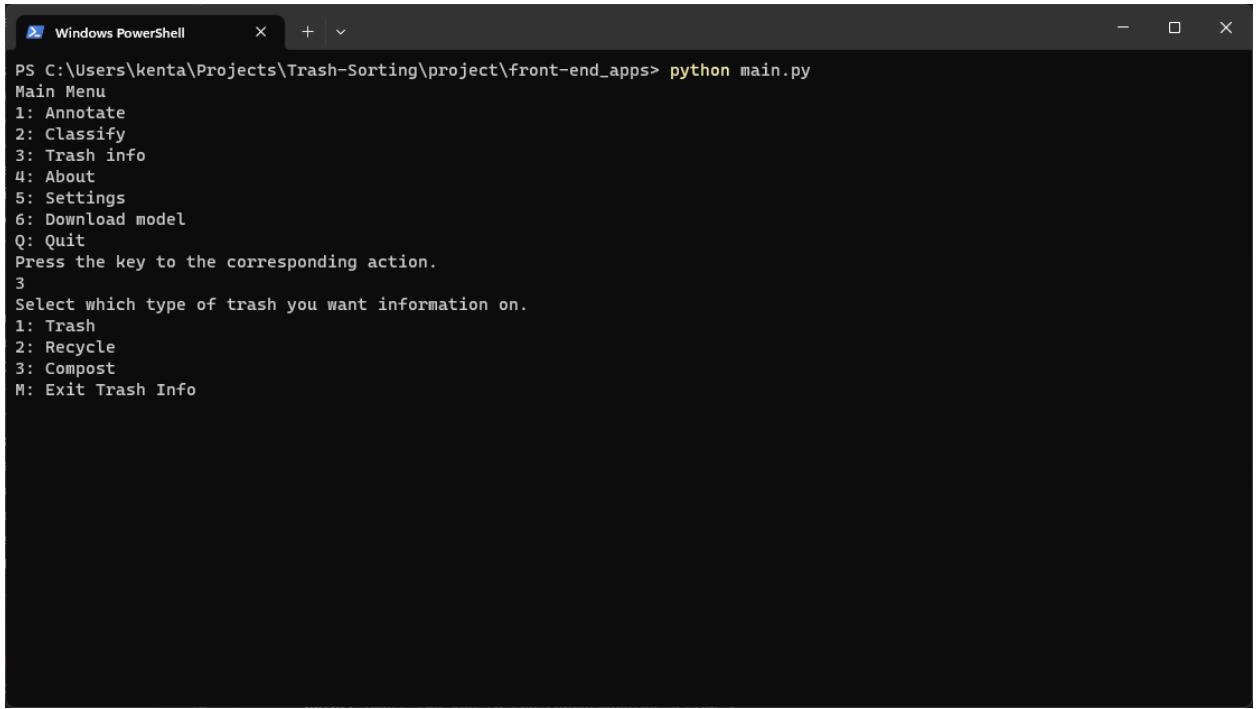
i. Trash-Sorting\project\front-end_apps\classify.py



```
PS C:\Users\kenta\Projects\Trash-Sorting\project\front-end_apps> python main.py
Main Menu
1: Annotate
2: Classify
3: Trash info
4: About
5: Settings
6: Download model
Q: Quit
Press the key to the corresponding action.
2
Classify trash
1: Open Camera and capture
2: Upload picture
3: Capture real time
M: Exit Classify
Press the key to the corresponding action
```

