CS551Q Database-Driven Django Application Design and Development Team Delta Group Report

A database-driven Django web application with an assigned carbon dioxide (CO2) emission dataset[1] is developed. There are five steps in the web development process assessing, planning, developing, testing and deploying[2]. In the assessing and planning stages, we worked as a group on requirements studying and model designing with a two-week limit for the whole project. Requirements from the professor were frequently revisited. Model and front-end sketches (Appendices 1.1-1.4) were done together to start with shared understanding.

With limited developing time, Agile Development Methodology[3] with continuous integration (CI) and continuous deployment (CD) were used to implement iterative and incremental development and delivery[4]. We searched for websites using similar datasets[5] and took inspiration from their data visualisation methods. Plan Do Check Act[4] cycle is used for continuous improvement. The project is split into small tasks with expected outcomes during Plan. After coding during Do, we compared the results obtained with the desired output in Check. After getting the desired result, we push the code to GitHub for version control and repeat with the next feature.

The developing stage is separated into two major parts which are the design and implementation of the database and frontend pages respectively. There are three versions of the database. To satisfy the 2000-7000 data rows requirement, 6032 rows of CO2 emission data for 235 countries for 26 years (1996-2021) in each Comma-Separated Values (CSV) file are chosen. These data are stored in 2 separate CSV files in the data folder for ease of data parsing. The country field also has a separate CSV file to save every unique country value from the other CSV files. In the first version, four models are designed where the country is a separate model which acts as the foreign key linking other models: total emission, per capita emission and source. In the second version, id is added to every model and country_id becomes the foreign key. In the third version, a year model storing numbers from 1996-2021 is added for filtering. Four Class objects are defined: Year, Country, TotalEmission and PerCapitaEmission. Year class is the year included with range 1996-2021. Country shows the countries, incidents or global with produced CO2 emissions. TotalEmission gives the total CO2 emissions by different sources (such as coal, gas, and oil) from different countries annually. PerCapitaEmission gives the per capita CO2 emissions by different sources (such as coal, gas, and oil) for different countries. parse_csv.py is written for parsing the data from CSV into the database after removing all data in the original tables. Pair programming[6] is used when writing models.py. Simple HTML & CSS templates are created to verify items parsed into the database (Appendix 2.1).

There are three versions for the front end. The website's basic layout is developed in the first version (Appendix 2.2) with a basic landing page, total and per capita CO2 emission page showing all the data in tables using pagination, with map and graph tabs. The second version (Appendix 2.3) was obtained after adding all the developed features such as filtering and graph display functions. The third version (Appendix 2.4) is obtained after changing the background image and solving some pagination issues from the search function, also removing the unused part like the map page and adding a data visualization part for showing both graphs together. The landing page composes of a navigation bar with a title. Data visualization was presented in form of tables, line graphs and bar charts. All forms are available for both total and per capita CO2 emission datasets. The table displays the CO2 emissions by different countries from 1996 to 2021; with a search button for year or country. The line graph and bar chart displays the total and per capita CO2 emissions by country over time. It is switched with a button toggle. By selecting a specific country and material, graphic historical CO2 data over time can be seen. Group collaboration is done explicitly in the front-end development, especially in the final development stages. Pair programming[6] during the development of the filter function (Appendix 1.5) at a faster pace[7] referencing some online reference materials[8].

Model-template-view(MTV) pattern, the Django version of the model-view-controller(MVC) pattern, is applied in the implementation. Model is the database of 5 tables, templates (frontend HTML) are created in the templates/emission folder in the emission application following the Django convention and view is the controller (views.py) controlling all functions implemented[9].

For the implementation stage, we created a Django project (mysite) and application (emission), configured Django settings to include Chart.js library, created models for emissions in countries and years with Django ORM, created views for line graph and bar chart using Chart.js library, created templates and URLs for the table, line graph and bar chart using HTML and Chart.js.

For the testing stage, unit tests that come with Django are mainly used. Five test types (model testing, index testing, graph testing and web views testing) and eleven test items are implemented. Five test types are separately stored in the tests folder. First, the index test tests whether different pages can be accessed and connected. The main method uses response.status_code=200[10] to verify the page connectivity. Second, model testing is for four models: Year, Country, TotalEmission, PerCapitaEmission. Models are tested with different data inputs, with importing models at the beginning. Third, the web view test focuses on the successful loading of the form and checking the form content, by creating test data and using assertContains to determine whether the form contains the test data.Fourthly, the filter test, is mainly to test that the selection component and the button component work properly and that the drop-down menu of the selection component contains the correct option data.The last test is the chart test, which is the same as the home page test, except that it adds the ability to jump to a chart with the correct id, e.g. id=myChart, and detects whether the page contains certain fields to jump to the correct page. All the testing is done based on Django's unit tests.

For the deployment stage, Render is used to deploy the Django application; a cloud platform that offers a simple way for web application deployments. Render deployment was done in the first week of website development for CD by solving some errors earlier(Appendix 5).

GitHub is used throughout the development process and everything is pushed to GitHub when finishing a version of the current feature we are working on, implementing CI. A separate master branch is created for storing the workable copy of the code that everyone agreed on (Appendix 3.3-3.4). However, we still encountered a major Git clash in the final development stage which is solved with the help of other classmates by deleting some messed up commits which are caused by pushing all code (git add .) to GitHub before pulling the latest version (git pull). We have learnt to only push the required or amended files to GitHub after this incident and always git pull before git add any files.

In conclusion, we have learnt everything from assessing to deploying a database-driven Django web application with a filtering function and showing visual graphs. Group collaboration and pair programming are practised during the development process. Data filtering and visualisation are studied and applied. The importance and practices of version control are reintroduced. Unit tests are applied and Render is used for website deployment.

