UNSW SYDNEY

COMP9900 Information Technology Project

T1, 2024

ESG MANAGEMENT SYSTEM (WEB APPLICATION) FOR FINTECH INDUSTRY

Team Name: StudentITP

Team Members:

Chia-Ching Lee z5351857@ad.unsw.edu.au **Scrum Master**

Yucheng Li z5440174@ad.unsw.edu.au **Group Member**

Ray Han z5432043@ad.unsw.edu.au **Group Member**

Zhengdong Chen z5293902@ad.unsw.edu.au **Group Member**

Jianze Li z5335404@ad.unsw.edu.au **Group Member**

Jingchen Xu z5471272@ad.unsw.edu.au **Group Member**

Submission Date: 19/04/2024

CONTENT

FIGURE TABLE

1 Overview

1.1 Background

With the continuously increasing focus on sustainability and environmental concerns(Kotsantonis and Serafeim, 2019), more investing resources(Li et al., 2021) are devoted to green finance, and sustainable business ethics are gradually adopted by different corporations (Broadstock et al., 2021). The emphasis on sustainable development and green finance not only reshapes the strategies of investment but also redefines the methodologies of company operation and governance(Dorfleitner et al., 2015). This conversion is driven by the gradual recognition of the concept of environmental protection along with financial risks and opportunities.

In response to the evolving landscape of sustainable finance and the pressing need for transparency and accountability to environmental, social, and governance (ESG)(Van Duuren et al., 2016), ESG performance evaluation has become a key indicator to measure a company’s long-term value and risk. ESG management systems are designed to help companies systematically collect, track, and report indicators and data related to environmental, social, and corporate governance(Escrig-Olmedo et al., 2010). Corporations can comprehensively understand their performance in ESG areas, identify potential risks and room for improvement through the system, and formulate corresponding strategies to improve the company's sustainable development level and social responsibility awareness.

An effective ESG management system not only helps companies establish a good impression and win the trust of stakeholders and customers but also obtains a long-term value of operation to the companies(Matos, 2020). Investors are increasingly inclined to incorporate ESG factors into investment decisions, and excellent ESG performance is often linked to lower risk premiums and higher valuations. Therefore, the ESG management system improves the competitiveness and profitability of a company and contributes to promoting sustainable finance and investment to support the transition to a more resilient and flexible global economy.

1.2 System architecture

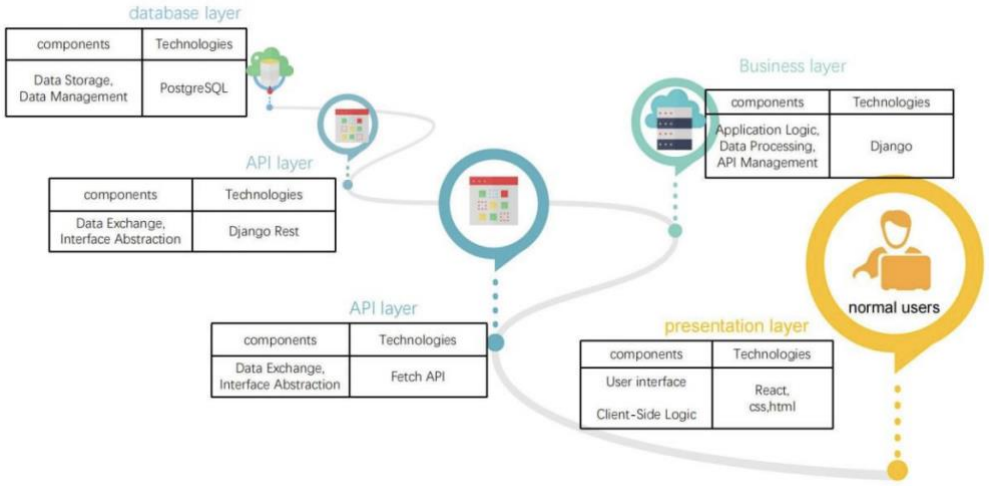

Figure 1 System architecture

In the front end, the project is mainly developed by JavaScript with React, facilitating the user interface with the back end. In this setup facility, the dataset is fetched from the back end and shown in the front end. In the project, various custom components are used such as a searching box, a selecting box, and different kinds of graphs, organised reasonably and integrated into the whole page. The version experience of skimming the website is enhanced through the react-bootstrap.

The project uses Django and Rest Framework as the back end. Requests are sent from the front end and processed. To be more specific, the back end obtains data from the database when receiving requests from the front end as well as calculates and then sends the result to the front end. The paper provides more details below.

1.2.1 Technologies and Third-party Functionality

User Management: Based on Django

The project is based on Django's original user management and support to add new users. That means the system can be used by different analytics at the same time.

The superuser of the system is loaded or created through the terminal when deployed for the first time. After that, clients can create their accounts and save their preferences such as metric weights or indicator weights. Furthermore, the superuser can pause or delete the user from the admin website.

Front-End: React

As a front-end library, React is used to build user interfaces, providing fast response and dynamic data rendering. It simplifies the creation of interactive and reusable UI components, optimizing performance and rendering speed through its virtual DOM mechanism, significantly enhancing the overall platform user experience.

Back-End: Django

This Django framework handles backend application logic and database interactions, simplifying data processing and HTTP request management, and making backend development more efficient and flexible.

API: Django REST framework

The framework provides an efficient development method and the code can be reused directly. Moreover, the API endpoints must be visited by valid users to avoid data leaks. Therefore, the project uses the session method to verify the request. That means the system returns 403 forbidden HTTP status if the user is invalid or visited anonymously to protect data integrity.

Database: PostgreSQL

As a storage solution, the PostgreSQL database securely stores user information and ESG-related data, ensuring efficient and safe data access.

Cloud Hosting Services: AWS

Choosing AWS cloud services provides our application with high availability, scalability, and security, ensuring stable operation and future growth potential.

1.2.2 Database Architecture

Together, the models in database build a complete data structure that stores various ESG-related information such as companies, frameworks, indicators, indicator values, and user-defined weights. Through this set of models, we can efficiently manage and calculate ESG assessment scores and provide data support for sustainable business decisions. Below are brief introductions to the models:

Location: Stores information about the company's geographical location.

Company: Stores basic information about the company, including name, description and location.

Framework: Stores information about ESG assessment frameworks. Each framework has a unique name and description.

Indicator: Stores information about ESG indicators, each indicator belonging to one of the three pillars of environment (E), society (S) or corporate governance (G).

Metric: stores specific ESG metrics values, including name, description, unit and data source.

DataValue: Stores the actual value of each company's corresponding indicator in a specific year.

FrameworkMetric: stores the predefined weight of each metric under a specific framework.

MetricIndicator: stores the predefined weight of each indicator value under each indicator.

UserMetricPreference: Stores custom weights set by the user for specific frames and metrics.

UserIndicatorPreference: stores the custom weights set by the user for each indicator value under a specific indicator.

2 Functionalities

2.1 Log-in page

2.1.1 Login

Here is the login screen, including the fill-in box of username and password, and then jump to the main page if these two inputs are matched.

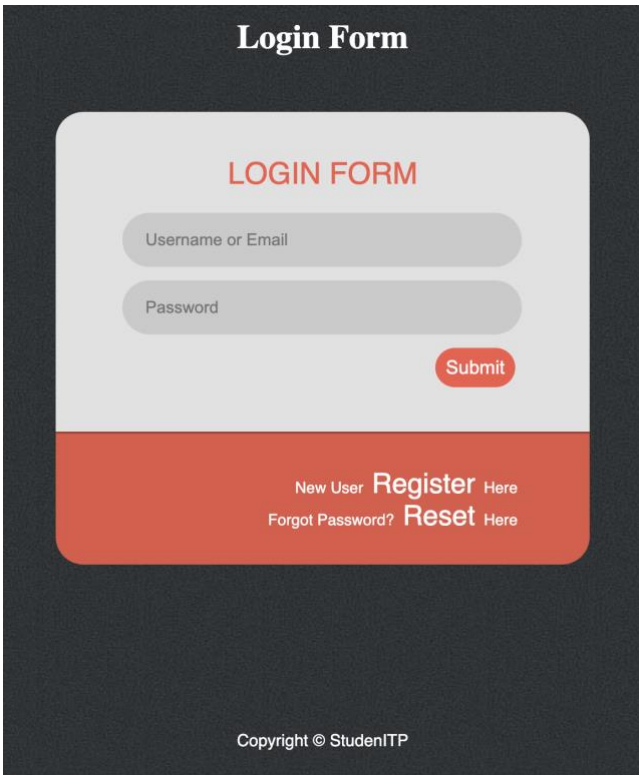

Figure 2 Login interface

It also integrates the registration page and a link to reset the password.

