EcoAct Platform

How Can Companies Engage in Data-Driven Carbon Emissions Compensation to Maximize Impact?

Programming Exam 2024

Submitted:	December 20th, 2024
By:	Johan Henri Schommartz – 175871
Lecturer:	Ole Torp Lassen
Course coordinator/ Lecturer:	Somnath Mazumdar
Number of characters:	25,903
Number of pages:	15

Abstract

The EcoAct platform addresses the urgent need of businesses to manage their carbon emissions in a data-driven and impactful way. Developed to support companies on their journey to net zero, the platform enables companies to calculate annual carbon emissions across scopes 1, 2, and 3, benchmark results against industry standards, and invest in high-quality carbon credits. Its object-oriented architecture ensures scalability and modifiability, incorporating features such as user-specific emissions data input, visualized reporting with Pandas and Matplotlib, and a integrated carbon credit marketplace. Challenges like data consistency and class interaction were resolved through incremental testing and modular design. Developing EcoAct up to the current status taught the essential need to effectively balance between code complexity and user benefit, when programming under resource constraints. By applying the course concepts to the real-world sustainability problem, EcoAct became a useful businesses tool and lays a solid foundation for future enhancements, such as cloud deployment and ERP integration.

Copenhagen Business School
Department of Business, Humanities and Law
2000 Frederiksberg
Phone: +45 (0) 3815 3630, Email: bhl@cbs.de

Table of Contents

List of Abbreviations

EU .....European Union

CSRD .....Corporate Sustainability Reporting Directive

VCM.....Voluntary Carbon Market

OTC .....Over-The-Counter

1 Introduction

Climate change poses one of the most pressing challenges of our time, and businesses are at the heart of this crisis (UNFCCC, 2023). As significant contributors to global carbon emissions, companies have a unique responsibility and opportunity to lead the charge in mitigating environmental harm (UNFCCC, 2023). Consequently, many companies have set themselves the ambitious target to achieve net zero emissions by 2030 (NewClimate Institute, 2023). Beyond just reducing their emissions, businesses can serve as a catalyzer for innovative climate solutions and as investors in negative emission projects. The EcoAct platform emerges within this application domain, offering companies the tools to assess and address their annual carbon emissions in a data-driven way.

The increasing urgency to combat climate change has brought forth new regulatory demands, particularly within the European Union (EU). The EU's Corporate Sustainability Reporting Directive (CSRD), effective from 2024, requires companies to disclose detailed sustainability metrics, including carbon emissions across scope 1, 2, and 3 (Official Journal of the European Union, 2022). However, the challenge lies not only in accurately assessing emissions but also in addressing residual emissions that cannot yet be eliminated. Despite these regulatory pressures, many companies fail to integrate impactful carbon compensation into their sustainability strategies.

The voluntary carbon market (VCM) offers a solution by providing businesses access to avoidance and removal credits. Yet, the market remains fragmented, with most transactions occurring over-the-counter (OTC) (McKinsey, 2021). Without a centralized platform, companies struggle to compare credits based on quality metrics and price, leading to the intransparency, that gives this industry its bad reputation. EcoAct bridges this gap by offering a comprehensive platform that transparently compares carbon credits, empowering businesses to make informed decisions.

This project aims to develop a functional prototype of EcoAct, blending practical utility with educational objectives. At its core, the platform is designed to help businesses streamline their carbon assessment and compensation processes. The application showcases fundamental programming concepts, including object-oriented design, algorithmic sorting, and integration of external libraries. For instance, the modular structure of the platform reflects imperative, declarative, and object-oriented programming principles, while leveraging external libraries like Pandas and Matplotlib ensures efficient data handling and visualization.

By building EcoAct, I aim to demonstrate my proficiency in Python and address key learning outcomes of the course, like implementing robust Python programs with clear syntax and appropriate features, showcasing a foundational understanding of algorithm complexity and data structures, leveraging external tools and libraries for real-world applications and producing scalable, modular code that can be expanded for commercial use. In doing so, EcoAct is not just a coding project, it's a step toward enabling businesses to make impactful, sustainable decisions and a step towards me becoming a Data Scientist.

2 Requirements Analysis

The EcoAct platform is designed to provide businesses with an efficient way to assess their carbon emissions, generate reports, and invest in carbon credits. The following analysis identifies the core requirements necessary for the successful implementation of the project, focusing on functional and non-functional needs, as well as the constraints shaping its development.