Website Links

CS511Q group delta submission folder in Pok Nga Ho's Team Assignment: cs551_delta

Render deployed website: https://delta-emission.onrender.com

Reference

[1] The Devastator, “Emissions by Country,” Kaggle. https://www.kaggle.com/datasets/thedevastator/global-fossil-co2-emissions-by-country-2002-2022?select=GCB2022v27_sources_flat.csv (accessed Mar. 31, 2023).

[2] N. Beacham and K.Musa (2023). CS551S - Web Development Lecture day 1 - Introduction to the Module & How the Internet Works [PowerPoint slides]. Available: https://abdn.blackboard.com/ultra/courses/_56779_1/outline/edit/document/_3587470_1?courseId=_56779_1&view=content

[3] A. Ahmed, S. Ahmad, N. Ehsan, E. Mirza, and S. Z. Sarwar, “Agile software development: Impact on productivity and quality,” in 2010 IEEE International Conference on Management of Innovation & Technology, 2010. Accessed: Feb. 10, 2023. [Online]. Available: http://dx.doi.org/10.1109/icmit.2010.5492703

[4] B. Scharlau (2022). CS551A - Software Engineering Week 1 Lecture 4 - Scrum and XP for Your Team [PowerPoint slides]. Available: https://abdn.blackboard.com/ultra/courses/_56808_1/outline/edit/document/_3570484_1?courseId=_56808_1&view=content

[5] H. Ritchie, M. Roser, and P. Rosado, “CO2 emissions,” Our World in Data. https://ourworldindata.org/co2-emissions (accessed Mar. 31, 2023).

[6] H. Hulkko and P. Abrahamsson, “A multiple case study on the impact of pair programming on product quality,” in Proceedings. 27th International Conference on Software Engineering, 2005. ICSE 2005. Accessed: Mar. 31, 2023. [Online]. Available: http://dx.doi.org/10.1109/icse.2005.1553595

[7] J. E. Hannay, T. Dybå, E. Arisholm, and D. I. K. Sjøberg, “The effectiveness of pair programming: A meta-analysis,” Information and Software Technology, vol. 51, no. 7, pp. 1110–1122, Jul. 2009, doi: 10.1016/j.infsof.2009.02.001.

[8] B. Kumar, “Python Django Search With Dropdown Filter,” Python Guides, Jan. 03, 2023. https://pythonguides.com/python-django-search-with-dropdown-filter/ (accessed Mar. 31, 2023).

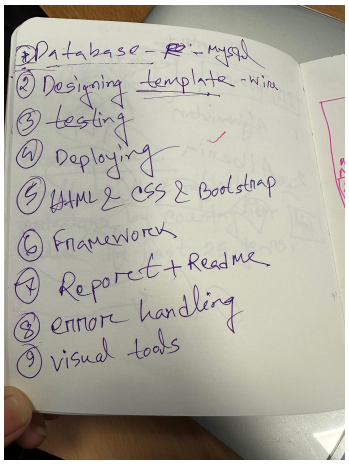
[9] B. Scharlau (2021). CS551Q - Enterprise Software Development Week 2 Lecture 6 - MVC with Django [PowerPoint slides]. Available: https://abdn.blackboard.com/ultra/courses/_56777_1/outline/file/_3572168_1

[10] B. Scharlau(2023). CS551Q - Enterprise Software Development Week 2 Lecture 7 - Testing Models in Django[PowerPoint slides]. Available: https://abdn.blackboard.com/ultra/courses/_56777_1/outline/file/_3574334_1