On the backend, Django receives the POST request sent by the front end and extracts the username and password data in the request body. The authentication method is called in the auth model to verify to the database whether the username and password are both correct.

If the authentication is passed, a user session will be created, and the user identification information such as the session and token will be returned to the front end. If authentication fails, the backend returns an error message, such as "username or password information incorrect" then redirects to the login page.

2.1.2 Register

On a register page, the username, email, password, and checked password are required to make a register. An email is required as a method to reset the password if the user forgets.

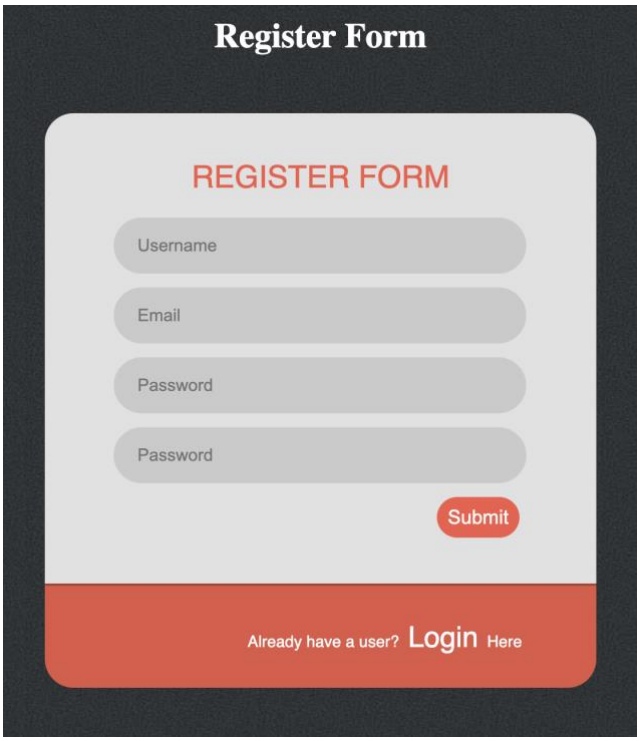

Figure 3 Register interface

The backend extracts the registration information data in the front-end POST request body and uses the model API for data verification. If the data is valid, for example, the username and email are available, a new user will be created through the create method and saved to the database using the save method.

However, the project will send a message “two passwords not same” and redirect the URL to the register page if both passwords are not the same. In addition, If the data is invalid, the backend returns error information, such as "username already exists".

2.1.3 Reset code

If the client unfortunately forgets his password, it still has a method to log in, the server will send an email that includes a check code. The client must provide the code received from the email then the website checks the code is the same and verifies the creation time to make sure is valid.

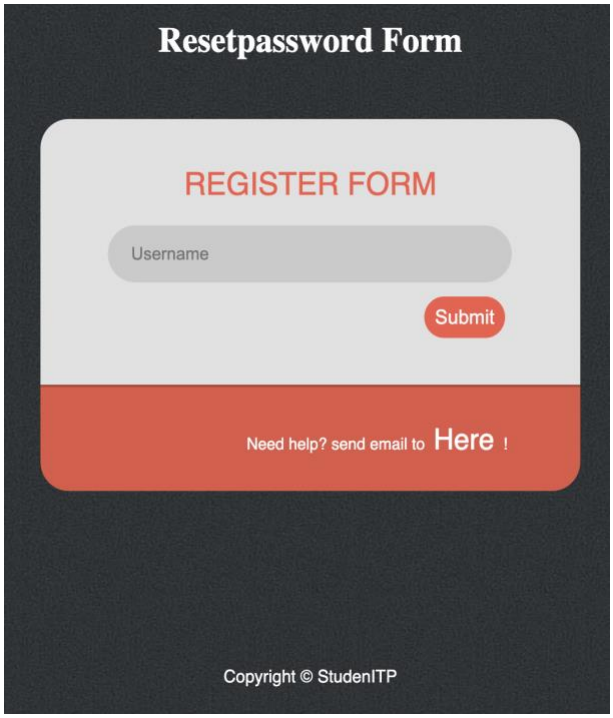

Figure 4 Reset password interface

The reset password in the back end is handled as two respective views, reset password send view and reset password view. The reset password send view is responsible for receiving the request to reset a user's password. Moreover, An email will be sent to the user's corresponding email address by a send-to-user method, which defines the email title, email body including the reset code and where to send, then sending to the address by SMTP method.

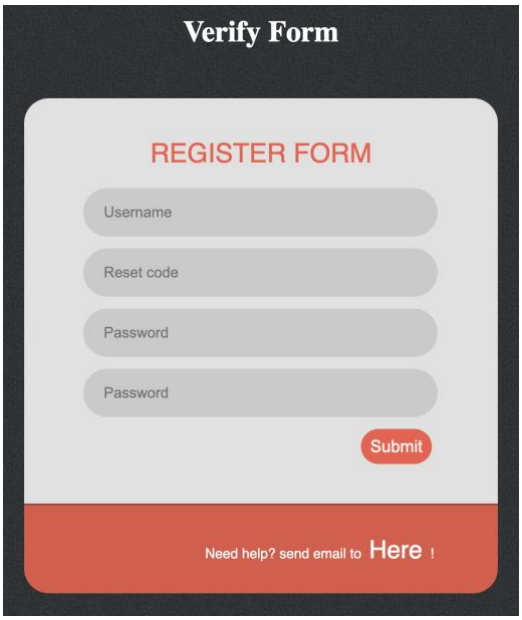

Figure 5 Verify form

The reset password view is responsible for resetting the password using the reset code the user received sent by the above sending function. In contrast, the code will be simultaneously recorded on the server. When the codes from the two places are identical, then a reset action will proceed. In the process, some validation will also run like checking if the new password and the new confirmation password are the same.

2.2 Main page

Here is the overview of the main page:

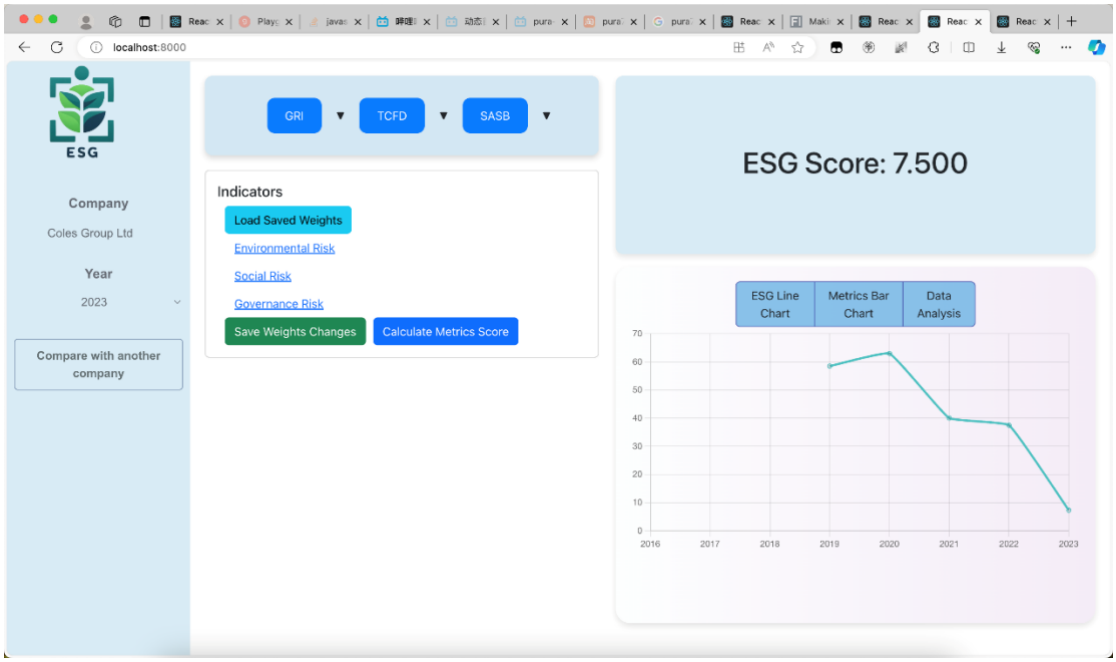

Figure 6 Main page overview

2.2.1 Sidebar

Front end

In the left part of the main page, the website is the sidebar part, it allows users to select the company they are interested in typing, and it will show all the company names that match the eligible letter combinations.

