НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
Навчально-науковий інститут атомної та теплової енергетики
Кафедра інженерії програмного забезпечення в енергетиці

ДО ЗАХИСТУ ДОПУЩЕНО
В.о. завідувача кафедри
_____ Олександр КОВАЛЬ
«___»_____2025 р.

Дипломна робота

на здобуття ступеня бакалавра
за освітньо-професійною програмою «Інженерія програмного забезпечення інтелектуальних кібер-фізичних систем в енергетиці»
спеціальності 121 Інженерія програмного забезпечення

на тему: «Вебсистема моделювання гібридної енергосистеми на основі Matlab Production Server»

Виконав:
студент IV курсу, групи ТВ-12
Мельник Ігор Борисович _____ _____
(прізвище, ім'я, по батькові) (підпис)

Керівник:
асистент Гейко Олег Олександрович _____ _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Рецензент:

_____ _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень із праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2025

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Навчально-науковий інститут атомної та теплової енергетики
Кафедра інженерії програмного забезпечення в енергетиці
Рівень вищої освіти перший (бакалаврський)
Спеціальність 121 Інженерія програмного забезпечення
Освітньо-професійна програма «Інженерія програмного забезпечення інтелектуальних кібер-фізичних систем в енергетиці»

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ Олександр КОВАЛЬ
«___»_____2025р.

ЗАВДАННЯ

на дипломну роботу студенту

Мельнику Ігорю Борисовичу
(прізвище, ім'я, по батькові)

1. Тема роботи: Вебсистема моделювання гібридної енергосистеми на основі Matlab Production Server
керівник роботи Гейко Олег Олександрович, асистент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «__» _____ 2025р. №_____
2. Строк подання студентом роботи «9» червня 2025р.
3. Вихідні дані до роботи: мова програмування MATLAB, мова программа Python, середовище розробки MATLAB, середовище розробки PyCharm
4. Зміст розрахунково-пояснювальної записки (перелік завдань, які потрібно розробити): розробити вебсистему для моделювання гбридної енергосистеми з інтеграцією MATLAB Production Server та реалізувати веб інтерфейс для клієнтського використання

5. Перелік ілюстративного матеріалу: Модель гібридної енергосистеми, Архітектура системи, Діаграма прецедентів, Файлова структура проєкту, Основні використані інструменти, Фрагмент основної функції обчислювального модуля, Алгоритм роботи обчислювального модуля, Користувацький інтерфейс
6. Дата видачі завдання «31» жовтня 2024р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Строки виконання етапів роботи	Примітка
1	Отримання завдання	30.10.2024	Виконано
2	Дослідження предметноїобласті	31.10.2024 – 01.11.2024	Виконано
3	Дослідження існуючих рішень	02.11.2024 – 01.12.2024	Виконано
4	Постановка вимог допроєктування системи	02.12.2024 – 12.01.2025	Виконано
5	Розробка програмного продукту	13.01.2025 – 11.05.2025	Виконано
6	Тестування	12.05.2025 – 15.05.2025	Виконано
7	Захист програмного продукту	12.05.2025 – 15.05.2025	Виконано
8	Оформлення дипломної роботи	19.05.2025 – 01.06.2025	Виконано
9	Передзахист	02.06.2025 – 06.06.2025	Виконано
10	Захист	16.06.2025 – 27.06.2025	

Студент _____ Мельник І. Б.
(підпис) (ім’я, прізвище)

Керівник роботи _____ Гейко О.О.
(підпис) (ім’я, прізвище)

РЕФЕРАТ

Структура та обсяг дипломної роботи. Робота містить 50 сторінок, 19 рисунків, 2 додатки та 5 посилань.

Метою роботи є розробка вебсистеми моделювання гібридної енергосистеми з можливістю обробки вхідних параметрів, виконання математичних обчислень за допомогою MATLAB Production Server та графічної візуалізації результатів у вебінтерфейсі.