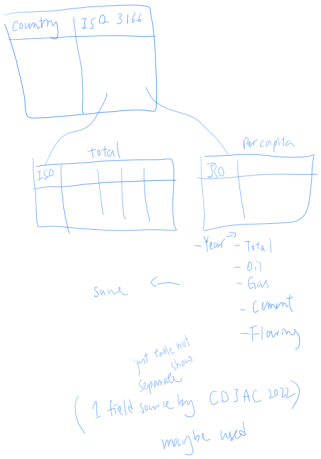
Appendix

1. Sketches

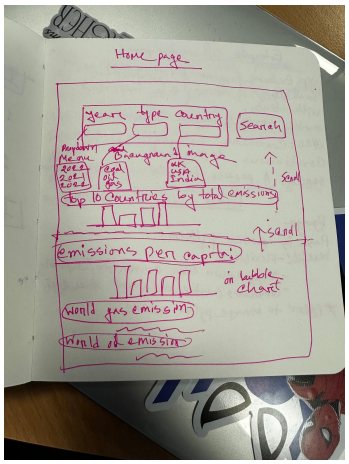
1.1.Project tasks splitting



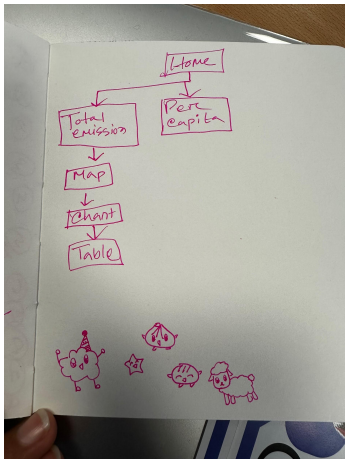
1.2.Database model sketch



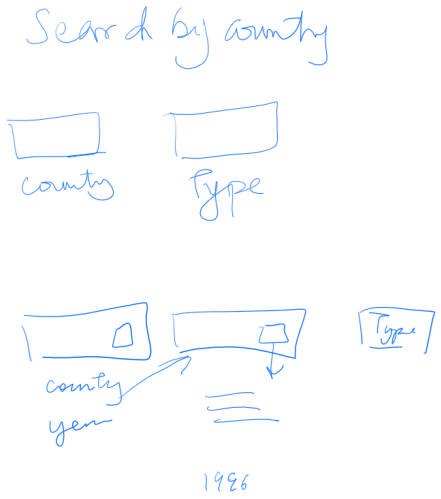
1.3.HTML template sketch



1.4.Navigation path sketch



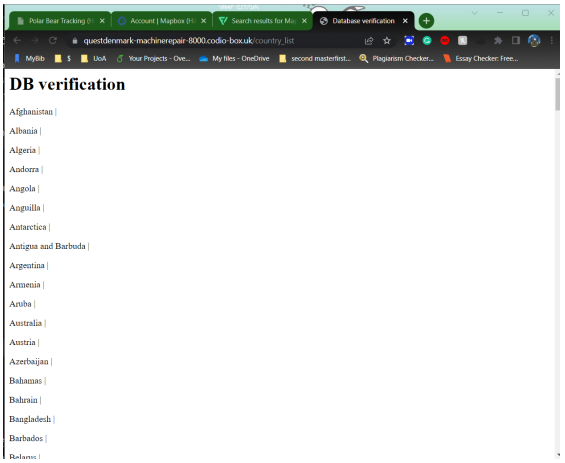
1.5.Filter bar design sketch



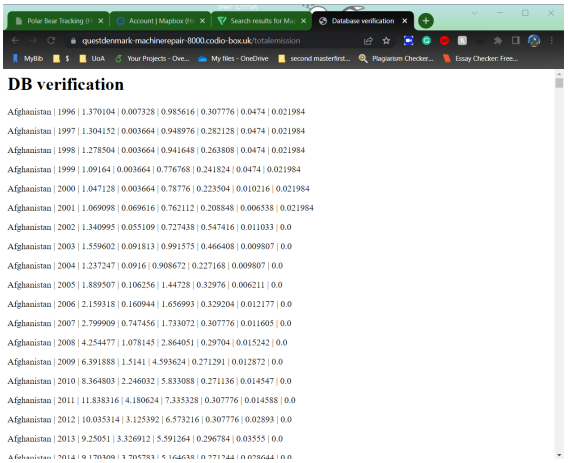