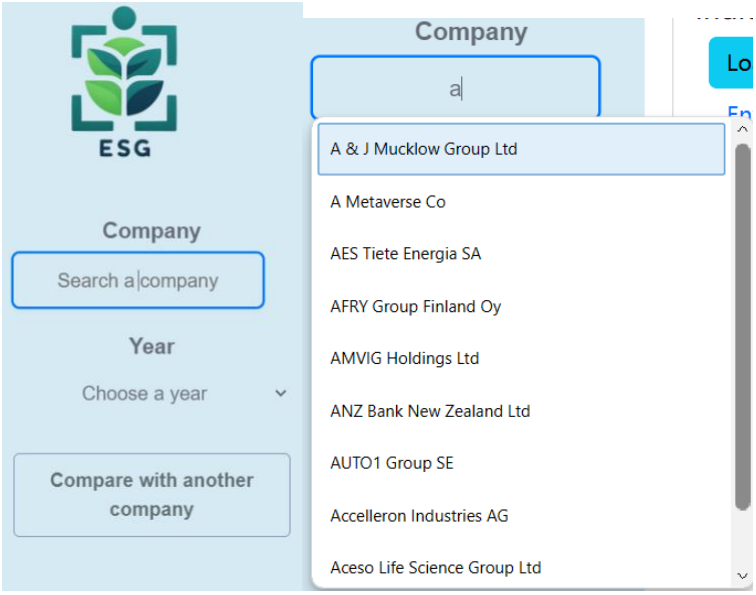

Figure 7 Search companies Function

The next selecting bar is for selecting the year, the dataset of the year is retrieved by the back-end which has possible valid data in a specific year (There is the missing year 2013 due to lack of data in that year) and as a parameter for calculating the ESG score.

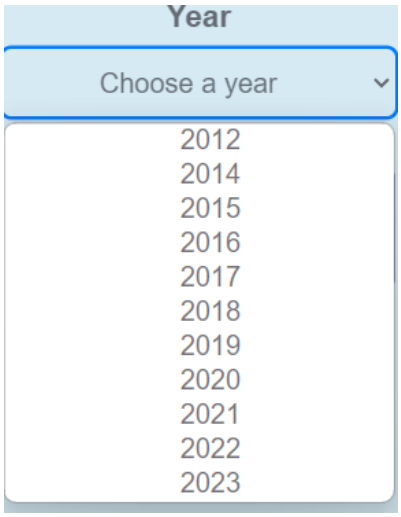

Figure 8 Select year

Back end

Figure 9 Company search

The company name search function is used here. The API endpoints return the first 10 company names by random choice from the database if the search name is not provided.

The second page shows the result when the front end provides the part of the name of the company. Moreover, the system searches any company name starts with the name and returns the first 10 results.

Figure 10 Year list

The function views of years are provided. This API endpoint is used in many places. Therefore, splitting the function in an endpoint can be reused in many pages to increase the software quality.

2.2.2 Frameworks

Front end

In this part, there are three choices of the framework: GRI, TCFD, and SASB provided.

Figure 11 Frameworks

Framework GRI focuses on comprehensive and multifaceted disclosure, providing a board framework containing various indicators. TCFD pays more attention to the influence of the climate with the fewest indicators in these three frameworks, while SASB is particularly suited to the specific industry ESG factors, paying more attention to the long-range operation.

All these three frameworks' brief introductions can be shown after clicking the nabla aside by its name. Here is an example of the framework GRI:

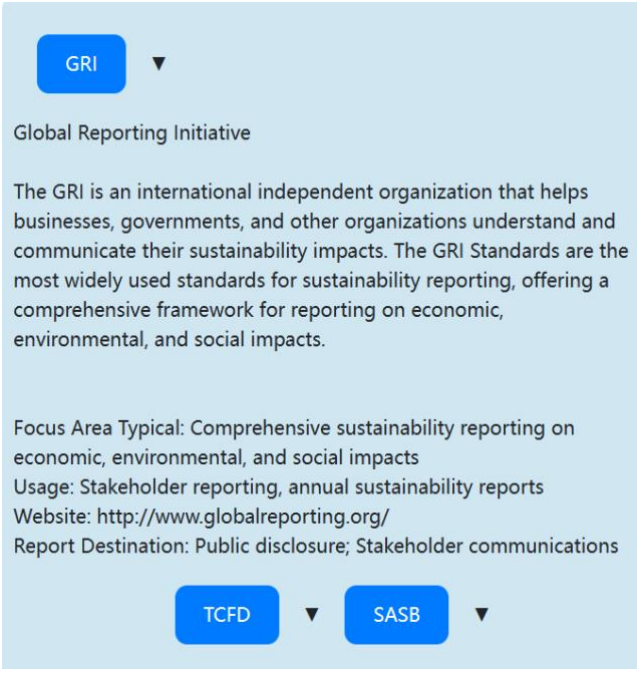

Figure 12 Briefly introduce of the framework

Back end

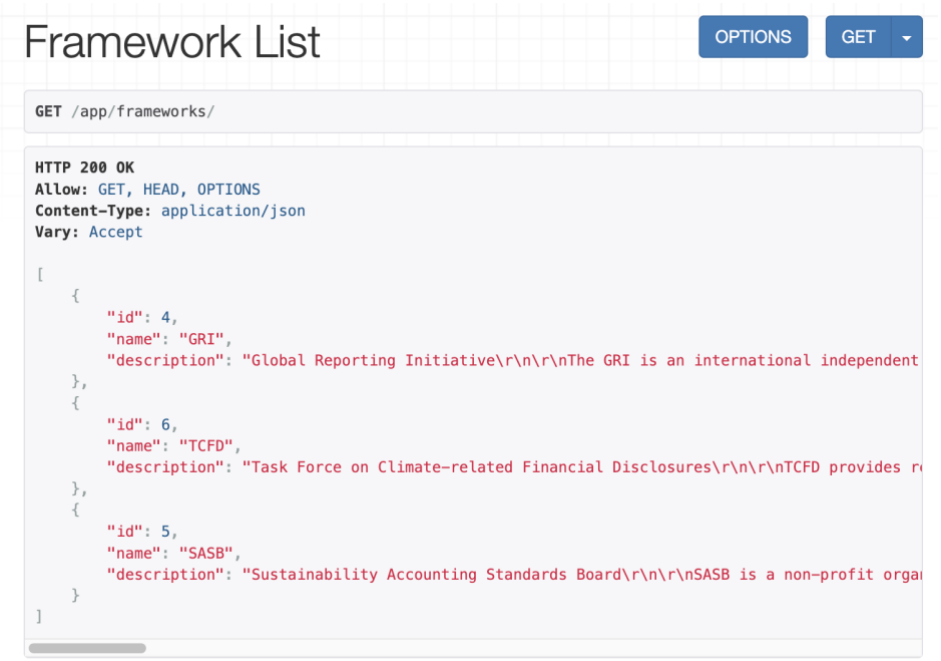

Figure 13 Framework list

This endpoint defines the basic query set of the view set and query set, to get all the framework data from the Framework model. Then the result returns different serializer classes based on different operations (action). The serializer converts the query set into JSON format data and finally returns the serialized data through Response.

To support obtaining a default list of indicators for a specified framework, a custom routing action list_framework_indicators is required by using the @action decorator. In the implementation method of this operation, The program first obtains the specified framework instance object and then filters out all indicators related to the framework from the FrameworkMetric model. If the request parameter contains the pillar parameter (the value is one of E, S, or G), further filter out the indicators related to the pillar. Finally, the filtered indicator query set is converted into JSON format data, and the serialized data is returned through Response.

2.2.3 Metrics and Indicators

Front end

After users select the company and the year, then choose one of the frameworks, it will show all the related data values of metrics that exist in the database belonging to

various indicators. It also has the functionality to show an overview of a global presentation of the entire default metrics and related indicators before confirming the frameworks with the company.