Практичне значення одержаних результатів полягає в створенні доступного інструменту для інженерів, дослідників та студентів, який дозволяє проводити симуляцію роботи гібридних енергосистем і аналізувати сценарії взаємодії сонячної генерації, батареї та дизельного генератора в умовах змінного навантаження.

Ключові слова: гібридна енергосистема, вебсистема, моделювання, MATLAB, MATLAB Production Server, симуляція, енергетика.

ABSTRACT

Structureandscope of thethesis. The work contains 50 pages, 19 figures, 2 appendices and 5 references.

The purpose of thework is to develop a web-based simulation system for a hybrid energy system with input processing, mathematical computation using MATLAB Production Server, and graphic visualization of results in a browser interface.

The practical value of the obtained results lies in providing an accessible tool for engineers, researchers, and students, enabling simulation of hybrid energy systems and analysis of scenarios involving solar generation, battery storage, and diesel backup under variable load conditions.

Keywords: hybrid energy system, web application, simulation, MATLAB, MATLAB Production Server, energy system modeling, solar panels**.**

ЗМІСТ

ВСТУП

На сьогоднішній день, при стрімкому розвитку енергетичної галузі особливу увагу приділяють питанням енергоефективності та надійності постачання електроенергії, і використання відновлюваних джерел енергії, зокрема сонячної, стає дедалі більш актуальним у всьому світі. Проте для забезпечення стабільної та ефективної роботи енергосистем з відновлюваними джерелами енергії необхідно створювати гібридні енергосистеми, які поєднують декілька джерел живлення, наприклад, сонячні панелі та дизельні генератори.

Оскільки генерація електроенергії з відновлюваних джерел значною мірою залежить від погодних умов, постає проблема забезпечення чіткого прогнозування та ефективного управління такими системами. Одним із способів вирішення цієї задачі є створення систем моделювання, які дозволяють на основі фактичних та прогнозних даних оцінити ефективність роботи енергосистеми при різних умовах.

Сучасні підприємства та наукові установи в енергетичній галузі активно впроваджують цифрові засоби моніторингу, прогнозування і моделювання гібридних систем. Такі системи дозволяють підтримувати інтеграцію сонячної генерації з резервними джерелами, наприклад, дизельними генераторами, і надають можливість візуалізувати результати та виконувати точні розрахунки на основі математичних моделей.

Метою даної дипломної роботи є розробка вебсистеми моделювання гібридної енергосистеми з можливістю збору, обробки та аналізу даних від сонячних панелей і дизельного генератора, та реалізацією обчислювальних функцій на основі Matlab Production Server. Основою поставленого завдання є забезпечення аналізу способів інтеграції відновлюваних джерел енергії у локальні мережі та розробка алгоритмів моделювання роботи гібридної енергосистеми в реальному часі. Також важливу роль зіграє побудова

інтуїтивного вебінтерфейсу з метою забезпечення доступу до результатів моделювання.

Подбна вебсистема може бути використана у декількох практичних напрямах, наприклад при дослідженні та проєктуванні у сфері енергетики для аналізу сценаріїв роботи гібридних систем. Також її можуть використовувати підприємства, які впроваджують локальні джерела генерації для оптимізації споживання та зменшення витрат. Крім того, дана система може стати базою для подальших наукових розробок у цій сфері.

Таким чином, актуальність дипломної роботи зумовлена потребою у доступних, адаптивних та ефективних цифрових рішеннях для моделювання гібридних енергосистем, а її новизна полягає у використанні Matlab Production Server для математичного моделювання в реальному часі у вебсередовищі на базі сучасних технологій.

РОЗДІЛ 1 ПОСТАНОВКА ЗАВДАННЯ

У цьому розділі сформульовано основну мету та завдання, які стояли перед розробкою вебсистеми для моделювання гібридної енергосистеми, та опис проблеми яку вирішує дана робота. Наведено обґрунтування вибору предметної області, визначено вимоги до функціональності, структури та взаємодії компонентів системи.