2.1 Functional Requirements

The functional requirements are at the heart of the EcoAct platform, defining what the application must do to meet its objectives. First, the platform must support the creation and management of company profiles, which includes capturing essential information such as the company name, industry, and number of employees. These details are vital for contextualizing emission calculations and benchmarking results against industry standards. The platform also requires a robust emissions assessment feature, capable of processing annual user-provided emission data. This assessment should contain a precise calculation of the company's total carbon footprint, expressed in metric tons of CO_2 equivalent.

The next critical functionality is the generation of carbon reports. These reports must not only summarize emissions and compensation data but also include a visual representation to enhance user comprehension. To achieve this, the platform requires integration with libraries like Matplotlib for data visualization. In addition, application must enable credit purchase integration, allowing businesses to filter through negative emission projects to find the optimal impact-to-price match.

2.2 Non-Functional Requirements

Beyond functional capabilities, the EcoAct platform must meet several non-functional requirements to ensure its effectiveness, scalability, modifiability, performance, security, availability, and integrability. Its current limitations provide a roadmap for future improvements, ensuring the platform evolves into a robust and scalable solution for carbon assessment and compensation.

Scalability is particularly important as the platform's user base grows. The modular object-oriented design allows for future expansion, including integration. Additionally, scalability is enhanced using the Company class, where company-specific data is stored as separate instances. To base able to handle a growing user base, the EcoAct platform requires efficient performance. Given that the emission calculations may involve large datasets, algorithms must be optimized to reduce latency. Report generation uses libraries like Pandas and Matplotlib, which handle data processing and visualization efficiently.

Modifiability ensures the platform can adapt to future changes with minimal effort. The object-oriented structure facilitates easy modifications. Integrating external APIs would only involve changes to a certain class or the main function without impacting the rest of the program. Additionally, the clear separation between input handling, data processing, and reporting simplifies the process of introducing new features. Security of this modular prototype is currently limited but remains a key consideration for future iterations. At present, user data is stored in the program's memory without encryption, which is sufficient for a local and academic prototype. However, in a production environment, secure data storage and encryption would be essential to prevent unauthorized access.

Availability refers to the platform's ability to remain operational and accessible to users. In its current form, the EcoAct platform runs locally, limiting its availability to a single computer. Deploying the program on cloud-based platforms would improve availability by ensuring 24/7 access. Independency on specific hardware raised the demand for integrability of the EcoAct platform with broader systems or databases. The current prototype lacks external integrations due to resource constraints but is structured to allow future enhancements. For instance, EcoAct could be integrated in a business ERP system like SAP, to allow for a direct link with accounting software. Additionally, integrating secure payment gateways would enable real-time credit transactions, enhancing usability.