d. Trash info

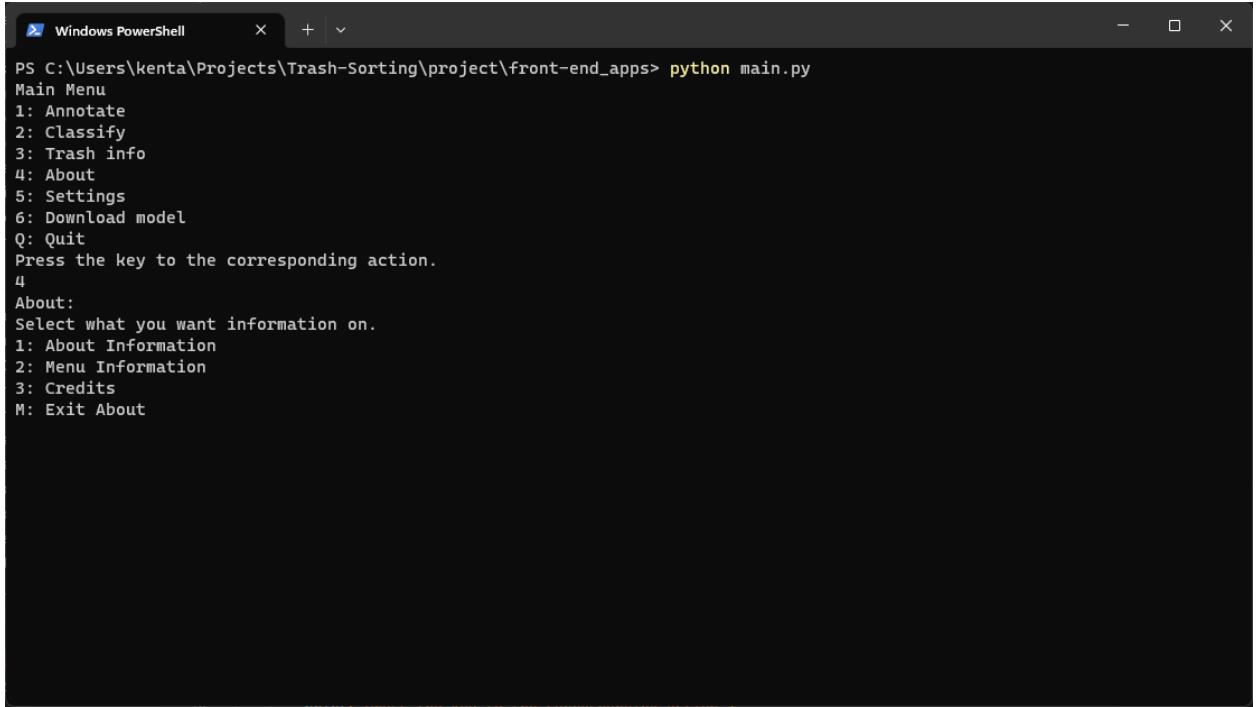
i. Trash-Sorting\project\front-end_apps\trash_info.py



```
PS C:\Users\kenta\Projects\Trash-Sorting\project\front-end_apps> python main.py
Main Menu
1: Annotate
2: Classify
3: Trash info
4: About
5: Settings
6: Download model
Q: Quit
Press the key to the corresponding action.
3
Select which type of trash you want information on.
1: Trash
2: Recycle
3: Compost
M: Exit Trash Info
```

e. About

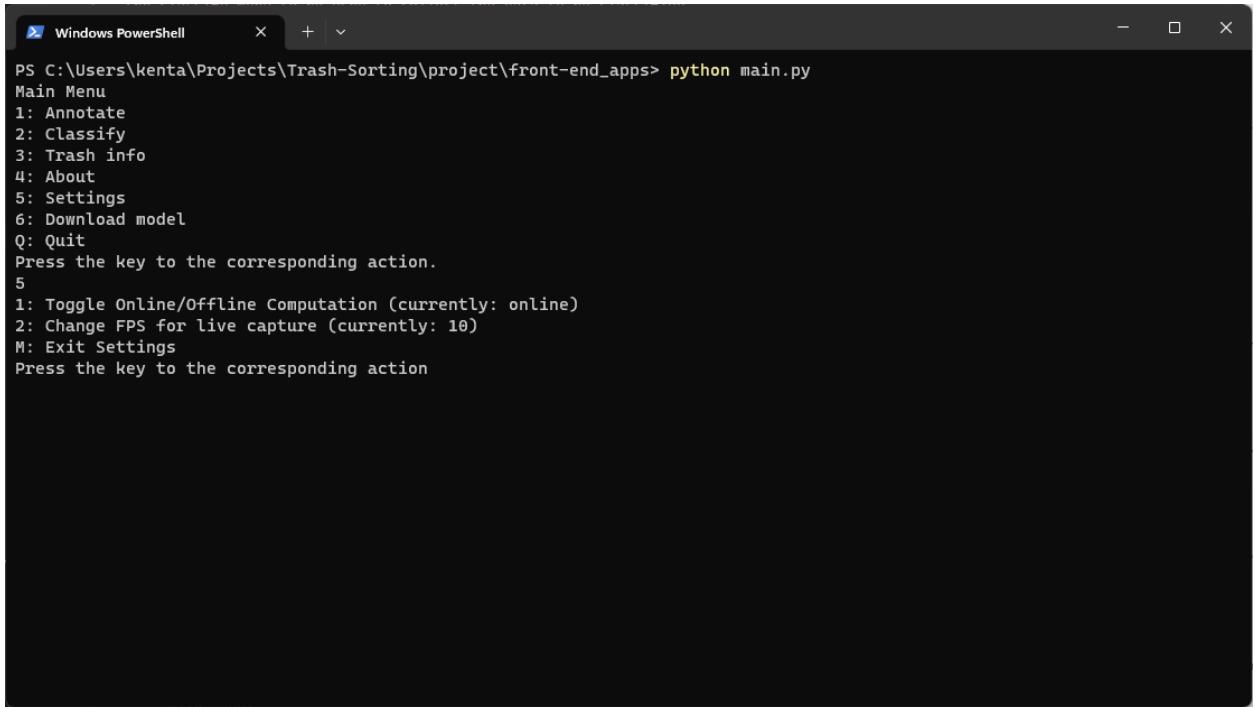
i. Trash-Sorting\project\front-end_apps\about.py



```
PS C:\Users\kenta\Projects\Trash-Sorting\project\front-end_apps> python main.py
Main Menu
1: Annotate
2: Classify
3: Trash info
4: About
5: Settings
6: Download model
Q: Quit
Press the key to the corresponding action.
4
About:
Select what you want information on.
1: About Information
2: Menu Information
3: Credits
M: Exit About
```