2.Templates

2.1.Database verification pages

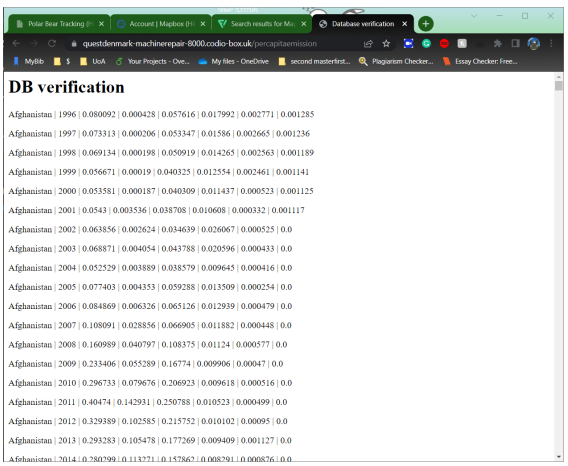
2.1.1.Country table



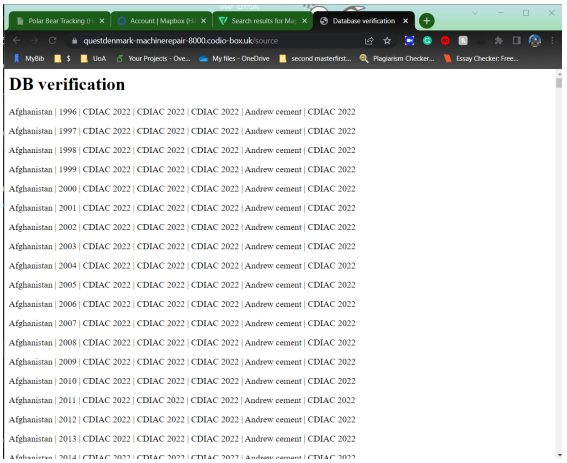
2.1.2.TotalEmission table



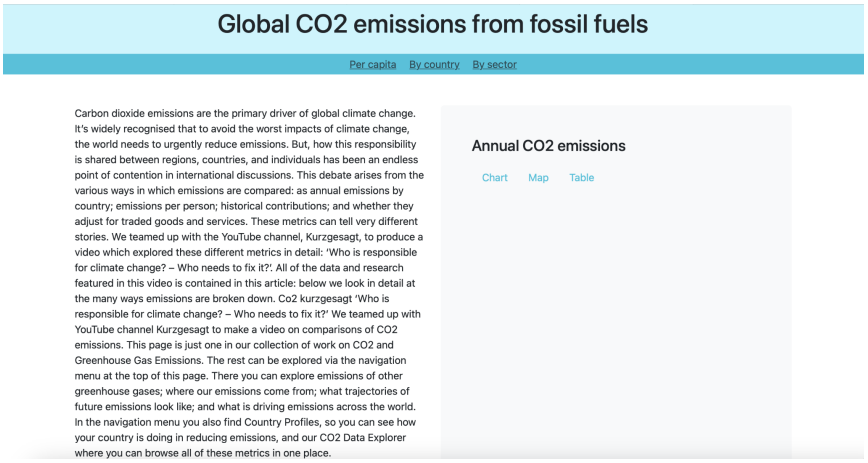
2.1.3.PerCapitaEmission table



2.1.4.Source table

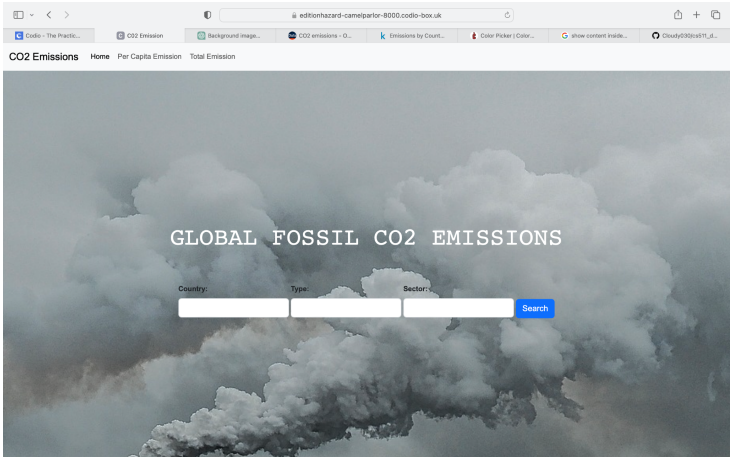


2.2.Version 1

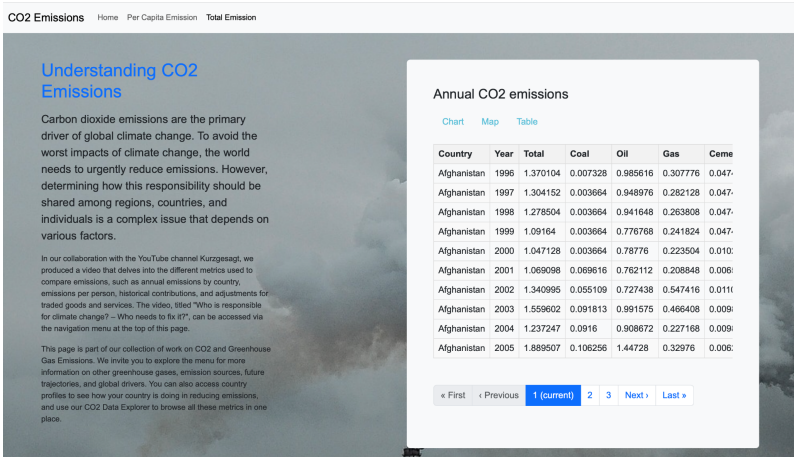


2.3.Version 2

