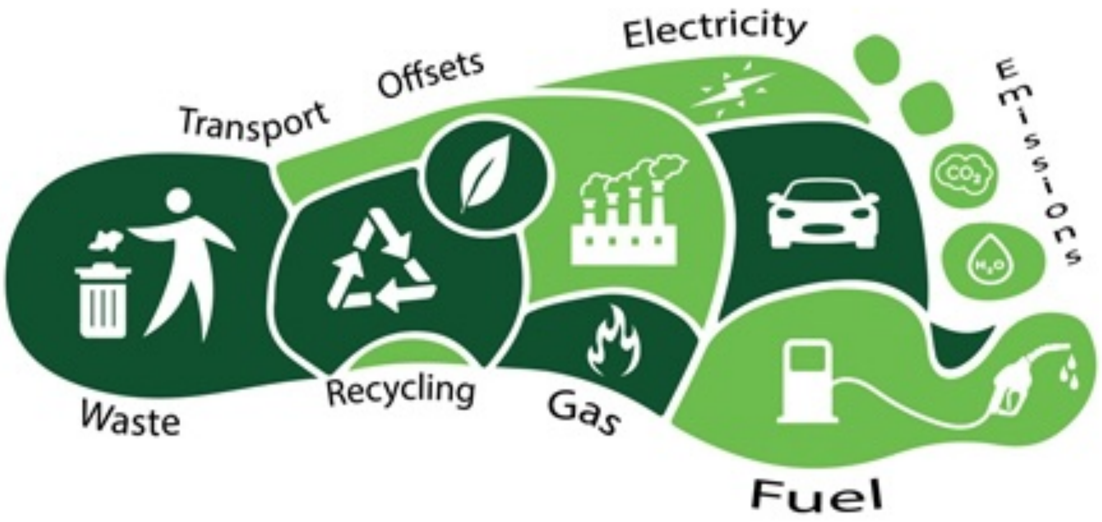
Computer Programming Final Report

Carbon footprint calculator



Seyyed Jalal Tabatabaee (GH1033801)

Winter 2024

Computer Programing Report

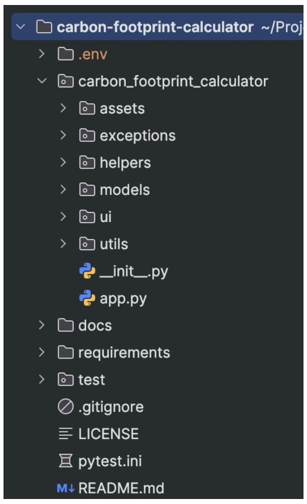
Carbon footprint calculator

Introduction

Nowadays carbon footprint is a global issue, and it needs more attention. A carbon footprint is calculated value or index that makes it possible to compare the total amount of greenhouse gases that an activity, product, company or country adds to atmosphere[1]. In this project, we aimed to develop an application which companies can input their data consists of *Energy usage, waste* and *business travel.* Based on the inputed data, the application calculates the total amount of produced carbons in kgCO2 unit per year. Finally, charts and comprehensive reports generates for the company and across companies which will be discussed further.

Project Structure

This project is written with *Python 3.13* and for the GUI part, *PyQt6* library is used. The Project is pushed into Github and can be found via this GitHub link : https://github.com/shayantabatabaee/carbon-footprint-calculator .
The project is structured as follows :

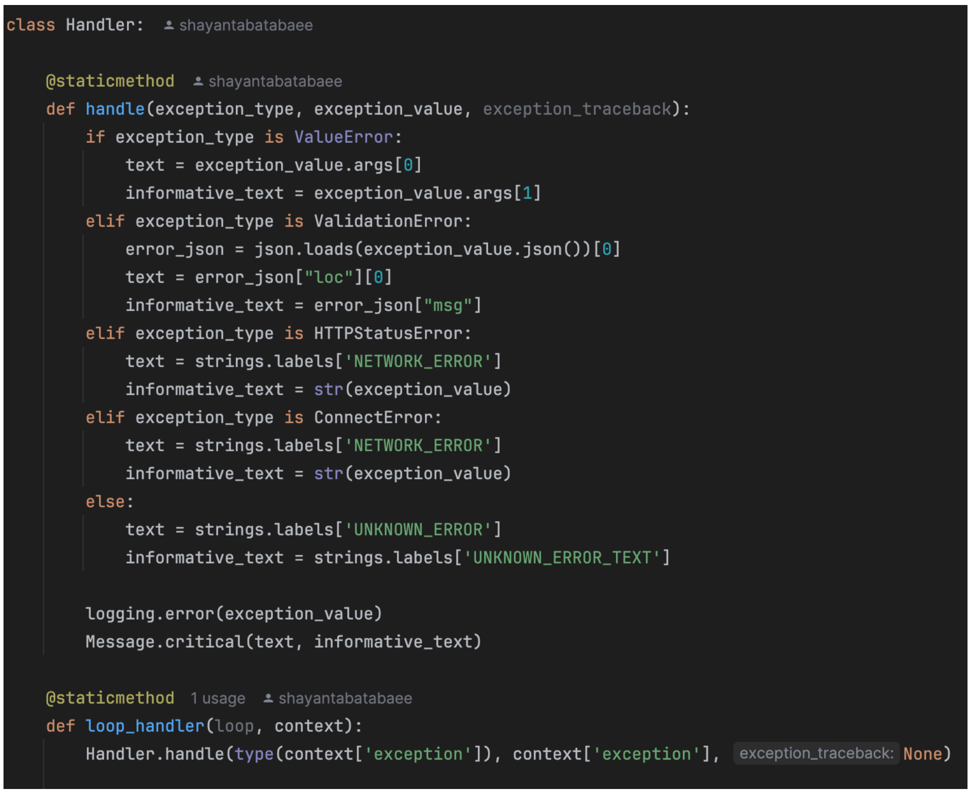


```
carbon-footprint-calculator ~/Proj
  .env
  carbon_footprint_calculator
    assets
    exceptions
    helpers
    models
    ui
    utils
    __init__.py
    app.py
  docs
  requirements
  test
  .gitignore
  LICENSE
  pytest.ini
  README.md
```

• carbon_footprint_calculator: This folder contains main codes which modularized based on categories of responsibilities.
• docs: The images and sample reports and anything related to docs goes into this folder.
• requirements: This folder includes requirement text files for main application and tests.
• test: All tests related to codes goes into this folder.

