МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет імені Петра Могили
Факультет комп’ютерних наук
Кафедра інтелектуальних інформаційних систем

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувач кафедри інтелектуальних інформаційних систем
_____Юрій КОНДРАТЕНКО
«____»_____ 2025 р.

КВАЛІФІКАЦІЙНА РОБОТА
НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ БАКАЛАВРА

АВТОМАТИЗОВАНА ІНФОРМАЦІЙНО-АНАЛІТИЧНА СИСТЕМА ПО ВИЗНАЧЕННЮ ПРІОРИТЕТІВ СТАЛОГО РОЗВИТКУ УНІВЕРСИТЕТСЬКОЇ СИСТЕМИ ОСВІТИ

Спеціальність 122 Комп’ютерні науки
Освітня програма «Комп’ютерні науки»

Здобувач _____ Євген МАЛЬОНА
«___»_____ 2025 р.
Керівник д-р пед.наук, професор _____ Олександр МЄЩАНІНОВ
«___»_____ 2025 р.

Миколаїв – 2025

Чорноморський національний університет імені Петра Могили
(повне найменування закладу вищої освіти)

Факультет	Комп’ютерних наук
Кафедра	Інтелектуальних інформаційних систем
Рівень вищої освіти	Перший (бакалаврський)
Освітній ступень	Бакалавр
Спеціальність	122 Комп’ютерні науки
Освітня програма	Комп’ютерні науки

ЗАТВЕРДЖУЮ

Завідувач кафедри інтелектуальних інформаційних систем

_____ Юрій КОНДРАТЕНКО

«____» _____ 2025 р.

ЗАВДАННЯ
на кваліфікаційну роботу здобувача

_____Мальони Євгена Івановича_____
((прізвище, ім’я, по батькові здобувача)

1. Тема кваліфікаційної роботи: «Автоматизована інформаційно-аналітична система по визначенню пріоритетів сталого розвитку університетської системи освіти».

Керівник роботи: Мєщанінов Олександр Павлович, професор кафедри інтелектуальних інформаційних систем, д-р пед.наук, професор.

Затверджена наказом ЧНУ ім. Петра Могили від «18» листопада 2024 р. № 321.

2. Строк представлення кваліфікаційної роботи «___» _____ 2025 р.

3. Очікуваний результат роботи та початкові дані, якщо такі потрібні: система по визначенню пріоритетів сталого розвитку університетської системи освіти; 8 показників сталого розвитку, ваги показників.

4. Перелік питань, що підлягають розробці: аналіз сучасного стану систем для сталого розвитку університетів; огляд існуючих систем орієнтованих на сталий

розвиток; Розробка втоматизованої інформаційно-аналітичної системи по визначенню пріоритетів сталого розвитку університетської системи освіти.

5. Перелік графічних матеріалів: презентація , рисунки , таблиці.

Керівник роботи _____ Олександр МЄЩАНІНОВ
(Особистий підпис) *(Власне ім'я ПРІЗВИЩЕ)*

Здобувач _____ Євген МАЛЬОНА
(Особистий підпис) *(Власне ім'я ПРІЗВИЩЕ)*

Дата видачі завдання «18» грудня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

кваліфікаційної роботи

Тема: Автоматизована інформаційно-аналітична система по визначенню пріоритетів сталого розвитку університетської системи освіти

№	Найменування роботи	Початок	Закінчення	Примітки
1	Отримання завдання на виконання КР	18.12.2024	20.12.2024	
2	Аналіз предметної області та постановка задачі	21.12.2024	30.01.2025	
3	Огляд літературних джерел за темою кваліфікаційної роботи, зокрема огляд публікацій та аналогічних систем, пов'язпних зі сталим розвитком	31.01.2025	01.03.2025	
4	Огляд існуючих засобів та методів для реалізаціх потрібної системи	02.03.2025	01.04.2025	
5	Реалізація обраних технологій з аналізом отриманих результатів	02.04.2025	15.05.2025	
6	Перший попередній захист КР на засіданні комісії кафедри	16.05.2025	16.05.2025	
7	Корегування роботи за результатами попереднього захисту	29.05.2025	29.05.2025	
8	Другий попередній захист КР на засіданні комісії кафедри	30.05.2025	30.05.2025	
9	Доробка та остаточне оформлення КР	31.05.2025	10.06.2025	
10	Подання КР, її електронної копії та інших документів (відгуку, рецензії) до захисту	16.06.2025	16.06.2025	

Керівник роботи _____ Олександр МЄЩАНІНОВ
(Особистий підпис) *(Власне ім'я ПРІЗВИЩЕ)*

Здобувач _____ Євген МАЛЬОНА
(Особистий підпис) *(Власне ім'я ПРІЗВИЩЕ)*

Дата складання календарного плану
«29» січня 2025 р.

АНОТАЦІЯ

до кваліфікаційної роботи
здобувача групи 401 ЧНУ ім. Петра Могили

Мальони Євгена Івановича

на тему: “АВТОМАТИЗОВАНА ІНФОРМАЦІЙНО-АНАЛІТИЧНА СИСТЕМА ПО ВИЗНАЧЕННЮ ПРІОРИТЕТІВ СТАЛОГО РОЗВИТКУ УНІВЕРСИТЕТСЬКОЇ СИСТЕМИ ОСВІТИ”

Актуальність даного дослідження полягає у зростаючій необхідності інтеграції принципів сталого розвитку в освітню та операційну діяльність університетів. Сучасні виклики вимагають від вищих навчальних закладів не лише адаптації до вимог сталості, але й проактивного визначення пріоритетів та ефективного моніторингу прогресу. Автоматизована інформаційно-аналітична система дозволяє забезпечити об'єктивну оцінку, виявити проблемні зони та сприяти прийняттю обґрунтованих рішень для підвищення рівня сталого розвитку університетського середовища.

Об'єктом роботи є процеси збору, обробки, аналізу та візуалізації даних, що характеризують сталий розвиток університетської системи.

Предметом роботи є методи проектування та програмної реалізації інформаційно-аналітичних систем для визначення пріоритетів сталого розвитку, включаючи методи інтегральної оцінки та візуалізації даних.

Метою роботи є розробка автоматизованої інформаційно-аналітичної системи, здатної ефективно збирати, аналізувати та візуалізувати показники сталого розвитку університетської системи освіти, що дозволить визначити пріоритетні напрямки для вдосконалення та підвищити загальний рівень сталості.

В результаті виконання роботи було досліджено сучасні підходи до оцінки сталого розвитку. Розроблено програмне забезпечення автоматизованої інформаційно-аналітичної системи, яка включає модулі для керування користувачами, імпорту та обробки даних, аналітики та візуалізації показників, а також інтерактивні модулі для взаємодії користувачів (чат, ідеї, опитування).

Дана робота складається з чотирьох розділів. У першому розділі проведено аналіз предметної області, огляд існуючих систем та сформульовано постановку задачі. Другий розділ присвячений огляду технологій та методів, що використані у розробці системи. У третьому розділі наведено результати проектування та моделювання інформаційно-аналітичної системи. В четвертому – описано програмну реалізацію, аналіз отриманих результатів, тестування системи.

Загальний обсяг роботи – 99 сторінок. Кваліфікаційна робота містить 2 додатки, 37 рисунків, 1 таблицю і 27 джерел посилання.

Ключові слова: сталий розвиток , інформаційно-аналітична система , система освіти , аналітика.

ABSTRACT

to the qualification work by the student of the group 401 of Petro Mohyla Black Sea National University

Malona Yevhen

" AUTOMATED INFORMATION AND ANALYTICAL SYSTEM FOR DETERMINING PRIORITIES FOR SUSTAINABLE DEVELOPMENT OF THE UNIVERSITY EDUCATION SYSTEM "

The relevance of this study lies in the growing need to integrate sustainable development principles into the educational and operational activities of universities. Modern challenges require higher education institutions not only to adapt to sustainability requirements, but also to proactively set priorities and effectively monitor progress. An automated information and analytical system allows for objective assessment, identification of problem areas, and facilitates informed decision-making to improve the level of sustainable development in the university environment.

The object of the work is the processes of collecting, processing, analyzing, and visualizing data that characterize the sustainable development of the university system.

The subject of the work is the methods of designing and implementing information and analytical systems for determining sustainable development priorities, including methods of integrated assessment and data visualization.

The goal of the work is to develop an automated information and analytical system capable of effectively collecting, analyzing, and visualizing indicators of sustainable development of the university education system, which will allow identifying priority areas for improvement and increasing the overall level of sustainability.

As a result of the work, modern approaches to assessing sustainable development were investigated. Software for an automated information and analytical system was developed, which includes modules for user management, data import and processing,

analytics and visualization of indicators, as well as interactive modules for user interaction (chat, ideas, surveys).

This work consists of four sections. The first section analyzes the subject area, reviews existing systems, and formulates the problem statement. The second section is devoted to reviewing the technologies and methods used in the development of the system. The third section presents the results of the design and modeling of the information and analytical system. The fourth section describes the software implementation, analysis of the results obtained, and system testing.

The overall scope of the work is 99 pages. Thesis contains 2 applications, 27 figures, 1 tables and 27 references in it.

Keywords: sustainable development, information and analytical system, education system, analytics.

ЗМІСТ

ВСТУП

Якщо брати до уваги глобальні виклики XXI століття, то одразу стає зрозумілим той факт що принципи сталого розвитку стають все більш важливими для будь-якої сфери нашого життя , а для освіти та науки найбільш важливими. Університети, як місця знань та інновацій, відіграють одну з найважливіших ролей у формуванні сталого майбутнього , додаючи відповідні програми та курси , які на це орієнтовані у основну навчальну програму, дослідження та власну операційну діяльність. Ефективний моніторинг та аналіз показників сталого розвитку університету є необхідною умовою для оцінки прогресу, виявлення слабких місць та прийняття обґрунтованих управлінських рішень, спрямованих на досягнення цілей сталості.

Наразі існує сильна потреба розробці або у модернізації існуючих інформаційних інструментів, які б могли реалізовувати функціонал систематизованого збору, зберігання, обробки та візуалізації інформації , яка пов’язана з багатьма аспектами сталого розвитку університетської системи освіти. Наявні підходи та програмні застосунку зачасту є частковими та неповноцінними у цьому питанні , так як вони не забезпечують необхідної гнучкості для аналізу даних у різних вимірах або взагалі не забезпечують аналіз будь-яких даних. Це зумовлює актуальність розробки спеціалізованих інформаційно-аналітичних систем, здатних вирішити завдання комплексного моніторингу та підтримки прийняття рішень у сфері сталого розвитку закладів вищої освіти.

1 АНАЛІЗ СУЧАСНОГО СТАНУ ПОСТАВЛЕНОЇ ЗАДАЧІ

1.1 Опис предметної сфери

Розробка автоматизованої інформаційно-аналітичної системи спрямована на визначення пріоритетів сталого розвитку університетської системи освіти. Сталий розвиток[1] у вищій та не тільки освіті є комплексним підходом, що поєднує різні показники такі як екологічну стійкість, соціальну відповідальність та економічну стабільність.

Університети, як основні одиниці для створення знань і просування інновацій у суспільство, відіграють провідну роль у реалізації глобальних ініціатив, таких як Цілі сталого розвитку ООН. Окремо можна виділити ціль 4[2] - якісна освіта , вона акцентує на забезпеченні доступної освіти, тоді як ціль 13[3] - кліматичні дії наполягає на необхіднісі зменшення екологічного впливу. Запропонована система має на меті бути саме тим веб-застосунком, який сприятиме формуванню спільного бачення сталого розвитку серед різних людей які будуть нею користуватись - адміністратори, викладачі та студенти - та визначенню різних шляхів для його досягнення.

Сталий розвиток охоплює кілька ключових напрямів діяльності університетів:

- екологічна стійкість: зменшення викидів вуглексилого газу, впровадження енергоефективних технологій, управління та переробка відходів, використання відновлюваних джерел енергії. Наприклад, покращення гуртожитків для зниження енергоспоживання чи переробка відходів є вкрай необхідними кроками;
- соціальна відповідальність: підтримка різноманітності, доступність освіти для різних груп людей , а також взаємодопомого місцевим громадам та співпраця з ними через освітні програми та соціальні ініціативи;
- економічна стабільність: оптимізовний розподіл фінансових витрат, залучення грантів і коштів спонсорів , розробка сучасних бізнес-моделей, таких як

створення стартапів.

Успіх сталого розвитку також неодмінно залежить від активної участі всіх користувачів :

– адміністрація: вона несе відповідальність за стратегічне планування, розподіл ресурсів і впровадження діяльностей які спрямовані на сталий розвиток;

– викладачі: впроваджують принципи сталого розвитку в навчальні програми навчальних закладів, проводять дослідження з екологічних і соціальних питань серед студентів і колег;

– студенти: беруть участь у зелених ініціативах, волонтерських проєктах і формуванні культури сталого розвитку.

В університетах україни перехід до сталого розвитку здебільшого ускладнений різними факторами. По-перше – через обмежене фінансування навчальних закладів зачасту не є можливим впроваджувати енергоефективні технології чи модернізувати інфраструктуру , наприклад гуртожитки. По-друге - відсутність системи збору та аналізу даних сталого розвитку ускладнює дослідження стану сталого розвитку на данний момент. Наприклад, дані про енергоспоживання чи соціальні ініціативи або ідеї про різні заходи часто зберігаються в різних місцях і не обробляються разом. По-третє, недостатні знання різних користувачів про принципи сталого розвитку не дають можливість підтримувати мотивацію до участі в таких ініціативах. Нарешті, брак автоматизованих інструментів для прогнозування та моделювання сценаріїв розвитку обмежує здатність університетів адаптуватися до глобальних змін.

Розроблена система у вигляді веб-застосунку здатна вирішити всі проблеми завдяки застосуванню:

– централізованого збору даних: інтеграція інформації про енергоспоживання, викиди вуглексилого газу , ідеї громад, фінансові показники;

– автоматизованого аналізу: використання алгоритмів для визначення показнику сталого розвитку і виявлення слабких місць;

– прогнозування та моделювання: створення альтернативних шляхів розвитку таких як наприклад, аналіз впливу використання сонячних панелей або оновлених навчальних програм;

– інтерактивної співпраці: надання платформи для того щоб люли могли ділитися ідеями, обговорювати стратегії і спільно приймати рішення.

Розробка такої системи є актуальною і незамінною для суспільства у наші часи стрімкого розвитку інформаційних технологій. Університети не тільки випускають готових до роботи фахівців, але й формують цінності суспільства та впливають на економічний розвиток регіонів і допомогають підтримувати екологічну стабільність. У нас в Україні, вища освіта зазнає реформ, тому впровадження принципів сталого розвитку університетськоб системи освіти може покращити конкурентоспроможність університетів на фоні міжнародних. Автоматизована система, що реалізує аналіз данних стане важливим інструментом для досягнення всіх цілей, відштовхуючись віж сучасних потреб вищої освіти та міжнародних стандартів.

1.2 Огляд та аналіз наявних аналогів і публікацій

Для того щоб оцінити сучасне становище для розробки інформаційно-аналітичних систем у сфері сталого розвитку університетської освіти , потрібно обов’язково провести детальний огляд наявних аналогічних систем і різних наукових публікацій , міжнародних та вітчизняних . Аналіз повинен поєднувати у собі 2 основні напрямки , це програмні рішення, які використовуються для оцінки сталого розвитку, а також теоретичні дослідження, що висвітлюють потреби та виклики у цій сфері.

Перший аналог це - Sustainability Tracking, Assessment & Rating System (STARS) [4].

Sustainability Tracking, Assessment & Rating System (STARS) (див. рис. 1.1), розроблена Асоціацією сприяння сталому розвитку у вищій освіті (AASHE), є

однією з найпоширеніших систем для оцінки сталого розвитку університетів.

STARS використовує комплексний набір методів, які дозволяють університетам оцінювати свої досягнення в чотирьох основних категоріях: освіта та дослідження, операційна діяльність, адміністрування та планування, залучення стейкхолдерів. Наприклад, у категорії "освіта та дослідження" ця система аналізує, скільки курсів містять у собі теми сталого розвитку, а в категорії "операційна діяльність" аналізує енергоспоживання університету, викиди вуглекислого газу та що університет робить з відходами. Університети отримують бали за кожен показник, що формують рейтинг від бронзового до платинового рівня (див. рис. 1.2).

Функціонал STARS базується на онлайн-порталі, де університети вводять дані, генерують звіти та порівнюють свої результати з іншими університетами. Наприклад, університет може побачити скільки вуглекислого газу він викидує в атмосферу, відстежити відсоток перероблених відходів або проаналізувати соціальне рівноправ'я. Звіти STARS є цінними , так як їх університет може використати для свого внутрішнього планування, налагодження зв'язків з партнерами та залучення студентів до зелених ініціатив. Система набула популярності в США, Канаді та Європі завдяки своїй прозорості та широкому охопленню показників. Для прикладу можна пригадати Університет Британської Колумбії , він використовує STARS для відстеження своїх екологічних ініціатив, і через це його енергоспоживання зменшилось приблизно 20% за останні роки.

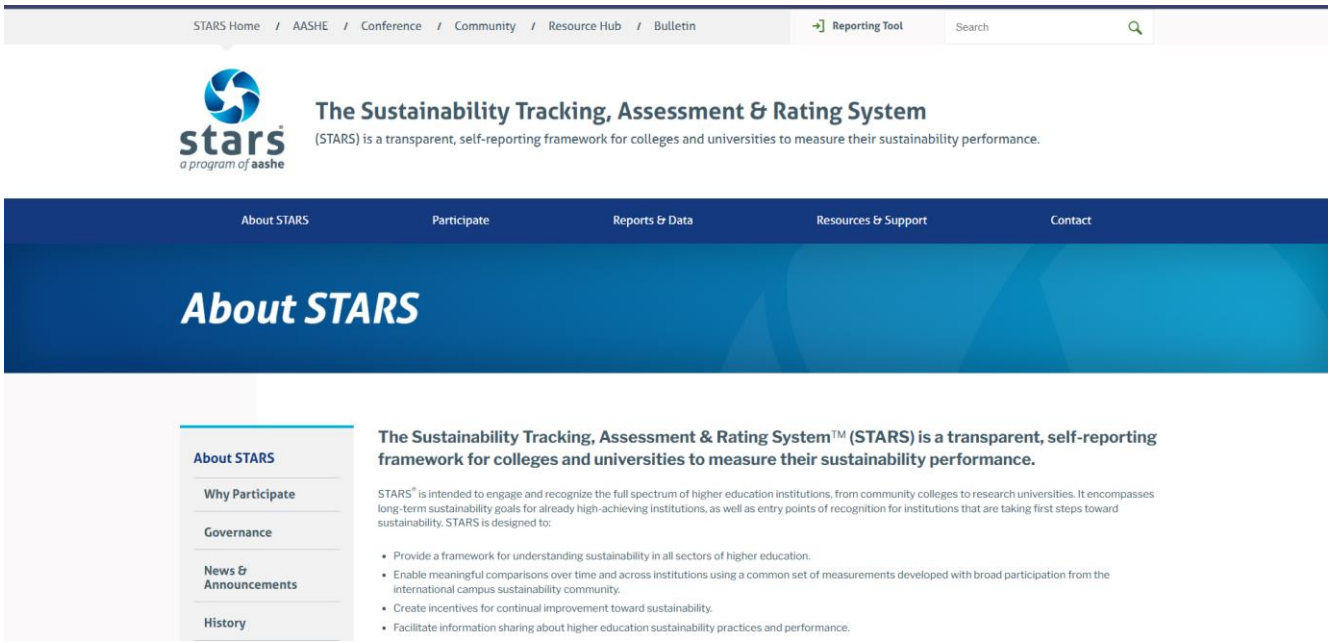


Рисунок 1.1 – Головна сторінка STARS

Рисунок 1.2 – Рейтинг

Незважаючи на свої переваги, STARS має і недоліки , але це скоріш обмеження. По-перше, система STARS не підтримує функціонал який давав би змогу прогнозувати чи моделювати сценарії розвитку, що суттєво зменшує її цінність у нашому питанні. По-друге, введення даних є трудомістким процесом, оскільки всі операції по введеннб даних виконується вручну , мало автоматизації. По-третє, STARS не пропонує інтерактивної платформи для співпраці стейкхолдерів та усіх користувачів системи , наприклад звичайного чату, що ускладнює формування спільного бачення.

Другий розглянутий аналог це - GreenMetric World University Rankings [5].

GreenMetric World University Rankings, створений Університетом Індонезії у 2010 році, є глобальним рейтингом, який оцінює екологічну стійкість університетів (див. рис. 1.3).

Рисунок 1.3 – Головна сторінка GreenMetric World University Rankings

Система строго орієнтована на шість категорій: інфраструктура, енергія та зміна клімату, управління відходами, водокористування, транспорт і освіта (див. рис. 1.4).

Університети завантажують свої дані за допомогоб онлайн-платформи, яка аналізує їх і формує рейтинг на основі цих даних. Це можна пояснити на прикладі категорії "енергія та зміна клімату" , вона оцінює використання саме відновлюваних джерел енергії, тоді як "освіта" аналізує кількість курсів , на яких піднімається питання екологічних проблем. GreenMetric є популярним у країнах Азії, Європи та Латинської Америки, де університети використовують його для підвищення своєї екологічної репутації та можна сказати конкурують один з одним за вище місце в рейтингу.Основна перевага GreenMetric — це простота та інтуїтивність інтерфейсу.

Університети можуть без зайвих проблем швидко вводити свої дані та отримувати результати, що робить цю систему легкою у використанні навіть для закладів із обмеженими ресурсами та фінансуванням. Наприклад Університет Вагенінгена в Нідерландах використовує функціонал системи GreenMetric для оцінки та аналізу своїх зелених ініціатив і саме це відіграло ключову роль у впровадженні системи переробки відходів в кампусі. Рейтинг також підтримує здорову конкуренцію сереж ніверситетів, що мотивує кожного з них до впровадження різних інновацій , таких як встановлення сонячних панелей чи створення велосипедної інфраструктури для того щоб виділятися на фоні інших.

Проте GreenMetric має суттєві недоліки. Система сфокусована тільки на екологічних аспектах сталого розвитку, ігноруючи соціальні та економічні аспекти сталого розвитку. Вона також як і перша розглянута система не підтримує прогнозування чи моделювання, що обмежує її використання. Також GreenMetric не забезпечує стейкхолдерів та користувачів функціоналом для взаємодії, що є ключовою особливістю розробленої системи. Розроблений веб-застосунок більш гнучкий через те що його функціонал для аналітики не зусереджений тільки на екологічних аспектах , а є більш розширеним , маючи змогу аналізувати соціальні і економічні показники разом з екологічними.

Третій аналог який був розглянутий це - Times Higher Education (THE) Impact Rankings [6].

Times Higher Education (THE) Impact Rankings відомий як глобальний рейтинг, який оцінює внесок університетів для досягнення 17 Цілей сталого розвитку ООН (див. рис. 1.5).