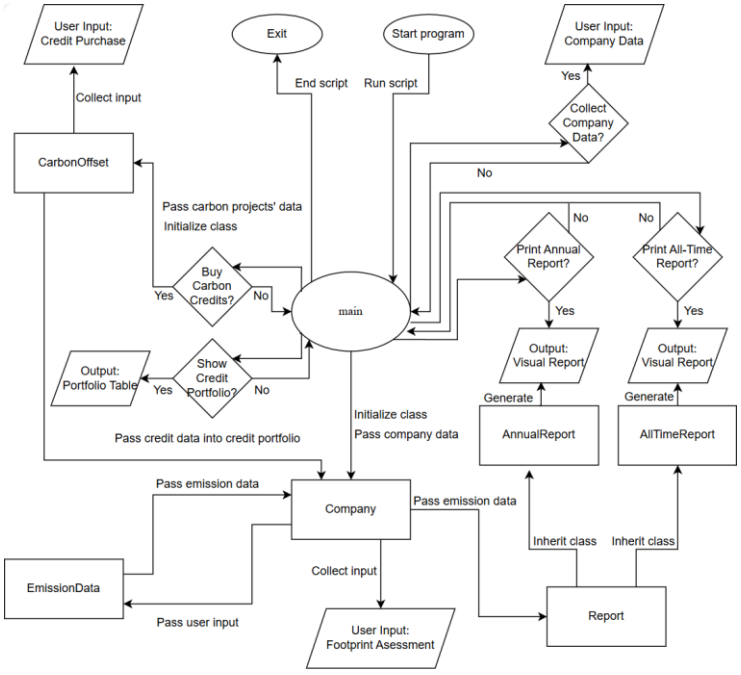
2.3 Constraints

The development of the EcoAct platform is constrained by several factors. The prototype is currently designed to run locally, which limits its capacity to handle large datasets or multiple concurrent users, as mentioned before. Another significant constraint is the project's reliance on free tools and libraries, as paid APIs of carbon credit registries or payment providers are beyond the scope of this academic assignment. Without these integrations, transactions cannot be processed, and the data input of negative emission projects remains manual. Time is another constraint, as the project must be completed within a limited period, restricting the scope of features that can be implemented. Permanent data storage also posed a challenge, as the code must function independently in the context of the exam. While data inputs could be stored in a CSV file using read/write functions, this approach carries the risk of compatibility issues, such as differing file paths on different computers. To avoid this potential complication, inputs are instead stored in the program's memory, ensuring the code runs consistently across systems.

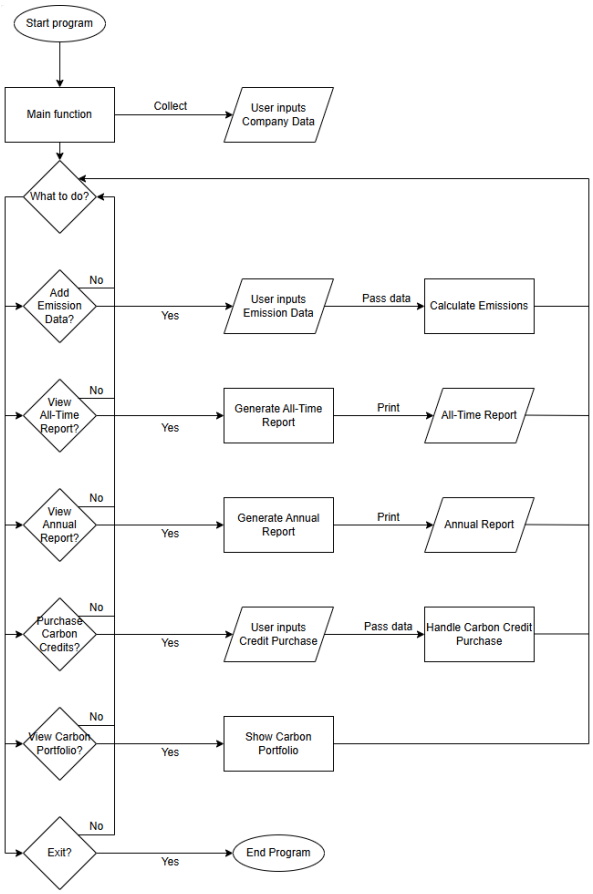
3 System Design

3.1 Architectural Design

The system architecture of the EcoAct platform integrates user input, data processing through classes, decision-making processes, and outputs such as reports, and carbon offset purchases. The system's design is illustrated by two main diagrams.



The block chart diagram provides a high-level overview of the EcoAct platform's relationships between the program's main script and its core objects. The main script serves as the central hub, initializing and directing the flow of execution. User inputs, such as company data, emissions data, and credit purchase requests, are collected and passed to the respective classes. Each class handles specific functionalities. The Company class stores company dat. The EmissionData class calculates the carbon footprint, the CarbonOffset class manages carbon credit purchases, and the Report class with its child classes AnnualReport and AllTimeReport generates reporting outputs. The arrows in the block chart indicate the flow of data between objects. For example, emissions data flows from Company to EmissionData and back, before being passed into the Report classes to be outputted.



The flow chart, in contrast, focuses on the user's interaction with the program. Users begin by inputting company data, which is collected by the main function. The main function directs the user to the navigation menu and realises the questions of "What would you like to do?". At this point the user must decide whether to add emission data, to view some reports, or to purchase carbon credits. These choices direct users to specific processes like emissions calculations or report generation. Input nodes, like filtering for carbon credits, represent stages where the user provides data, and output nodes represent results such as emissions reports or carbon credit portfolios. The flowchart highlights when user inputs are required, when data is processed, and how outputs are generated.