Main Code

To have a modularized code, I used classes and tried to consider OOP[1] and SOLID[2] programming best practices. Also it is worth mentioning that to prevent from breaking changes and keep the stability of the application, tests which include *unit-test* and *integration-test* added. The library which is used for writing tests names *pytest*.
To setup and run application please kindly follow the README.md file beside the project. In the following part, examples of some part of code will be discussed.
For example as the following image demonstrates, to prevent crashes for application and show appropriate messages to the user, centralized exception handler used which for some known exceptions show message to user and for the unknown exceptions log errors in the console and shows general message.



```
class Handler:  shayantabatabaee

    @staticmethod  shayantabatabaee
    def handle(exception_type, exception_value, exception_traceback):
        if exception_type is ValueError:
            text = exception_value.args[0]
            informative_text = exception_value.args[1]
        elif exception_type is ValidationError:
            error_json = json.loads(exception_value.json())[0]
            text = error_json["loc"][0]
            informative_text = error_json["msg"]
        elif exception_type is HTTPStatusError:
            text = strings.labels['NETWORK_ERROR']
            informative_text = str(exception_value)
        elif exception_type is ConnectError:
            text = strings.labels['NETWORK_ERROR']
            informative_text = str(exception_value)
        else:
            text = strings.labels['UNKNOWN_ERROR']
            informative_text = strings.labels['UNKNOWN_ERROR_TEXT']

        logging.error(exception_value)
        Message.critical(text, informative_text)

    @staticmethod  1 usage  shayantabatabaee
    def loop_handler(loop, context):
        Handler.handle(type(context['exception']), context['exception'], exception_traceback: None)
```

[1] https://www.geeksforgeeks.org/introduction-of-object-oriented-programming/

[2] https://www.digitalocean.com/community/conceptual-articles/s-o-l-i-d-the-first-five-principles-of-object-oriented-design

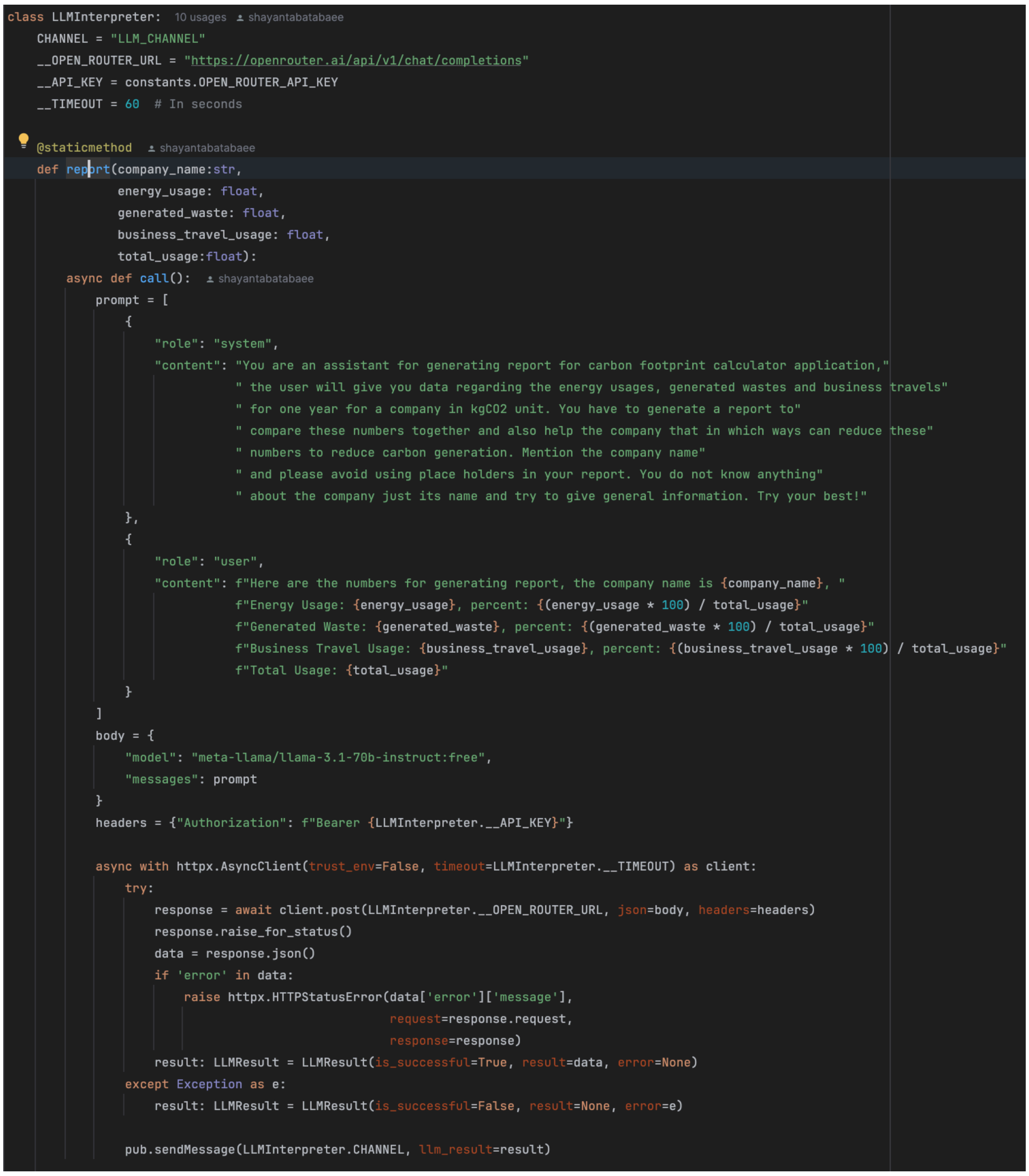
For generating report for a single company, the application communicate to an external LLM which uses *meta llama 3.1* with *70B* parameters that is also *instruction tuned*. This model is hosted at openrouter.ai and can be used freely. Just note that the application needs an api key for calling service, this api key should be set with OPEN_ROUTER_API_KEY name via environment variable. Here is the image of LLMInterpreter class which is responsible for sending prompt to model and get the result.
Generally, calling services is a blocking operation which can lead to block the entire UI, to prevent there are only two options for that :