1.1 Формулювання завдянь

Проєктування та моделювання гібридних енергосистем потребують не лише якісного обладнання, але й точного математичного підходу до аналізу та оптимізації їхньої роботи. У реальних умовах, де кожен об'єкт має унікальні характеристики особливо важливим є попередній аналіз можливих сценаріїв функціонування системи. Такий аналіз дозволяє не лише передбачити поведінку системи в різних умовах, але й оптимізувати налаштування її компонентів до початку фізичного впровадження.

Одним із дієвих інструментів такого аналізу є моделювання, яке дає змогу математично описати поведінку енергосистеми з урахуванням усіх необхідних параметрів. У межах цієї дипломної роботи вирішується задача побудови програмного середовища, що дозволяє користувачу формувати набір параметрів гібридної енергосистеми, запускати моделювання роботи цієї системи на основі заданих характеристик та аналізувати результати й робити висновки щодо ефективності роботи компонентів.

На відміну від систем, які працюють виключно з теоретичними моделями або обмежені певними наборами сценаріїв, розроблювана система має дозволити задавати будь-які допустимі параметри, наприклад, площу сонячних панелей, рівень сонячної радіації, ємність батареї, стан заряду, навантаження тощо. Кінцева мета — забезпечити можливість швидкої

перевірки та оцінки результатів моделювання, адаптованої під конкретні умови користувача.

Важливою особливістю задачі є необхідність поєднання доступності інтерфейсу для користувача з високоточними обчисленнями на основі інженерної моделі. Тобто, система повинна працювати у форматі вебзастосунку, який доступний з браузера, але обчислення повинні виконуватись у середовищі, що підтримує складну математичну логіку — у цьому випадку MATLAB.

У процесі постановки задачі були сформульовані такі ключові технічні цілі:

- реалізувати вебінтерфейс для взаємодії користувача з моделлю;
- забезпечити передачу параметрів моделювання на сервер;
- розробити обчислювальний модуль, який виконує симуляцію згідно з прийнятою структурою гібридної енергосистеми;
- повернути результати у зручному форматі для подальшого аналізу;
- забезпечити авторизацію користувачів, захист даних і можливість масштабування функціоналу в майбутньому.

Таким чином, задача даної роботи не обмежується створенням математичної моделі. Вона полягає у реалізації повноцінного функціонального програмного комплексу, який поєднує сучасний вебінтерфейс, серверну логіку та розрахунковий модуль.

1.2 Структура та основні вимоги

Розроблена система має трирівневу архітектуру, яка включає клієнтську частину, серверну логіку та обчислювальний модуль. Кожен з цих компонентів виконує чітко визначену функцію і взаємодіє з іншими через чітко прописаний інтерфейс. Такий підхід дозволяє зробити систему більш надійною, масштабованою та простою у супроводі.

Клієнтська частина реалізована за допомогою бібліотеки React, що дозволяє створювати інтерактивні вебінтерфейси. Використання TypeScript забезпечує строгішу типізацію, що знижує кількість помилок і підвищує якість коду. Користувач взаємодіє із системою через браузер, має можливість зареєструватися, увійти в обліковий запис, сформувати параметри для моделювання, запускати симуляцію та переглядати результати. Інтерфейс передбачає як ручне введення даних, так і автоматичну генерацію сценаріїв для аналізу роботи системи в різних умовах. Крім того, реалізовано можливість перемикання між ручним і автоматичним режимом запуску симуляцій.

Серверна частина побудована на основі Python з використанням фреймворку Flask. Вона відповідає за прийом запитів від клієнта, валідацію вхідних параметрів, авторизацію користувачів за допомогою токенів JWT та передачу даних до обчислювального модуля. Крім того, сервер обробляє відповіді від MATLAB Production Server, здійснює логування, обробку помилок і надає відповіді у форматі JSON, які потім обробляються на фронтенді. В якості бази даних для зберігання облікових записів користувачів використано SQLite, оскільки вона є легкою та достатньою для невеликої вебсистеми.