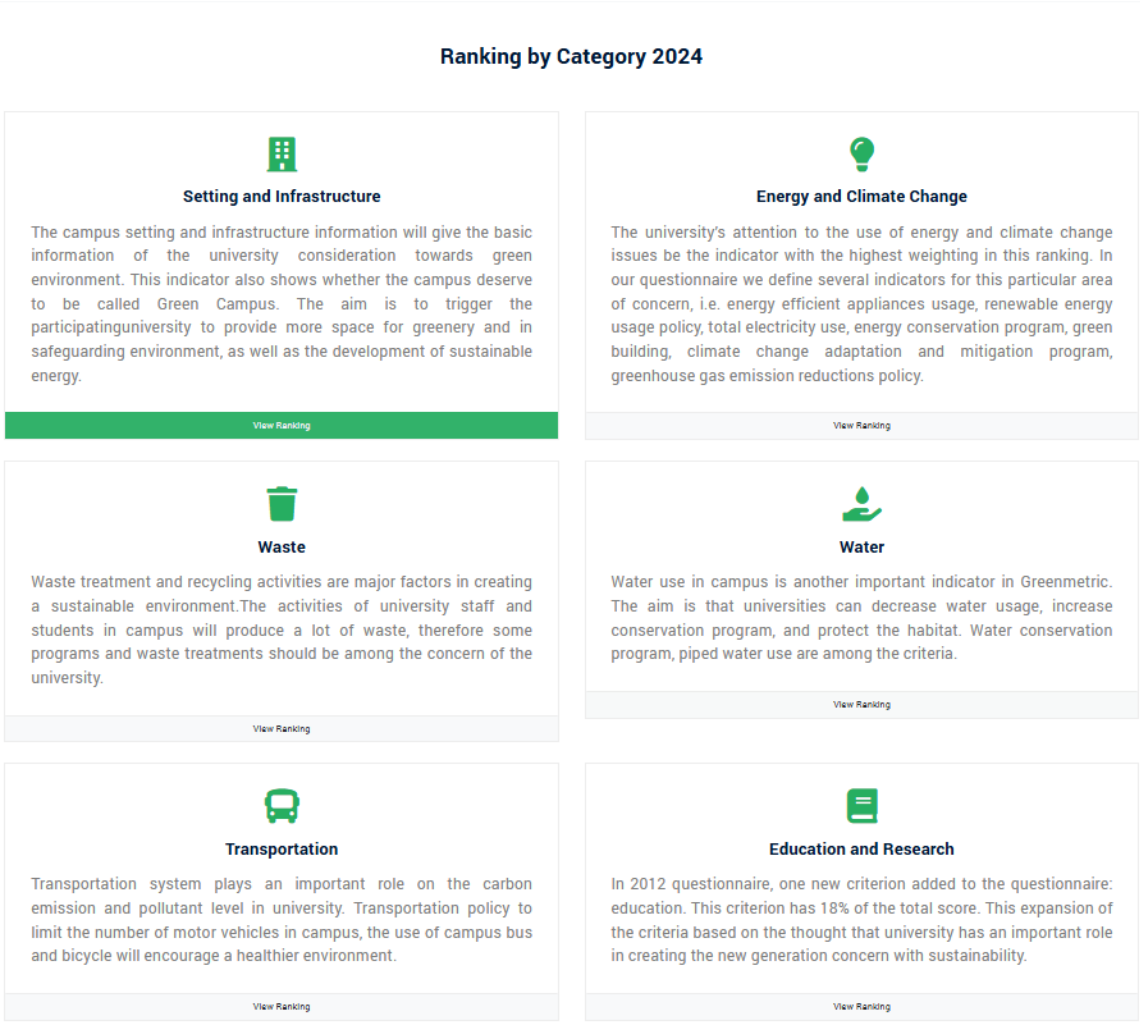


Рисунок 1.4 – Категорії GreenMetric

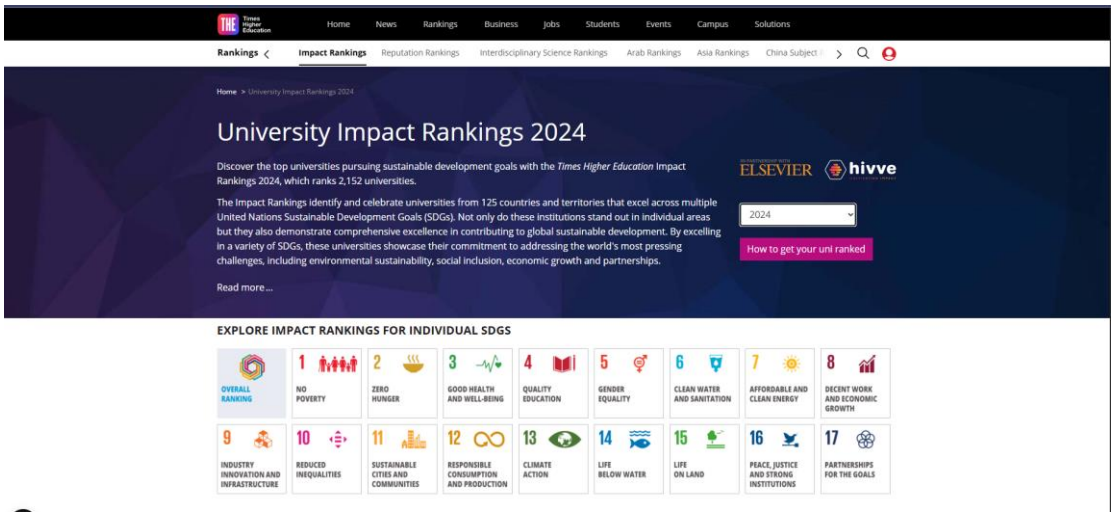


Рисунок 1.5 – Головна сторінка imes Higher Education (THE) Impact Rankings

Цей рейтинг працює з широким переліком показників, включаючи якісну освіту, гендерну рівність, кліматичні дії, партнерство для досягнення цілей тощо. Університети подають свої дані у цей рейтинг і потім вже ці дані аналізуються експертами THE, після чого автоматично створюється рейтинг, що відображає їхній вплив на досягнення цілей на глобальному рівні. Наприклад, університет може оцінити свою роботу яка пішла на користь досягнення цілі 4 - якісна освіта завдяки кількіості впроваджених безкоштовних навчальних курсів.

THE Impact Rankings виділяється на фоні інших своїм комплексним підходом, оскільки працює з екологічними, соціальними та економічними аспектами сталого розвитку. Рейтинг є важливим інструментом який можна використати для підвищення репутації університетів і залучення партнерів , як міжнародних так і вітчизняних. Університет Окленда в Новій Зеландії використовує THE Impact Rankings для того щоб більше зацікавлених людей у сталому розвитку побачили їх ініціативи у сфері інклюзивної освіти, і це спрацювало так як це спричинило залучення іноземних студентів. Дані, зібрані в рамках рейтингу, також можуть використовуватися для внутрішнього аналізу та планування, що робить систему універсальною.

Використовуючи THE Impact Rankings можна дослідити рейтинги університетів по окремому критерію який вас цікавить на данний момент (див. рис. 1.6). А їх тут використовується цілих 17 різних критеріїв , також можна дивитись і загальні рейтинги по декількома критеріями і так далі.

Однак THE Impact Rankings не є повноцінною інформаційно-аналітичною системою. Вона зосереджена вцілому на порівнянні університетів , що сприяєї конкуренції між ними , але не надає інструментів для автоматизованого збору чи аналізу даних. Крім того, система також не має функціоналу для інтерактивної взаємодії між користувачами та стейкхолдерами, що обмежує її можливості у формуванні спільного бачення пріоритетів сталого розвитку. Розроблена система у цьому контектсті перевершує THE Impact Rankings завдяки інтеграції

автоматизованого аналізу, прогнозування та інтерактивної платформи для співпраці та взаємодії.

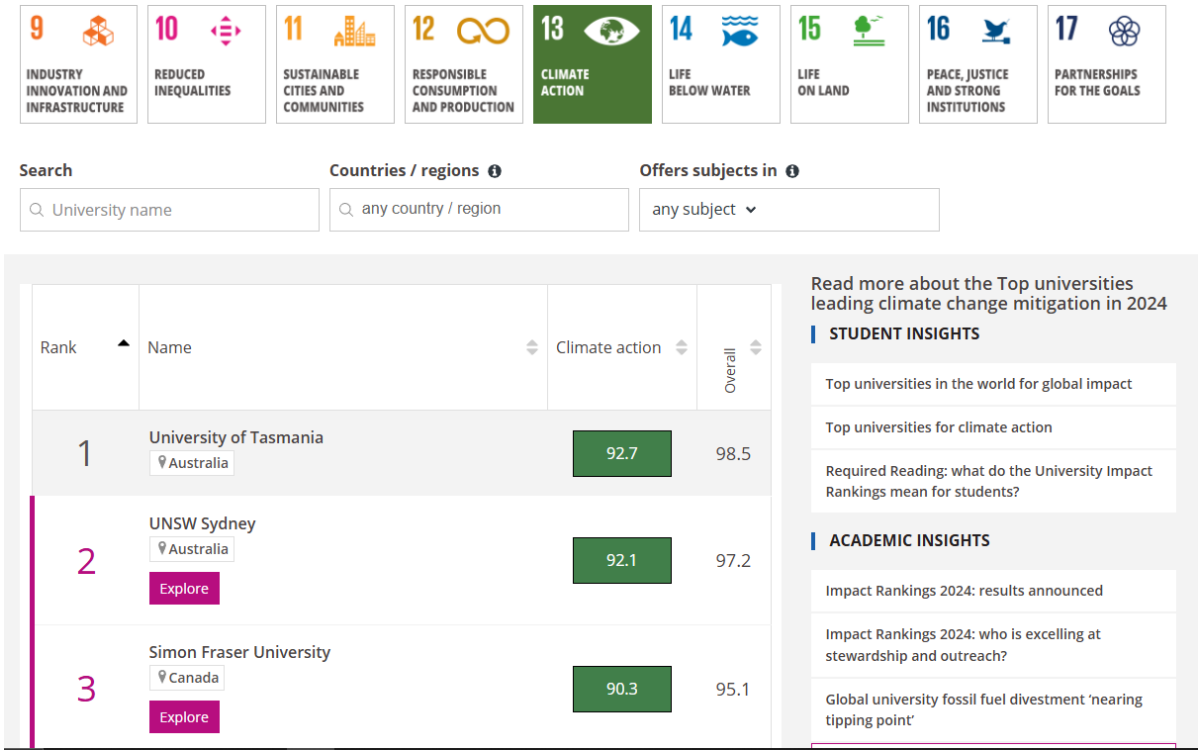


Рисунок 1.6 – Рейтинг за 13 критерієм

Проаналізуємо публікації , дослідження Leal Filho et al. у статті Sustainable Development in Higher Education: Challenges and Opportunities[7] наголошують на необхідності автоматизованих інструментів для оцінки прогресу сталого розвитку в університетах та системі освіти вцілому. Автори стверджують, що додавання принципів сталості в освіту, а також дослідження та управління потребує єдиного і злагодженого підходу, який включає аналіз даних і координацію зусиль стейкхолдерів з простими користувачами. Більше всього вони звертають увагу на те, що без автоматизованих систем університети не можуть ефективно оцінювати свої внески у Цілі сталого розвитку, що робить розробку нашої системи критично важливою.

Findler et al. у статті Sustainability Assessment Tools in Higher Education

Institutions[8] розповідають прообмеження існуючих систем, таких як STARS і GreenMetric. Вони критикують їх за дуже поганий рівень автоматизації та відсутність функціоналу для співпраці між стейкхолдерами та користувачами. Автори наполягають на високій необхідності в розробці систем, що поєднують кількісний і якісний аналіз для того щоб більш точні оцінки сталого розвитку стали реальністю.

У документі UNESCO Education for Sustainable Development: A Roadmap[9] зусереджують увагу на ключовій ролі освіти в досягненні цілей сталого розвитку. У дослідженні пишеться про потреби в автоматизованих системах або платформах для відстеження та аналізу внеску університетів у досягнення цілей ООН, зокрема через інтеграцію даних і аналітичний функціонал. Цей підхід підтверджує актуальність розробленої системи , яка поєднує в собі аналіз і функціонал комунікації між користувачами для підтримки сталого розвитку.

Sterling у статті Transformative Learning for a Sustainable Future[10] акцентує на необхідності інтерактивних платформ для залучення студентів і викладачів до формування культури сталого розвитку. Автор зазначає, що цифрові інструменти, які сприяють співпраці та обміну ідеями, є важливими для трансформації освіти, що підтримує ідею створення інтерактивного веб-застосунку в цьому дослідженні.

Це були міжнародні публікації які стосуються теми сталого розвитку всі вони так чи інакше говорять про те що автоматизовані системи або хоча б автоматизовані якісь певні аспекти для сталого розвитку є вкрай необхідними , звісно це все буде марно якщо викладачі , студенти та інші люди не будуть брати у цьому участь , все має бути налогоджено на кожному рівні.

В Україні Коренева І. М. та співавтори у колективній монографії Науковий та педагогічний супровід сталого розвитку[11] досліджують освіту як інструмент сталого розвитку. Вони вказують, що інформаційні системи можуть значно підвищити ефективність відстежування виконання цілей сталого розвитку в школах і університетах. Проте автори підкреслюють проблему браку автоматизації

в Україні, що ускладнює збір і аналіз даних, і пропонують розробляти цифрові платформи для вирішення цих питань, що узгоджується з цілями цього дослідження.

Радкевич В. О. та співавтори (2021) у праці Професійна освіта України в контексті євроінтеграційних процесів[12] активно обговорюють ролі інформаційно-аналітичних систем у покращенні та модернізації освіти. Вони зазначають, що автоматизовані системи які орієнтовані на збір даних про професійну підготовку можуть бути націлені і на відстеження сталого розвитку, через аналіз соціальних і економічних показників які завантажуються закладами освіти. Ця ідея напряму наголошує на розробці універсальної системи, яка охоплює всі аспекти сталого розвитку.

Биков В. Ю. та співавтори (2017) у статті Проблеми та завдання сучасного етапу інформатизації освіти[13] акцентують свою увагу на необхідності розробки єдиного інформаційного простіру в Україні. Вони стверджують , що цифрові та інформаційні засоби, такі як аналітичні системи, є ключем для реформування освіти, особливо якщо говорити про досягнення цілей сталого розвитку. Автори зазначають, що автоматизація збору та аналізу даних може підвищити ефективність управління освітою.

Національна доповідь «Цілі сталого розвитку: Україна» (2017), підготовлена Міністерством освіти і науки[14], наголошує на важливості освіти для досягнення цілей сталого розвитку. Хоча доповідь не бере до уваги саме інформаційні системи, вона розповідає про потребу в інструментах відслітковування прогресу досягнення цілей через автоматизацію.

1.3 Постановка задачі

Університети є основними одиницями сталого розвитку, даючи відпір глобальним викликам, такі як різка зміна клімату, соціальна нерівність і економічна нестабільність. В Україні ці виклики посилюються обмеженим фінансуванням, і

низькою автоматизацією. Розробка інформаційно-аналітичної системи є актуальною, і вона відповідає і працює на досягнення Цілей сталого розвитку ООН зокрема, ціль 4 - якісна освіта та ціль 13 - кліматичні дії і реформам освіти в Україні[15], спрямованим на євроінтеграцію. Завдяки цій системі зросте конкурентоспроможність університетів і їх внесок в національні та глобальні ініціативи сталого розвитку.

Метою роботи є розробка автоматизованої інформаційно-аналітичної системи у вигляді веб-застосунку, який забезпечить формування спільного бачення та визначення альтернативних шляхів переходу до сталого розвитку університетської системи освіти шляхом аналітики даних університетів та взаємодії стейкхолдерів і користувачів. Система має проводити аналіз даних, і забезпечити співпрацю стейкхолдерів і користувачів на комфортному рівні , а головне щоб це було максимально інформативно для створення стратегій, що поєднують екологічну стійкість, соціальну відповідальність і економічну стабільність.

Об’єктом дослідження є процеси для керування та аналізу даних у контексті сталого розвитку університетської системи освіти. Ці всі процеси охоплюють не малий перелік діяльності університетів, включаючи відстежування екологічних показників , таких як енергоспоживання, викиди вуглексислого газу та управління відходами, аналіз соціальних ініціатив та ідей , а саме інклюзивність, доступність освіти, співпраця з людьми та аналіз економічних аспектів , наприклад фінансові потоки, залучення грантів, оптимізація ресурсів. В Україні це все часто дуже складно реалізується через розподіл даних між різними частинами університету, також у нас немає єдиного стандарту за допомогою якого ми б могли збирати всю необхідну інформацію і вцілому зараз для цього всього ми маємо дуже низький рівень автоматизації [16]. Об’єкт роботи також містить взаємодію стейкхолдерів та звичайних користувачів для об’єднання зусиль формування стратегій сталого розвитку. Таким чином, дослідження фокусується на вдосконаленні цих процесів

шляхом створення системи, яка забезпечить інтеграцію і аналіз даних , а також взаємодію всіх користувачів між собою.

Предметом дослідження є автоматизована інформаційно-аналітична система, реалізована у вигляді веб-застосунку, призначена для збору, обробки і аналізу даних та надавання функціоналу для взаємодії користувачів для підтримки сталого розвитку університетської системи освіти. Система включає модулі для введення даних, їх автоматизованого аналізу і візуалізації результатів, а також інтерактивну платформу яка складаєть з декількох частин , кожна з яких виконує свою функцію для співпраці стейкхолдерів та користувачів. Вона має забезпечити централізоване управління інформацією про екологічні, соціальні та економічні показники, дозволяючи оцінювати поточний стан університету та шляхом обговорення або опитувань виявляти пріоритети сталого розвитку.

Унікальність предмета полягає в інтеграції кількісного та якісного аналізу, а також у створенні інтерактивного середовища, яке сприятиме обміну ідеями та спільному прийняттю рішень.

Завдання:

- провести аналіз вимог стейкхолдерів та простих користувачів;
- розробити модель даних для показників сталості;
- спроектувати архітектуру веб-застосунку;
- реалізувати інтерактивний інтерфейс;
- провести тестування на реальних даних.

Висновки до 1 розділу

Аналіз предметної сфери показав, що сталий розвиток є стратегічно важливим напрямком для університетів та системи освіти вцілому, особливо в наш час коли відбуваються глобальні зміни , такі як зміна клімату, соціальна нерівність і економічна нестабільність. Огляд аналогічних систем таких як STARS, GreenMetric, THE Impact Rankings виявив їхні обмеження в автоматизації та інтерактивності, тоді як міжнародні та українські публікації підтверджують необхідність в негайній розробці інформаційно-аналітичних систем, які мають у своєму функціоналі якісний аналіз, підтримують співпрацю стейкхолдерів і користувачів. Зокрема, українські дослідження підкреслюють необхідність цифрових платформ для моніторингу Цілей сталого розвитку в освіті, що відповідає національним реформам і процесам.

2 ДОСЛІДЖЕННЯ ТА ВИБІР МЕТОДІВ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ РЕАЛІЗАЦІЇ СИСТЕМИ

2.1 Методи для вирішення поставленої задачі

Для реалізації автоматизованої інформаційно-налітичної системи, спрямованої на визначення пріоритетів сталого розвитку університетської системи освіти, необхідно використати різні методи, які забезпечать збір, обробку та звичайно ж аналіз даних. У моїй системі , методи головним чином сфокусовані на аналітичних підходах для розрахунку єдиного показника сталого розвитку, без використання надто складних для цього технологій таких як машинне навчання, але залишаючи можливість їхньої інтеграції в майбутньому. Методи поділяються на аналітичні , вони використовуються для обробки даних та їх оцінки для сталого розвитку та організаційні , вони вже використовуються для співпраці стейкхолдерів з користувачами і управління проєктом.

Аналітичні методи використовуються для виконання основного завдання системи , а саме оцінювання показника сталого розвитку університету за екологічними, соціальними та економічними аспектами з подальшим формуванням єдиної оцінки. Метод агрегації даних для цього ідеально підходить [17], він включає базові статистичні операції, такі як обчислення суми та середнього значення. Застосовуючи його система може на прикладі екологічного показника порахуватим загальне енергоспоживання кампусу за місяць , рік або середній рівень переробки відходів. Ці операції забезпечують узагальнення даних, необхідне для подальшого аналізу.

Головним методом для аналітичного модуля системи є розрахунок композитного індексу, який дозволяє системі об’єднувати різнорідні показники в єдину оцінку сталого розвитку. Процес включає три етапи:

– нормалізація даних: показники з різними одиницями виміру такі як наприклад енергоспоживання , яке вимірюється у кВт/год та відсоток перероблених відходів приводяться до спільної шкали (0–100%) шляхом

перетворення фактичних значень у бали. Це усуває вплив різних одиниць виміру[18];

– зважування показників: кожен показник на етапі розробки отримав свій ваговий коефіцієнт, що відображає його важливість самого по собі та на фоні інших показників. В розробленій системі ваги можуть бути призначені двома шляхами , експертним шляхом , наприклад через опитування стейкхолдерів або за стандартами, такими як Цілі сталого розвитку ООН. Щоб пояснити як це працює то наприклад якщо в якомусь університеті високі викиди вуглексислого газу то екологічні показники можуть мати вищу вагу;

– об’єднання в композитний індекс: нормалізовані та зважені показники підсумовуються за формулою зваженої суми, формуючи єдину оцінку.

Використання такого методу гарантує об’єктивну оцінку сталого розвитку університету і вже потім ця оцінка може бути використана адміністрацією разом з стейкхолдерами для визначення пріоритетів розвитку.У майбутньому систему можна розширити методами аналізу ієрархій (AHP) для складнішого ранжування чи машинним навчанням для прогнозування трендів.

2.2 Технології розробки системи

Для реалізації запланованої системи у вигляді веб-застосунку нам знадобляться сучасні технології, які будуть гарантувати швидкість розробки, надійність та найголовніш зручність використання розробленою системою кінцевими користувачами. План реалізації поділений на кілька коючових компонентів , серед них є фронтенд, бекенд, база даних і принципи розгортання.

Як ми всі знаємо за створення зручного та адаптивного інтерфейсу , відповідає фронтенд , і у розробленій системі він буде використаний для реалізакації інтерфейсу для різних задач , таких як введення даних, перегляд звітів і співпраці стейкхолдерів. Для цього використовуються стандартні веб-технології: HTML для структури сторінок, CSS - для стилізації та JavaScript для інтерактивності. Щоб прискорити розробку та забезпечити сучасний дизайн,

застосовується Bootstrap - CSS фреймворк[19], його дуже зручно використовувти через заздалегідь готові компоненти , такі як кнопки, форми, таблиці , а також він бонусом підтримує адаптивність для різних пристроїв - комп’ютери, планшети, смартфони.