1. Using Threads
2. Using asynchronous programming

For the sake of simplicity, I decided to go with async approach and publish the result into a channel whenever the result is ready which is called observer design pattern. In the UI, the user hit the submit button and a loading will be shown, the generated PDF which includes the report and chart will pop up whenever the result is ready. Communication to the model is done via a POST request. The inputs include headers for api key and body which has prompt for two different roles :

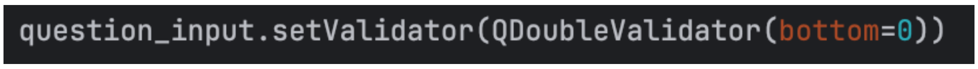
1. System
2. User

For publishing result into a channel *pypubsub* library is used. The full code for LLMInterpreter class can be found in the following image. On the other hand, generating report across companies has been done using a static template. In both ways, the content is written as markdown language, convert to pdf and finally rendered in a web view.



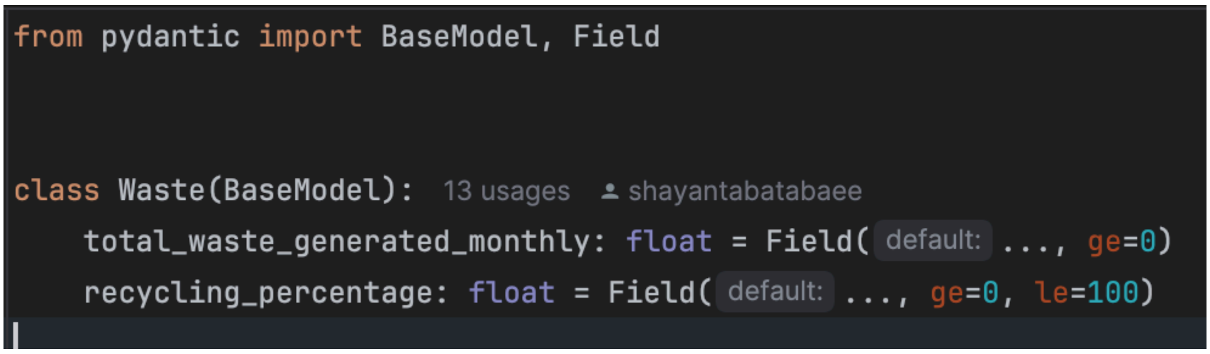
```python
class LLMInterpreter:  10 usages  shayantabatabaee
    CHANNEL = "LLM_CHANNEL"
    __OPEN_ROUTER_URL = "https://openrouter.ai/api/v1/chat/completions"
    __API_KEY = constants.OPEN_ROUTER_API_KEY
    __TIMEOUT = 60  # In seconds

    @staticmethod  shayantabatabaee
    def report(company_name:str,
               energy_usage: float,
               generated_waste: float,
               business_travel_usage: float,
               total_usage:float):
        async def call():  shayantabatabaee
            prompt = [
                {
                    "role": "system",
                    "content": "You are an assistant for generating report for carbon footprint calculator application,"
                               " the user will give you data regarding the energy usages, generated wastes and business travels"
                               " for one year for a company in kgCO2 unit. You have to generate a report to"
                               " compare these numbers together and also help the company that in which ways can reduce these"
                               " numbers to reduce carbon generation. Mention the company name"
                               " and please avoid using place holders in your report. You do not know anything"
                               " about the company just its name and try to give general information. Try your best!"
                },
                {
                    "role": "user",
                    "content": f"Here are the numbers for generating report, the company name is {company_name}, "
                               f"Energy Usage: {energy_usage}, percent: {(energy_usage * 100) / total_usage}"
                               f"Generated Waste: {generated_waste}, percent: {(generated_waste * 100) / total_usage}"
                               f"Business Travel Usage: {business_travel_usage}, percent: {(business_travel_usage * 100) / total_usage}"
                               f"Total Usage: {total_usage}"
                }
            ]
            body = {
                "model": "meta-llama/llama-3.1-70b-instruct:free",
                "messages": prompt
            }
            headers = {"Authorization": f"Bearer {LLMInterpreter.__API_KEY}"}

            async with httpx.AsyncClient(trust_env=False, timeout=LLMInterpreter.__TIMEOUT) as client:
                try:
                    response = await client.post(LLMInterpreter.__OPEN_ROUTER_URL, json=body, headers=headers)
                    response.raise_for_status()
                    data = response.json()
                    if 'error' in data:
                        raise httpx.HTTPStatusError(data['error']['message'],
                                                    request=response.request,
                                                    response=response)
                    result: LLMResult = LLMResult(is_successful=True, result=data, error=None)
                except Exception as e:
                    result: LLMResult = LLMResult(is_successful=False, result=None, error=e)

                pub.sendMessage(LLMInterpreter.CHANNEL, llm_result=result)
```

Since it is crucial to have validation on user inputed data in each application, I tried to have different stage of validation on data. Firstly, because inputing negative numbers is prohibited, I limited the input type to only positive double variables in the UI with the following code snippet.