Figure 14 All Social indicators

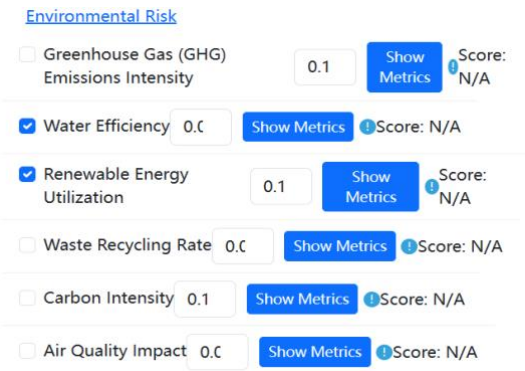

Figure 15 All Environmental indicators

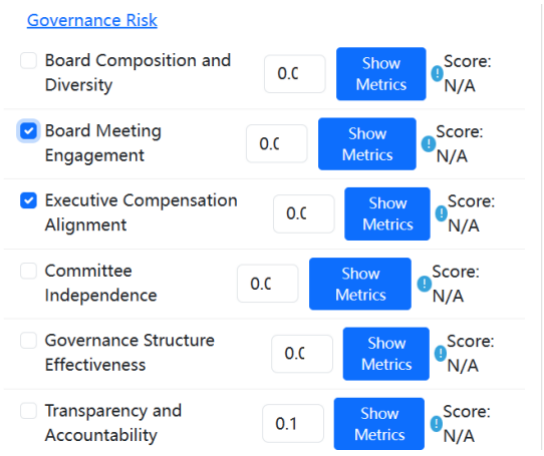

Figure 16 All Governance indicators

Metrics along with certain indicators, can be shown after clicking the “show metrics”. It will show a list containing metrics and their weights to fill in. All the indicators can be selected with setting weight and discarded.

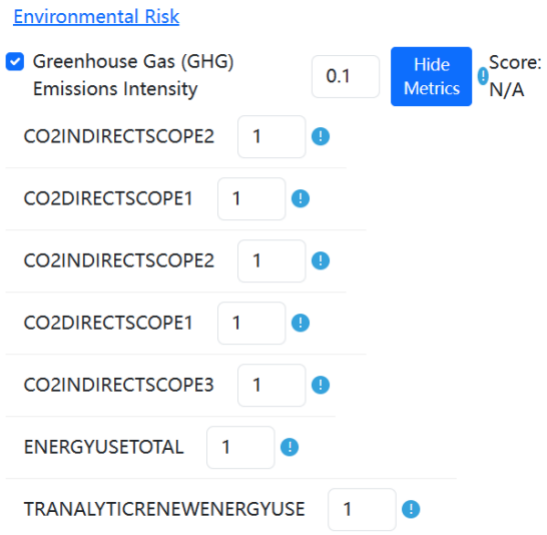

Figure 17 The metrics of individual indicators

There are three remaining buttons: “Load Saved Weights”, “Save Weight Change”, and “Calculate Metrics Score”. If the user clicks on "Load saved weights," the system will retrieve both indicators and metrics weights previously saved by the user from the backend into the weights field. If the user clicks on "Save weights changes," the system will store the current weights in the fields in the database for future retrieval, and update the ESG score. If the user clicks on "Calculate indicator score," the system will calculate the scores for all selected metrics.

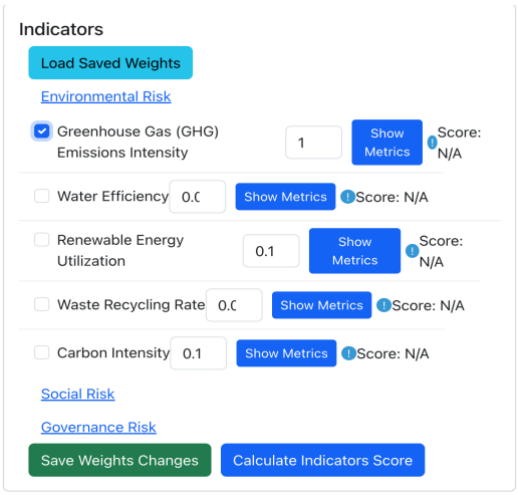

Figure 19 Before click calculate indicator score

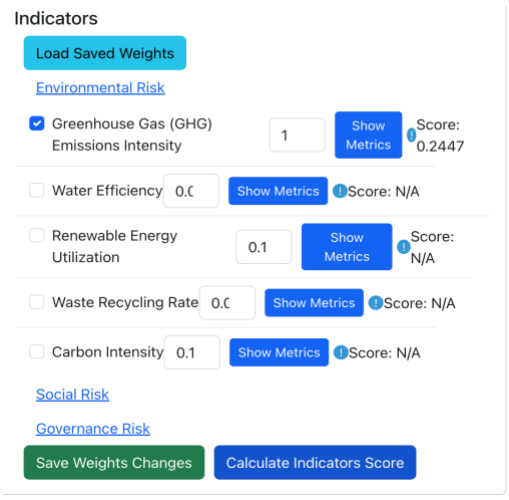

Figure 18 After click calculate indicator score

When the user clicks on "Load saved weights," the system will retrieve the metric weights previously set by the user from the backend. When the user clicks on "Save weights changes," the system will send the user-input data to the backend for storage, updating the ESG score accordingly.

Back end

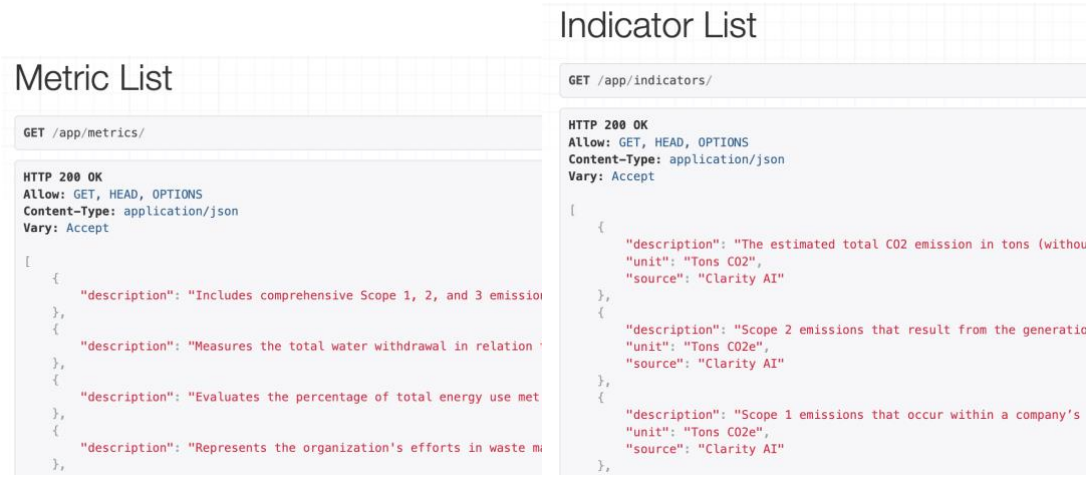

Figure 20 Metric and indicator list

Two view sets IndicatorViewSet and MetricViewSet are developed, which are used to obtain descriptions for metrics and units as well as data sources for indicators respectively. The front end can obtain a list of metrics or indicators through these view sets or can filter specific indicators or indicator group data based on ID. This data can be used to display and manipulate information about metrics and indicators on the front-end interface.

```
GET /app/metricsdatavalue/?companies=1&framework=4&metrics=97
```

```
HTTP 200 OK
Allow: GET, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

{
    "data": [
        {
            "company_id": 1,
            "company_name": "Toyota Motor Corp",
            "metrics_scores": [
                {
                    "year": 2012,
                    "metric_id": 97,
                    "metric_name": "Greenhouse Gas (GHG) Emissions Intensity",
                    "score": -1
                },
                {
                    "year": 2014,
                    "metric_id": 97,
                    "metric_name": "Greenhouse Gas (GHG) Emissions Intensity",
                    "score": -1
                },
                {
                    "year": 2015,
                    "metric_id": 97,
                    "metric_name": "Greenhouse Gas (GHG) Emissions Intensity",
                    "score": -1
                },
```

Figure 21 Metrics data view set (1)

```
GET /app/metricsdatavalue/?companies=1&framework=4&metrics=97&year=2022
```

```
HTTP 200 OK
Allow: GET, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

{
    "data": [
        {
            "company_id": 1,
            "company_name": "Toyota Motor Corp",
            "metrics_scores": [
                {
                    "year": "2022",
                    "metric_id": 97,
                    "metric_name": "Greenhouse Gas (GHG) Emissions Intensity",
                    "score": 0.11241007194244604
                }
            ]
        }
    ]
}
```

Figure 22 Metrics data view set (2)

In addition, the MetricsDataViewSet view set is provided, which is used to calculate the performance of each specified company under the specified year, framework and indicator based on the parameters provided by the front end (such as company ID list, framework ID, indicator ID list and year). After calculating the score, the API

endpoint returns these data to the front end for display or other operations. This view set provides core metric score data support for the front end. However, the result is manually set to -1 when some data are missing.