Інтерфейс системи складатиметься з різних компонентів , які будуть відповідати за свої індивідуальні завдання , так наприклад форми будуть використовуватися для введення даних, дашборди для відображення композитного індексу та графіків оцінки сталого розвитку університетів впродовж місяців або років, також буде розділ для співпраці стейкхолдерів та користувачів. Як вже зазначалось Bootstrap ідеально підходить для створення таких елементів, а JavaScript відповідає за динамічні функції, які будуть використовуватися для валідації введених даних чи оновлення графіків. Звісно ж для повного розуміння результатів аналізу нам потрібні графіки , вони в системі знаходяться в дашбордах але щоб вони працювали потрібна бібліотека Chart.js, яка інтегрується з Bootstrap і підтримує створення інтерактивних графіків[20].

Тепер поговоримо про бекенд системи , він буде використаний для обробки даних, виконання аналітичних розрахунків і управління користувачами. Для його реалізаціїї відмінно підходить мова програмування Python із веб-фреймворком Django[21]. Python[22], є потужною мовою з розвиненою екосистемою бібліотек для обробки даних , що допомагає в написанні складних функцій , так як їх не потрібно самому писати з повного нуля , достатньо просто використати відповідну бібліотеку і все, а Django прискорює розробку завдяки вбудованим рішенням, таким як ORM або по іншому - об’єктно-реляційне відображення , для роботи з базою даних і Admin-панеллю для управління даними або більш простимим словами всіма данними в системі можна керувати з однієї панелі , видаляти , додавати і для цього не потрібно беспосередньо бути у системі.

Для реалізації аналітичної логіки , яка в нас складається з агрегації, нормалізації, зважування та композитного індексу , доцільно буде використати

бібліотеку Pandas[23]. Чому саме Pandas , за допомогою цієї бібліотеки можна без проблем нормалізувати дані про енергоспоживання та дані кількість заходів які присвячуються сталому розвитку в єдині бали від до 0 до 100 і потім розрахувати зважену суму для індексу. Для того щоб у розробленій системі можна було логінитись користувачам , та кожному типу користувачу був доступен певний функціонал використовується Django Authentication , він забезпечує аутентифікацію користувачів із різними ролями таких як адміністрація для редагування та наприклад викладачі/студенти для перегляду.

Не одна система не може нормально функціонувати без бази даних , для цієї системи була обрана реляційна база даних PostgreSQL[24] , вона використовується разом із адаптером psycopg2 для злагодженої роботи з фреймворком Django. PostgreSQL вирізняється високою надійністю, підтримкою складних запитів, а також відкритою ліцензією, що знижує витрати.

Для оптимізації запитів використовуватимуться індекси, а для безпеки - шифрування паролів і захист з’єднання через SSL. Ще однією важливою ознакою PostgreSQL є здатність працювати з великим обсягом даних, що важливо для подальшого масштабування системи в майбутньому з додаванням все більш гнучкого функціоналу.

Про візуалізацію результатів аналізу на фронтенді вже було сказано , тепер розглянемо візуалізацію результатів на бекенді. Для неї у розробленій системі використовуються дві відомі бібліотеки - Matplotlib[25] і Seaborn[26], які генеруватимуть графіки , а потім вже ці графіки будуть прийматись на фронтенд завдяки розглянутому вище Chart.js.

Введення даних для аналізу здійснюється на пів автоматично , вручну адміністрації потрібно лише обрати файл з даними , зазвичай у подібних системах використовують CSV-файли , наша система не виняток і коли потрібний файл завантажили у систему , вона автоматично його считує та проводить одразу аналітичні розрахунки. Користувачі , які увійшли до системи під своїм логіном і

паролем , і права співробітника , які видаються у панелі адміністрації можуть завантажувати файли з даними за допомогою окремої сторінки на якій є форма створена за допомогою Django і Bootstrap.

Система розгортатиметься в локальному середовищі на моєму персональному комп'ютері з використанням вбудованого сервера Django і PostgreSQL. Локальне середовище включає Python, Django, PostgreSQL і стандартний веб-браузер для доступу до системи.

В майбутньому розроблена система може бути розгорнута на сервері , нехай для прикладу розглянемо сервер з операційною системою Ubuntu, для цього знадобиться:

- веб-сервер: Nginx або Apache для обробки HTTP-запитів і передачі їх до Django;
- сервер додатків: Gunicorn або Waitress для виконання Python-коду Django;
- СУБД: PostgreSQL для зберігання даних.

Nginx забезпечує високу продуктивність і підтримку HTTPS, а Gunicorn - стабільну роботу Django. Для спрощення розгортання може використовуватися Docker, який пакує систему яка складається з Django, PostgreSQL та Nginx у контейнери, забезпечуючи переносимість.

Висновки до 2 розділу

Для реалізації веб-застосунку використано Python із фреймворком Django для створення бекенду, що обробляє завантажені на пів автоматично дані , далі вже автоматично виконує аналітичні розрахунки такі як агрегація, нормалізація, зважування. PostgreSQL із адаптером psycopg2 використано для надійного зберігання структурованих даних. Фронтенд який реалізовано за допомогою HTML, CSS, JavaScript і Bootstrap дає в результаті лаконічний інтерфейс для завантаження даних і відображення результатів аналізу ,а Chart.js генерує інтерактивні графіки оцінки сталого розвитку. Ці методи й технології дозволяють оцінювати сталий розвиток і формувати спільне бачення для університетів.

3 РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ ТА АНАЛІЗ ОТРИМАНИХ РЕЗУЛЬТАТІВ

3.1 Опис вхідних даних та структури системи

Функціонування та безперебійна робта всх інформаційних систем залежить від ефективності та швидкості опрацювання інформаційних ресурсів, вхідні дані у цьому питанні є ключовою одиницею. Для того щоб система виконувалв усі поставлені перед нею завдання з аналізу сталого розвитку університету, у ній налогоджена робота одразу з двома типами вхідних даних, які формують її інформаційну базу та забезпечують подальші обчислення та візуалізацію.

Основним типом вхідних даних є дані для аналітики сталого розвитку університету. Ці дані представляють собою кількісні або якісні виміри певних аспектів сталості університету за певний період часу. Кожен запис про показник містить наступну ключову інформацію:

– Назви 8 показників: "енергоспоживання", "площа зелених насаджень", "кількість утворених відходів";

– фактичне значення: числове значення виміру за певний період;

– дата періоду: дата, що фіксує період, за який внесено значення наприклад місяць або рік;

– користувач, який вніс дані: показує хто несе відповідальність за внесені дані.

Ці дані завантажуються у систему шляхом імпорту спеціальних файлів з зовнішніх джерел , таких як звичайний ПК на якому цей файл був створений. У системі за замовчуванням використовуються файли формату CSV[27]. Для кращого розуміння як це все налаштовано , приблизний вигляд вхідних даних які завантажуються у систему показаний у вигляді таблиці, де кожен запис відповідає за дані які завантажені за певний період (таблиця 3.1).

Таблиця 3.1 – Приклад завантажених даних

Рік	Місяць	8 Показників(для кожного окремий стовпчик зі значенням)	Користувач
2025	Січень	X	admin
2025	Лютий	X	співробітник 1
2025	Березень	X	студент 1

Наступними важливими данними для сисеми є дані які містять в собі інформацію про користувачів які є або будуть в майбутньому зареєстрован у цій системі. Цей тин данних використовуються для ідентифікації усіх користувачів системи , адміністрації , студентів , викладачів і так далі. Щоб було видно що зробив той чи інший користувач , як у будь-якій соціальній мережі або наприклад месенджері. Кожен користувач в системі описується такими атрибутами:

– логін (username): унікальне ім'я користувача для входу в систему;

– пароль: секретний набір символів для автентифікації;

– email: адреса електронної пошти користувача;

– тип користувача: категорія, що визначає роль користувача в університеті , доступні такі як студент, викладач, співробітник, адміністрація;

– фото профілю: зображення, що дозволяє легко ідентифікувати користувача;

– опис профілю: додаткова інформація про користувача яку він може за бажанням додати про себе.

Ці дані з’являються у системі коли нові користувачі проходять процедуру реєстрації в системі або якщо ці дані заздалегідь підготувала адміністрація системи за допомогою пенелі адміністрування (рис.3.1).

Управління профілями користувачів дає змогу адаптувати взаємодію з системою під кожного окремого користувача та відстежувати їхні дії. Це дозволяє розуміти хто саме вніс певні дані або хто є автором певної ідеї чи коментаря в системі.

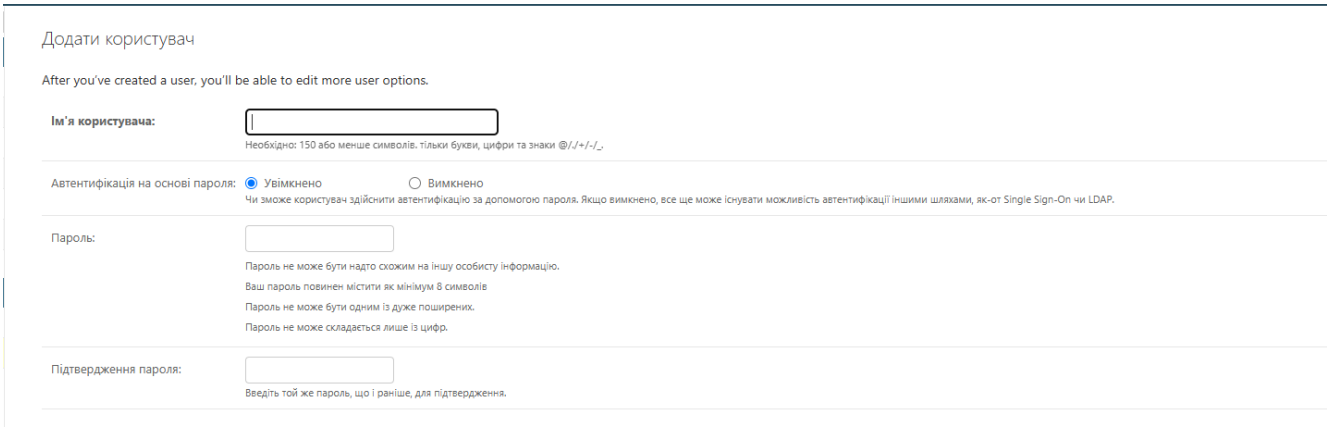


Рисунок 3.1 – Додавання нового користувача адміністрацією

Кожна інформаційна система маю свою унікальну труктуру яка дає зрозуміти як налаштовані та організовані всі компоненти та як вони взаємодіють між собою для виконання всіх поствлених завдань , таких як обробка вхідних даних та навпаки генерація вихідної інформації яка є доступною в цій системі , а саме аналітика, списки запропонованих ідей від користувачів тощо. Розроблена система побудована на основі архітектурного патерну Model-View-Template (MVT), який є базовим так як для розробки був обраний веб-фреймворк Django. Цей патерн забезпечує чіткий та інтуітивний розподіл відповідальностей між різними частинами застосунку, що робить його структурованим , зрозумілим та легким для масштабування та підтримки.

Ось основні 3 компонента цього паттерну:

– models;
– view;
– templates.

Перші , а саме моделі представляють структуру усіх даних які так чи інакше

використовуються у системі та описують , як ці дані зберігаються, управляються та взаємодіють з базою даних системи. Кожна модель являє собою таблицю в реляційній базі даних у моєму випадку це PostgreSQL, а поля моделі відповідають стовпцям таблиці.

У моделях описана вся логіка яка пов’язана з даними , також завдяки ним стає можливим визначити інтерфейс запитів та різних дій з даними за допомогою відомого вже нам об'єктно-реляційного відображення (ORM) Django, через це відпадає необхідність розробнику власноруч писати SQL-запити. У системі визначені усі необхдіні моделі для реалізації усього функціоналу пов’язаного зі сталим розвитком , а саме моделі записів даних - DataEntry, користувачів - User - стандартна модель Django, профілів користувачів - Profile, ідей – Idea та інші.

Наступний компонент нашого паттерну це - view , його можна охарактеризувати наступним чином - набір Python-функцій, які отримують запити від користувача потім обробляють їх , паралельно взаємодіючи з моделями для отримання або збереження даних. Також вони використовуються для роботи з формами для валідації вхідних даних від користувача, і, врешті-решт, вирішують, яку відповідь відправити користувачу.

За замовчуванням відповіддю цих функцій є рендеринг певного HTML шаблону, який пов’язаний саме з конкретною функцією але це може бути і перенаправлення на іншу сторінку , наприклад зі сторінки з ідеями на сторінку де детально описується конкретно обрана ідея. View містять основну бізнес-логіку додатку. Наприклад, view для введення даних обробляє POST-запит з форми, валідує дані, створює об'єкт моделі DataEntry та зберігає його. View для аналітики отримує дані з моделі DataEntry, виконує розрахунки та передає результати в шаблон.

І нарешті шаблони - це текстові файли , написані за домогою HTML, які визначають структуру та вигляд інтерфейсу системи для цільового користувача. Шаблони містять статичний HTML-код , котрий вже у собі містить спеціальні теги

та фільтри мови шаблонів Django, які критично необхідні для того щоб ми могли отримувати динамічні дані з View та мати можливість реалізовувати різні сценарії користування системою або прості операції такі як цикли та умови, включати інші шаблони такі як базовий шаблон base.html у всі інші шаблони системи , такий підхід є дуже зручним для дотримання стандартної структури сторінки всюди.

Шаблони відповідають за представлення інформації користувачу. На данний момент в системі всього є 15 фашбонів , написаних на HTML (рис.3.2).

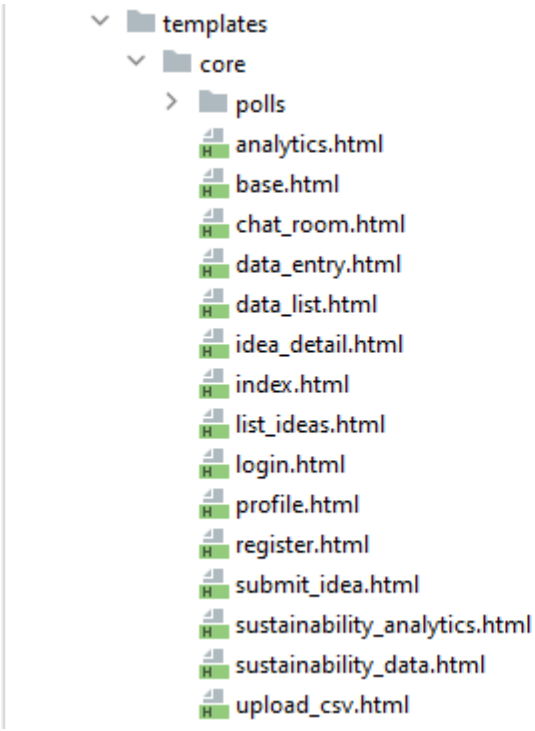


Рисунок 3.2 – Шаблони системи

3.2 Проєктування системи

Проектування інформаційної системи яка орієнтована на сталий розвиток базується на трирівневій архітектурі "клієнт-сервер", завдяки чому стає реальним чітке та інтуітивно зрозуміле розділення відповідальностей між всіма компонентами та бонусом через це ми отримаємо високий рівень гнучкості нашої розробленої системи.

Ця архітектура як можна зрозуміти з назви складається з 3 рівнів , а саме рівня презентації , по іншому відомий як клієнтська частина, рівня застосунку - серверна логіка та рівня даних - безпосередньо наша база даних. Рівень презентації використовується для того щоб користувачі взаємодіяли з системою через реалізований веб-інтерфейс, розроблений за допомогою стандартних веб-технологій таких як HTML, CSS та JavaScript. Цей підхід забезпечує кросплатформність та доступність системи з будь-якого пристрою, на якому всановлений будь-який сучасний веб-браузер.

Вибір технологічного стеку обґрунтований потребою в ефективній розробці, надійності та можливості подальшого розширення системи і саме через це в якості основного фреймворку для розробки серверної частини , як вже було сказано було обрано фреймворк Django , а для зберігання та керування даними обрана реляційна система управління базами даних PostgreSQL.

Як ця трирівнева архітектура працює можна побачити на (рис.3.3).

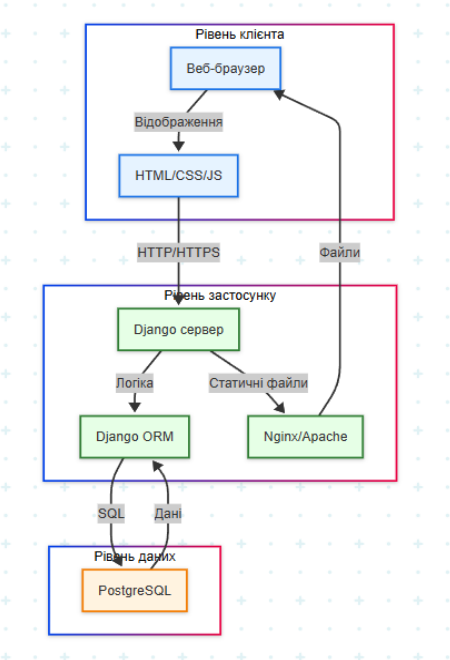


Рисунок 3.3 – Трирівнева архітектура

Проектування інформаційної системи для визначення пріоритетів сталого розвитку включало детальний аналіз вимог та розробку структур даних, що

забезпечують ефективне зберігання, обробку та доступ до інформації. Ключовим етапом стало моделювання бази даних, яка слугує основним сховищем всієї інформації системи. Реляційна модель бази даних виділяється на фоні інших завдяки можливості надати цілістний, та ефективний доступу до даних, що є критично важливим для розробленої системи. База даних була розроблена таким чином, щоб можна було показати всі сутності та те як вони взаємозв'язані. Основні сутності та їхні атрибути, а також зв'язки між ними, описані нижче.

Центральною сутністю системи є User, що є стандартною моделлю користувача використаного веб-фреймворку Django. Вона містить базові дані для автентифікації, такі як ім'я користувача, хешований пароль та електронна пошта. Для того щоб ця модель відповідала усім вимогам для нашої системи до неї було додано розширення у вигляді сутності Profile. Модель Profile створює зв'язок який називають один-до-одного - One-to-One з моделью User та зберігає додаткову інформацію про користувача, що не входить до стандартної моделі , наприклад тип користувача:

- студент;
- викладач;
- співробітник;
- адміністрація.

Також до цієї інформаціїї налажить фото профілю яке є у кожного користувача , а якщо його не додали то використовується фото за замовчуванням також короткий опис належить до цієї інормації. Таке розділення дозволяє гнучко розширювати атрибути користувача без модифікації базової таблиці User.

Для зберігання та використання в подальшому введених даних які містять у собі інформацію щодо сталого розвитку, спроектовано сутність DataEntry. Ця модель є основою для аналітичної частини системи та містить числові показники за певні періоди. Головними атрибутами цієї моделі є year та month де атрибут month може бути NULL , тобто порожнім , це використовується тоді коли

вводяться річні дані, що дозволяє зберігати як річні, так і місячні дані в одній таблиці. Також в цій моделі є ті самі 8 показників сталого розвитку:

- energospozhyvannya_zagalne;
- vodospozhyvannya_zagalne;
- utvorennya_vidhodiv_zagalne;
- vykydy_co2_pryami;
- ploscha_zelenyh_nasadzhen;
- vidsotok_vidsortovanyh_vidhodiv;
- kilkist_zahodiv_stalyi_rozvytok;
- vytraty_energoefektyvnosti.

Кожен запис DataEntry пов'язаний із сутністю користувача – User способом який називається - багато-до-одного або його ще називають Many-to-One через поле uploaded_by, що вказує на те , який користувач завантажив дані. Це забезпечує відслідковування джерел даних у системі. Додатково, DataEntry містить uploaded_at для зберігання часу завантаження. Для забезпечення унікальності записів застосовано обмеження unique_together на поля year та month, що запобігає дублюванню даних за один і той же період.

Модуль інтерактивної взаємодії між стейкхолдерами та користувачами розроблений із використанням кількох основних сутностей. Наприклад для групового чату розроблено модель Message. Вона містить такі поля:

- sender;
- content;
- created_at;
- media_file;
- file_type.

Зв'язок багато-до-одного з User дозволяє одному користувачу відправляти безліч повідомлень.

Система також підтримує функціонал управління ідеями та опитуваннями. Сутність Idea призначена для зберігання користувацьких пропозицій та ініціатив. Її атрибути включають:

- title;
- description;
- author;
- created_at;
- likes_count.

Завдяки використанню зв'язку багато-до-одного з моделью User кожен користувач системи може запропонувати багато ідей та ініціатив. Для взаємодії з ідеями існують дві додаткові сутності: Comment та Like. Завдяки сутності Comment користувачі можуть коментувати ідеї та пропозиції. Like є допоміжною сутністю, яка реалізує функціонал коли якась ідея сподобалась користувачу і він хоче її відмітити лайком. Ця сутність також додатково містить ForeignKey до User та Idea, тобто багато користувачів можуть ставити лайк і лайків на одній ідеї може бути багато від різних користувачів. Також до цих двох полів застосоване обмеження unique_together і через це система може гарантувати , що один користувач може лайкнути ідею або аніціативу тільки 1 раз , не можна залишити багато лайків одній ідеї від одного користувача.

Модуль опитувань реалізовано за допомогою трьох моделей. Головна сутність Poll описує саме опитування і має наступні поля:

- title;
- description;
- pub_date (дата публікації);
- created_by;
- is_active (для контролю активності опитування).

До Poll також додано поле для прикріплення картинок - image. У системі як і будь-де кожне опитування має свої варіанти вибору , їх може бути різна кількість , вони реалізовані в системі за допомогою Choice, яка пов'язана зв’язком багато-до-одного з Poll та містить choice_text.

Нарешті, ми підійшли до останньої сутності яка використовується для інтерактивної взаємодії стейкхолдерів і користувачів і це - сутність Vote , вона використовується для фіксації голосів користувачів. Вона має зв'язок ForeignKey з User та Choice, що дає змогу відстежувати, хто з користувачів віддав свій голос за конкретний варіант. Хоча безпосередньо в моделі Vote атрибут unique_together не налаштовано для обмежень, але логіка яка вбудована у views.py, гарантує, що один користувач може взяти участь в опитуванні лише один раз. Діаграму класів з описом усіх цих класів можна побачити на відповідному рисунку (Додаток А).

Поговоримо більш детально про кожен модуль інтерактивної взаємодії окремо і почнемо з модуля так званого чату в нашій системі.