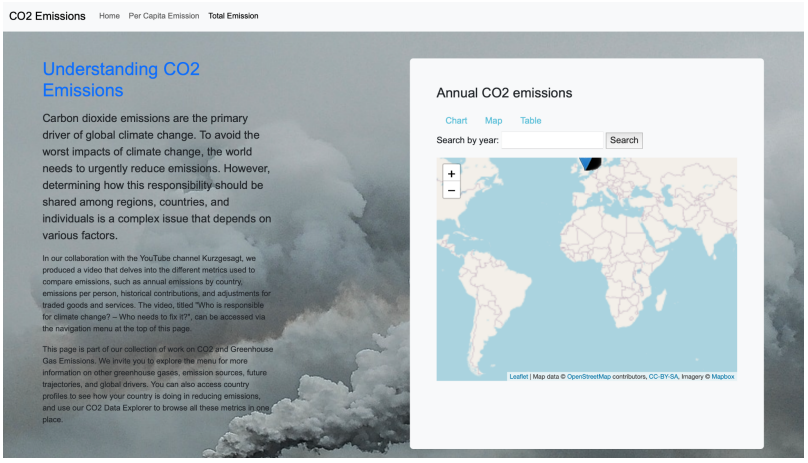
2.3.1.Landing page with search bar



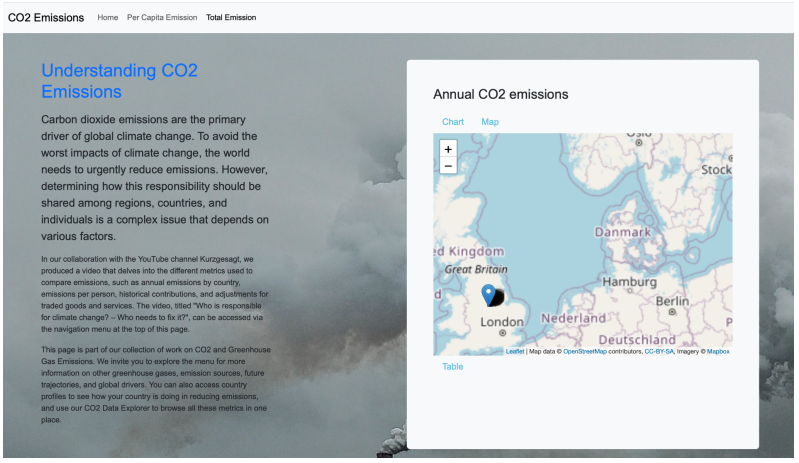
2.3.2.Table page



2.3.3.Map page

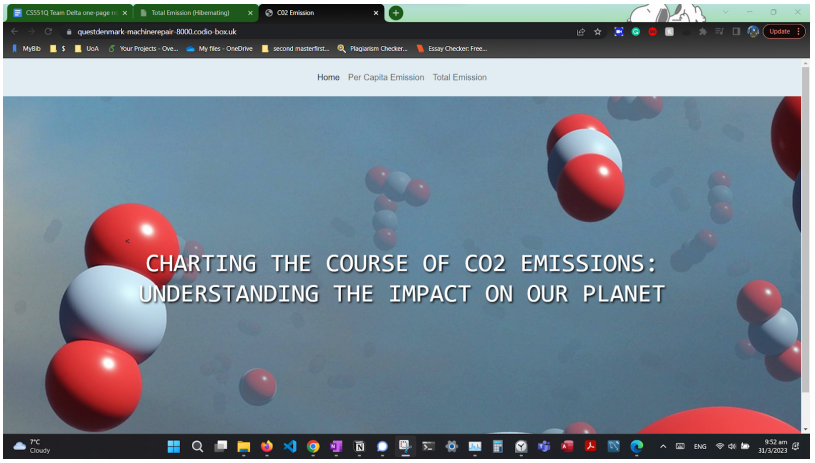


2.3.4. Map page locating in UK

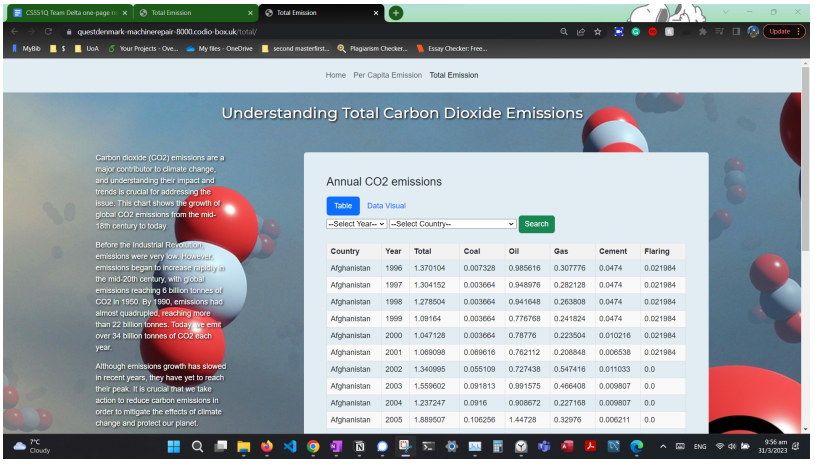


2.4. Version 3

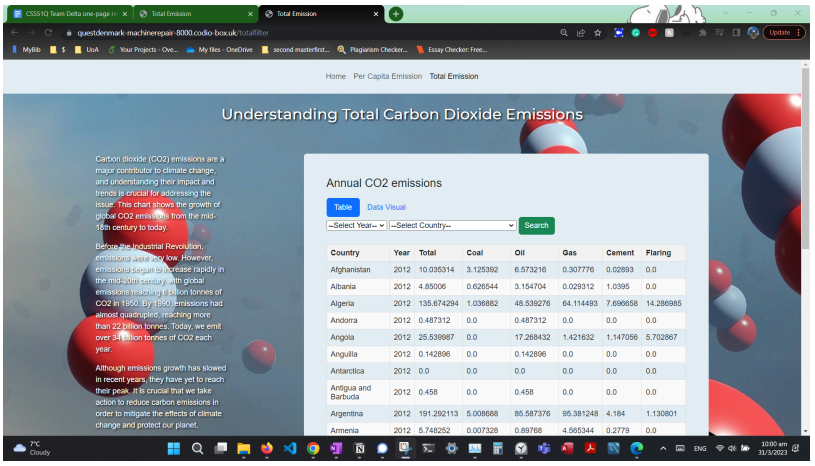
2.4.1.Landing page



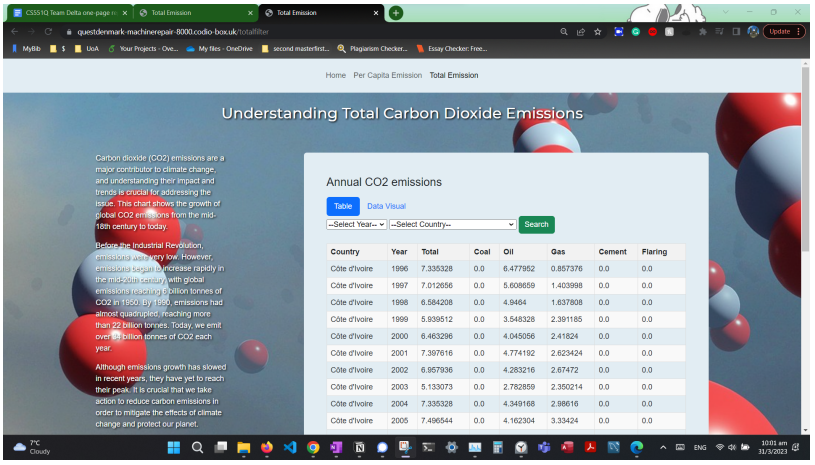
2.4.2.TotalEmission page table