Through these view sets and views, a comprehensive API interface is provided for the front end, supporting the acquisition and manipulation of various data such as frames, indicators, indicator groups, indicator values, and indicator scores. The front end can flexibly call these API interfaces according to needs, obtain the required data, and perform corresponding display and interaction on the interface. At the same time, because we use the Django REST Framework, these API interfaces have good scalability and maintainability and can be adjusted and expanded according to business needs.

2.2.4 Estimate score

Front end

The ESG score is calculated by assigning weights and aggregating all the single estimations, and here is the single score output with selected indiators and metrics in the front end.

Figure 23 The main page after the score is shown

In the ESG assessment system, the backend provides a key function - calculating the company's comprehensive performance score in different years and frameworks. This function is completed by several core modules working together. It will be explained in detail in subsequent chapters.

Back end

```
{
  "GRI": {
    "2016": {
      "total_score": 17.2163322545847,
      "metrics": {
        "Carbon Intensity": 0.0816612729234088,
        "Labor Relations Quality": 86
      }
    },
    "2017": {
      "total_score": 17.5484017278618,
      "metrics": {
        "Water Efficiency": 0.657667386609071,
        "Carbon Intensity": 0.084341252699784,
        "Labor Relations Quality": 87
      }
    },
    "2018": {
      "total_score": 17.5302575107296,
      "metrics": {
        "Water Efficiency": 0.651287553648069,
        "Labor Relations Quality": 87
      }
    },
    "2019": {
      "total_score": 2.398,
      "metrics": {
        "Workforce Diversity": 11.99
      }
    },
    "2020": {
      "total_score": 54.2196356399686,
      "metrics": {
        "Water Efficiency": 0.628571428571429,
        "Renewable Energy Utilization": 0.00822857142857143,
        "Carbon Intensity": 0.0811428571428571,
        "Diversity in Leadership": 11.605,
        "Employee Turnover Rate": 1.1,
        "Labor Relations Quality": 91,
        "Board Composition and Diversity": 22.22,
        "Board Meeting Engagement": 99.2,
        "Governance Structure Effectiveness": 44.44,
        "Transparency and Accountability": 0.8152353427
      }
    },
```


Figure 24 Total scores of the year

First, the apply_metric_formula function calculates the company's specific indicator values based on the specified framework and indicators and generates the corresponding single indicator score. This function contains calculation rules for multiple frameworks and indicators, ensuring the accuracy of score calculation.

Next, the aggregate_scores function aggregates multiple indicator scores from apply_metric_formula and combines them with the framework weights to calculate the company's comprehensive score in a specific year and framework. At the same time, it also records the individual scores of each indicator to provide data support for subsequent analysis.

Finally, the CompanyPerformance view obtains the required original data such as indicator values and weights through database queries and applies user-defined weight preferences. Then, it iterates through the year and frame combinations, calling aggregate_scores to perform the score calculation. The calculation results are returned in JSON format for front-end presentation and use.

2.2.5 Charts and Analysis

There are two different types of graphs to give a more visual representation of the dataset analyzed with a variety of emphases. When the user clicks on the ESG line chart, it will display the trend of the selected company's ESG scores over the years. When the user clicks on the Metrics bar chart, it will show a comparison of the selected company's metrics data for the chosen year.

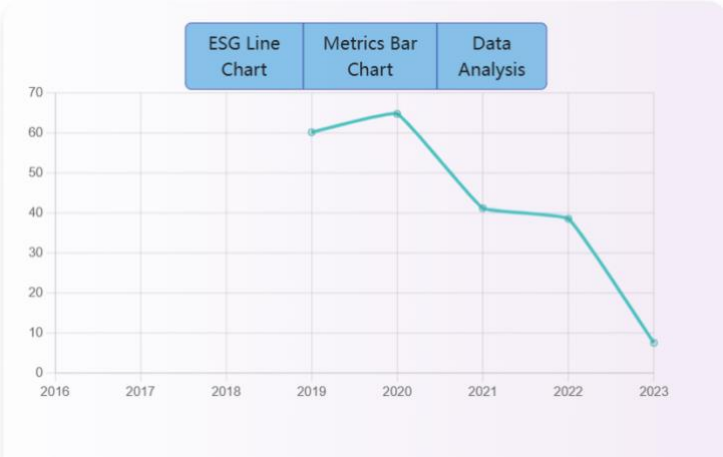

Figure 25 Line chart of the ESG scores for a certain company across different years

Figure 26 Bar chart for the metrics value for a company

User could also click data analysis to use AI to analyze the dataset with the current relationship of the E, S, and G parts and their scores. Like the analysis of the behaviour based on the given dataset.

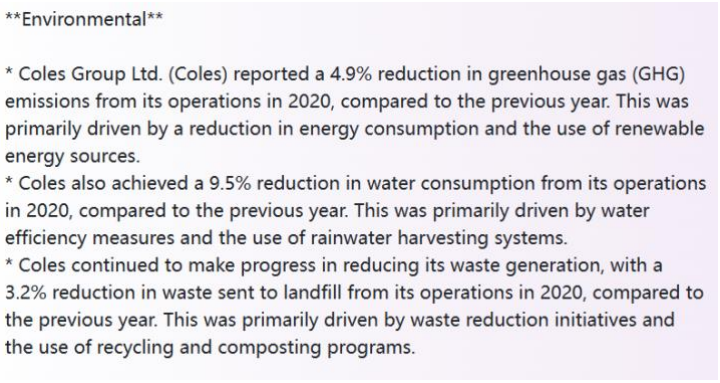

Figure 27 Data analysis

For the back end part, the MetricsDataViewSet view is reused here which was mentioned before. This view can provide the metric score which is used to render different graphs.

2.3 Compare-page

2.3.1 Front end

The compare page is going to compare two different companies with their ESG score, specific indicators, and their metrics line graph in a version site. From the example page, we can see two famous companies, coles and Woolworths with the comparison in different metrics like carbon dioxide scope estimate and energy use in total.

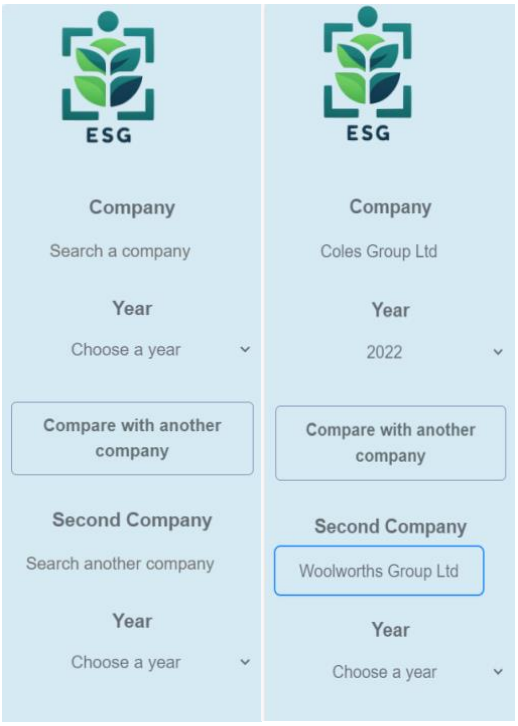

Figure 28 The function of comparing companies

This page shows the ESG scores of these two companies, and below are all the selected indicators if you click one of these bars, it shows all the metrics in this indicator and shows the difference.