Обчислювальний модуль реалізований як окрема функція у середовищі MATLAB і виконується через MATLAB Production Server. Він приймає параметри, сформовані користувачем, у вигляді структури, обробляє їх згідно з розробленою математичною моделлю гібридної енергосистеми, і повертає результати, що включають згенеровану потужність, стан заряду батареї, споживання пального дизельним генератором, непокрите навантаження тощо. MATLAB Production Server виступає в ролі науково-обчислювального модуля системи, що забезпечує високу точність результатів.

До ключових вимог, які висувалися до цієї системи, можна віднести зручність використання через вебінтерфейс, підтримку захищеного доступу, швидку відповідь сервера на запити, стабільність роботи обчислювального

модуля та можливість розширення функціоналу в майбутньому. Усі компоненти системи мають бути легко інтегрованими, змінюваними та адаптивними до нових задач.

1.3 Реалізація функцій

Реалізація функціоналу системи охоплює весь алгоритм взаємодії користувача — від першого входу до отримання результатів моделювання. Для початку, користувач має змогу створити обліковий запис або увійти до системи за допомогою логіна та пароля якщо аккаунт уже існує. Весь процес авторизації реалізовано з дотриманням сучасних принципів безпеки, включаючи хешування паролів та використання токенів для захищеного доступу до API. Після входу дані про користувача зберігаються у локальному сховищі браузера, що дозволяє безперервно використовувати систему без повторного входу.

Після авторизації користувач потрапляє на основну сторінку моделювання. Тут він може заповнити форму з параметрами гібридної енергосистеми: вказати величину навантаження, площу сонячних панелей, ефективність генерації, рівень сонячної радіації, параметри батареї, максимальну потужність дизельного генератора тощо. Також передбачено автоматичну генерацію параметрів з використанням генератора випадкових значень у допустимих межах. Автоматичний режим дозволяє симулювати зміну стану системи в реальному часі з інтервалом оновлення в 30 секунд, з урахуванням попереднього рівня заряду акумулятора SOC, що забезпечує реалізм симуляції.

Усі зібрані дані передаються на серверну частину в якій формуються запити до MATLAB Production Server. У MATLAB запускається функція simulateHybridSystem, яка обробляє параметри, виконує моделювання та повертає набір результатів. Ці результати містять інформацію про обсяг згенерованої сонячної енергії, потік потужності до або з батареї, стан заряду

акумулятора, кількість спожитого пального, загальну потужність, яку вдалося подати, та величину непокритого навантаження.

Після отримання результатів від обчислювального модуля система відображає їх на інтерфейсі користувача. Для зручності реалізовано форматування даних, їх виведення у вигляді таблиць або JSON-блоків, а також логування останніх результатів автоматичного режиму. У подальшому можлива реалізація графічного представлення даних за допомогою відповідних графіків та діаграм, а також модулів для збереження історії запитів і аналізу динаміки.

Таким чином, реалізована система надає повний функціонал для аналізу роботи гібридної енергосистеми на основі змінних параметрів, дозволяє взаємодіяти з математичною моделлю та забезпечує інтеграцію MATLAB у зручному вебформаті.

Висновки до розділу 1

У першому розділі, у результаті аналізу предметної області було сформовано чітке завдання — створити вебсистему, яка дозволяє моделювати роботу гібридної енергосистеми на основі взаємодії сонячних панелей, акумулятора та дизельного генератора. Визначено ключові вимоги до системи, її архітектуру, базові функції та принципи побудови з урахуванням інженерної специфіки.

РОЗДІЛ 2 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

При розробці програмного забезпечення для моделювання гібридних енергосистем важливо попередньо ознайомитися з існуючими рішеннями в цій галузі. Це дозволяє не тільки зрозуміти загальні підходи до проєктування подібних систем, але й визначити можливі проблеми, які можна врахувати при розробці власної системи.