Груповий чат було реалізовано за допомогою функціоналу обміну текстовими повідомленнями та медіа-файлами у форматі стіни спілкування. Користувачі можуть надсилати текстові повідомлення - Message.content або прикріплювати файли - Message.media_file, використовуючи для цього спеціальну форму - MessageForm. Логіка валідації форми забезпечує, що користувач не надсилає пустий вміст , а надсилає або текст, або файл або одразу і те і те. Повідомлення відображаються у хронологічному порядку біля повідомлень зазначається автор , а саме його фото профілю та ім’я ,також зазначається час надсилання повідомлення. Зберігання медіа-файлів реалізовано з використанням налаштувань MEDIA_ROOT та MEDIA_URL Django, що дозволяє безпечно завантажувати та обслуговувати файли.

Ідеї/Ініціативи: цей функціональний модуль надає всім користувачам системи розповісти про власні пропозиції та ініціативи щодо сталого розвитку їх університету або системи освіти вцілому. Користувачі можуть створювати нові ідеї через IdeaForm, заповнюючи відповідні поля title та description. Система автоматично приписує ідею поточному авторизованому користувачу за допогою request.user , тобто який користувач увійшов у систему , значить такий і додав , все дуже просто.

Перегляд ідей реалізовано у вигляді списку, який може бути сортований наприклад, за датою створення або за кількістю лайків. Для кожної ідеї в системі задумана окрема сторінка з додатковими деталями , яка носить назву - idea_detail_view, у файлі views.py де відображається повний опис, а також реалізовано механізм додавання коментарів. Функція лайків - like_idea_view дозволяє користувачам висловлювати свою підтримку ідеям, при цьому гарантується, що кожен користувач може поставити лайк одній ідеї лише один раз , про це вже згадувалось , це завдяки моделі Like та обмеженню unique_together. Коментарі додаються через CommentForm і автоматично пов'язуються з конкретною ідеєю та автором.

Опитування: цей модуль дозволяє адміністраторам або уповноваженим користувачам створювати опитування, збирати голоси та відображати результати, щоб таким стати , адміністратори у панелі адміністрування мають дати необхідні привілегії.

Створення опитування здійснюється за допомогою PollForm , яка дозволяє задати заголовок , опис та якщо потрібно , зображення. Після створення опитування, до нього можна додавати варіанти відповідей – ті самі Choice. Користувачі можуть переглядати список активних опитувань, а на сторінці деталей опитування голосувати за обраний варіант. Система забезпечує, що один користувач може проголосувати в одному опитуванні лише один раз. Результати опитувань відображаються користувачу одразу після голосування, надаючи швидкий огляд думок інших користувачів.

Саме проектування інтерфейсу користувача було без зайвої роботи , це з самого початку уявлялось як простий але сучасний і інтуітивно зрозумілий інтерфйес для користувачів від початкового до просунутого рівня володіння персональним комп’ютером.

Також не слід забувати про адаптивність до різних пристроїв. Веб-інтерфейс розроблено з використанням стандартних HTML-шаблонів, з простим але

сучасним і зрозумілим для сприйняття CSS дизайном та JavaScript для інтерактивності.

Головним елементом інтерфейсу системи є навігаційна панель , через яку можна потрапляти на всі сторінки системи:

- дашборд;
- дані;
- аналітика;
- ідеї;
- опитування;
- чат;
- профіль;
- вхід/вихід.

Також в кожному з цих розділів системи присутні і свої спеціальні форми та елементи інтерфейсу для введення даних , наприклад функціонал для завантаження CSV-файлів, публікації ідей , написання коментарів , всі вони використовують вбудовані можливості Django Forms для валідації та рендерингу. Не слід забувати і про таблиці для представлення структурованих даних з можливістю їх фільтрації та сортування , а також про візуалізацію даних , таку як графіки для наочного відображення аналітичних результатів.

Принципи адаптивного дизайну були застосовані для забезпечення коректного відображення та функціональності системи на персональних комп’ютерах, планшетах та мобільних пристроях, що підвищує доступність та зручність використання для широкого кола користувачів.

Як вже ось було згадано завантаження файлів CSV , то слід про нього розповісти детальніше , адже це головний модуль саме для аналітики , без нього просто не було б даних на основі яких ця аналітика і виконувалась.

Цей модуль відповідає за централізоване надходження та обробку сирих даних про сталий розвиток. Його ключовою функцією є завантаження даних у

форматі CSV-файлів. Для цього розроблена спеціалізована форма CSVUploadForm, яка приймає файли.

На серверній стороні у файлі views.py відбувається валідація завантаженого файлу , а саме перевірка формату .csv та його парсинг. Зміст CSV-файлу читається рядок за рядком, при цьому відбувається зіставлення даних з колонок файлу з відповідними полями моделі DataEntry такими як year, month, energospozhyvannya_zagalne тощо , згідно із заздалегідь визначеними назвами.

Особлива увага приділяється коректній обробці числових значень та порожніх значень для поля month, що дозволяє розрізняти місячні та річні дані. Механізм оновлення/додавання записів реалізовано таким чином, що перед створенням нового об'єкта DataEntry система перевіряє наявність запису за комбінацією year та month. Якщо запис для даного періоду вже існує, він оновлюється новими даними, в іншому випадку створюється новий запис. Це запобігає дублюванню даних та забезпечує актуальність інформації.

Весь процес супроводжується повідомленнями через систему messages Django, про успіх завантаження або про виявлені помилки ,наприклад некоректний формат даних у файлі, надаючи користувачу зворотний зв'язок. Після імпорту даних, користувачі з відповідними правами можуть переглядати завантажені дані через інтерфейс, який передбачає гнучку фільтрацію за роком та місяцем, що дозволяє деталізувати або агрегувати представлену інформацію.

3.3 Аналіз отриманих результатів

Інформаційна система моніторингу сталого розвитку розроблена не лише для збору та зберігання даних, але й для їхнього глибокого аналізу та перетворення на інструмент прийняття обґрунтованих рішень за допомогою різних інтерактивних модулів. Основна мета аналітичного модуля - надати користувачам комплексну та легко інтерпретовану оцінку стану сталого розвитку за визначені періоди.

Принципи аналізу ґрунтуються на концепції єдиного показника, який

дозволяє об'єднати різнорідні дані в єдину, цілісну оцінку. Такий підхід вирішує проблему коли у показників різні одиниці виміру і кардинально різні значення для аналізу оцінки сталості. Механізм аналізу є гнучким і адаптивним, що досягається завдяки використанню конфігураційного словника INDICATOR_CONFIG, визначеного у файлі core/constants.py. Цей словник є ключовим елементом, оскільки він містить всі необхідні додаткові дані для аналізу і для коректного розрахунку: verbose_name - зрозуміла назва для відображення, target - цільове значення, weight - ваговий коефіцієнт та direction - напрямок бажаних змін – lower_is_better або higher_is_better.

Завдяки такій параметризації, система може бути легко адаптована до зміни пріоритетів або додавання нових показників без необхідності модифікації програмного коду ядра аналітики.

Процес трансформації сирих даних, завантажених у модель DataEntry, на індивідуальні бали складається з кількох етапів. Спочатку, для кожного окремого показника , наприклад для енергоспоживання або площа зелених насаджень система обчислює індивідуальний бал у діапазоні від 0 до 100. Цей бал є результатом порівняння фактичного значення показника з його встановленим цільовим значенням, з урахуванням напрямку - direction. Якщо показник належить типу - lower_is_better , як показник викиді вуглексилого газу і фактичне значення є низьким або нульовим то в результаті , бал буде високим і близьким до 100. І навпаки, якщо higher_is_better , такий показник як кількісті заходів які присвячені сталому розвитку і фактичне значення високе, бал також буде високим. Така нормалізація дозволяє уніфікувати оцінки для показників з різними одиницями виміру та діапазонами значень.

На другому етапі, ці індивідуальні бали агрегуються для формування показника сталого розвитку. Кожен індивідуальний бал множиться на свій ваговий коефіцієнт - weight, визначений у INDICATOR_CONFIG.

Вагові коефіцієнти відображають відносну важливість кожного показника у

загальній системі цінностей сталого розвитку організації чи спільноти. Сума всіх вагових коефіцієнтів дорівнює одиниці, що забезпечує коректність зваженого підсумовування та дозволяє отримати фінальний бал, який також знаходиться у діапазоні від 0 до 100. Такий механізм дозволяє системі не тільки показувати абсолютні значення показників, а й надавати комплексну, зважену оцінку прогресу в напрямку сталості. Крім того, система підтримує гнучкий порівняльний аналіз, дозволяючи користувачам зіставляти показники за різні роки або місяці, виявляючи динаміку змін та ефективність вжитих заходів.

Можливості візуалізації та інтерпретації показників є досить обширними так як фективність аналізу значною мірою залежить від того, наскільки зрозуміло та наочно представлені отримані результати. Система забезпечує візуалізацію аналітичних даних через комбінацію таблиць та графіків, що дозволяє користувачам швидко інтерпретувати складну інформацію та виявляти ключові тенденції.

Цінність та практичне застосування отриманих результатів. Основна цінність розробленої інформаційної системи полягає в її здатності перетворювати сирі дані на значущі для адміністраціїї та стейкхолдерів результати, що є основою для ефективного управління та прийняття рішень у сфері сталого розвитку. Отримані аналітичні результати дозволяють ідентифікувати проблемні зони, оцінювати ефективність впроваджених заходів та формувати майбутні стратегії.

Наприклад, аналіз динаміки нашого показника сталого розвитку дозволяє оцінити загальну успішність ініціатив зі сталого розвитку, а деталізація до індивідуальних показників вказує на конкретні аспекти, де необхідно зосередити зусилля. Якщо, наприклад, система показує стабільне зниження балу за показником відсоток відсортованих відходів, це є чітким сигналом для перегляду та оптимізації процесів управління відходами.

Окрім аналітичного функціоналу , важливу роль відіграють і інтерактивні модулі системи:

- ідеї;
- опитування;
- чат.

Модуль ідей є платформою для генерації нових пропозицій від спільноти, дозволяючи виявляти інноваційні підходи до вирішення існуючих проблем сталості. Можливість лайкати і коментувати ідеї сприяє їхньому колективному обговоренню та оцінці. Опитування надають інструмент для систематичного збору думок користувачів щодо конкретних ініціатив або загальних питань сталості, що може бути використано для коригування стратегій та визначення пріоритетів. Груповий чат сприяє оперативній комунікації та обміну досвідом, створюючи динамічне середовище для вирішення поточних питань. Всі ці модулі разом сприяють підвищенню обізнаності та залученості користувачів до питань сталого розвитку, перетворюючи систему не просто на інструмент визначення рівня сталого розвитку, а на платформу для спільної дії та колективного внеску у досягнення цілей сталого розвитку. Такий синергетичний підхід забезпечує неперервний цикл покращення та адаптації до мінливих умов.

Висновки до 3 розділу

Вхідні дані які інформаційно-аналітична система використовує для аналітики є логічно структурованими і не мають в собі нічого зайвого окрім необхідних значень для аналітики та простого відображення в системі. Сама система складається з багатьох модулів і кожен з них унікальний , один з них для аналітики , другий для спілкування , третій для визначення найкращих і оптимальних рішень для сприяння сталого розвитку і так далі. Усі моделі системи або як їх ще можна назвати класи , які в свою чергу використовуються для проєктування бази даних системи для її нормального функціоналу мають усі необхідні для цього поля з логічними назвами.

Вцілому система має не малий функціонал та багато сценаріїв використання

і додатково ці сценаріїї залежать напряму від того ким я користувач , для одного типу користувача може бути більше варіантів користування ніж для іншого. Це тому і називається система , тому що можна робити абстолютно різні речі для досягнення або руху до поставленої мети.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ

4.1 Опис програмної реалізації

Програмна реалізація автоматизованої інформаційно-фаналітичної системи по визначенню пріоритетів сталого розвитку університетської системи освіти базується повністю на використанні веб-фреймворку Django, який забезпечує високий рівень абстракції, порівняно з іншими підходами високу швидкість розробки та підтримку ключових аспектів веб-додатків, таких як управління базами даних, маршрутизація запитів, автентифікація користувачів та панель адміністратора. Розроблена система має стандартну для Django структуру проєкту, яка складається з корневого проєкту та одного або декількох додатків , у даному випадку, основний додаток core.

Кореневий проєкт Django sustainability_system містить глобальні налаштування та конфігурацію , які відповідають за налаштування всієї системи. Основним функціонуючим компонентом системи є додаток core, який реалізує всю логіку, пов'язану з моделями даних, обробкою запитів, формами та шаблонами сторінок, що безпосередньо стосуються функціоналу системи та управління користувачами. Така організація коду за принципом додатків робить систему модульною, що спрощує її розробку, тестування та подальше масштабування. Нижче наведений вигляд структури усіх файлів та директорій системи (рис.4.1). Повний зміст ключових файлів наведений у Додатку Б.

На цьому рисунку видно схему дерева файлів та директорій проєкту розробленої автоматизованої інформаційно-аналітичної системи , де присутня корнева папка university_sustainability_system у якій є наступні директорії:

- core , це основний додаток;
- media , це директорія де зберігається усі медіа файли необхідні системі;
- sustainability_system , це корневий проєкт;
- templates , це директорія де містяться шаблони системи.

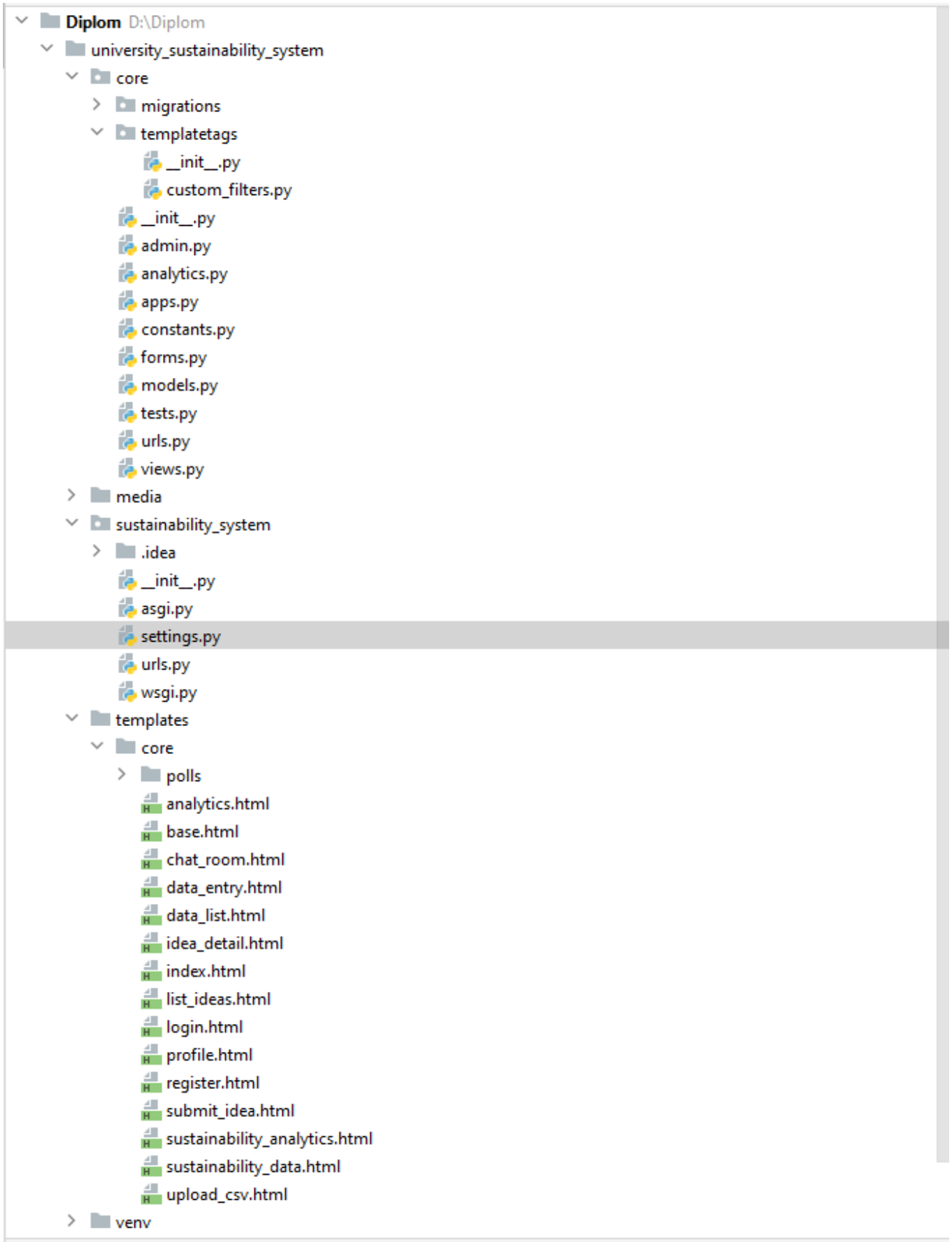


Рисунок 4.1 Структура усіх файлів проєкту

Файл manage.py присутній для того щоб використовувати командний рядок для управління проектом. Він дозволяє виконувати різноманітні завдання, такі як запуск сервера розробки, створення міграцій бази даних, застосування міграцій, створення суперкористувача та інші адміністративні дії безпосередньо з терміналу.

Файл sustainability_system/settings.py містить усі необхідні для проєкту налаштування. Тут визначаються параметри завдяки яким стає можливим підключення до бази даних PostgreSQL, список встановлених додатків , включаючи core та вбудовані додатки Django, налаштування для статичних та медіа-файлів які використовуються у системі, конфігурація автентифікації користувачів , а саме URL для входу/виходу та звісно ж перенаправлення, налаштування мови та часового поясу, а також інші не менш важливі параметри, що впливають на роботу системи та на її функціонал . Приклад блоку з файлу settings.py , який відповідає за налаштування бази даних проєкту наведено на (рис.4.2).

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': 'diplomdb',       # <--- НАЗВА бази, яку я створив
        'USER': 'zhenya',   # <--- ІМ'Я користувача, якого я створив
        'PASSWORD': '1234', # <--- ПАРОЛЬ цього користувача
        'HOST': 'localhost',              # Зазвичай 'localhost'
        'PORT': '',                       # Замовчки порожній
    }
}
```

Рисунок 4.2 – Налаштування бази даних

Файл core/models.py містить визначення усіх моделей даних які потрібні системі для повного її функціоналу , який був з самого початку визначений. Кожен клас у цьому файлі являю собою таблицю в базі даних та визначає поля цієї

таблиці, їх тип даних, обмеження та зв'язки з іншими моделями наприклад ForeignKey, OneToOneField , це може використовуватися наприклад для реалізаціїї того щоб 1 користувач міг голосувати тільки 1 раз.

Моделі є основою для роботи ORM Django, вони дозволяють взаємодіяти з базою даних за допомогою об'єктно-орієнтованого підходу. У цьому файлі описано багато моделей , серед них моделі для показників - IndicatorType, записів даних - DataEntry, профілю користувача – Profile та ідей – Idea та інші. Як цей файл працює та як визначається кожен клас можна побачити на прикладі моделі для опису коментарів (рис.4.3).

```python
class Comment(models.Model):
    # Зв'язок Many-to-One: багато коментарів належить ОДНІЙ ідеї
    # related_name='comments' дозволяє отримати всі коментарі до ідеї через idea.comments.all()
    idea = models.ForeignKey(Idea, on_delete=models.CASCADE, related_name='comments', verbose_name="Ідея")
    # Зв'язок Many-to-One: багато коментарів написані ОДНИМ автором (користувачем)
    author = models.ForeignKey(User, on_delete=models.CASCADE, related_name='comments', verbose_name="Автор коментаря")
    # Текст коментаря
    text = models.TextField(verbose_name="Текст коментаря")
    # Автоматично фіксуємо дату і час створення коментаря
    created_at = models.DateTimeField(auto_now_add=True, verbose_name="Дата створення")

    class Meta:
        verbose_name = "Коментар"
        verbose_name_plural = "Коментарі"
        ordering = ['created_at'] # Сортуємо коментарі від старих до нових

    def __str__(self):
        # Показуємо початок тексту коментаря та автора
        return f"Коментар до '{self.idea.title[:20]}...' від {self.author.username}"
```

Рисунок 4.3 – Модель для опису коментарів

Файл core/views.py містить основну логіку обробки запитів і складається з набору view-функцій які нам потрібні на різних сторінках системи. Кожна функція у цьому файлі відповідає за обробку свого індивідуального типу запиту до певного URL. View-функції взаємодіють з моделями через ORM для отримання або зберігання даних, використовують потрібні форми для обробки інформації яку вводить користувач, виконують необхідні розрахунки , наприклад в View , який використовується для сторінки на якій відбувається аналітика показника сталого

розвитку університету, викликають функцію render() для відображення відповідного HTML-шаблону, передаючи йому необхідні дані. Також в цьому файлі реалізована логіка яка відповідає за управління користувачами та обробку повідомлень від системи.

Як це все працює показано на прикладу view-функції для реєстрації (рис.4.4).

```python
# View для реєстрації користувачів
# Використовуємо стандартну форму UserCreationForm
def register_view(request):
    if request.method == 'POST':
        # Створюємо форму UserCreationForm і заповнюємо її даними з POST-запиту
        form = UserRegistrationForm(request.POST)
        if form.is_valid():
            # Якщо форма валідна, зберігаємо нового користувача в базу даних
            user = form.save()
            # TODO: Можна створити об'єкт Profile для нового користувача тут, якщо потрібно
            # Profile.objects.create(user=user)

            # Виводимо повідомлення про успішну реєстрацію
            messages.success(request, "Реєстрація успішна! Тепер Ви можете увійти.")

            # Перенаправляємо користувача на сторінку входу
            return redirect('login')
    else:
        # Якщо це GET-запит, створюємо порожню форму UserCreationForm
        form = UserRegistrationForm()

    # Рендеримо шаблон 'register.html', передаючи форму в контекст
    return render(request, 'core/register.html', {'form': form})
```

Рисунок 4.4 – View для реєстраціїї користувачів

Файл core/urls.py визначає URL-маршрутизатори для додатку core. Тут вказується, які конкретні шляхи є у межах додатку ,наприклад:

– /ideas/;
– /profile/;
– /analytics/.

І всі оці шляхи мають бути пов’язані з відповідними View-функціями,

визначеними у core/views.py. Кожен URL-шаблон має своє унікальне ім'я , що дозволяє зручно та легко під час розробки генерувати посилання на ці сторінки в шаблонах та коді функцій View. Цей файл відносно не великий тож повний його вміст наведений на (рис.4.5).