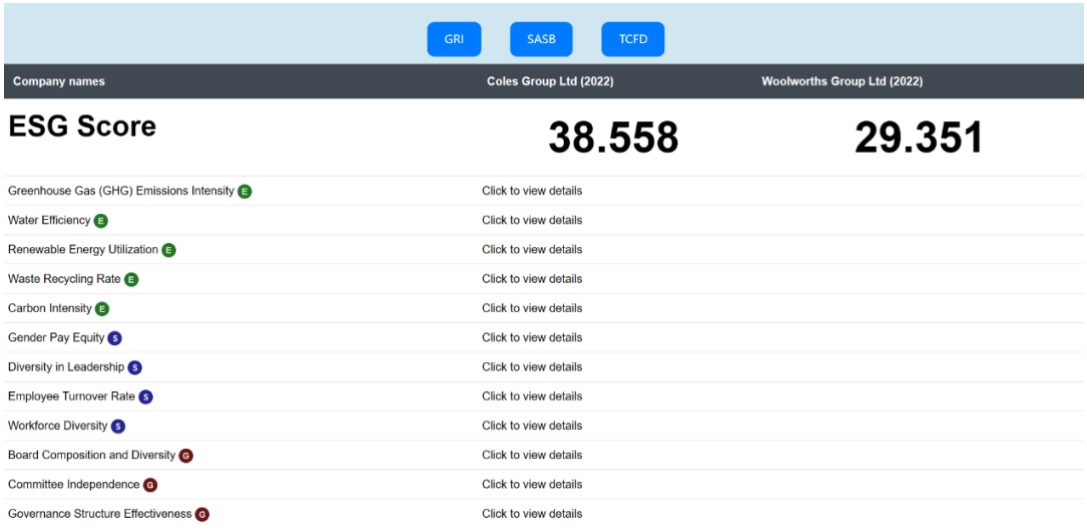

Figure 29 Comparing page

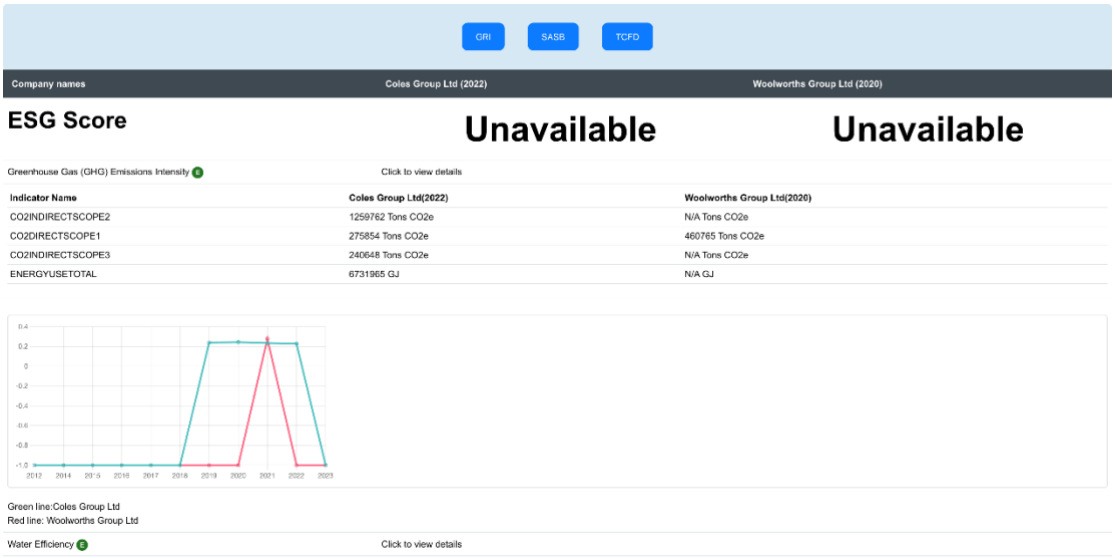

Figure 30 Comparing page for selecting one indicator

On the comparison page, users can compare the ESG scores of two companies across multiple dimensions. Initially, users can visually see the total ESG score. When a user clicks on a specific metric, the system will display detailed information about that metric. Firstly, it will show a comparison of values for different indicators. Then, it will display a line chart comparing the performance of the two companies for this metric across different years.

2.3.2 Back end

Figure 31 Indicator value

The ListIndicatorValue view, which uses native SQL queries to get detailed indicator value data for a specified company, frame, and year is provided. The native SQL is used because the performance has to increase when the whole function is developed through Rest Framework Serializer. This is because the number of data values is around 700000 which is stored in our database. Therefore, a function that is used to render JSON data is required to replace the serializer. The data returned by this view includes the indicator value itself and some metadata, such as indicator ID, name, pillar, weight, unit, and source which provides rich data support for the front end and can be used for further display, analysis and calculation.

2.4 Save user preferences

2.4.1 Front end

When a user modifies certain weightings and clicks the "Save Weights Change" button, as shown in the diagram below, the system will notify the user that the weightings have been successfully updated. Additionally, it will update the ESG score with the new weighted calculation and display the updated score to the user.

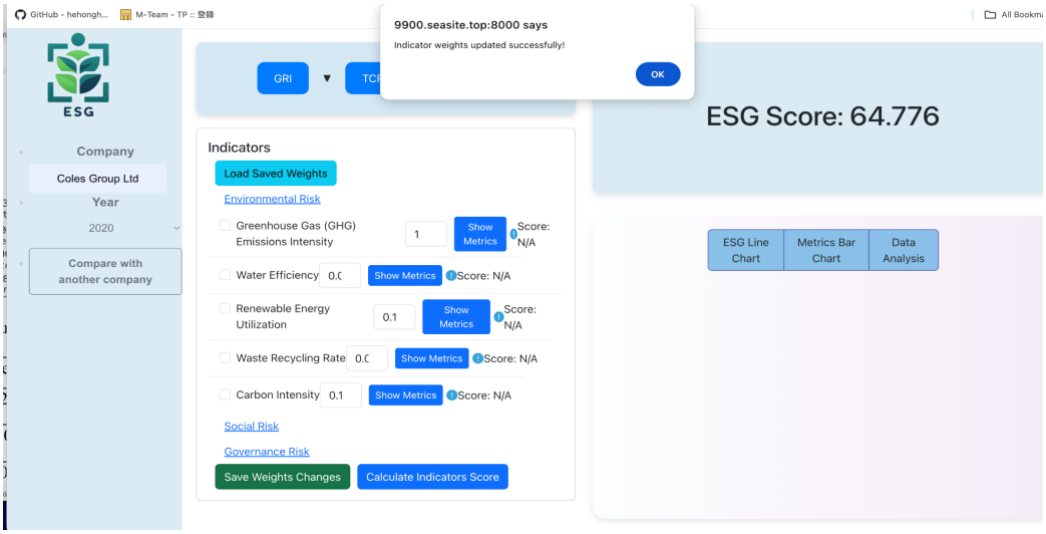

Figure 32 Prompt box after clicking "Save Weights Change"

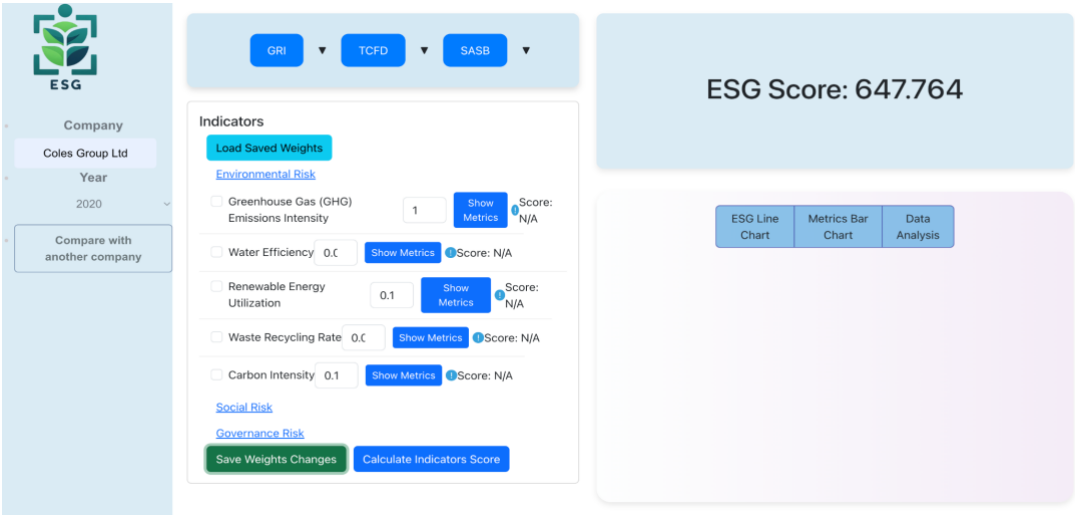

Figure 33 The weight change after clicking "Save Weights Change"

2.4.2 Back end

On the one hand, the back end provides an important feature - saving user-defined weighting preferences from the front end. This function allows users to adjust the weight distribution of the metric and indicator values according to their own needs, making the evaluation results closer to actual scenarios.

Specifically, the backend provides two API views `SaveMetricPreferences` and `SaveIndicatorPreferences`, which are used to process metric weight and indicator value weight change requests sent by the front end respectively.

When the front end sends a request, these two views will first use the serializer to verify the request data to ensure that the data is legal and valid. Once authenticated, they perform the following core operations:

1. Query the database to obtain existing user-defined weight records.

2. Perform batch update and create operations in a database transaction:

- For existing weight records, update their `custom_weight` field to reflect the user's latest preferences.