3.2 Module Design

The EcoAct platform is built using a modular design approach, where each class is responsible for a specific process and each function for a specific task. This modular structure ensures that the code is organized, maintainable, and scalable. Below is a detailed explanation of each module, including its purpose, functionality, and key methods.

The main function serves as the entry point of the program. It orchestrates the user interaction and directs the flow between various classes based on user choices. The main function initializes objects such as Company and CarbonOffset and relies on a while True loop to maintain continuous user interaction. Within the loop, user inputs determine the flow of execution, such as whether to add emissions data, view reports, or purchase credits. Decision points are implemented using conditional statements, ensuring that the program responds to the user's choices.

The Company class is responsible for storing and organizing company-specific information. It initializes attributes like the company details, emission data and credit portfolio through its init method. This data is later passed to other classes for further processing. The Company class ensures that user inputs are structured and accessible to other modules. The EmissionData class handles emissions calculations. The class collects user-provided emissions data and processes this data with the calculate_emissions method. The total emissions are saved within the class instance of a year and to be accessible to other classes and the main script. The CarbonOffset class manages the purchase of carbon credits ulitilizing the offset project data from the main script. The process_transaction function handles credit purchases and reduces the total available credit amount within the offset project data after executing a transaction.

4 Implementation

4.1 Tools and Technologies

The EcoAct platform leverages several Python libraries and tools to handle data processing, visualization, and user interaction. Among these, Pandas plays a pivotal role in managing and manipulating company's credit portfolio data and the carbon offset project data. Panda's intuitive data structures and built-in operations make it ideal for tasks like aggregating and editing emissions credit and project data. For data visualization, Matplotlib is utilized to generate clear and professional charts, which are integral to the platform's reporting capabilities. These charts help users interpret their emissions data and the impact of selected carbon offsets more effectively. Additionally, IPython.display is used to format and style the table outputs for the offset project data and the credit portfolio. This library ensures that the application remains user-friendly and readable while being very text and data heavy.

4.2 Code Structure

The code structure follows a modular design, with the directory structure in the main function emphasizing the separation of functionalities into the different classes for clarity and scalability. This design ensures that individual components can be updated or extended without affecting the entire application. Usually, the code would not just be into different cells within one python document but be divided into different documents, owning different functionalities, that direct towards each other. This was disregarded in this project to ensure easiness of executing the code.

4.3 Key Code Snippets

One of the most critical challenges was ensuring that carbon offset project data was updated correctly across multiple transactions. Each project has a finite balance of available credits and this balance needs to decrease as credits are purchased. Initially, the CarbonOffset class reinitialized project data every time it was called, overwriting the reduced balance. To address this, the initialization of project data was decoupled from the instantiation of the CarbonOffset class. Instead, project data is loaded once during the main script's initialization and passed to the class.

```
# Define the offset projects with inital project total credit amounts (can be reduced later)
offset_projects = [
    {'project_id': 'VCS1001', 'name': 'UNDO Biochar Rwanda', 'project_type': 'Biochar', 'country': 'Rwanda', 'duration': 1000, 'risk_factor': 2,
    {'project_id': 'VCS1002', 'name': 'Ginco Biochar New Dehli', 'project_type': 'Biochar', 'country': 'India', 'duration': 1000, 'risk_factor':
    {'project_id': 'VCS1003', 'name': 'Green Leaf Biochar Brazil', 'project_type': 'Biochar', 'country': 'Brazil', 'duration': 1500, 'risk_factor

]

# Convert the offset projects into a pandas DataFrame
offset_projects_df = pd.DataFrame(offset_projects)
projects_df = offset_projects_df.copy()
```

```
    while True:
        print("\nWhat would you like to do?")
        print("1. Add yearly emission data")
        print("2. View all-time report")
        print("3. View yearly report")
        print("4. Purchase carbon offsets")
        print("5. View carbon portfolio")
        print("6. Exit")
        choice = input("Enter your choice: ")

elif choice == "4":
    # Purchase carbon offsets
    try:
        print("\nPurchasing Carbon Offsets:")
        offset = CarbonOffset(company, projects_df)
        offset.display_offset_options()

        # Proceed with the purchase
        cost = offset.purchase_offset(project_id)
        print(f"Offset purchased successfully. Total cost: ${cost:.2f}\n")
    except ValueError:
        print("Invalid input. Please enter a valid input.")
```