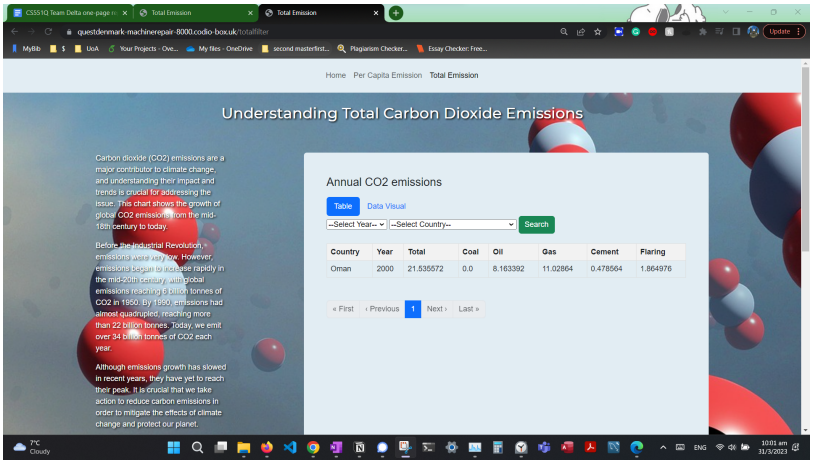
2.4.3.TotalEmission page filter year



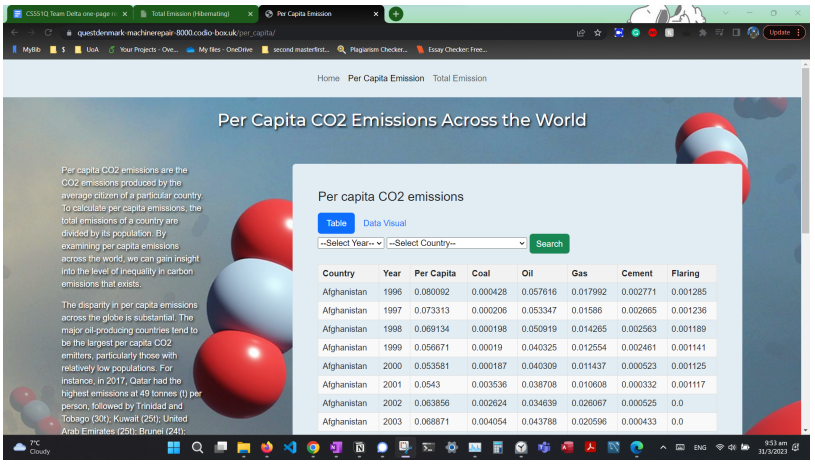
2.4.4.TotalEmission page filter country



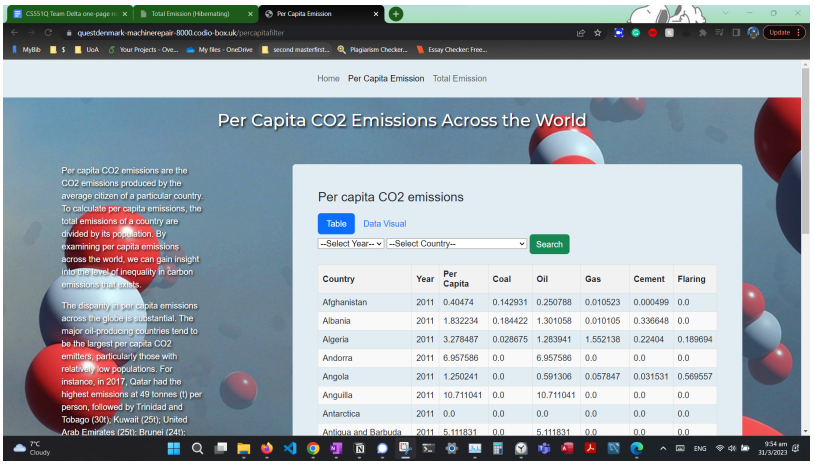
2.4.5.TotalEmission page filter country and year



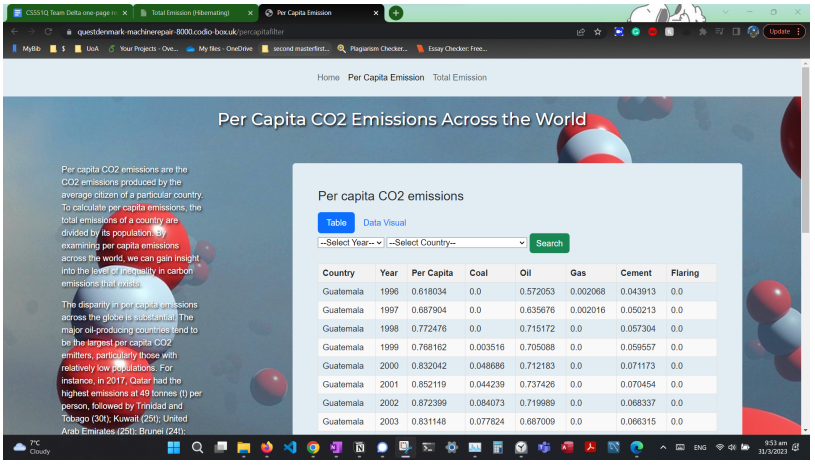
2.4.6.PerCapitaEmission page table



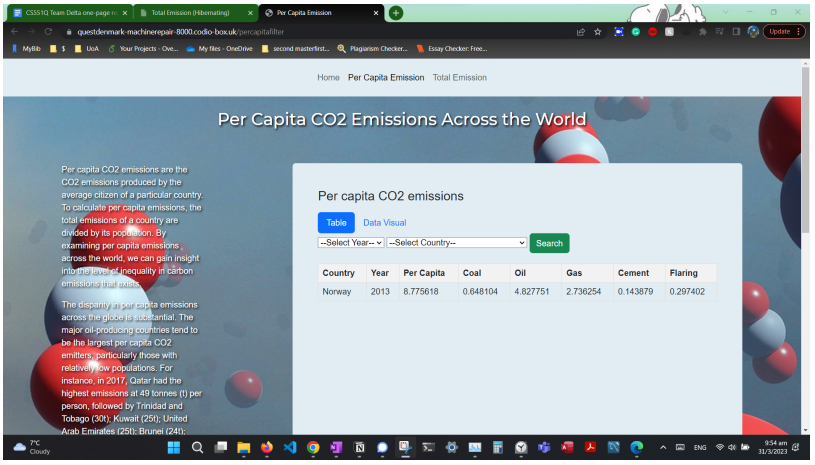
2.4.7.PerCapitaEmission page filter year