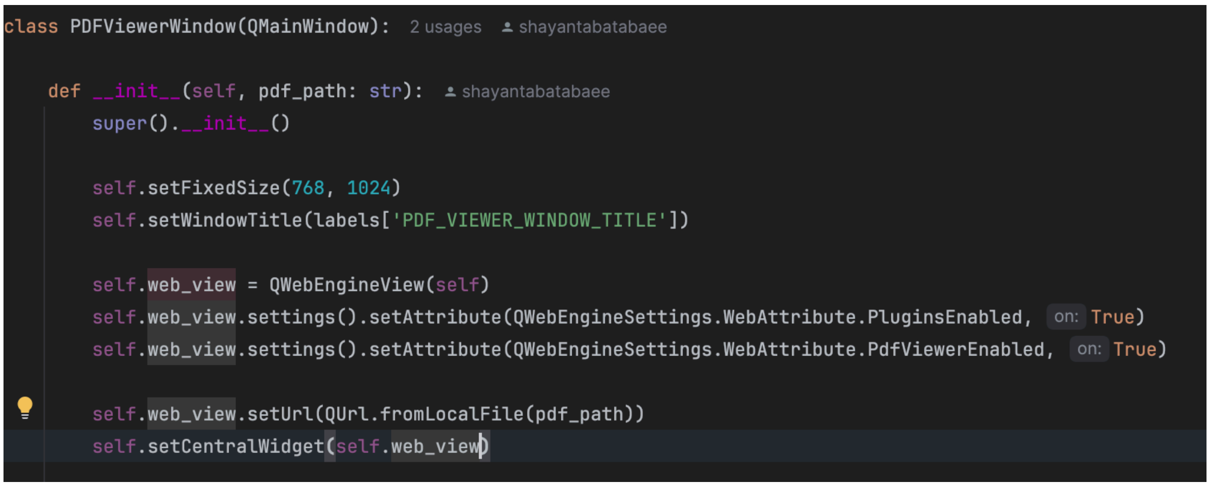
```python
question_input.setValidator(QDoubleValidator(bottom=0))
```

Secondly, the validation should be implemented also in the backend which leads to use *pydantic* library. Here is the sample image for class *Waste* that all of the validations have applied to that.



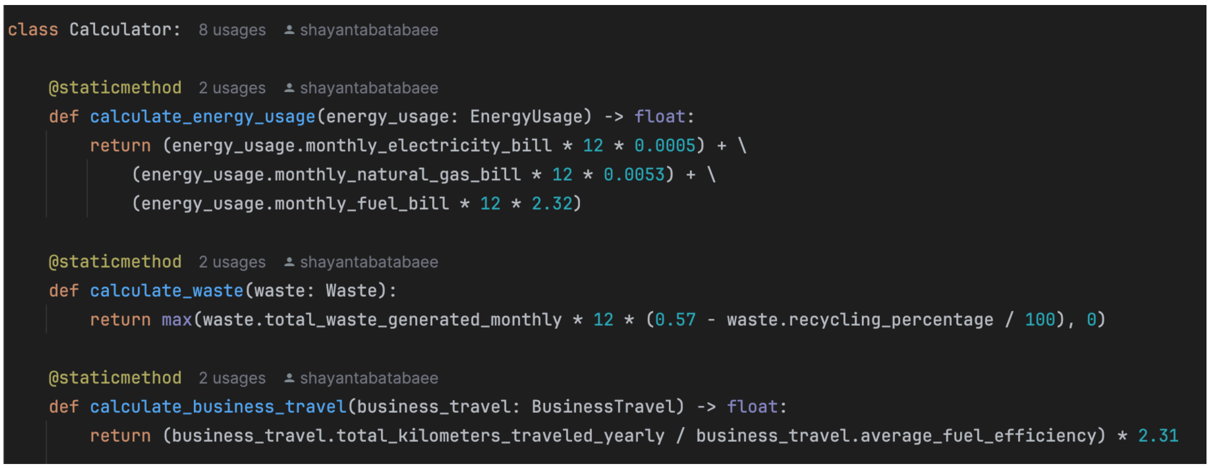
```
from pydantic import BaseModel, Field


class Waste(BaseModel):  13 usages  shayantabatabaee
    total_waste_generated_monthly: float = Field( default: ..., ge=0)
    recycling_percentage: float = Field( default: ..., ge=0, le=100)
```

As we can see through the image, *recycling_percentage* can be only between 0 and 100.
UI module includes all of the related code for the UI. For example for showing generated PDF to user, another window will pop up which includes a web view to render generated PDF. Here is the related image for *PDFViewerWindow* class.



```
class PDFViewerWindow(QMainWindow):  2 usages  shayantabatabaee

    def __init__(self, pdf_path: str):  shayantabatabaee
        super().__init__()

        self.setFixedSize(768, 1024)
        self.setWindowTitle(labels['PDF_VIEWER_WINDOW_TITLE'])

        self.web_view = QWebEngineView(self)
        self.web_view.settings().setAttribute(QWebEngineSettings.WebAttribute.PluginsEnabled,  on: True)
        self.web_view.settings().setAttribute(QWebEngineSettings.WebAttribute.PdfViewerEnabled,  on: True)

        self.web_view.setUrl(QUrl.fromLocalFile(pdf_path))
        self.setCentralWidget(self.web_view)
```

For the last part of project structure section, I would refer to *utils* package. *Utils* package includes all of the utility classes. For example consider *Calculator* class which is responsible to get models for each usage section and calculate the energy usage. Here is the image of this class.