This ensures that the remaining credit balance is updated correctly with each transaction without being reset when the CarbonOffset class is called for a next transaction. The following snippets demonstrates this final implementation.

```python
def purchase_offset(self, selected_id):  # Use project ID for selection
    # Find the selected project by ID
    selected_project = self.projects_df[self.projects_df['project_id'] == selected_id]
    if selected_project.empty:
        print("Project ID doesn't exists. Please try again.")
        return None

    # Input the desired amout of carbon credits
    while True:
        try:
            # Input the desired amount of carbon credits
            amount = int(input("Enter the amount of CO2 credits (tons): "))
            total_credit_amount = selected_project['total_credit_amount'].iloc[0]
            if 0 <= amount <= total_credit_amount:
                # Reduce the total amount of credits of the selected project
                self.projects_df.loc[self.projects_df['project_id'] == selected_id, 'total_credit_amount'] -= amount
                display(self.projects_df)#REMOVE AFTER CORRECTED
                break  # Exit the while loop if input is valid
            else:
                print("The amount of CO2 credits must be between zero and the projects' total credit amount. Please try again.")
        except ValueError:
            print("Invalid input. Please enter a integer number.")
```

This logic allows the program to manage remaining credits effectively, preventing over-allocation and ensuring accurate reporting.

Another noteworthy feature of the EcoAct platform is the implementation of a bubble sort algorithm that sorts the filtered credits based on ascending prices. Although bubble sort is not the most computationally efficient sorting method, its use demonstrates the general understanding of algorithmic design in real-world applications.

```python
def bubble_sort_by_price(df):
    # Bubble sort to sort by 'price_per_ton' in ascending order
    n = len(df)
    for i in range(n):
        for j in range(0, n-i-1):
            if df.iloc[j]['price_per_ton'] > df.iloc[j+1]['price_per_ton']:
                # Swap the rows if they are in the wrong order
                df.iloc[j], df.iloc[j+1] = df.iloc[j+1], df.iloc[j]
    return df
```

5 Results and Discussion

5.1 Results

The EcoAct platform delivers results that align with its intended purpose of helping companies calculate and compensate for their carbon emissions. The following outputs illustrate key functionalities and user interactions.

After starting the program, the user is welcomed and asked to input company details such as name, industry, and number of employees. This setup initializes the company instance, storing relevant information for subsequent processes.

```
main()

Welcome to EcoAct!
Your Carbon Management Partner from Footprint Assessment to Climate Action.
Your are a CSO or Sustainability manager?
We're going to help you assess your company's carbon footprint and directly compensate for those emissions effectively!

Enter the company's name:  SAP AG

Enter the company's industry. Pick one of the following:
Technology, Finance, Hospitality, Transportation, Services, Healthcare, Retail, Education, Construction, Manufacturing
You picked:
 Technology

Enter the company's number of employees:  112000
Company 'SAP AG' in the 'Technology' industry with 112000 employees has been created.


What would you like to do?
1. Add yearly emission data
2. View all-time report
3. View yearly report
4. Purchase carbon offsets
5. View carbon portfolio
6. Exit
Enter your choice:  ↑↓ for history. Search history with c-↑/c-↓
```

When the user selects to add yearly emissions, the program collects emissions data categorized into scopes 1, 2 and 3. The input process is user-friendly, asking for specific measures like fuel consumption or electricity usage to calculate a detailed carbon footprint for the company.

```
Enter your choice:  1

Enter the year for which you are adding data:  2023

Adding data for the year 2023.

Enter Scope 1: Direct Emissions Data
Enter fuel consumption (liters):  25000
Enter natural gas consumption (cubic meters):  5600
Enter raw material usage for industrial processes (tons):  0
Enter refrigerant leakage (kg):  0

Enter Scope 2: Indirect Emissions Data
Enter electricity usage (kWh):  185000
Enter purchased heat/steam usage (MWh):  1200

Enter Scope 3: Indirect Emissions Data

Upstream Emissions:
Enter business travel distance (km):  450000
Enter average employee commuting distance per day (km):  8
Enter purchased goods and services emissions (tons):  500
Enter waste disposal emissions (tons):  0

Downstream Emissions:
Enter product use emissions (tons):  0
Enter product end-of-life emissions (tons):  0
Enter distribution and transportation emissions (tons):  ↑↓ for history. Search history with c-↑/c-↓
```