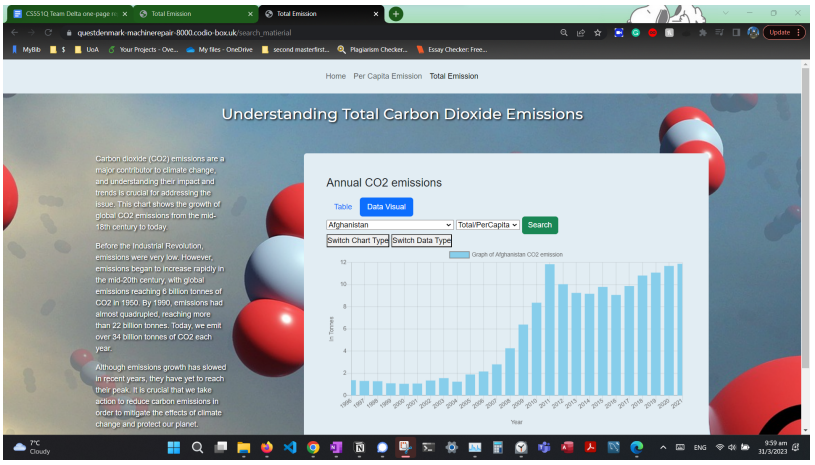
2.4.8.PerCapitaEmission page filter country



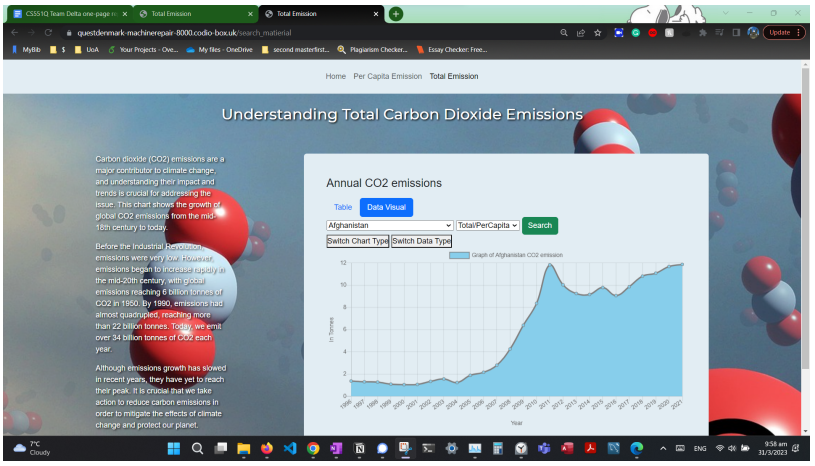
2.4.9.PerCapitaEmission page filter country and year



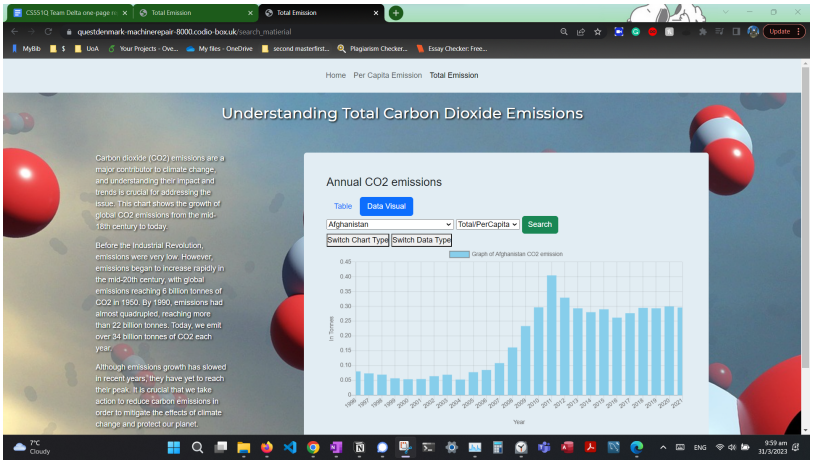
2.4.10.TotalEmission bar graph



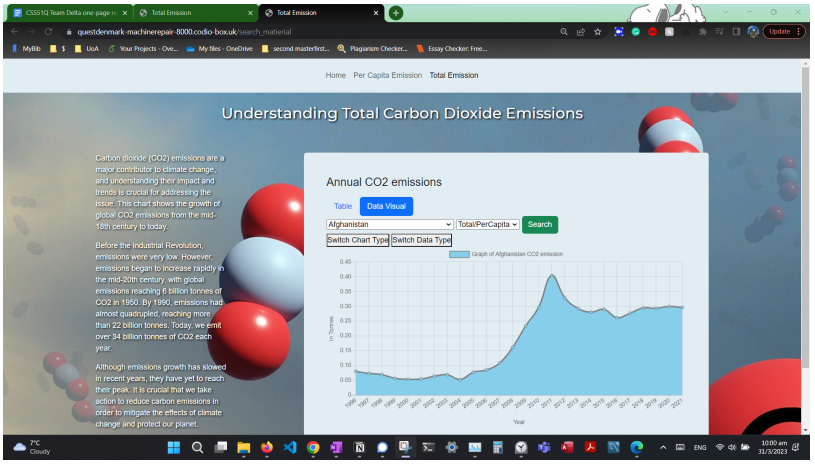
2.4.11.TotalEmission line graph



2.4.12.PerCapitaEmission bar graph



2.4.13.PerCapitaEmission line graph



3. Coding excerpts

3.1.pyenv update

```
codio@questdenmark-machinerepair:~/workspace/cs511_delta$ pyenv update
Updating /home/codio/.pyenv...
remote: Enumerating objects: 22711, done.
remote: Counting objects: 100% (22708/22708), done.
remote: Compressing objects: 100% (5950/5950), done.
remote: Total 22147 (delta 15327), reused 21691 (delta 14916), pack-reused 0
Receiving objects: 100% (22147/22147), 4.16 MiB | 5.43 MiB/s, done.
Resolving deltas: 100% (15327/15327), completed with 299 local objects.
From https://github.com/pyenv/pyenv
 * branch            master     -> FETCH_HEAD
   943015eb..4ef81b5c master     -> origin/master
Updating 943015eb..4ef81b5c
Fast-forward
```