f. Settings

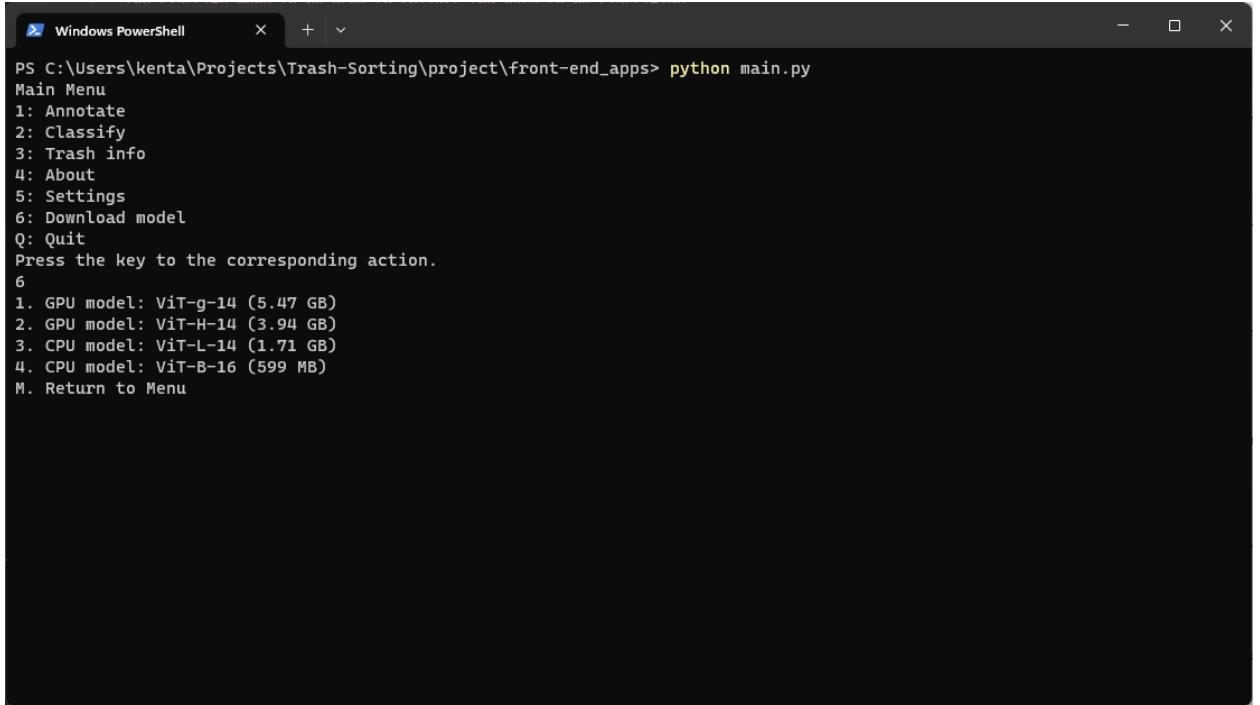
i. Trash-Sorting\project\front-end_apps\settings.py



```
PS C:\Users\kenta\Projects\Trash-Sorting\project\front-end_apps> python main.py
Main Menu
1: Annotate
2: Classify
3: Trash info
4: About
5: Settings
6: Download model
Q: Quit
Press the key to the corresponding action.
5
1: Toggle Online/Offline Computation (currently: online)
2: Change FPS for live capture (currently: 10)
M: Exit Settings
Press the key to the corresponding action
```

g. Download Model

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



```
PS C:\Users\kenta\Projects\Trash-Sorting\project\front-end_apps> python main.py
Main Menu
1: Annotate
2: Classify
3: Trash info
4: About
5: Settings
6: Download model
Q: Quit
Press the key to the corresponding action.
6
1. GPU model: ViT-g-14 (5.47 GB)
2. GPU model: ViT-H-14 (3.94 GB)
3. CPU model: ViT-L-14 (1.71 GB)
4. CPU model: ViT-B-16 (599 MB)
M. Return to Menu
```

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