```python
# core/urls.py

from django.urls import path
from . import views # Імпортуємо функції View з поточного додатку


urlpatterns = [

    # URL для сторінки перегляду списку даних: /data/list/
    path('sustainability-data/', views.view_sustainability_data, name='sustainability_data'),
    # URL-шаблон для головної сторінки
    path('', views.index, name='index'), # 'name="index"' дає ім'я цьому URL-шаблону
    path('sustainability-analytics/', views.view_sustainability_analytics, name='sustainability_analytics'),
    # Зараз додамо їх заглушки, щоб посилання в навігації працювали
     path('register/', views.register_view, name='register'),
     path('login/', views.login_view, name='login'),
     path('logout/', views.logout_view, name='logout'),      # URL для виходу
     path('profile/', views.profile_view, name='profile'),
    # URL для сторінки зі списком всіх ідей
    path('ideas/', views.list_ideas_view, name='ideas_list'),
    # URL для сторінки з формою надсилання нової ідеї
    path('ideas/submit/', views.submit_idea_view, name='submit_idea'),

    path('ideas/<int:idea_id>/', views.idea_detail_view, name='idea_detail'),
    # URL для опитувань (ДОДАНО ДЛЯ ДНЯ 12)
    # URL-и для модуля опитувань
    path('polls/', views.poll_list_view, name='poll_list'),
    path('polls/<int:poll_id>/', views.poll_detail_view, name='poll_detail'),
    path('polls/create/', views.create_poll_view, name='create_poll'),
    # URL для лайків
    path('ideas/<int:idea_id>/like/', views.like_idea_view, name='like_idea'),
    # URL для завантаження CSV-файлів (доступно тільки адміністраторам)
    path('data/upload-csv/', views.upload_csv_view, name='upload_csv'),

    path('chat/', views.chat_room_view, name='chat_room'),

]
```

Рисунок 4.5 – Файл urls.py

Файл core/admin.py використовується для налаштування та кастомізації панелі адміністратора Django для моделей додатку core. Тут реєструються моделі (DataEntry, Profile, Idea, Comment), визначається які поля відображати у списку

об'єктів, за якими полями можна фільтрувати та шукати, а також налаштовується вигляд форм додавання та редагування об'єктів. Для прикладу з чого складається файл admin.py , на (рис.4.6) наведено шматок коду , який відповідає за реєстрацію моделі profile в адмін панелі.

```
# Реєструємо модель Profile в адмін-панелі як окремий розділ
@admin.register(Profile)
class ProfileAdmin(admin.ModelAdmin):
    # Які поля показувати в загальному списку Профілів в адмінці
    # Додали 'photo' та 'description'
    list_display = ('user', 'user_type', 'photo', 'description')
    # Поля для фільтрації в адмінці
    list_filter = ('user_type',)
    # Поля для пошуку
    search_fields = ('user__username',)
```

Рисунок 4.6 – Реєстрація моделі profile

Папка core/templates/ містить всі HTML-шаблони системи , які використовуються View-функціями для формування веб-сторінок. Тут розташовані файли шаблонів, такі як base.html - базовий шаблон з основною структурою сторінки та навігацією, index.html - головна сторінка, шаблони для форм введення списку даних, аналітики, сторінок автентифікації, профілю, списку ідей , чату тощо.

Використання системи шаблонів дозволяє окремо реалізовувати логіку системи та окремо представлення і потім завдяки цьому повторно використовувати блоки коду. Приклад того як виглядають шаблони сторінок , наведено на (рис.4.7) , на ньому шаблон який відповідає за додавання ідеї або ініціативи до системи для просування ідей сталого розвитку.

```
{# templates/core/submit_idea.html #}
{% extends 'core/base.html' %} {# Наслідуємо наш базовий шаблон #}

{% block title %}Запропонувати ідею{% endblock %} {# Задаємо заголовок сторінки #}

{% block content %} {# Контент всередині блоку 'content' базового шаблону #}
    <h2>Запропонувати нову ідею</h2>

    <p>Поділіться своїми пропозиціями щодо сталого розвитку університету.</p>

    <form method="post"> {# Форма для відправки даних #}
        {% csrf_token %} {# Обов'язковий токен безпеки #}

        {# Відображення помилок форми, якщо вони є #}
        {% if form.non_field_errors %}
            <div class="alert alert-danger">
                <strong>Помилки форми:</strong>
                <ul>
                {% for error in form.non_field_errors %}
                    <li>{{ error }}</li>
                {% endfor %}
                </ul>
            </div>
        {% endif %}

        {# Рендеримо поля форми IdeaForm #}
        {# TODO: Використовувати ручний рендеринг для кращого вигляду з Bootstrap #}
        {{ form.as_p }}

        <button type="submit" class="btn btn-primary mt-3">Надіслати ідею</button> {# Кнопка відправки форми #}
    </form>

    <p class="mt-3">
        <a href="{% url 'ideas_list' %}">Переглянути всі ідеї</a> {# Посилання на список ідей #}
    </p>