3.2.pyenv install 3.10.7

```
codio@questdenmark-machinerepair:~/workspace/cs511_delta$ pyenv install 3.10.7
Downloading Python-3.10.7.tar.xz...
-> https://www.python.org/ftp/python/3.10.7/Python-3.10.7.tar.xz
Installing Python-3.10.7...
```

3.3.Git push to master branch 1

```
(.venv) codio@questdenmark-machinerepair:~/workspace/cs511_delta$ git checkout main
M       git-log.txt
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
(.venv) codio@questdenmark-machinerepair:~/workspace/cs511_delta$ git pull
Already up to date.
(.venv) codio@questdenmark-machinerepair:~/workspace/cs511_delta$ git checkout master
M       git-log.txt
Switched to branch 'master'
(.venv) codio@questdenmark-machinerepair:~/workspace/cs511_delta$ git add git-log.txt
(.venv) codio@questdenmark-machinerepair:~/workspace/cs511_delta$ git commit -m 'update git log'
[master ba41657] update git log
 1 file changed, 16 insertions(+)
(.venv) codio@questdenmark-machinerepair:~/workspace/cs511_delta$ git push
fatal: The current branch master has no upstream branch.
To push the current branch and set the remote as upstream, use

    git push --set-upstream origin master

(.venv) codio@questdenmark-machinerepair:~/workspace/cs511_delta$  git push --set-upstream origin master
Counting objects: 3, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 966 bytes | 966.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To github.com:Cloudy030/cs511_delta.git
   754a7e8..ba41657  master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.
(.venv) codio@questdenmark-machinerepair:~/workspace/cs511_delta$
```

3.4.Git push to master branch 2

```
(.venv) codio@questdenmark-machinerepair:~/workspace/cs511_delta$ git checkout master
Switched to branch 'master'
Your branch is up to date with 'origin/master'.
(.venv) codio@questdenmark-machinerepair:~/workspace/cs511_delta$ git merge main
Auto-merging git-log.txt
CONFLICT (content): Merge conflict in git-log.txt
Automatic merge failed; fix conflicts and then commit the result.
(.venv) codio@questdenmark-machinerepair:~/workspace/cs511_delta$ git pull origin master
error: Pulling is not possible because you have unmerged files.
hint: Fix them up in the work tree, and then use 'git add/rm <file>'
hint: as appropriate to mark resolution and make a commit.
fatal: Exiting because of an unresolved conflict.
(.venv) codio@questdenmark-machinerepair:~/workspace/cs511_delta$ git branch
  main
* master
(.venv) codio@questdenmark-machinerepair:~/workspace/cs511_delta$ ls
commands.txt  db.sqlite3  git-log.txt  manage.py  README.md
data          emission    LICENSE      mysite
(.venv) codio@questdenmark-machinerepair:~/workspace/cs511_delta$ git add git-log.txt
(.venv) codio@questdenmark-machinerepair:~/workspace/cs511_delta$ git commit -m 'test merge
branch'
[master f83c37e] test merge branch
(.venv) codio@questdenmark-machinerepair:~/workspace/cs511_delta$ git merge origin main
Already up to date.
(.venv) codio@questdenmark-machinerepair:~/workspace/cs511_delta$ git push
Counting objects: 3, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 346 bytes | 346.00 KiB/s, done.
Total 3 (delta 2), reused 0 (delta 0)
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To github.com:Cloudy030/cs511_delta.git
   ba41657..f83c37e  master -> master
(.venv) codio@questdenmark-machinerepair:~/workspace/cs511_delta$
```

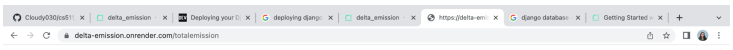
3.5.Git reset --hard

```
(.venv) codio@questdenmark-machinerepair:~/workspace/cs511_delta$ git reset --hard
HEAD is now at 32bc4f2 country.csv
(.venv) codio@questdenmark-machinerepair:~/workspace/cs511_delta$ git pull
Updating 32bc4f2..f45435d
Fast-forward
 .../__pycache__/__init__.cpython-310.pyc           |   Bin 7593 -> 7593 bytes
 .../asgiref/__pycache__/__init__.cpython-310.pyc   |   Bin 202 -> 202 bytes
 .../current_thread_executor.cpython-310.pyc        |   Bin 2715 -> 2715 bytes
 .../asgiref/__pycache__/local.cpython-310.pyc      |   Bin 4134 -> 4134 bytes
 .../asgiref/__pycache__/sync.cpython-310.pyc       |   Bin 12662 -> 12662 bytes
 .../django/__pycache__/__init__.cpython-310.pyc    |   Bin 1000 -> 1000 bytes
 .../django/__pycache__/shortcuts.cpython-310.pyc   |   Bin 4411 -> 4411 bytes
 .../apps/__pycache__/__init__.cpython-310.pyc      |   Bin 291 -> 291 bytes
 .../apps/__pycache__/config.cpython-310.pyc        |   Bin 6317 -> 6317 bytes
 .../apps/__pycache__/registry.cpython-310.pyc      |   Bin 13074 -> 13074 bytes
 .../conf/__pycache__/__init__.cpython-310.pyc      |   Bin 10332 -> 10332 bytes
```

3.6.python3 manage.py dbshell

```
(.venv) codio@questdenmark-machinerepair:~/workspace/cs511_delta$ py
thon3 manage.py dbshell
SQLite version 3.22.0 2018-01-22 18:45:57
Enter ".help" for usage hints.
sqlite> .table
auth_group                  django_content_type
auth_group_permissions      django_migrations
auth_permission             django_session
auth_user                   emission_country
auth_user_groups            emission_percapitaemission
auth_user_user_permissions  emission_source
django_admin_log            emission_totalemission
sqlite>
```

4. Error pages getting EOF when first deployed to Render


delta-emission.onrender.com/totalemission

[]