- For the new weight record, create a new record in the database.

These operations are performed within transactions, ensuring data consistency and integrity.

Through this function, users can flexibly adjust the weight of the framework, indicators and indicator values, so that the final evaluation results are more in line with their judgment and needs. At the same time, the back-end data persistence operation also provides reliable data support for the front end, ensuring the permanent storage and application of user preferences.

This function is an important reflection of user personalization and flexibility in the entire system, making our ESG assessment not only objective but also adaptable to the special needs of different users and scenarios.

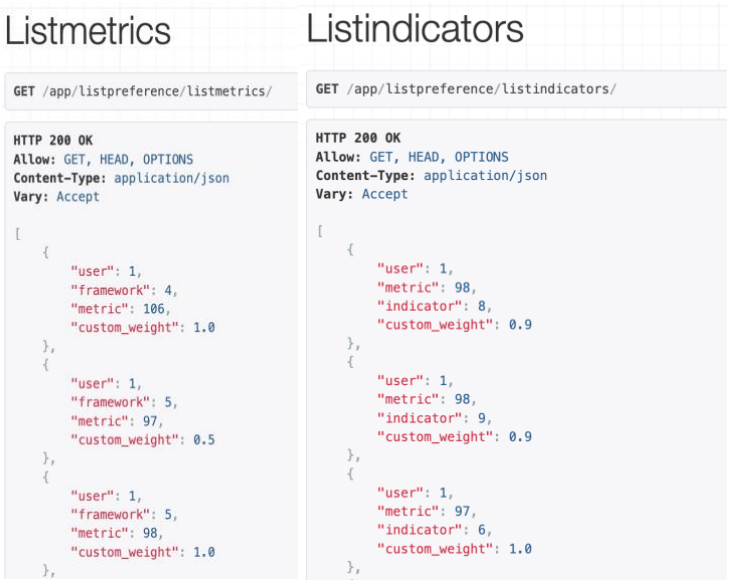

Figure 34 User preferences

On the other hand, the Listmetrics and Listindicators API endpoints are used to return client definition weight after the client clicks the “Load saved weight” button. The back end returns personal weight associated with the login user and the front end replaces those values.

3 Implementation challenge

3.1 Front-end part

3.1.1 Data visualization

As a result of the ESG score, we introduce two different graphs to visualize the ESG score more intuitively and show the relationship between years and selected metrics. On the compare page, you can see the comparison of the two companies in one line graph.

Figure 35 Line chart of two companies

3.1.2 AI Introduce

By utilizing AI to analyze the statistics of the ESG score on various sides while considering the behavior and history of the specific company to make the conclusion more accurate. Different from other investigating websites, our site gives an alternative and rational assessment to avoid the limitation of a single piece of information.

3.2 Back-end part

3.2.1 Implement complex logical judgments of multiple models

By using dictionary mapping to replace traditional if-else statements, code readability and maintainability are significantly improved. The dictionary data structure is lifted, where the conditions to be evaluated are used as keys, and the corresponding operations or values are used as values. By looking up the dictionary keys and returning the corresponding values, large if-else statement blocks can be avoided to use, making the code more readable, robust and easier to maintain.

3.2.2 Implementing an automated deployment environment

Docker and Docker Compose are combined to use to automate the deployment and management of databases and websites, streamlining the deployment process. Docker allows for the packaging of applications and their dependencies into lightweight, self-contained containers, ensuring consistent behavior across different environments. Docker Compose, on the other hand, enables the orchestration of multi-container applications, simplifying the process of defining, running, and managing multiple containers simultaneously.

3.2.3 Performance bottleneck caused by processing large amounts of data

To alleviate the performance bottleneck caused by processing large amounts of data, the database is split into multiple smaller tables. By reducing the complexity of a single table, we can significantly improve query efficiency. This approach not only enhances overall system performance but also facilitates better data organization and management.

3.2.4 Improve front-end and back-end data exchange efficiency while ensuring data security

The front-end and back-end are merged by rendering the front-end through Django, a high-level Python web framework. This approach allows us to generate pages directly on the server-side, eliminating unnecessary data transmission between the front-end and back-end. Consequently, we can achieve improved efficiency and prevent potential data leakage, thereby enhancing bot

3.3 Database part

3.3.1 Processing and quantifying unstructured data sources

Since the data source is in the form of csv, it cannot be directly quantified in the database. Therefore, it is necessary to refine the split database schema and design multiple tables to extract and transform this unstructured data. Besides clever schema design, unstructured data is converted and loaded into the database by Django command as script, laying the foundation for subsequent data processing and analysis. Although this mode splitting method increases the complexity of the database, it enables the system to improve the flexibility and scalability of the system.

4 Installation and user documentation

The README.md has shown how to manually install the project on a machine. For the first time installing it, it is recommended to download the program to the "product" folder by the command "git clone xxx product". Then the below steps are executed.

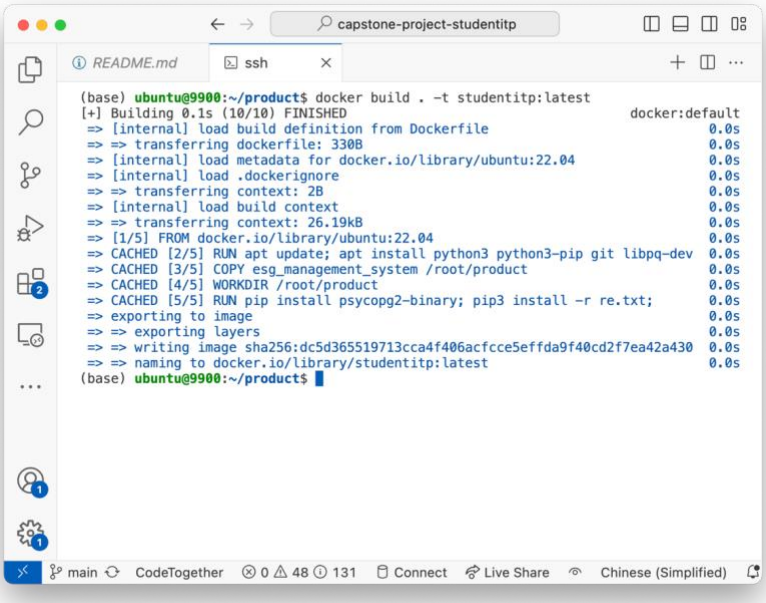

Figure 36 Installation (1)

docker build . -t studentitp:latest let the machine build the docker mirror of the project.

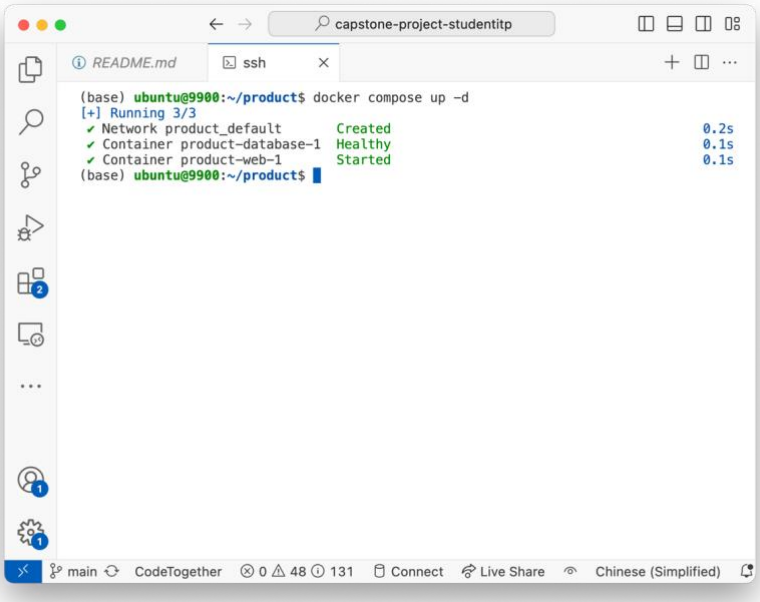

Figure 37 Installation (2)

docker compose up -d let the docker container start to run the web front and back end.