{% endblock %} {# Кінець блоку content #}
```

Рисунок 4.7 – Шаблон submit_idea.html

4.2 Керівництво користувача

Для початку роботи з системою необхідно відкрити веб-браузер та ввести її адресу. Після цього на зустрічає головна сторінка системи , яка надає загальну інформацію про себе та навігаційне меню для доступу до основних розділів. Вигляд головної сторінки наведено на (рис.4.8).

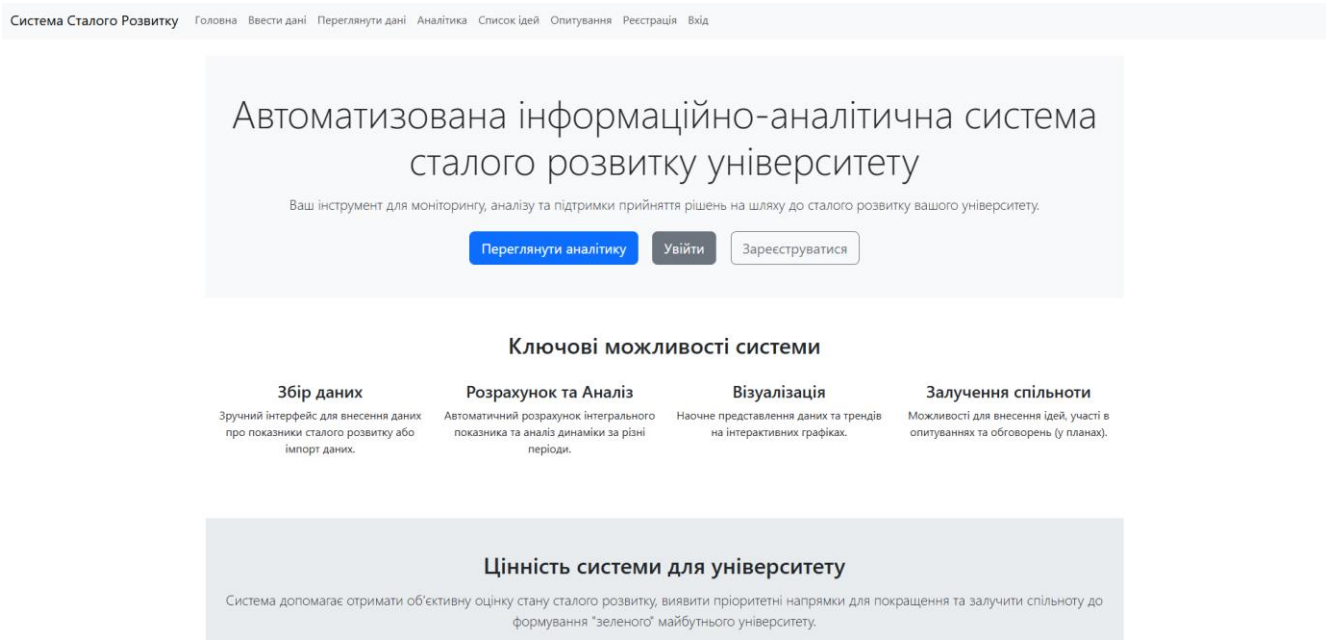


Рисунок 4.8 – Головна сторінка системи

Система передбачає розмежування прав доступу та персоналізацію взаємодії через облікові записи користувачів. Нові користувачі можуть пройти процедуру реєстрації щоб почати користуватися усіма можливостями розробленої системи. Для цього на головній сторінці одразу є 2 відповідні кнопки: увійти та зареєструватися. Якщо ви новий користувач який до цього моменту не користувався системою то потрібна саме кнопка реєстрації , після її натискання ви потрапляєте на сторінку реєстрації (рис.4.9).

Після реєстраціїї вас знову пускає на головну сторінку але вже буде видно що ви створили свій профіль і зверху сторінки з’явиться ваш аватар та ваше ім’я(рис.4.10).

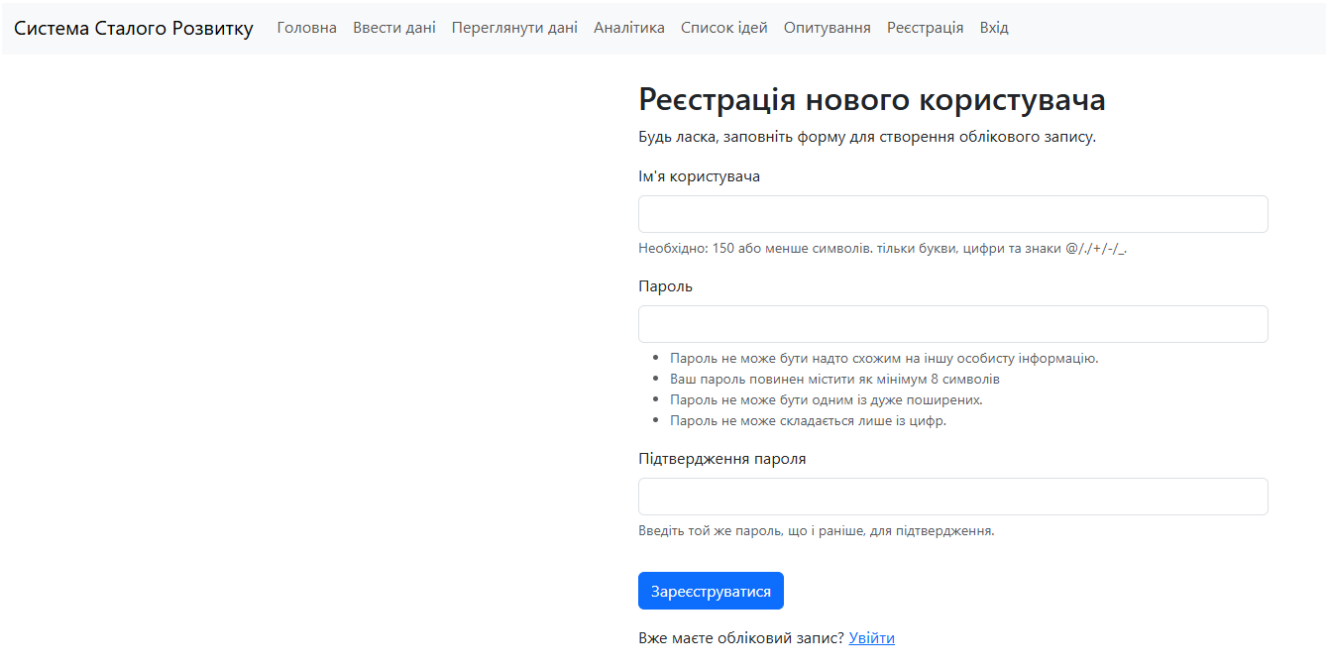


Рисунок 4.9 – Сторінка реєстрації

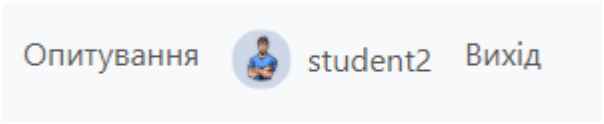


Рисунок 4.10 – Іконка профілю

Якщо ж навпаки ви користувались цією системою раніше і у вас є свій обліковий запис , який ви реєстрували раніше тоді вам потрібна саме кнопка увійти , після натискання якої вас перекине на сторінку входу до системи(рис.4.11).

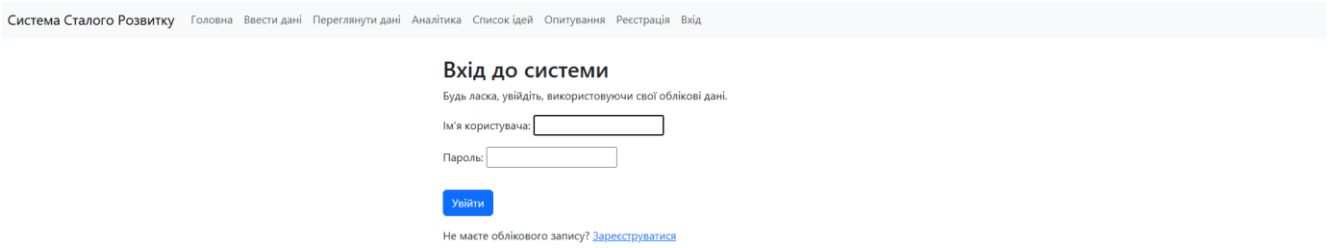


Рисунок 4.11 – Сторінка входу до системи

Також після реєєстрації або входу у систему у вас є свій профіль до якого можна зайти і подивитись інформацію або редагувати її , в майбутньому можна буде реалізувати більш складнішу структуру для інформаціїї про користувача та розробити більш гнучку персоналізація власного профілю. Профіль користувача зображено на (рис.4.12).

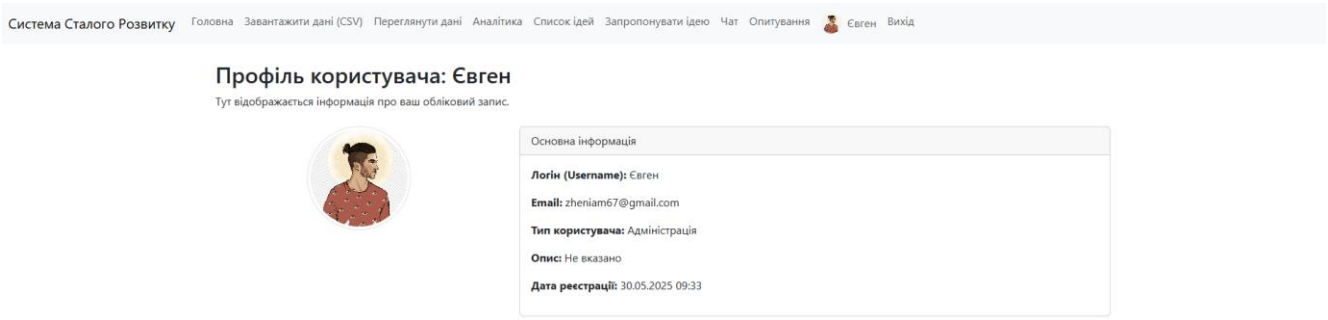


Рисунок 4.12 – Профіль користувача

Після реєстрації або входу до системи ви можету запропонувати свою ідею або ініціативу щодо покращення сталого розвитку університету. Наприклад ви можете запропонувати створення різноманітних гуртків які будуть орієнтовані на сталий розвиток. Модуль ідей для цього і призначений , і потрапити до нього можна з головної сторінки , біля іконки профілю.

Коли ви перейшли в цей модуль ви можете почати додавати свої ідеї та ініціативи , заповнивши відповідну форму на сторінці (рис.4.13). В майбутньому форму можна буде розширити для того щоб коже користувач розкрив свою ідею на всі 100% і її підтримали як умого більшн користувачів , зараз це проста та зрозуміла форма без зайвого функціоналу , з яким без зайвого клопоту кожен користувач розбереться. Для цього в системі також реалізована сторінка для перегляду усіх запропонованих ідей від різних користувачів (рис.4.14). Якщо ініціатива дійсно класна її можна оцінити натиснувши кнопку лайку і ініціативи які набирають найбільшу кількість лайків завжди перші на цій сторінці.

Рисунок 4.13 – Сторінка для публікаціїї ідей

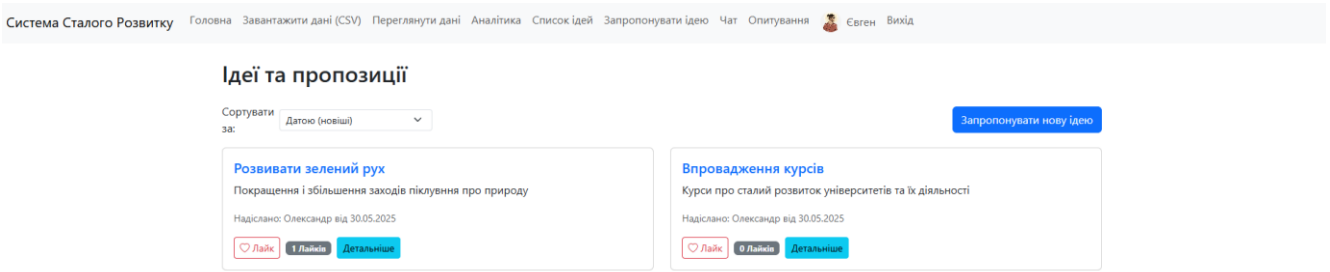


Рисунок 4.14 – Список всіх ідей від користувачів

Також кожен користувач може переглянути аналітику університу , а саме його показник оцінки сталого розвитку. Це все знаходить на сторінці аналітика , потрапити на неї знову ж таки можна через головну сторінку , її вигляд наведено на (рис.4.15). На цій сторінці також зображений графік(рис.4.16) який показує у якому напрямку рухається університет у плані сталого розвитку , чим вищий показник тим звісно краще. Відповідно якщо показник зібльшується то все рухається у правильному напрямку , якщо ні , потрібно як умога більше ідей та ініціатив зі сталого розвитку та їх негайне впровадження для покращення показника.

Користувач може бачити аналітику по рокам , а може вибрати окремий рік який його цікавить та подивитись аналітику тільки за окремий рік , для кожного місяця вибраного року для оцінки.

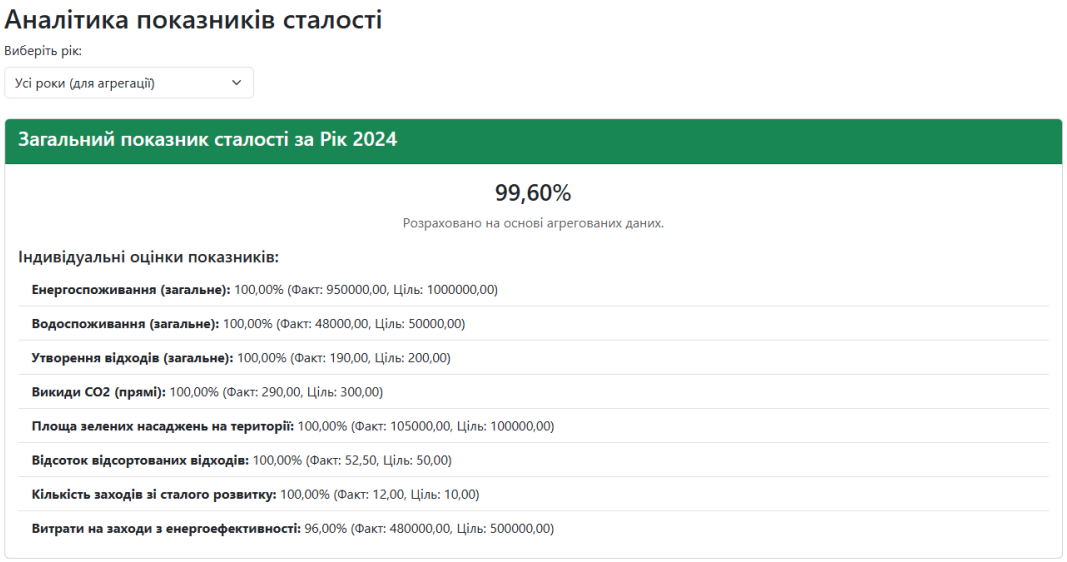


Рисунок 4.15 – Сторінка аналітики

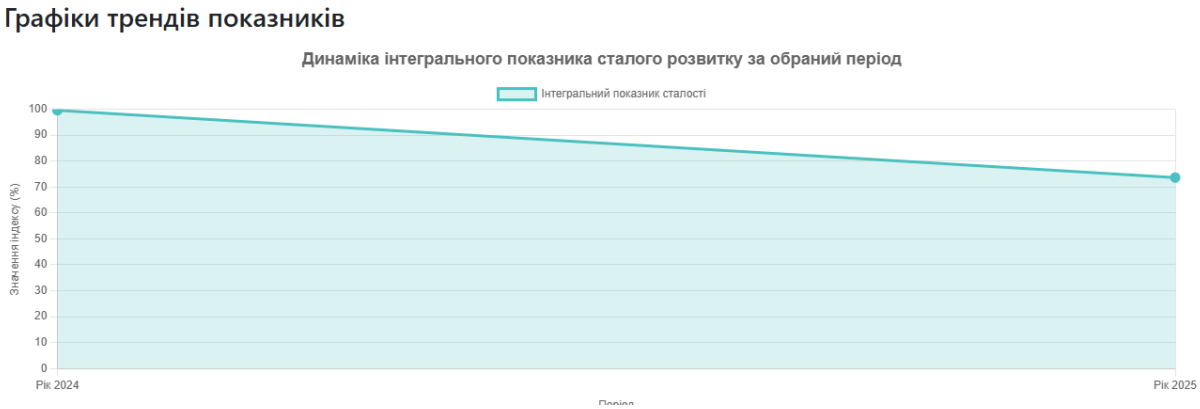


Рисунок 4.16 – Графік показника сталого розвитку

Звісно вся аналітика не може працювати без відповідниз даних , дані завантажуються за допомогою звичайних CSV-файлів , які мають необхідну структуру для системи , щоб вона їх розпізнавала, звичайні користувачі не можуть їх завантажувати , тільки переглядати ті дані які ввели адміністратори чи користувачі яким видали на це дозвіл через панель адміністрації. Дані можна переглянути на відповідній сторінці , вона так і називається , переглянути дані (рис.4.17).

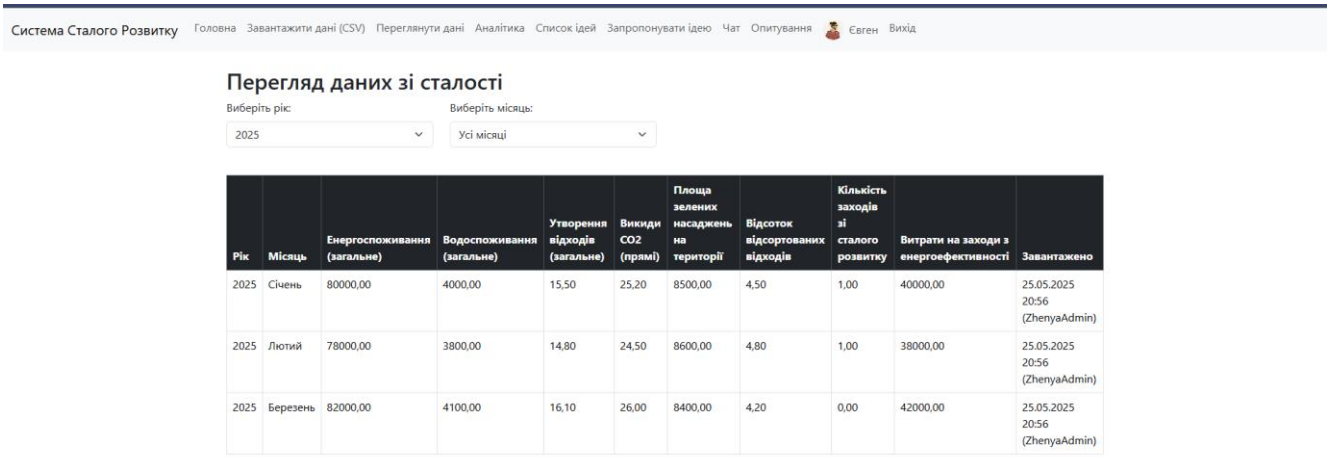


Рисунок 4.17 – Дані для аналітики

Ще один дуже важлива частина реалізованої системи , це сторінка опитувань , їх проходження та створення. Як і в будь-якій системі або соц. Мережі для їх проходження потрібно спочатку створити профіль та зареєструватися і потім користувач може потрапити на сторінку з опитуваннями які тривають прямо зараз(рис.4.18).

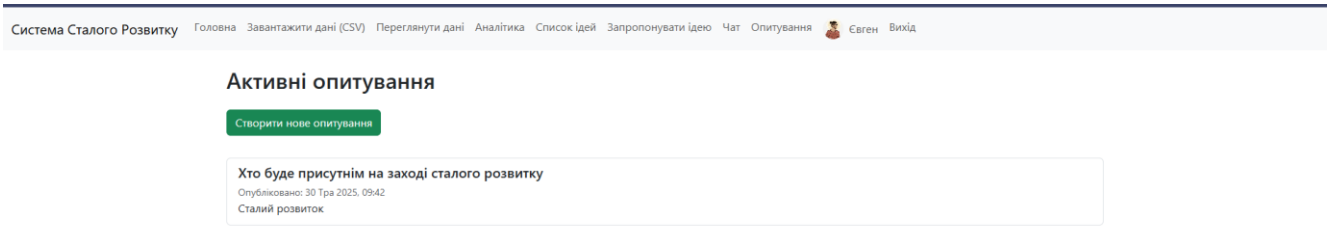


Рисунок 4.18 – Сторінка з актуальними опитуваннями

Для проходження опитування користувач повинен обрати те опитування яке доступне або якщо їх багато то те в якому він хоче взяти участь. Після натискання на опитування користувач потрапляє на сторінку саме цього опитування де вже і може проголосувати(рис. 4.19).

Після обрання свого варіанту відповіді система покаже скільки голосів на кожному з варіантів та потім буде показано що ви вже брали в цьому опитування участь(рис.4.20).

Рисунок 4.19 – Сторінка опитування

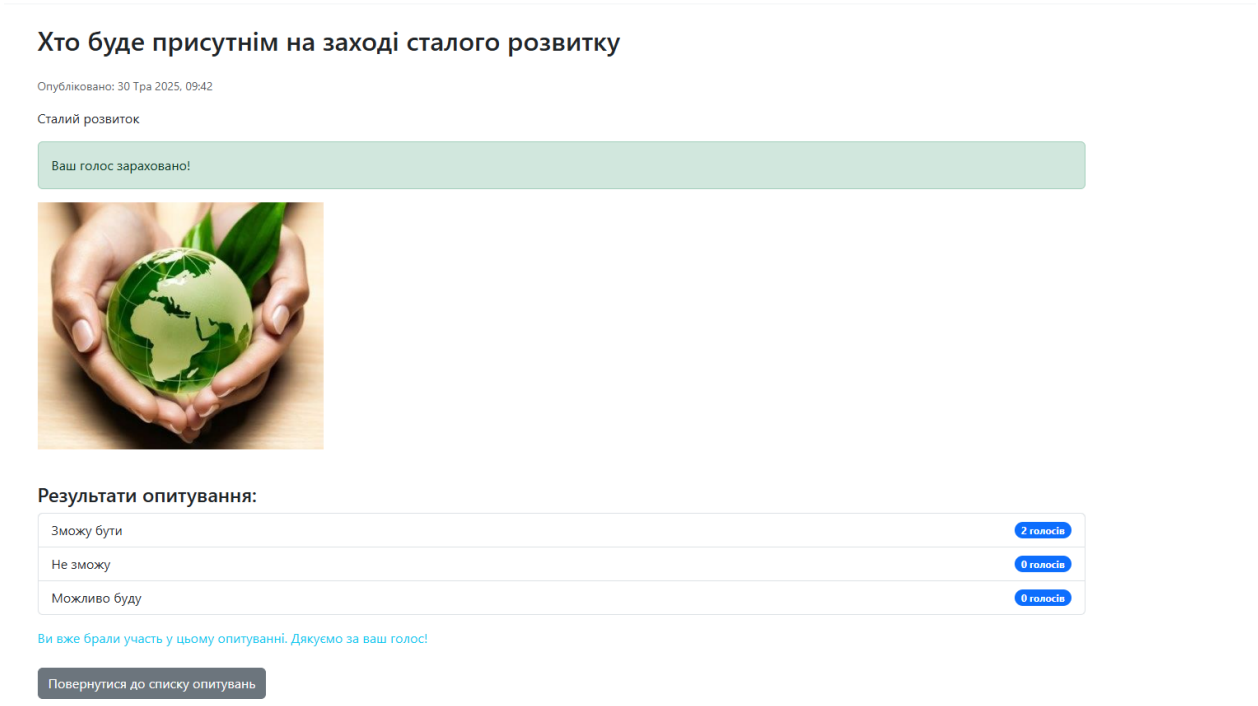


Рисунок 4.20 – Результат голосування для 1 користувача

Ну і як на мене 2 після аналітики найважливіший функціонал системи це чат з усіма користувачами які зареєстровані у цій системі , адже ідеї і голосування це звісно також позитивно впливає на сталий розвиток університету і на швидкість пошуку шляхів покращення становище університету в цілому , але спілкування на пряму це звісно те що дає найбільший профіт у цій справі. Потрапити на цю сторінку також можна через головне меню у верхній частині головної сторінки(рис.4.21).

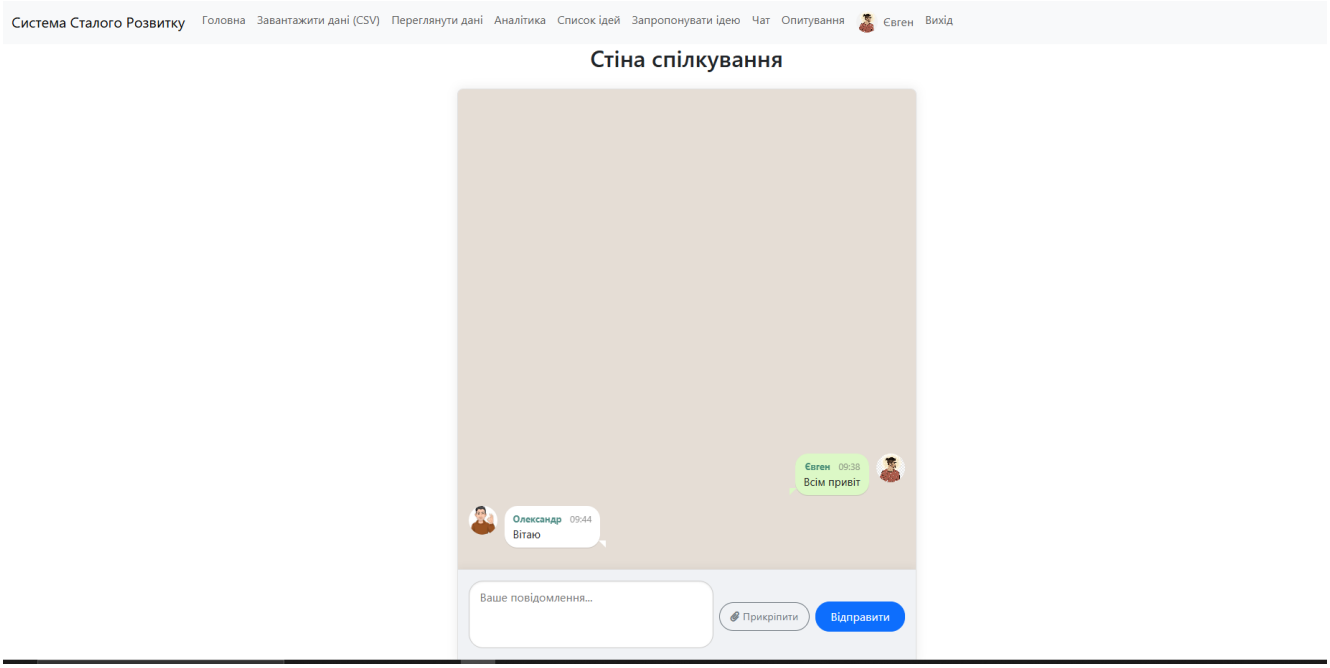


Рисунок 4.21 – Стіна спілкування (Чат)

Як ми бачимо користувачі можуть не тільки писати прості повідомлення а і прикріпляти різні файли , наприклад фото або відео , так набагато легше все пояснити усім користувачам , якщо вони просто подивляться відеоматеріал який ви надішлете , все що потрібно для цього це натиснути кнопку прикріпити біля кнопки відправити та обрати файл на власному персональному комп’ютері і потім надіслати його , нічого складного , як у всіх соціальних мережах , обрав і відправив(рис.4.22).

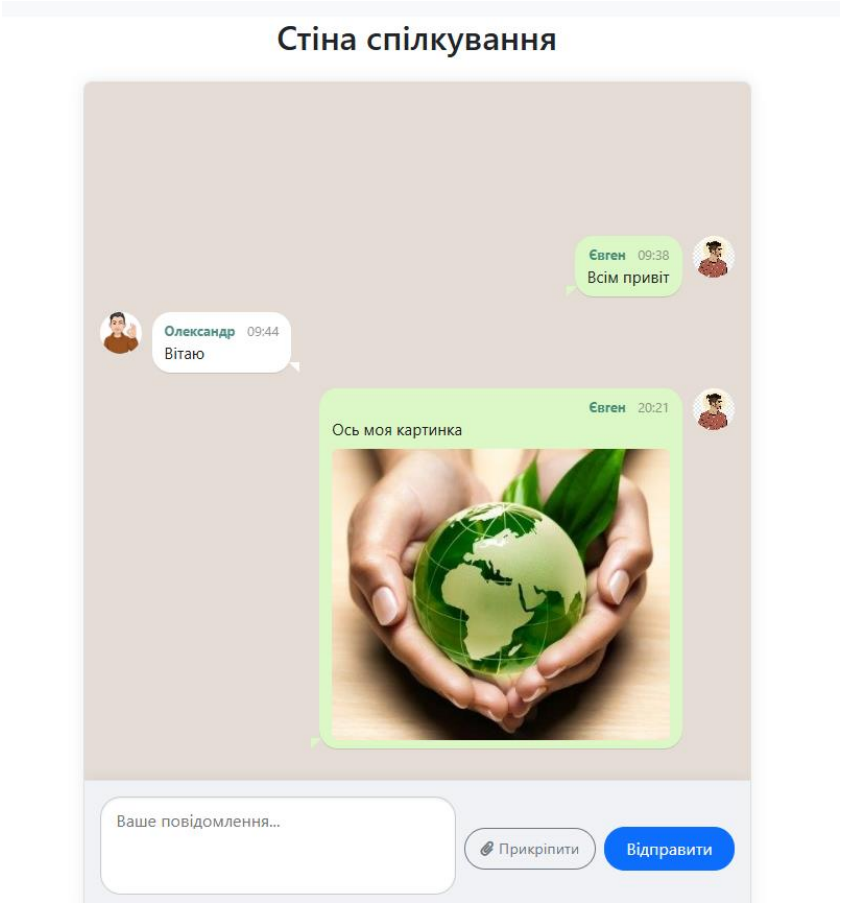


Рисунок 4.22 – Прикріплений файл до повідомлення

4.3 Тестування

Тестування розробленої інформаційної системи є критично важливим етапом розробки, що дозволяє переконатись у її працездатності, відповідності поставленим вимогам та виявити потенційні помилки.

Методика тестування полягала у послідовній перевірці всіх реалізованих функціональних можливостей з точки зору кінцевого користувача. Були розроблені тестові сценарії, що охоплювали основні операції системи, такі як реєстрація та автентифікація користувачів.

В процесі тестування використовувались як валідні, так і, в деяких випадках, невалідні вхідні дані наприклад при тестуванні валідації форм реєстрації, щоб перевірити коректність обробки помилок системою.

Отже тестування реєстрації користувачів відбувалось за 2 сценаріями: перший сценарій , це введення дуже простого пароля , який не підхдить під критерії для його створення , наприклад (12345678). Та другий сценарій де було використано надійний пароль , який система має прийняти. Результат тестування за першим сценарієм зображений на (рис.4.23) , за другим на (рис.4.24).

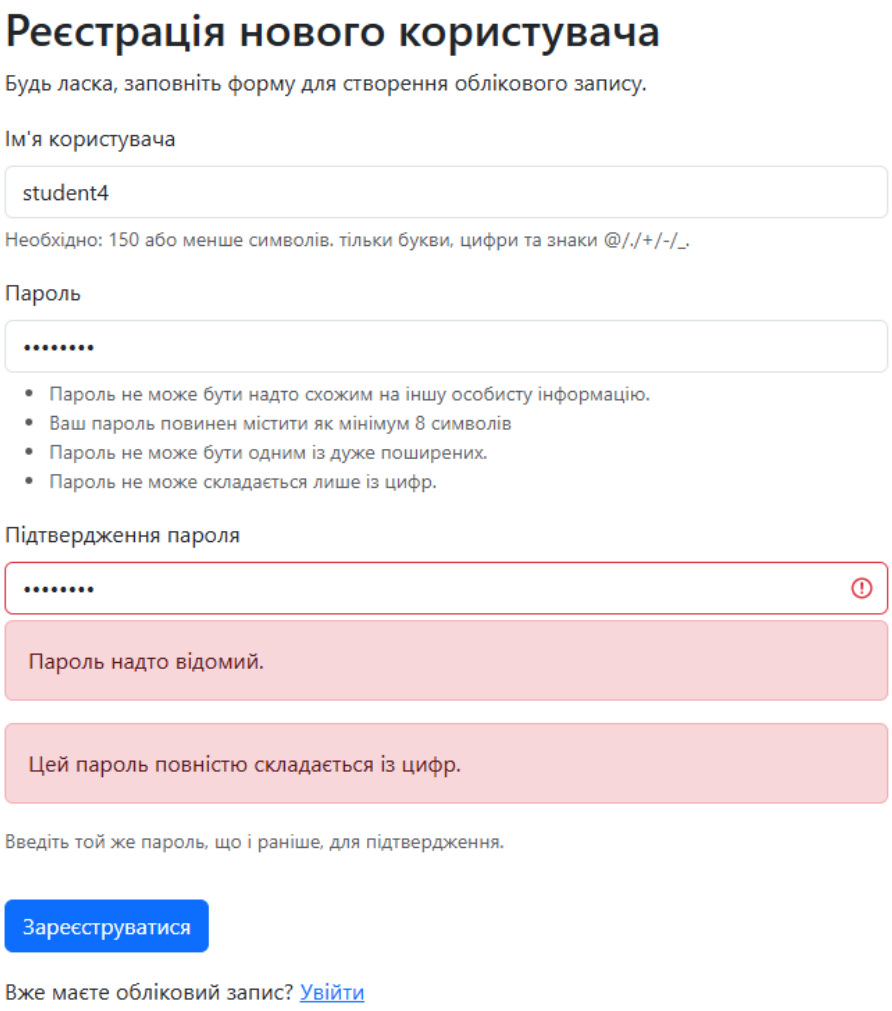


Рисунок 4.23 – Тестування форми реєстрації

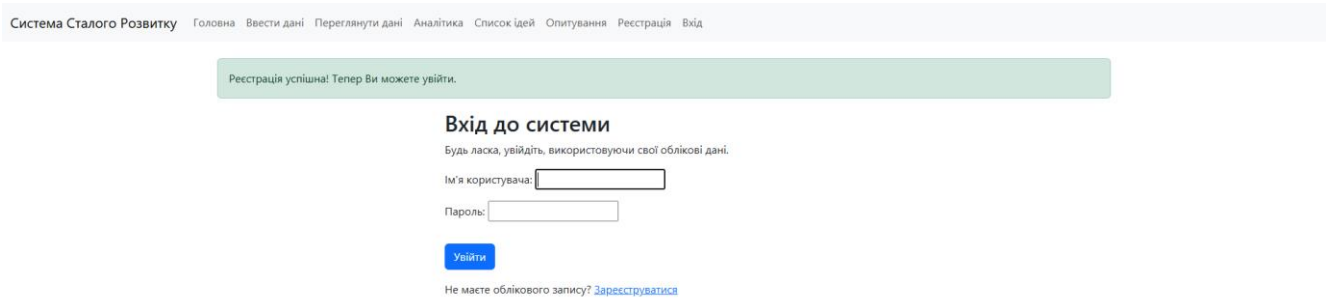


Рисунок 4.24 – Тестування форми реєстрації

Тепер проведемо тестування форми входу до системи , цей процес нічим не відрізняється від тестування форми реєстрації , введемо спочатку невірний логін або пароль , потім введемо корректні дані. Результат цього тестування зробрежний на (рис.4.25 - 4.26).

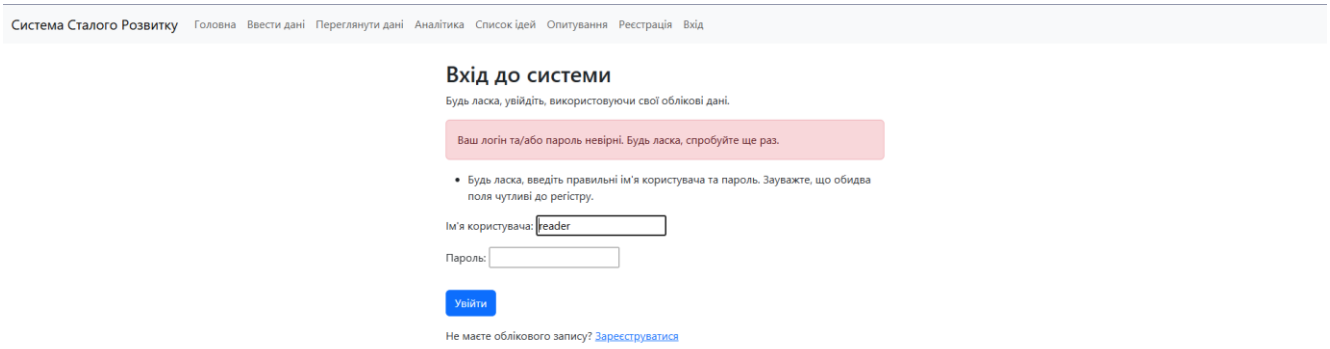


Рисунок 4.25 – Тестування форми входу

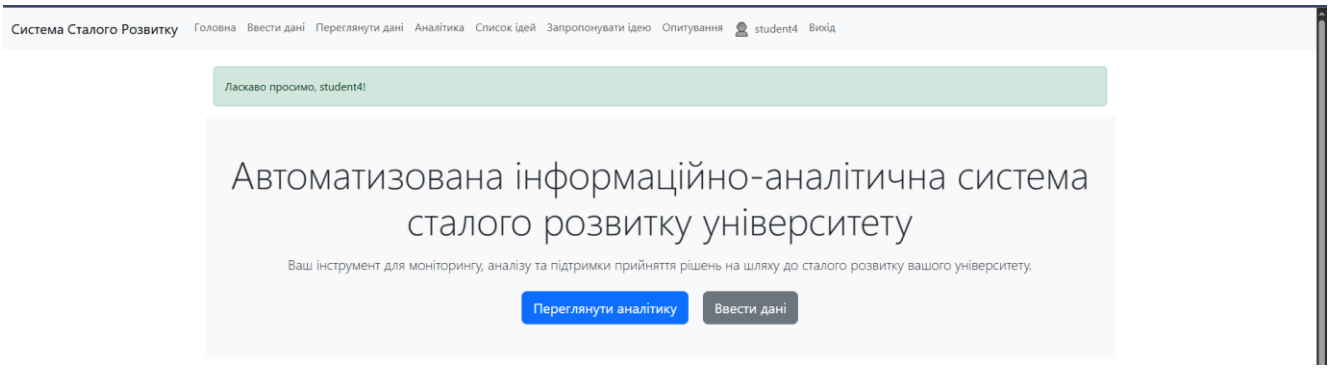


Рисунок 4.26 – Тестування форми входу

Наступним протестуємо процес завантаження даних до системи , через розглянуті раніше CSV-файли , для цього створимо CSV-файл який буде відповідати вимогам які прописані на самій сторінці завантаження цих файлів(рис. 4.27). А саме щоб у нього були відповідні назви стовпців першим рядком, а потім йшли числові значення у потрібному порядку для нормального розрахунку показників на сторінці з аналітикою.

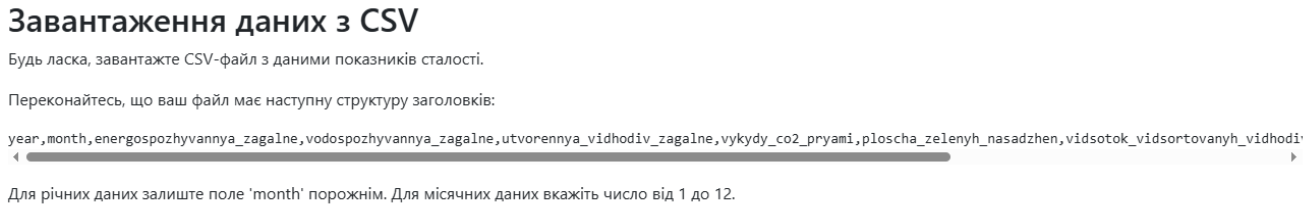


Рисунок 4.27 – Необхідна структура CSV-файлу для завантаження

Створений CSV-файл(рис.4.28) завантажуємо до системи і якщо все добре то отримаємо відповідне повідомлення і на сторінці з даними та аналітикоб будуть доступні ці дані та відповідна аналітика до них

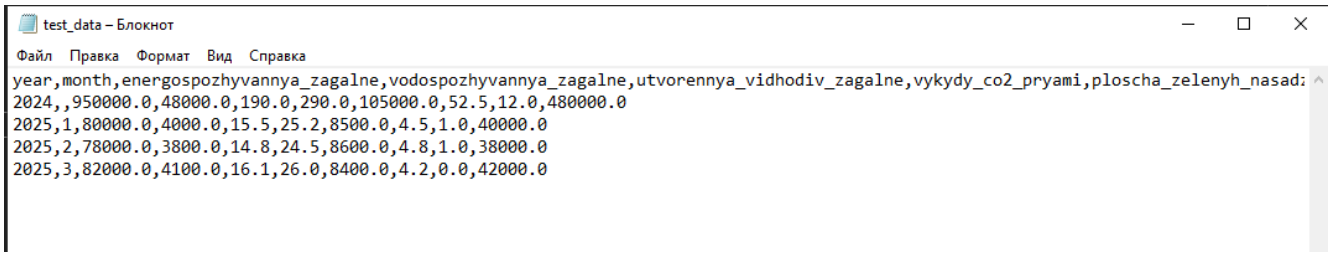


Рисунок 4.28 – Тестові дані

Висновки до 4 розділу

Розроблена система складається з багатьох файлів і директорій , кожний файл виконує свої завдання , є файли для налаштування проєкту вцілому , є файли для налаштування бази даних системи , є директорія в якій зберігаються усі HTML

шаблони системи і так далі , це складна але водночас зрозуміло структурована система в якій все виконує свою роботу окремо. Користуватися системою дуже легко , інтерфейс простий і зрозумілий , без зайвих деталей , на самому початку не є доступним увесь функціонал системи , для цього потрібна реєстрація про що система одразу каже користувачу на головному екрані , просивши його зареєструватися. Тестування показало що всі ключові моменти і сценарії використання системи працюють без помилок.

ВИСНОВКИ

В ході виконання кваліфікаційної роботи була розроблена автоматизована інформаційно-аналітична система по визначенню пріоритетів сталого розвитку університетської системи освіти , яка дозволяє збирати, зберігати, оброблювати та візуалізовувати дані, пов'язані з різноманітними аспектами сталого розвитку. Також розроблена система має багато шляхів взаємодіїї користувачів між собою , це опитування , спеціальний чат , та інструменти для додавання ідей та ініціатив.

У першому розділі були розглянуті схожі системи та публікаціїї пов’язані з темою сталого розвитку у відношенні університетів та освіти в цілому. Серед розглянутих систем була система STARS , потім система GreenMetric World University Rankings і останнім 3 аналогом була не зовсім система а рейтинг на основі сталого розвитку - Times Higher Education (THE) Impact Rankings.

Другий розділ був присвячений дослідженням усіх методів на інструментів які необхідні для реалізаціїї данної системи. Це і методи для аналітичного модуля системи і мова програмування вцілому , для цієї системи була обрана Python з її відомим фреймворком Django. Також було не обійись без HTML та CSS , так як логіка це одне , відображати сторінки якось потрібно. Для зберігання даних була використана база даних PostgreSQL.

У третьому розділі було розглянуто структура системи , було описано вхідні дані які використовує система для аналізу показника сталого розвитку. Було описано з чого взагалі складається реалізована система , всі її модулі та яку роль відіграє кожен модуль у цій системі та як вони реалізовані.

У останньому четвертому розділі демонструється програмна реалізація системи , для цього були розгялнуті всі основні файли системи на прикладах шматків коду з них було показано за що відповідає кожен файл і яка в нього структура. Також у цьому розділі описано як користуватися системою та було проведено тестування основних функцій системи , серед них процес реєстрації та входу у систему.

Вцілому можна сказати що реалізована система повністю відповідає усім вимогам та виконує усі реалізовані в ній задачі чітко і без помилок , але звісно для повноцінної її справної роботи потрібно дотримуватися певних правил , наприклад під час реєстрації або завантаження файлів для аналізу , це вже залежить від користувача , він повинен чітко дотримуватись інструкцій та вказівок які прописані у системі.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1) Бєлянінова, Г. Г., Драз, О. М., Ткачов, В. М., & Чала, Л. Е. (2021). Особливості досягнення Цілей сталого розвитку у профільних закладах вищої освіти на прикладі Харківського національного університету радіоелектроніки.

2) Воробієнко, П. П. (2023). Роль освіти в досягненні цілей сталого розвитку ООН. Вісник Національної академії педагогічних наук України, 5(1), 1-8.

3) Левченко, І. В. (2023). Адаптація екологічного законодавства України до стандартів сталого розвитку ООН (Doctoral dissertation, Поліський національний університет, м. Житомир).

4) STARS, Sustainability Tracking Assessment & Rating System. URL: https://stars.aashe.org/ (Дата звернення 25.04.2025).

5) UI GreenMetric World University Rankings - Ui.ac.id. URL: https://greenmetric.ui.ac.id/ (Дата звернення 25.04.2025).

6) University Impact Rankings 2024. URL: https://www.timeshighereducation.com/impactrankings (Дата звернення 25.04.2025).

7) Leal Filho, W., Manolas, E., & Pace, P. (2015). The future we want: Key issues on sustainable development in higher education after Rio and the UN decade of education for sustainable development. International Journal of Sustainability in Higher Education, 16(1), 112-129.

8) Caeiro, S., Leal Filho, W., Jabbour, C., & Azeiteiro, U. (2013). Sustainability assessment tools in higher education institutions: mapping trends and good practices around the world. Springer International Publishing.

9) Ebzeeva, Y. N., & Smirnova, Y. B. (2023). Contemporary trends in educational policy: UNESCO higher education roadmap. RUDN Journal of Sociology, 23(2), 329-337.

10) Blake, J., Sterling, S., & Goodson, I. (2013). Transformative learning for a sustainable future: An exploration of pedagogies for change at an alternative college. Sustainability, 5(12), 5347-5372.

11) Армен, Л., Бернацька, Н., Бородіна, К., Бутова, В., Горшкова, Л., Грицай, Н., ... & Шибанова, А. (2019). Автори.

12) Радкевич, В. О., Бородієнко, О. В., & Кравець, С. Г. (2021). Професійна (професійно-технічна) освіта України в контексті євроінтеграційних процесів (порівняльний аналіз).

13) Биков, В. Ю., Спірін, О. М., & Пінчук, О. П. (2017). Проблеми та завдання сучасного етапу інформатизації освіти. Наукове забезпечення розвитку освіти в Україні: актуальні проблеми теорії і практики (до 25-річчя НАПН України), 191-198.

14) Міністерство освіти і науки України. (2017). Національна доповідь «Цілі сталого розвитку: Україна». Отримано з https://www.kmu.gov.ua/storage/app/sites/1/natsionalna-dopovid-csr-Ukrainy.pdf

15) Кобєлєва, Т. О., & Погорєлова, Т. О. (2025). Розвиток системи вищої освіти в Україні.

16) Azarova, I. (2019). ЗАКОНОДАВЧІ ТА НОРМАТИВНІ МЕТОДИ ОЦІНЮВАННЯ ЕКОЛОГІЇ В УКРАЇНІ. Вісник Львівського державного університету безпеки життєдіяльності, 19, 115-121.

17) Защепкіна, Н., Рудницький, Р., Наконечний, О., & Маркіна, О. (2021). ЗАСТОСУВАННЯ МЕТОДІВ АГРЕГАЦІЇ ДАНИХ В ІНФОРМАЦІЙНІЙ СИСТЕМІ КОНТРОЛЮ НАВЧАЛЬНОГО ПРОЦЕСУ. *Measuring and computing devices in technological processes*, (2), 12-20.

18) Шингалов, Д. В., Мелешко, Є. В., & Босько, В. В. (1980). Методи нормалізації даних для моделей машинного навчання. Редакційна колегія, 68, 187.

19) Шкабара, В. С. (2016). Розробка адаптивного сайту з використанням Bootstrap. Актуальні питання сучасної інформатики, 2, 122-128.

20) Створення динамічних графіків та діаграм з використанням Chart.js : вебсайт. URL: https://it-rating.ua/stvorennya-dinamichnih-grafikiv-ta-diagram-z-vikoristannyam-chartjs (дата звернення: 15.05.2025).

21) Кучер, В. В. (2017). Особливості використання фреймворку Django для написання веб-додатків. Актуальні питання сучасної інформатики, (4), 19-25.

22) Васильєв, О. М. (2022). Програмування мовою Python. Bohdan Books.

23) Pandas 2.3.0 documentation: вебсайт. URL: https://pandas.pydata.org/docs/ (дата звернення: 15.05.2025).

24) Чумак, В. (2020). Оптимізація роботи СКБД PostgreSQL.

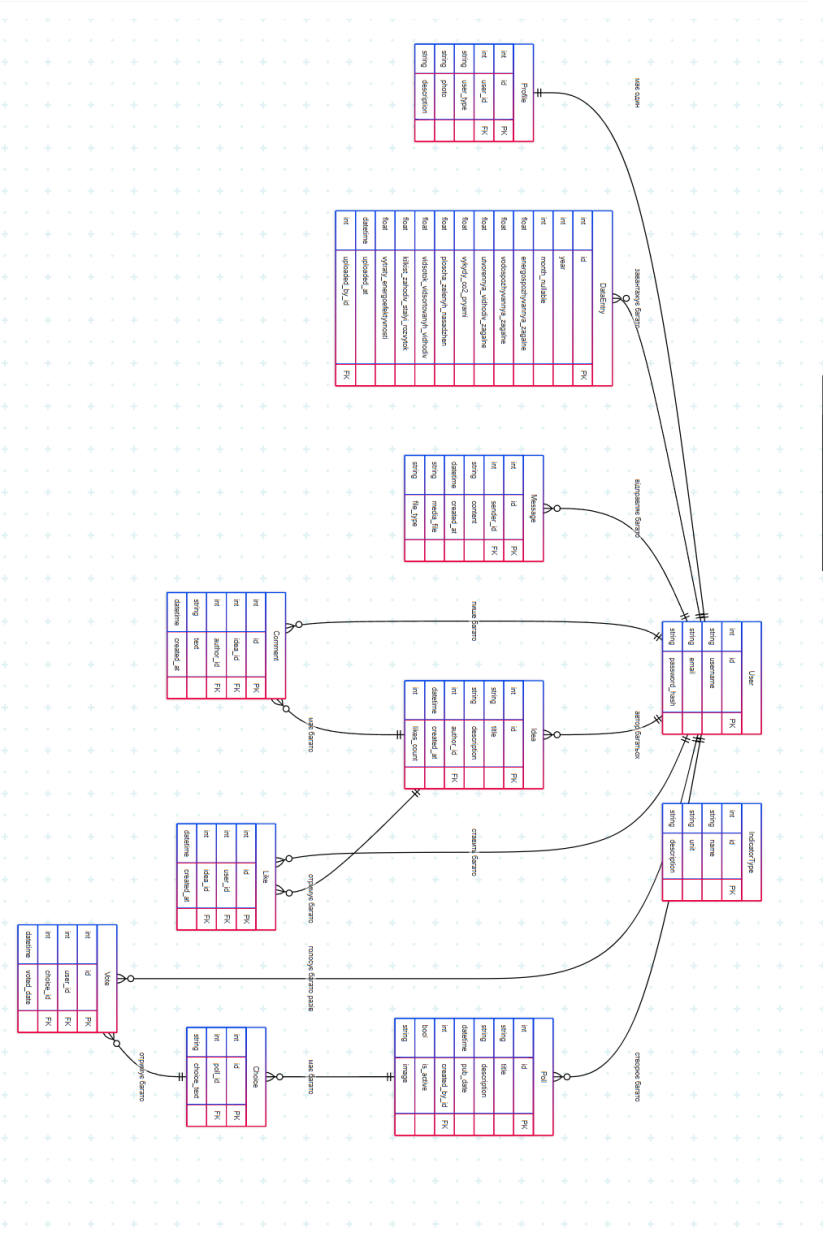
25) Нескородєва, А. Р., & Данильчук, О. М. (2022). Використання бібліотеки Matplotlib для візуалізації даних. Прикладні інформаційні технології, 115-119.

26) Візуалізація даних у Python із Seaborn: вебсайт . URL: https://codelabsacademy.com/uk/blog/data-visualization-in-python-with-seaborn/ (дата звернення:16.05.2025).

27) CSV-файл: вебсайт. URl: https://products.aspose.com/gis/uk/net/gis-formats/csv/ (дата звернення: 14.05.2025).

ДОДАТОК А

Діаграма класів



ДОДАТОК Б
Ключові файли системи

Файл models.py:

```
# core/models.py

from django.db import models
from django.contrib.auth.models import User # Імпортуємо стандартну модель користувача
from django.db.models.signals import post_save
from django.dispatch import receiver
from django.utils import timezone

# Модель для опису ТИПУ показника сталого розвитку (напр., "Споживання води")
class IndicatorType(models.Model):
    name = models.CharField(max_length=200, verbose_name="Назва показника")
    unit = models.CharField(max_length=50, verbose_name="Одиниця виміру")
    description = models.TextField(blank=True, null=True, verbose_name="Опис")

    class Meta:
        verbose_name = "Тип показника"
        verbose_name_plural = "Типи показників"

    def __str__(self):
        return f"{self.name} ({self.unit})"

# Модель для імпортованих даних показників
class DataEntry(models.Model):
    year = models.IntegerField(verbose_name="Рік")
    month = models.IntegerField(null=True, blank=True, verbose_name="Місяць (для місячних даних)") # null=True,
blank=True для річних даних

    # Показники (використовуємо "чисті" назви з CSV)
    energospozhyvannya_zagalne = models.FloatField(verbose_name="Енергоспоживання (загальне)")
    vodospozhyvannya_zagalne = models.FloatField(verbose_name="Водоспоживання (загальне)")
    utvorennya_vidhodiv_zagalne = models.FloatField(verbose_name="Утворення відходів (загальне)")
    vykydy_co2_pryami = models.FloatField(verbose_name="Викиди CO2 (прямі)")
    ploscha_zelenyh_nasadzhen = models.FloatField(verbose_name="Площа зелених насаджень на території")
    vidsotok_vidsortovanyh_vidhodiv = models.FloatField(verbose_name="Відсоток відсортованих відходів")
    kilkist_zahodiv_stalyi_rozvytok = models.FloatField(verbose_name="Кількість заходів зі сталого розвитку")
```

```
    vytraty_energoefektyvnosti = models.FloatField(verbose_name="Витрати на заходи з енергоефективності")

    # Автоматично додаємо дату створення запису в БД
    uploaded_at = models.DateTimeField(auto_now_add=True)
    uploaded_by = models.ForeignKey(User, on_delete=models.SET_NULL, null=True, blank=True,
verbose_name="Завантажено користувачем")


    class Meta:
        verbose_name = "Запис даних"
        verbose_name_plural = "Записи даних"
        # Гарантуємо унікальність запису для певного року/місяця
        # Якщо month = null, то запис унікальний по року.
        # Якщо month заповнений, то унікальний по року і місяцю.
        # Цей унікальний обмеження може бути складним для null=True в month.
        # Простіше обробляти перевірку у view, якщо дозволяємо кілька річних записів,
        # або примусово унікальність, якщо один запис на рік.
        # Для простоти, давайте спочатку не додавати unique_together для month=null
        # і керувати цим на рівні логіки імпорту.
        # For now, let's just make year and month unique together.
        # This means only one entry per year for annual data (month=None)
        # and one entry per month for monthly data.
        unique_together = ('year', 'month')


    def __str__(self):
        if self.month:
            return f"Дані за {self.month:02d}.{self.year}"
        return f"Дані за {self.year} рік"


# ##############################################################################
# Модель Профілю користувача (ОБ'ЄДНАНО ДЛЯ ДНЯ 9)
# Зберігає додаткову інформацію про користувача, включаючи тип, фото та опис
# ##############################################################################
class Profile(models.Model):
    # Зв'язок One-to-One з моделлю User.
    # related_name='profile' дозволяє звертатись до профілю з об'єкта користувача як user.profile
    user = models.OneToOneField(User, on_delete=models.CASCADE, related_name='profile')
```

```
    # Поле для Типу користувача (з твого існуючого коду)
    USER_TYPE_CHOICES = [
        ('student', 'Студент'),
        ('faculty', 'Викладач'),
        ('staff', 'Співробітник'),
        ('admin', 'Адміністрація'),
        ('other', 'Інше'),
    ]
    user_type = models.CharField(
        max_length=20,
        choices=USER_TYPE_CHOICES,
        default='student',
        verbose_name="Тип користувача"
    )
    # TODO: Якщо будемо додавати підрозділи, тут знадобиться ForeignKey до моделі Підрозділу

    # Поле для фотографії профілю (з плану Дня 9)
    # Upload_to вказує підпапку в MEDIA_ROOT, куди будуть завантажуватись файли
    # null=True, blank=True - робить поле необов'язковим
    photo = models.ImageField(upload_to='profile_pics/', null=True, blank=True, verbose_name="Фото профілю")

    # Поле для опису або біографії (текстове, з плану Дня 9)
    description = models.TextField(max_length=500, blank=True, null=True, verbose_name="Опис")
    # TODO: Додати інші поля профілю, якщо потрібно (напр., Університет, факультет тощо)



    class Meta:
        verbose_name = "Профіль користувача"
        verbose_name_plural = "Профілі користувачів"

    def __str__(self):
        # Використовуємо get_user_type_display() для отримання читабельної назви типу
        return f"Профіль користувача {self.user.username} ({self.get_user_type_display()})"



class Message(models.Model):
    """Модель для повідомлень у груповому чаті/стіні спілкування."""
    sender = models.ForeignKey(User, on_delete=models.CASCADE, related_name='sent_messages',
verbose_name="Відправник")
    content = models.TextField(blank=True, verbose_name="Текст повідомлення")
```

```
    created_at = models.DateTimeField(auto_now_add=True, verbose_name="Дата та час")

    media_file = models.FileField(upload_to='chat_media/', blank=True, null=True, verbose_name="Медіа-файл")
    file_type = models.CharField(max_length=10, blank=True, null=True, verbose_name="Тип файлу")

    class Meta:
        verbose_name = "Повідомлення чату"
        verbose_name_plural = "Повідомлення чату"
        ordering = ['created_at']

    def __str__(self):
        return f"Повідомлення від {self.sender.username} ({self.created_at.strftime('%Y-%m-%d %H:%M')})"

    def save(self, *args, **kwargs):
        if self.media_file and not self.file_type:
            extension = self.media_file.name.split('.')[-1].lower()
            if extension in ['jpg', 'jpeg', 'png', 'gif', 'bmp', 'webp']:
                self.file_type = 'image'
            elif extension in ['mp4', 'webm', 'ogg']:
                self.file_type = 'video'
            elif extension in ['mp3', 'wav', 'ogg']:
                self.file_type = 'audio'
            elif extension in ['pdf', 'doc', 'docx', 'txt', 'xls', 'xlsx']:
                self.file_type = 'document'
            else:
                self.file_type = 'other'
        super().save(*args, **kwargs)

# ###########################################################################
# Signal для автоматичного створення профілю при створенні нового користувача (з плану Дня 9)
# Це гарантує, що кожен User матиме пов'язаний Profile
# ###########################################################################
from django.db.models.signals import post_save # Імпортуємо сигнал
from django.dispatch import receiver # Імпортуємо декоратор receiver

# Функція, яка буде викликатись ПІСЛЯ збереження (створення) об'єкта User
@receiver(post_save, sender=User)
def create_user_profile(sender, instance, created, **kwargs):
    if created:
        # Якщо користувача (instance) тільки що створили (created=True),
```

```
        # створюємо для нього об'єкт Profile
        # При створенні автоматично встановлюється user=instance
        Profile.objects.create(user=instance)

# Цей сигнал може бути корисним, якщо редагувати User в адмінці,
# але якщо основне створення профілю йде через create_user_profile,
# він може бути некритичним або реалізованим інакше.
# @receiver(post_save, sender=User)
# def save_user_profile(sender, instance, **kwargs):
#     try:
#         # Намагаємось зберегти пов'язаний профіль
#         instance.profile.save()
#     except Profile.DoesNotExist:
#         # Якщо профілю чомусь немає (напр., старий користувач без профілю до застосування міграцій),
#         # можемо його створити або проігнорувати залежно від логіки.
#         # Якщо create_user_profile працює, цей except не має викликатись для нових користувачів.
#         pass

# ########################################################################
# Модель Ідеї / Ініціативи (ДОДАНО ДЛЯ ДНЯ 10)
# Зберігає інформацію про запропоновані користувачами ідеї
# ########################################################################
class Idea(models.Model):
    title = models.CharField(max_length=200, verbose_name="Заголовок ідеї") # Заголовок ідеї
    description = models.TextField(verbose_name="Опис ідеї") # Детальний опис
    # Зв'язок Many-to-One: одна ідея належить ОДНОМУ автору (користувачу)
    # on_delete=models.CASCADE - якщо видалити користувача, видаляться і його ідеї
    author = models.ForeignKey(User, on_delete=models.CASCADE, related_name='ideas', verbose_name="Автор")
    # Автоматично фіксуємо дату і час створення ідеї
    created_at = models.DateTimeField(auto_now_add=True, verbose_name="Дата створення")
    # TODO: Додати інші поля пізніше (напр., статус, категорія, теги, поле для коментарів/обговорень)
    likes_count = models.IntegerField(default=0, verbose_name="Кількість лайків")

    class Meta:
        verbose_name = "Ідея"
        verbose_name_plural = "Ідеї"
        ordering = ['-created_at'] # Сортуємо за замовчуванням від нових до старих

    def __str__(self):
        return self.title # У представленні об'єкта показуємо заголовок ідеї
```

```
# ########################################################################
# Модель Коментаря (ДОДАНО ДЛЯ ДНЯ 11)
# Зберігає коментарі до ідей
# ########################################################################
class Comment(models.Model):
    # Зв'язок Many-to-One: багато коментарів належить ОДНІЙ ідеї
    # related_name='comments' дозволяє отримати всі коментарі до ідеї через idea.comments.all()
    idea = models.ForeignKey(Idea, on_delete=models.CASCADE, related_name='comments', verbose_name="Ідея")
    # Зв'язок Many-to-One: багато коментарів написані ОДНИМ автором (користувачем)
    author = models.ForeignKey(User, on_delete=models.CASCADE, related_name='comments', verbose_name="Автор коментаря")
    # Текст коментаря
    text = models.TextField(verbose_name="Текст коментаря")
    # Автоматично фіксуємо дату і час створення коментаря
    created_at = models.DateTimeField(auto_now_add=True, verbose_name="Дата створення")

    class Meta:
        verbose_name = "Коментар"
        verbose_name_plural = "Коментарі"
        ordering = ['created_at'] # Сортуємо коментарі від старих до нових

    def __str__(self):
        # Показуємо початок тексту коментаря та автора
        return f"Коментар до '{self.idea.title[:20]}...' від {self.author.username}"


# ########################################################################
# Моделі опитувань  (ДОДАНО ДЛЯ ДНЯ 12)
# ########################################################################
class Poll(models.Model):
    """Модель для представлення опитування."""
    title = models.CharField(max_length=255, verbose_name="Заголовок опитування")
    description = models.TextField(blank=True, null=True, verbose_name="Опис опитування")
    pub_date = models.DateTimeField(default=timezone.now, verbose_name="Дата публікації")
    created_by = models.ForeignKey(User, on_delete=models.SET_NULL, null=True, related_name='polls_created', verbose_name="Створено")
    is_active = models.BooleanField(default=True, verbose_name="Активне") # Чи активне опитування для голосування
    image = models.ImageField(upload_to='poll_images/', blank=True, null=True, verbose_name="Зображення до
```