The user can purchase carbon credits by applying filters like project type, country, duration, or risk factor. The program retrieves a list of available projects sorted by price, allowing the user to make informed decisions about carbon offset purchases. This output demonstrates the program's ability to manage and display filtered project data dynamically.

```
Enter your choice:  4

Purchasing Carbon Offsets:
Check the available Negative Emission Projects:
We suggest, that you filter the projects to find the right credits for your high-impact carbon portfolio!

You can filter by:
1. Project Type
2. Country
3. Duration
4. Risk Factor
5. Show All
6. Continue to Purchase
Enter the number of the filter option (or 6 to continue):  1
Enter the project type (e.g., Biochar, Reforestation, Renewable Energy):  Biochar
```

```
These are your filtered credit options filtered by price (ascending):
```

Available Carbon Offset Projects

Project ID	Project Name	Project Type	Country	Duration (Years)	Risk Factor	Total Credit Amount	Price per Ton (USD)
VCS1004	Eco Future Biochar Kenya	Biochar	Kenya	2000	1	5500	200.450000
VCS1001	UNDO Biochar Rwanda	Biochar	Rwanda	1000	2	5000	214.560000
VCS1003	Green Leaf Biochar Brazil	Biochar	Brazil	1500	1	7000	221.340000
VCS1005	Sunshine Biochar Indonesia	Biochar	Indonesia	1800	2	4800	234.890000
VCS1002	Ginco Biochar New Dehli	Biochar	India	1000	2	4420	245.670000

After selecting a project and specifying the number of credits, the program confirms the purchase. It reduces the project's available credits accordingly, ensuring accurate transaction management. The total cost of the credits is calculated and displayed to the user.

```
Choose the project ID:  VCS1004
Enter the amount of CO2 credits (tons):  4000

Purchasing 4000 tons of CO2 offset from Eco Future Biochar Kenya for $801800.0.
The total amount of credits prochasable from Eco Future Biochar Kenya has been reduced by 4000.
Offset purchased successfully. Total cost: $801800.00
```

Users can review their purchased carbon credits in a portfolio table, showcasing details like project ID, name, duration, and remaining credits. This feature reflects the user's offset transactions and provides transparency.

```
Enter your choice:  5

Viewing Carbon Portfolio:
```

Available Carbon Offset Projects

Project ID	Project Name	Project Type	Country	Duration (Years)	Risk Factor	Total Credit Amount	Price per Ton (USD)
VCS1004	Eco Future Biochar Kenya	Biochar	Kenya	2000	1	4000	200.450000

The program generates an all-time emissions report summarizing the company's total carbon footprint, offsets purchased, and net carbon balance. The report also benchmarks emissions against an industry average, helping users assess their environmental performance.

```
Enter your choice:  2

All-Time Report:
Company: SAP AG
Industry: Technology
All-Time Totals:
Total Footprint: 537.60 tons CO2
Below benchmark by 18.99519999642857 tons CO2 per employee.Total Offsets: 4000.00 tons CO2
Net Carbon Balance: -3462.40 tons CO2
```

The final screen confirms the program's closure with a farewell message, reminding the user to return next year to manage emissions.

```
Thank you for using EcoAct.
See you next year to compensate your emissions again and make this world a better place!
```

Overall, the outputs align with both the functional and non-functional requirements of the platform. Emissions data is calculated and displayed accurately, carbon credits are filtered and managed dynamically, and the user receives clear, actionable reports.

5.2 Discussions

The development of the EcoAct platform came with several challenges, particularly around managing data flow between multiple classes and ensuring accessibility of shared variables. A major issue arose when determining which variables should remain local which needed broader access across classes. For example, the projects_df dataframe, which stores carbon credit project data, needed to be accessible to both the main script and the CarbonOffset class. Initially, defining it within the CarbonOffset class caused conflicts when other functions attempted to modify the data. This was resolved by initializing projects_df in the main script and passing it as a parameter to the relevant classes and methods.