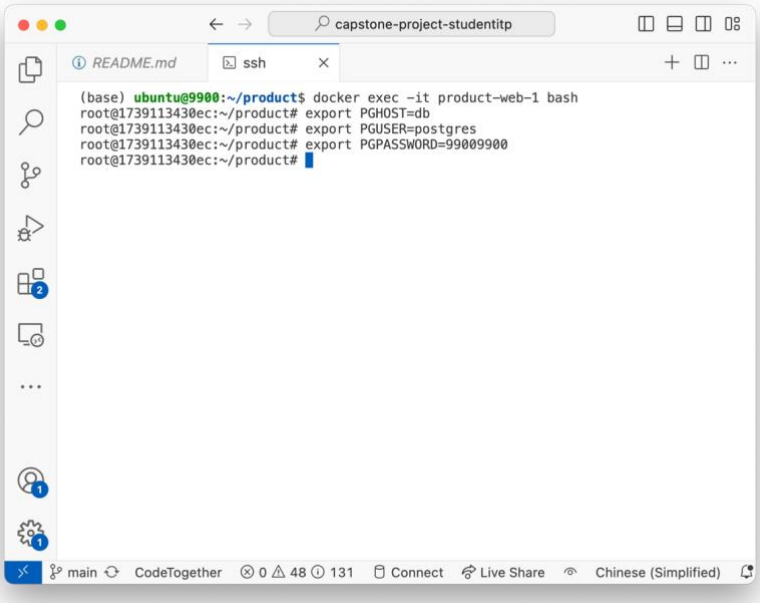

Figure 38 Installation (3)

```

export PGHOST=db

export PGUSER=postgres
```

export PGPASSWORD=99009900

```

These commands create the environment variables and let PostgreSQL know the username, password and the database host.

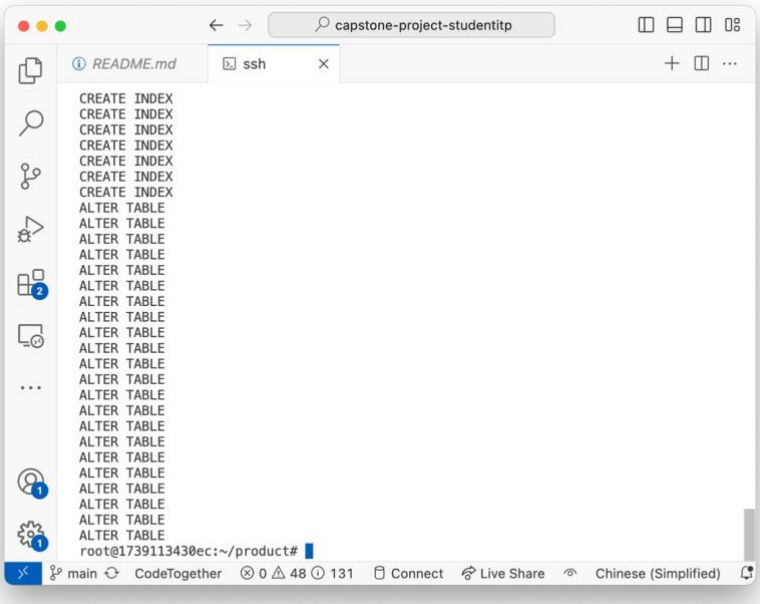


Figure 39 Installation (4)

Finally, psql esg < database.sql command initializes all the tables in the database using the ESG data sets that we saved previously for the installation.

After all the above operations, the website now starts to run normally.
```

5 Reference

BROADSTOCK, D. C., CHAN, K., CHENG, L. T. & WANG, X. 2021. The role of ESG performance during times of financial crisis: Evidence from COVID-19 in China. *Finance research letters,* 38**,** 101716.

DORFLEITNER, G., HALBRITTER, G. & NGUYEN, M. 2015. Measuring the level and risk of corporate responsibility–An empirical comparison of different ESG rating approaches. *Journal of Asset Management,* 16**,** 450-466.

ESCRIG-OLMEDO, E., MUÑOZ-TORRES, M. J. & FERNANDEZ-IZQUIERDO, M. A. 2010. Socially responsible investing: sustainability indices, ESG rating and information provider agencies. *International journal of sustainable economy,* 2**,** 442-461.

KOTSANTONIS, S. & SERAFEIM, G. 2019. Four things no one will tell you about ESG data. *Journal of Applied Corporate Finance,* 31**,** 50-58.

LI, T.-T., WANG, K., SUEYOSHI, T. & WANG, D. D. 2021. ESG: Research progress and future prospects. *Sustainability,* 13**,** 11663.

MATOS, P. 2020. ESG and responsible institutional investing around the world: A critical review.

VAN DUUREN, E., PLANTINGA, A. & SCHOLTENS, B. 2016. ESG integration and the investment management process: Fundamental investing reinvented. *Journal of business ethics,* 138**,** 525-533.

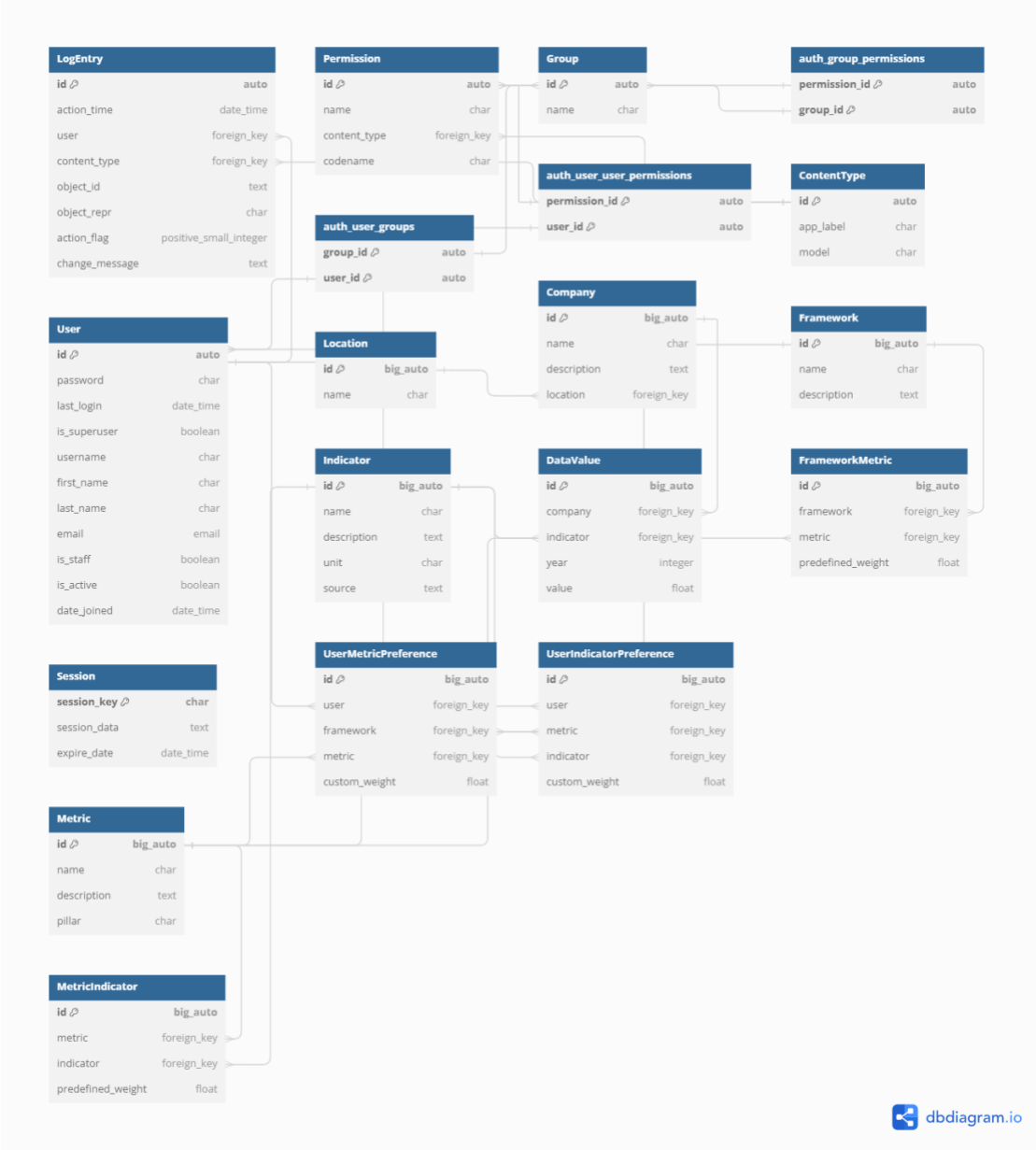

Figure 40 (Schema generate by dbml using dbdiagram.io)