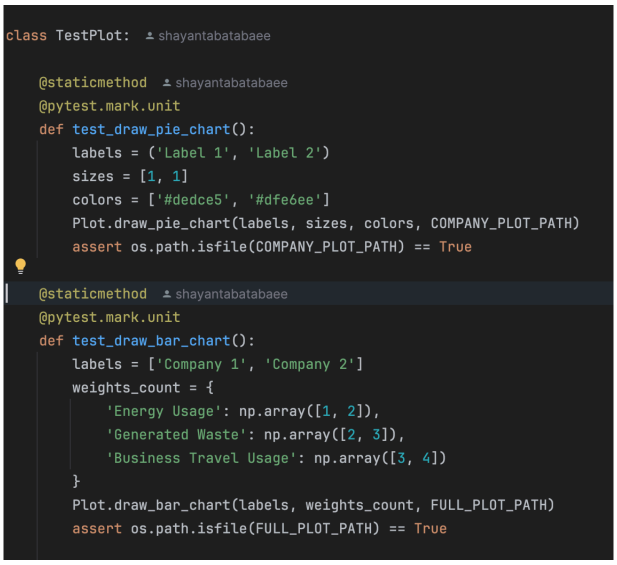

```python
class Calculator:  8 usages  shayantabatabaee

    @staticmethod  2 usages  shayantabatabaee
    def calculate_energy_usage(energy_usage: EnergyUsage) -> float:
        return (energy_usage.monthly_electricity_bill * 12 * 0.0005) + \
            (energy_usage.monthly_natural_gas_bill * 12 * 0.0053) + \
            (energy_usage.monthly_fuel_bill * 12 * 2.32)

    @staticmethod  2 usages  shayantabatabaee
    def calculate_waste(waste: Waste):
        return max(waste.total_waste_generated_monthly * 12 * (0.57 - waste.recycling_percentage / 100), 0)

    @staticmethod  2 usages  shayantabatabaee
    def calculate_business_travel(business_travel: BusinessTravel) -> float:
        return (business_travel.total_kilometers_traveled_yearly / business_travel.average_fuel_efficiency) * 2.31
```

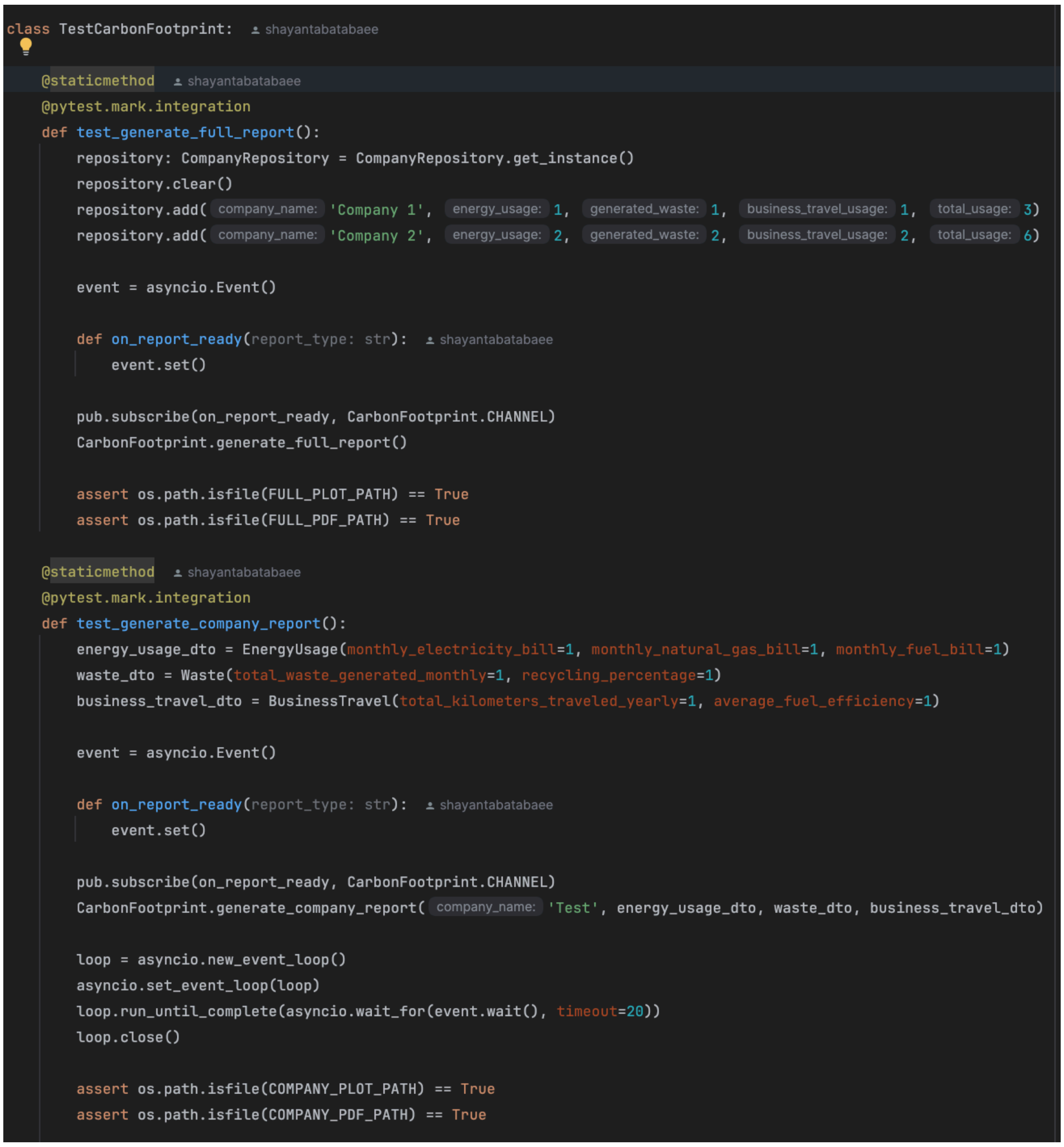
Test Section

As mentioned before, writing test for every application is crucial to prevent from breaking changes. In this application, I used *pytest* library which is a quite familiar library in python to write tests. Tests for this application divide into two categories:

1. Unit-tests
2. Integration-test

Unit-tests belongs to each unit, for example the following image demonstrates test written for *Plot* class which is a unit responsible to draw a pie chart based on the inputted data.


```python
class TestPlot:  shayantabatabaee

    @staticmethod  shayantabatabaee
    @pytest.mark.unit
    def test_draw_pie_chart():
        labels = ('Label 1', 'Label 2')
        sizes = [1, 1]
        colors = ['#dedce5', '#dfe6ee']
        Plot.draw_pie_chart(labels, sizes, colors, COMPANY_PLOT_PATH)
        assert os.path.isfile(COMPANY_PLOT_PATH) == True

    @staticmethod  shayantabatabaee
    @pytest.mark.unit
    def test_draw_bar_chart():
        labels = ['Company 1', 'Company 2']
        weights_count = {
            'Energy Usage': np.array([1, 2]),
            'Generated Waste': np.array([2, 3]),
            'Business Travel Usage': np.array([3, 4])
        }
        Plot.draw_bar_chart(labels, weights_count, FULL_PLOT_PATH)
        assert os.path.isfile(FULL_PLOT_PATH) == True
```


```
class TestCarbonFootprint:  shayantabatabaee

    @staticmethod  shayantabatabaee
    @pytest.mark.integration
    def test_generate_full_report():
        repository: CompanyRepository = CompanyRepository.get_instance()
        repository.clear()
        repository.add( company_name: 'Company 1',  energy_usage: 1,  generated_waste: 1,  business_travel_usage: 1,  total_usage: 3)
        repository.add( company_name: 'Company 2',  energy_usage: 2,  generated_waste: 2,  business_travel_usage: 2,  total_usage: 6)

        event = asyncio.Event()

        def on_report_ready(report_type: str):  shayantabatabaee
            event.set()

        pub.subscribe(on_report_ready, CarbonFootprint.CHANNEL)
        CarbonFootprint.generate_full_report()

        assert os.path.isfile(FULL_PLOT_PATH) == True
        assert os.path.isfile(FULL_PDF_PATH) == True

    @staticmethod  shayantabatabaee
    @pytest.mark.integration
    def test_generate_company_report():
        energy_usage_dto = EnergyUsage(monthly_electricity_bill=1, monthly_natural_gas_bill=1, monthly_fuel_bill=1)
        waste_dto = Waste(total_waste_generated_monthly=1, recycling_percentage=1)
        business_travel_dto = BusinessTravel(total_kilometers_traveled_yearly=1, average_fuel_efficiency=1)

        event = asyncio.Event()

        def on_report_ready(report_type: str):  shayantabatabaee
            event.set()

        pub.subscribe(on_report_ready, CarbonFootprint.CHANNEL)
        CarbonFootprint.generate_company_report( company_name: 'Test', energy_usage_dto, waste_dto, business_travel_dto)

        loop = asyncio.new_event_loop()
        asyncio.set_event_loop(loop)
        loop.run_until_complete(asyncio.wait_for(event.wait(), timeout=20))
        loop.close()

        assert os.path.isfile(COMPANY_PLOT_PATH) == True
        assert os.path.isfile(COMPANY_PDF_PATH) == True
```

On the other hand, Integration-tests relates to modules that use different sub modules and integrated into one class. The above image is an example of Integration-test written for the *CarbonFootprint* class.

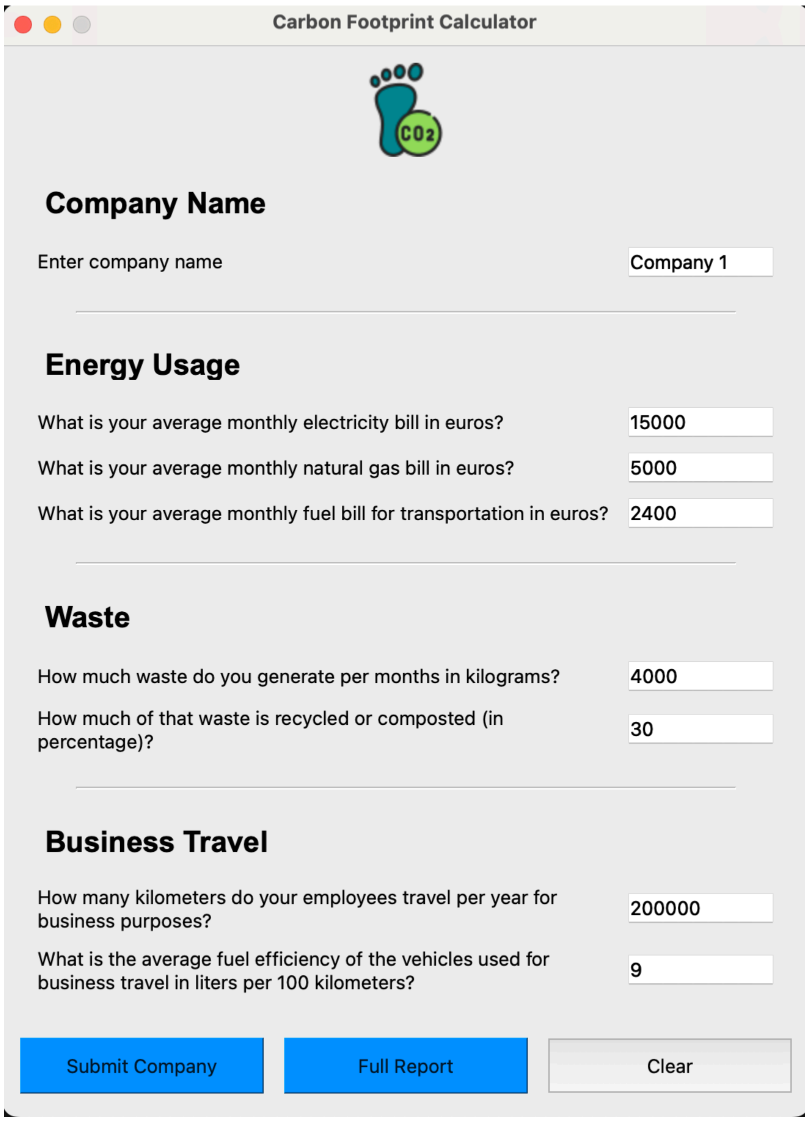
Summaries of The Analysis

After the required information entered by user, a pie chart and a report generates. The final pdf report will pop up into a different window. Here is the sample screen shots of the application and the generated report for a company with these values :