```
опитування")

    class Meta:
        verbose_name = "Опитування"
        verbose_name_plural = "Опитування"
        ordering = ['-pub_date'] # Сортувати за датою публікації, від нових до старих

    def __str__(self):
        return self.title

class Choice(models.Model):
    """Модель для варіантів відповідей у опитуванні."""
    poll = models.ForeignKey(Poll, on_delete=models.CASCADE, related_name='choices', verbose_name="Опитування")
    choice_text = models.CharField(max_length=200, verbose_name="Текст варіанта відповіді")

    class Meta:
        verbose_name = "Варіант відповіді"
        verbose_name_plural = "Варіанти відповідей"

    def __str__(self):
        return self.choice_text

class Vote(models.Model):
    """Модель для запису голосу користувача."""
    user = models.ForeignKey(User, on_delete=models.CASCADE, verbose_name="Користувач")
    choice = models.ForeignKey(Choice, on_delete=models.CASCADE, verbose_name="Обраний варіант")
    voted_date = models.DateTimeField(default=timezone.now, verbose_name="Дата голосування")

    class Meta:
        verbose_name = "Голос"
        verbose_name_plural = "Голоси"
        # Обмеження, щоб один користувач міг проголосувати лише один раз в одному опитуванні.
        # Це обмеження ми будемо перевіряти у View, бо уникати дублювання у unique_together для зв'язків
складніше
        # unique_together = ('user', 'choice') # <- ЦЕ НЕ ТРЕБА, БУДЕМО ПЕРЕВІРЯТИ У В'Ю

    def __str__(self):
        return f"{self.user.username} проголосував за {self.choice.choice_text} у опитуванні {self.choice.poll.title}"
```

```
# Модель для лайків (додавання до ідей)
class Like(models.Model):
    user = models.ForeignKey(User, on_delete=models.CASCADE, related_name='likes')
    idea = models.ForeignKey('Idea', on_delete=models.CASCADE, related_name='likes_on_idea')
    created_at = models.DateTimeField(auto_now_add=True)

    class Meta:
        # Гарантує, що один користувач може поставити лайк лише одній ідеї один раз
        unique_together = ('user', 'idea')
        verbose_name = "Лайк"
        verbose_name_plural = "Лайки"

    def __str__(self):
        return f"Лайк від {self.user.username} до ідеї '{self.idea.title}'"
```

Файл views.py:

```
# core/views.py

import csv

import io

from django.shortcuts import render, redirect , get_object_or_404 # Функції для рендерингу шаблонів та перенаправлення

from django.contrib.auth.decorators import login_required # Декоратор для обмеження доступу

from django.contrib import messages # Для виведення повідомлень користувачу (напр., про успіх)

from django.contrib.auth.models import User # Імпортуємо стандартну модель користувача

# ДОДАНО для управління користувачами

from django.contrib.auth import authenticate, login, logout # Імпорт для логіну/логауту

from django.contrib.auth.forms import AuthenticationForm, UserCreationForm # Форми для логіну/реєстрації

from django.conf import settings # Для доступу до налаштувань з settings.py

from .forms import IdeaForm , CommentForm , PollForm, UserRegistrationForm , CSVUploadForm , MessageForm # Імпортуємо нашу форму

from .models import DataEntry, IndicatorType, Profile, Idea , Comment , Poll, Choice, Vote , Like , Message# Імпортуємо модель даних

# ЗМІНЕНО: Імпортуємо гнучку функцію розрахунку замість річної

from django.contrib.auth.decorators import login_required

from django.db.models import Count , Q # Для підрахунку голосів
```

```
from django.http import HttpResponseRedirect , JsonResponse# Для перенаправлення

from django.urls import reverse # Для генерації URL-адрес

from .constants import INDICATOR_CONFIG

from .analytics import calculate_sustainability_index_flexible




# ================================================================================

# НОВИЙ МОДУЛЬ: Груповий чат / Стіна спілкування (Chat Room)

# ================================================================================


@login_required # Доступно тільки для авторизованих користувачів

def chat_room_view(request):

    """

    Відображає груповий чат, дозволяє надсилати нові повідомлення.

    """

    messages = Message.objects.select_related('sender__profile').order_by('created_at')

    # .select_related('sender__profile') оптимізує запити до бази даних,

    # завантажуючи профілі користувачів заздалегідь.


    if request.method == 'POST':

        form = MessageForm(request.POST, request.FILES)

        if form.is_valid():

            message = form.save(commit=False) # Не зберігаємо одразу, бо треба додати відправника

            message.sender = request.user

            message.save()

            # Після успішного відправлення перенаправляємо на ту ж сторінку,

            # щоб уникнути повторного відправлення при оновленні сторінки.

            return redirect('chat_room')

    else:
```

```
        form = MessageForm() # Порожня форма для GET-запиту


    context = {
        'messages': messages,
        'form': form,
    }
    return render(request, 'core/chat_room.html', context)



# ДОДАНО View для головної сторінки (ми її створимо пізніше)
def index(request):
    # Ця view буде відображати нашу головну сторінку index.html
    # TODO: Можливо, передавати якісь дані на головну сторінку пізніше
    return render(request, 'core/index.html') # Потрібно створити файл index.html



# ############################################################################
# View для сторінки профілю користувача (ДОДАНО ДЛЯ ДНЯ 9) - ПОВНИЙ КОД ФУНКЦІЇ
# ############################################################################
@login_required # <- Цей декоратор перевіряє авторизацію
def profile_view(request):
    # request.user автоматично містить об'єкт поточного користувача завдяки @login_required
    user = request.user

    # Отримуємо пов'язаний об'єкт Профілю. Завдяки сигналу, він має існувати для увійшовшого користувача.
    # Звертаємось через related_name='profile', який ми визначили в моделі Profile
    # Можна додати try...except Profile.DoesNotExist для безпеки, але зазвичай user.profile працює
    # try:
    #    profile = user.profile
```

```
    # except Profile.DoesNotExist:
    #     # Наприклад, перенаправити на головну або сторінку помилки
    #     # return redirect('index')
    #     profile = None # Або створити новий профіль, якщо його немає



    profile = user.profile # <- Отримуємо об'єкт профілю

    # Готуємо дані для передачі в шаблон
    context = {
        'user': user,      # Сам об'єкт користувача (містить username, email тощо)
        'profile': profile, # Об'єкт профілю (містить photo, description, user_type)
    }

    # Рендеримо шаблон profile.html, передаючи контекст
    return render(request, 'core/profile.html', context)



# View для введення даних CSV
@login_required # Доступ тільки для авторизованих
def upload_csv_view(request):
    # Перевірка дозволів: тільки для адміністраторів (is_staff)
    if not request.user.is_staff:
        messages.warning(request, "У вас немає дозволу для завантаження CSV-файлів.")
        return redirect('home') # Перенаправлення на домашню сторінку або іншу відповідну

    if request.method == 'POST':
        form = CSVUploadForm(request.POST, request.FILES)
        if form.is_valid():
```

```
    csv_file = request.FILES['csv_file']

    # Перевіряємо, чи це CSV-файл
    if not csv_file.name.endswith('.csv'):
        messages.error(request, "Завантажувати можна тільки CSV-файли.")
        return redirect('upload_csv')

    # Зчитуємо файл у текстовому режимі
    data_set = csv_file.read().decode('utf-8')
    io_string = io.StringIO(data_set)
    next(io_string) # Пропускаємо перший рядок (заголовки стовпців)

    # Визначаємо заголовки, щоб зіставити їх з полями моделі
    # Важливо: використовуємо "чисті" назви полів як у моделі DataEntry
    field_names = [
        'year', 'month', 'energospozhyvannya_zagalne', 'vodospozhyvannya_zagalne',
        'utvorennya_vidhodiv_zagalne', 'vykydy_co2_pryami', 'ploscha_zelenyh_nasadzhen',
        'vidsotok_vidsortovanyh_vidhodiv', 'kilkist_zahodiv_stalyi_rozvytok',
        'vytraty_energoefektyvnosti'
    ]

    # Додаємо список для збору нових об'єктів для масового створення
    new_data_entries = []
    updated_data_entries = [] # Для оновлення існуючих записів

    csv_reader = csv.reader(io_string, delimiter=',')

    # Ітеруємося по кожному рядку в CSV
    for row in csv_reader:
```

```
        # Пропускаємо порожні рядки
        if not row or all(not cell.strip() for cell in row):
            continue

        data = {}
        # Зіставляємо дані з рядка з іменами полів
        for i, value in enumerate(row):
            if i < len(field_names): # Перевіряємо, щоб не вийти за межі списку полів
                field_name = field_names[i]
                # Обробляємо порожні значення для 'month'
                if field_name == 'month':
                    try:
                        data[field_name] = int(value) if value.strip() else None
                    except ValueError: # Якщо значення не є числом і не порожнє
                        messages.error(request, f"Помилка формату місяця в рядку: {row}. Місяць має бути числом або порожнім.")
                        return redirect('upload_csv')
                else:
                    try:
                        # Конвертуємо числові значення, ігноруємо порожні
                        data[field_name] = float(value) if value.strip() else None
                    except ValueError: # Якщо значення не є числом і не порожнє
                        messages.error(request, f"Помилка формату числа для '{field_name}' в рядку: {row}. Переконайтесь, що це числове значення.")
                        return redirect('upload_csv')

        # Перевіряємо, чи є обов'язкові поля year
        if 'year' not in data or data['year'] is None:
            messages.error(request, f"Відсутній або некоректний рік в рядку: {row}. Рядок пропущено.")
            continue # Пропускаємо цей рядок
```

```
    # Перевіряємо, чи запис для даного року/місяця вже існує
    try:
        existing_entry = DataEntry.objects.get(year=data['year'], month=data['month'])
        # Якщо існує, оновлюємо його
        for key, value in data.items():
            setattr(existing_entry, key, value)
        updated_data_entries.append(existing_entry)
        month_str = f".{data['month']}" if data['month'] else ""
        messages.info(request, f"Дані за {data['year']}{month_str} оновлено.")

    except DataEntry.DoesNotExist:
        # Якщо не існує, створюємо новий об'єкт DataEntry
        # Видаляємо null-значення, щоб DataEntry.objects.create працював коректно
        # (не передавайте None для FloatField, якщо не очікується)
        entry_data = {k: v for k, v in data.items() if v is not None}
        new_data_entries.append(DataEntry(**entry_data, uploaded_by=request.user))

# Масове створення нових записів (якщо є)
if new_data_entries:
    DataEntry.objects.bulk_create(new_data_entries)
    messages.success(request, f"Успішно завантажено {len(new_data_entries)} нових записів даних.")

# Масове оновлення існуючих записів (якщо є)
if updated_data_entries:
    # Оновлюємо поля вручну, бо bulk_update потребує список полів
    # Тут можна оптимізувати, але для початку так
    for entry in updated_data_entries:
        entry.save() # Це буде викликати UPDATE запит для кожного оновленого об'єкта
```

```
            messages.success(request, f"Успішно оновлено {len(updated_data_entries)} існуючих записів даних.")

        if not new_data_entries and not updated_data_entries:
            messages.info(request, "Не знайдено нових або оновлених даних у CSV-файлі.")

        return redirect('upload_csv') # або інша сторінка після успішного завантаження
    else:
        messages.error(request, "Будь ласка, виберіть CSV-файл.")
else:
    form = CSVUploadForm()

return render(request, 'core/upload_csv.html', {'form': form})


# View для перегляду списку внесених даних
@login_required
def view_sustainability_data(request):
    available_years = DataEntry.objects.order_by('-year').values_list('year', flat=True).distinct()

    selected_year = request.GET.get('year')
    if not selected_year and available_years.exists():
        selected_year = available_years.first()
    elif selected_year:
        selected_year = int(selected_year)

    data_entries = DataEntry.objects.all() # Починаємо з усіх даних

    if selected_year:
        data_entries = data_entries.filter(year=selected_year)
```

```
    # Отримуємо доступні місяці, виключаючи річні дані для списку MonthSelect
    available_months = DataEntry.objects.filter(year=selected_year)\
                                .exclude(month__isnull=True)\
                                .order_by('month')\
                                .values_list('month', flat=True).distinct()
else:
    available_months = []



selected_month_param = request.GET.get('month') # Отримуємо параметр з URL


if selected_month_param:
    if selected_month_param == "all_months":
        # Вибрано "Усі місяці": показуємо всі місячні дані (month IS NOT NULL) за обраний рік
        data_entries = data_entries.filter(month__isnull=False)
        selected_month = selected_month_param # Для відображення в select
    elif selected_month_param == "annual_data":
        # Вибрано "Річні дані": показуємо тільки річні дані (month IS NULL) за обраний рік
        data_entries = data_entries.filter(month__isnull=True)
        selected_month = selected_month_param # Для відображення в select
    else:
        # Вибрано конкретний місяць (1-12)
        try:
            selected_month = int(selected_month_param)
            data_entries = data_entries.filter(month=selected_month)
        except ValueError:
            # Якщо month_param не число і не "all_months"/"annual_data", ігноруємо його
            selected_month = None
else:
```

```
    # Якщо month_param взагалі не передано (або він порожній), це дефолтний стан для обраного року.
    # Згідно вашої логіки: це має бути "Усі місяці", тобто month__isnull=False.
    if selected_year: # Якщо рік обрано
        data_entries = data_entries.filter(month__isnull=False)
        selected_month = "all_months" # Встановлюємо як обране за замовчуванням у контексті
    else:
        selected_month = None # Якщо рік не обрано, місяць теж не має бути обраний

  context = {
    'data_entries': data_entries,
    'available_years': available_years,
    'selected_year': selected_year,
    'available_months': available_months,
    'selected_month': selected_month, # Тепер може бути "all_months", "annual_data", або число
    "INDICATOR_CONFIG": {
        field: props['verbose_name'] for field, props in INDICATOR_CONFIG.items()
    }
  }
  return render(request, 'core/sustainability_data.html', context)
```

Файл analyticks.py:

```
# core/analytics.py
from .models import DataEntry
from collections import defaultdict # Може бути корисно для групування даних у словник
from django.db.models.functions import ExtractYear, ExtractMonth # <- ДОДАНО ExtractMonth
from django.db.models import Q , F , Sum , Avg # <- ДОДАНО Q
import calendar # <- ДОДАНО імпорт модуля calendar
from .constants import INDICATOR_CONFIG
```

```
# ##############################################################################
# ФУНКЦІЯ ГНУЧКОГО РОЗРАХУНКУ ІНТЕГРАЛЬНОГО ПОКАЗНИКА ЗА ПЕРІОДАМИ
# ##############################################################################

# ДОДАНО/ЗМІНЕНО: Функція для отримання назви місяця за номером (1=Січень)
def month_name_ua(month_num):
    """Повертає назву місяця українською за його номером."""
    # calendar.month_name[month_num] повертає назву місяця
    # Важливо: для коректної локалізації назв місяців може знадобитись налаштування локалі в системі/програмі
    # Але для більшості сучасних систем це працює коректно
    try:
        # calendar.month_name - це список, індексація починається з 1 для місяців
        return calendar.month_name[month_num]
    except IndexError:
        return f"Місяць {month_num}"



# ##############################################################################
# ФУНКЦІЯ ГНУЧКОГО РОЗРАХУНКУ ІНТЕГРАЛЬНОГО ПОКАЗНИКА ЗА ПЕРІОДАМИ
# АДАПТОВАНО ДЛЯ НОВОЇ МОДЕЛІ DataEntry ТА ЇЇ ПОЛІВ
# ##############################################################################



def calculate_sustainability_index_flexible(period_type='year', year=None, month=None):
    """
    Розраховує інтегральний показник сталого розвитку та індивідуальні бали
    за обраним типом періоду (рік, місяць) та роком/місяцем.
```

```
    :param period_type: 'year' для річних агрегацій або 'month' для місячних/помісячних.
    :param year: Обраний рік (int).
    :param month: Обраний місяць (int 1-12), 0 для month__isnull=True, None для month__isnull=False (всі місяці).
    """
    queryset = DataEntry.objects.all()

    if year:
        try:
            year = int(year)
            queryset = queryset.filter(year=year)
        except (ValueError, TypeError):
            return {'periods_data': [], 'verbose_indicator_names': {}}

    # Фільтрація за місяцем відповідно до переданого значення
    if month is not None:
        if month == 0:  # Означає "річні дані" (month__isnull=True)
            queryset = queryset.filter(month__isnull=True)
        # Якщо month це None, ми не фільтруємо month__isnull, що означає "всі місяці" (month IS NOT NULL)
        # тобто, month__isnull=False.
        # Якщо month це 1-12, то фільтруємо за конкретним місяцем.
        elif isinstance(month, int) and 1 <= month <= 12:
            queryset = queryset.filter(month=month)
        # else: якщо month не int і не 0, або не None, це невідомий параметр.
        # Можна обробити як помилку або ігнорувати. Наразі ігноруємо, тобто фільтрація за місяцем не застосовується.

    if not queryset.exists():
        return {'periods_data': [], 'verbose_indicator_names': {}}

    # Визначаємо, як групувати дані
```

```
group_by_fields = []
display_format_func = None

if period_type == 'year':
    # Якщо period_type 'year', групуємо тільки за роком
    group_by_fields = ['year']

    def display_format_func(item):
        return f"Рік {item['year']}"
elif period_type == 'month':
    # Якщо period_type 'month', групуємо за роком і місяцем.
    # Якщо filter_month=0 (річні дані), то місяць буде None
    # Якщо filter_month=None (усі місяці), то це також буде month__isnull=False
    group_by_fields = ['year', 'month']  # Групуємо за month, щоб побачити окремо місяці

    def display_format_func(item):
        if item['month'] is None:  # Це означає, що це були річні дані
            return f"Річні дані ({item['year']})"
        return f"{month_name_ua(item['month'])} {item['year']}"
else:
    # Невідомий period_type
    return {'periods_data': [], 'verbose_indicator_names': {}}

results = []
verbose_indicator_names = {field: props['verbose_name'] for field, props in INDICATOR_CONFIG.items()}

# Збираємо словник для агрегації SUM для кожного показника
aggregation_expressions = {
    field_name: Sum(field_name)
```

```
    for field_name in INDICATOR_CONFIG.keys()
}

# Групуємо і агрегуємо дані
aggregated_data = queryset.values(*group_by_fields).annotate(**aggregation_expressions).order_by(*group_by_fields)

for item in aggregated_data:
    current_period_data = {}
    for field_name in INDICATOR_CONFIG.keys():
        current_period_data[field_name] = item.get(field_name)

    total_score_this_period = 0.0
    individual_scores_this_period = {}

    for indicator_field, props in INDICATOR_CONFIG.items():
        actual_value = current_period_data.get(indicator_field)
        target_value = props['target']
        weight = props['weight']
        direction = props['direction']

        score = 0.0
        if actual_value is None or actual_value == 0:  # Якщо даних немає або значення 0
            if target_value == 0:  # Якщо ціль теж 0, і факт 0 - ідеально
                score = 100.0
            elif direction == "lower_is_better":  # Якщо ціль > 0, а факт 0 (ідеально)
                score = 100.0
            else:  # Якщо ціль > 0, а факт 0 (для positive) - погано
                score = 0.0
        else:  # actual_value > 0
```

```
        if target_value == 0:  # Якщо ціль 0, а факт > 0 (напр. ціль CO2=0, а факт=100)
            if direction == "lower_is_better":  # Погано
                score = 0.0
            else:  # Дуже добре, якщо ціль 0 (напр. заходів сталого розвитку)
                score = 100.0
        else:  # Обидва (факт і ціль) > 0
            if direction == "higher_is_better":
                score = (actual_value / target_value) * 100.0
                score = min(score, 100.0)
            elif direction == "lower_is_better":
                score = (target_value / actual_value) * 100.0
                score = min(score, 100.0)

    total_score_this_period += (score / 100.0) * weight * 100.0

    individual_scores_this_period[indicator_field] = {
        "score": round(score, 2),
        "verbose_name": props['verbose_name'],
        "actual_value": actual_value,
        "target_value": target_value,
        "direction": direction
    }

period_label = display_format_func(item)
results.append({
    'period_label': period_label,
    'total_score': round(total_score_this_period, 2),
    'individual_scores': individual_scores_this_period
})
```

```
return {
    'periods_data': results,
    'verbose_indicator_names': verbose_indicator_names
}
```

Файл urls.py:

```
# core/urls.py

from django.urls import path
from . import views # Імпортуємо функції View з поточного додатку


urlpatterns = [

    # URL для сторінки перегляду списку даних: /data/list/
    path('sustainability-data/', views.view_sustainability_data, name='sustainability_data'),

    # URL-шаблон для головної сторінки
    path('', views.index, name='index'), # 'name=''index'''' дає ім'я цьому URL-шаблону

    path('sustainability-analytics/', views.view_sustainability_analytics, name='sustainability_analytics'),

    # Зараз додамо їх заглушки, щоб посилання в навігації працювали
     path('register/', views.register_view, name='register'),
     path('login/', views.login_view, name='login'),
     path('logout/', views.logout_view, name='logout'),      # URL для виходу
     path('profile/', views.profile_view, name='profile'),

    # ################################################################################
```

```
    # URL-шаблони для Модуля "Ідеї / Ініціативи" (ДОДАНО ДЛЯ ДНЯ 10) -
    # ##############################################################################
    # URL для сторінки зі списком всіх ідей
    path('ideas/', views.list_ideas_view, name='ideas_list'),
    # URL для сторінки з формою надсилання нової ідеї
    path('ideas/submit/', views.submit_idea_view, name='submit_idea'),

    # ##############################################################################
    # URL для перегляду деталей ОДНІЄЇ ідеї (ДОДАНО ДЛЯ ДНЯ 11)
    # <int:idea_id> - це динамічна частина URL, яка захоплює ціле число (ID ідеї)
    # ##############################################################################
    path('ideas/<int:idea_id>/', views.idea_detail_view, name='idea_detail'),

    # ##############################################################################
    # URL для опитувань (ДОДАНО ДЛЯ ДНЯ 12)
    # URL-и для модуля опитувань
    path('polls/', views.poll_list_view, name='poll_list'),
    path('polls/<int:poll_id>/', views.poll_detail_view, name='poll_detail'),
    path('polls/create/', views.create_poll_view, name='create_poll'),
    # URL для лайків
    path('ideas/<int:idea_id>/like/', views.like_idea_view, name='like_idea'),
    # URL для завантаження CSV-файлів (доступно тільки адміністраторам)
    path('data/upload-csv/', views.upload_csv_view, name='upload_csv'),
    # ==============================================================================
    # НОВА URL-АДРЕСА ДЛЯ ЧАТУ / СТІНИ СПІЛКУВАННЯ
    # ==============================================================================
    path('chat/', views.chat_room_view, name='chat_room'),
]
```