Another challenge was ensuring seamless interaction between child and parent classes when generating reports. The AnnualReport and AllTimeReport classes inherit from the Report parent class but required additional attributes and methods for their specialized outputs. I addressed this by carefully implementing inheritance and overriding methods in child classes as needed, ensuring modularity and code reusability. Error handling also posed a challenge during implementation. To address this, I adopted a method of constantly testing sub-parts of the code before integrating them into the main program. By implementing intelligent error-handling mechanisms, I could pinpoint errors quickly and accurately without solely relying on default Python error messages.

5.3 My learnings

This project taught me that nearly any feature can be built with enough time and resources, but as a developer, I must prioritize the balance between functionality, complexity, and user benefit. I realized that while certain features would be interesting to implement, they might add unnecessary complexity without significantly increasing the program's value for the core user, the CSO. For instance, while the program generates both all-time and annual emissions reports, its primary benefit lies in enabling easy access to high-impact carbon compensation. Another key learning was the importance of modular and scalable design. By structuring the program around separate classes and passing data between them, I created a flexible system that can be extended in the future. Additionally, I learned the significance of error handling and incremental testing, which allowed me to isolate problems and ensure smooth development without being overwhelmed by debugging code that was already integrated in the main code.

6 Conclusion and Future Work

6.1 Conclusion

The EcoAct platform successfully meets its intended purpose of helping businesses calculate their carbon emissions, generate insightful reports, and manage carbon offset purchases. Through the development of this project, I was able to address the key learning objectives described in the introduction. I demonstrated my understanding of fundamental Python programming concepts by building a functional application that integrates data input, processing, and output generation in a cohesive structure. The program reflects my ability to design and implement a solution using Python, incorporating appropriate syntax, features, and external libraries such as Pandas and Matplotlib for data handling and visualization.

The platform uses modular classes which contain distinct functionalities and interact through clearly defined methods. This modular design demonstrates a solid grasp of imperative programming for step-by-step task execution, while object-oriented features ensured maintainability and scalability. I also implemented algorithmic logic, such as the bubble sort algorithm for sorting carbon offset projects, showcasing a foundational understanding of algorithm design and complexity.

The project provided a deeper understanding of Python's capabilities and limitations. For instance, I made deliberate choices to balance program independence and user experience. While the program avoids reliance on external storage files, this design decision simplifies its execution but limits long-term data persistence. Independent of the limitations, the EcoAct platform aligns with the project's functional requirements. Users can input emissions data, filter and purchase carbon offsets, and generate reports. In reflecting on my personal learnings, I realized that while Python enables limitless feature possibilities, it is essential to focus on delivering functionalities that align with user's primary needs.

6.2 Future Work

While the EcoAct platform fulfills its initial objectives, there are several opportunities to improve and expand its capabilities in future iterations. Next to the obvious improvements, like integrating emission calculation and registry APIs, enabling multiple user and company instances and

implementing permanent data storage, the next development step that I am looking forward to the most, is the integration with existing business software providers.

Integrating EcoAct into the most prominent ERP systems from SAP S/4, Microsoft 365 and Oracle Fusion would allow sustainability managers and CSOs to assess and compensate the companies' emissions within its existing software infrastructure, direct linking departments like accounting or treasury.

The EcoAct platform represents a strong foundation for an impactful carbon management tool. With future improvements like API integration, data persistence, and user authentication, it has the potential to evolve into a robust, scalable solution for businesses striving to achieve true climate impact.

7 References

Directive (EU) 2022/2464 of the European Parliament and of the Council of 14 December 2022 Amending Regulation (EU) No 537/2014, Directive 2004/109/EC, Directive 2006/43/EC and Directive 2013/34/EU, as Regards Corporate Sustainability Reporting (Text with EEA Relevance), EP, CONSIL, 322 OJ L (2022). http://data.europa.eu/eli/dir/2022/2464/oj/eng

McKinsey. (2021). *Carbon credits: Scaling voluntary markets*. https://www.mckinsey.com/capabilities/sustainability/our-insights/a-blueprint-for-scaling-voluntary-carbon-markets-to-meet-the-climate-challenge

NewClimate Institute. (2023). *Corporate Climate Responsibility Monitor 2023*. https://newclimate.org/resources/publications/corporate-climate-responsibility-monitor-2023

UNFCCC. (2023). *Impacts of Emerging Industries and Businesses Hydrogen, Carbon Capture Utilisation and Storage and Artificial Intelligence*. https://unfccc.int/documents/632556