- Monthly electricity bill : 15.000 euros
- Monthly natural gas bill: 5.000 euros
- Monthly fuel bill: 2.400 euros
- Total waste generated monthly: 4000 kgs
- Recycling percentage : 30%
- Total kilometers traveled yearly: 200,000 kilometers
- Average fuel efficiency: 9 liters/100 km

The screenshot of application, pie chart and the generated report would be as follows.

Screenshot



Company Name

Enter company name	Company 1

Energy Usage

What is your average monthly electricity bill in euros?	15000
What is your average monthly natural gas bill in euros?	5000
What is your average monthly fuel bill for transportation in euros?	2400

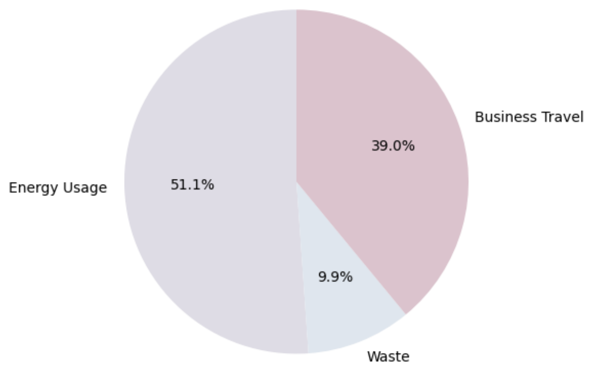
Waste

How much waste do you generate per months in kilograms?	4000
How much of that waste is recycled or composted (in percentage)?	30

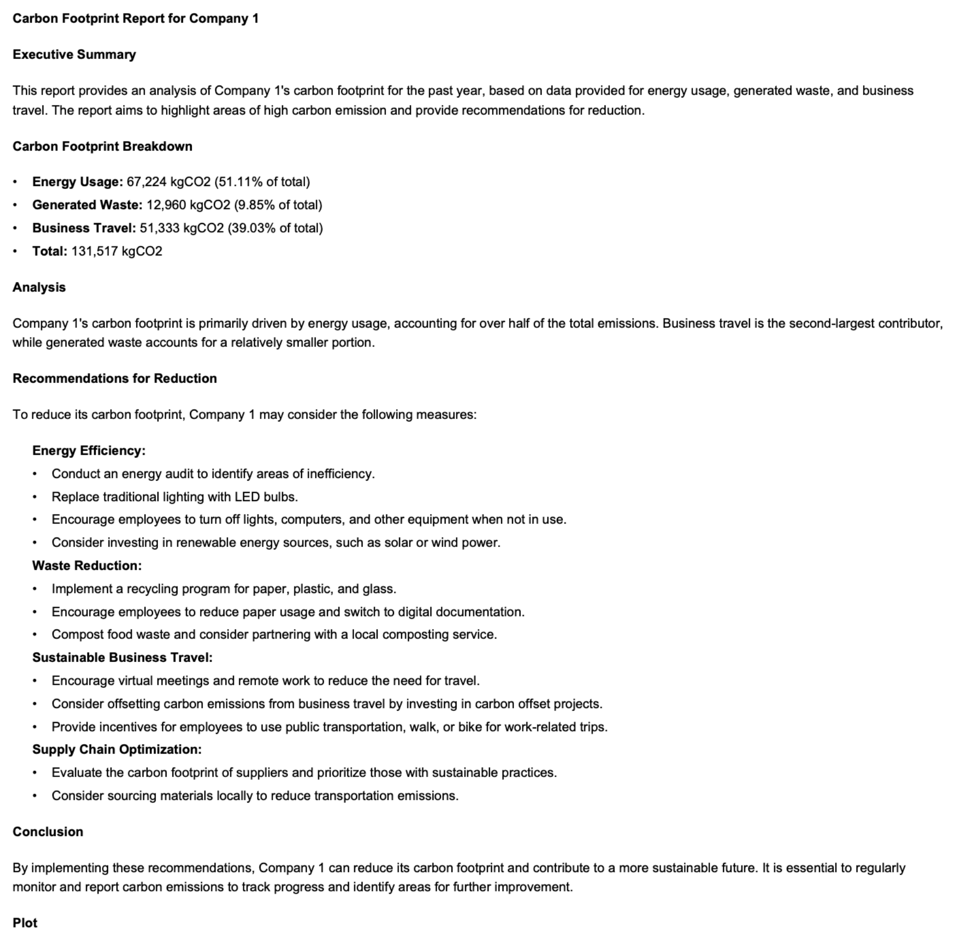
Business Travel

How many kilometers do your employees travel per year for business purposes?	200000
What is the average fuel efficiency of the vehicles used for business travel in liters per 100 kilometers?	9

Pie chart



Report



Carbon Footprint Report for Company 1

Executive Summary

This report provides an analysis of Company 1's carbon footprint for the past year, based on data provided for energy usage, generated waste, and business travel. The report aims to highlight areas of high carbon emission and provide recommendations for reduction.

Carbon Footprint Breakdown

- **Energy Usage:** 67,224 kgCO2 (51.11% of total)
- **Generated Waste:** 12,960 kgCO2 (9.85% of total)
- **Business Travel:** 51,333 kgCO2 (39.03% of total)
- **Total:** 131,517 kgCO2

Analysis

Company 1's carbon footprint is primarily driven by energy usage, accounting for over half of the total emissions. Business travel is the second-largest contributor, while generated waste accounts for a relatively smaller portion.

Recommendations for Reduction

To reduce its carbon footprint, Company 1 may consider the following measures:

Energy Efficiency:
- Conduct an energy audit to identify areas of inefficiency.
- Replace traditional lighting with LED bulbs.
- Encourage employees to turn off lights, computers, and other equipment when not in use.
- Consider investing in renewable energy sources, such as solar or wind power.

Waste Reduction:
- Implement a recycling program for paper, plastic, and glass.
- Encourage employees to reduce paper usage and switch to digital documentation.
- Compost food waste and consider partnering with a local composting service.

Sustainable Business Travel:
- Encourage virtual meetings and remote work to reduce the need for travel.
- Consider offsetting carbon emissions from business travel by investing in carbon offset projects.
- Provide incentives for employees to use public transportation, walk, or bike for work-related trips.

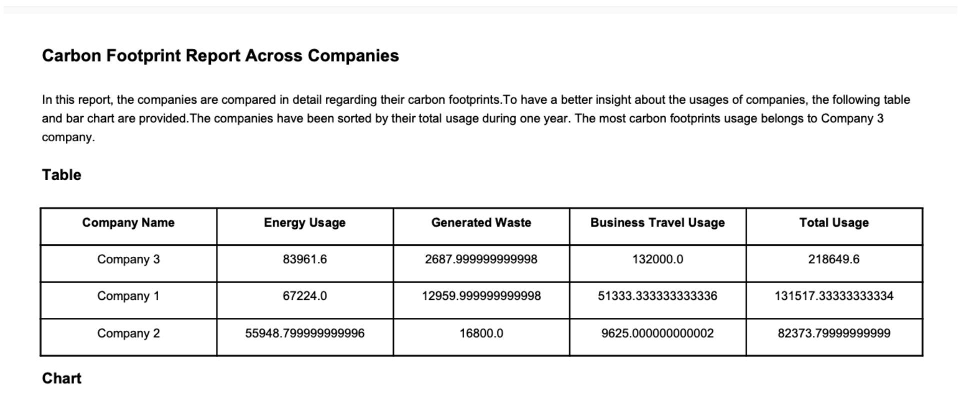
Supply Chain Optimization:
- Evaluate the carbon footprint of suppliers and prioritize those with sustainable practices.
- Consider sourcing materials locally to reduce transportation emissions.

Conclusion

By implementing these recommendations, Company 1 can reduce its carbon footprint and contribute to a more sustainable future. It is essential to regularly monitor and report carbon emissions to track progress and identify areas for further improvement.

Plot

After inputting companies data by user, the application saves data for each company in a singleton class named *CompanyRepository* which acts as an in-memory storage. The reason that singleton design pattern is used relates to the fact that there is only need one instance of this class across the entire application. Here would be the reports across the companies which is sorted based on the total usage of companies.



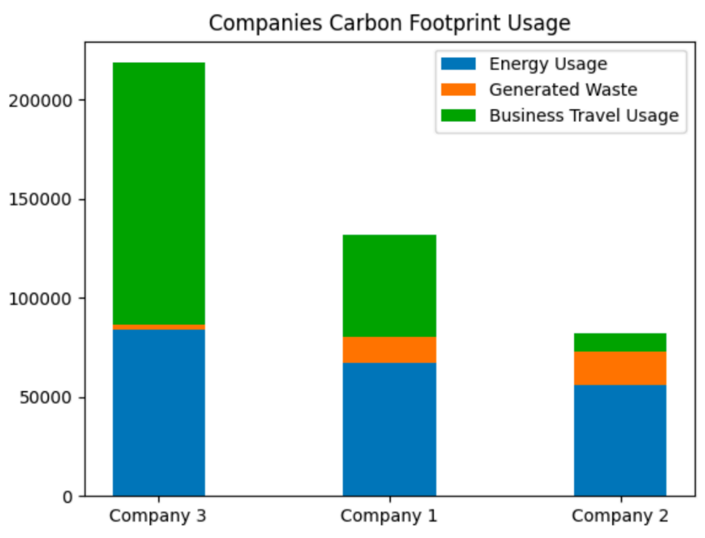
Carbon Footprint Report Across Companies

In this report, the companies are compared in detail regarding their carbon footprints.To have a better insight about the usages of companies, the following table and bar chart are provided.The companies have been sorted by their total usage during one year. The most carbon footprints usage belongs to Company 3 company.

Table

Company Name	Energy Usage	Generated Waste	Business Travel Usage	Total Usage
Company 3	83961.6	2687.999999999998	132000.0	218649.6
Company 1	67224.0	12959.999999999998	51333.333333333336	131517.33333333334
Company 2	55948.799999999996	16800.0	9625.000000000002	82373.79999999999

Chart



Also these reports and charts can be found in the *docs* folder in Github.

Conclusion

This application is designed to calculate carbon footprint for individual and across companies based on usage of electricity, waste and business travel values. The company inputs its data, and based on the provided data a pie chart and a comprehensive report would be generated. In this application I tried to keep everything simple and readable. Also I tried to use best practices in application development. However, it is worth mentioning that there are lots of areas for improvement that will be discussed further.
For example for generating reports, I used an external LLM which is pre-trained on general datasets. To improve the report quality it is crucial to fine tune this LLM to be an expert on suggesting ways to reduce carbon footprint usages. Indeed, for having a real world application it would be better to store each company data in a centralized database like PostgreSQL and have different users that should be authenticated and authorized.

Bibliography

3. Contributors, W. (13 November 2024 10:35 UTC). *Carbon footprint*. Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/w/index.php?title=Carbon_footprint&oldid=1257124007