2.1 Підходи до моделювання гібридних енергосистем

Існуючі програмні системи для аналізу та планування моделювання гібридних енергетичних систем можна умовно розділити на дві категорії: локальні десктопні застосунки та веборієнтовані платформи, що працюють через браузер. У більшості випадків їх основне призначення полягає у допомозі проєктувальникам і дослідникам під час вибору компонентів такої енергосистеми, її розгортання, порівняння різних сценаріїв генерації та споживання енергії, а також в економічному аналізі окупності тих чи інших рішень.

Десктопні системи, як правило, орієнтовані на розгорнуте інженерне моделювання з фіксованими наборами параметрів. Вони часто мають складний інтерфейс, вимагають встановлення та ліцензування, і працюють автономно, без інтеграції з іншими цифровими системами. Такі системи здебільшого підходять для попереднього проєктування або аналізу після завершення експлуатаційного циклу, але не дають змоги проводити моделювання у реальному часі з актуальними параметрами та даними в режимі онлайн.

Окрему групу складають високоспеціалізовані пакети для моделювання сонячних або вітрових установок, які зазвичай концентруються на оптимізації геометричних чи економічних параметрів. Їхня функціональність часто обмежується розрахунками для одного типу генерації та не охоплює

повністю всю логіку взаємодії між різними джерелами енергії, які зазвичай є компонентами гібридних енергосистем, такими як дизельні генератори або акумулятори. Це ускладнює їх застосування в контексті гібридних систем, де потрібна взаємодія між усіма компонентами.

З появою нових вебтехнологій також часто спостерігається тенденція до перенесення елементів моделювання у браузерні застосунки та надання певного доступу до таких симуляцій через веб-інтерфейси. Проте такі системи або орієнтовані на дуже широкі сценарії, наприклад національні енергомережі, або є лише фронтендами до статичних обчислень.

Деякі спрощені симулятори створюються університетами або дослідницькими лабораторіями для навчальних цілей, однак їх функціональність обмежена лише демонстрацією базових принципів. При цьому вони рідко інтегруються з високоточними обчислювальними модулями на основі інженерних рішень, таких як MATLAB.

Таким чином, у сфері веборієнтованих систем, які дозволяють здійснювати гнучке, точне моделювання гібридних енергосистем у режимі реального часу, зберігається суттєвий дефіцит інструментів, орієнтованих на користувача, що не має глибоких технічних знань у програмуванні або енергетиці.

2.2 Переваги створення власної системи з MATLAB

На основі розглянутих недоліків типових рішень даної проблеми виникає чітка потреба у створенні спеціалізованої системи, яка буде не лише адаптована під конкретну задачу моделювання гібридної енергосистеми, а й дозволить інтерактивно взаємодіяти з математичною моделлю в режимі реального часу. Така система повинна бути простою у використанні, технологічно гнучкою, а також здатною до інтеграції з іншими цифровими сервісами та інтерфейсами.

Ключовою перевагою розробленого рішення є його веборієнтованість. Усі дії — введення параметрів, запуск симуляцій, перегляд результатів — виконуються через веб браузер, без потреби встановлювати додаткове програмне забезпечення. Це відкриває доступ до системи з будь-якого сучасного пристрою, як і комп'ютера так і ноутбука, планшета або смартфона. Такий підхід забезпечує максимальну мобільність, зручність і доступність як для окремих користувачів, так і для команд, які працюють над спільними дослідженнями.

Система розробляється як відкрито масштабована: її архітектура дозволяє у майбутньому розширити набір моделей — наприклад, додати підтримку вітрових електростанцій, гідрогенераторів, економічних моделей або систем управління навантаженням. Код системи побудований за принципами модульності, що забезпечує простоту зміни обчислювальної логіки, графічного інтерфейсу або джерел вхідних даних.

Центральне місце в системі займає обчислювальний модуль на базі MATLAB. MATLAB — це провідне середовище технічних обчислень, яке широко використовується в сучасних інженерних, наукових та академічних сферах. Його головними перевагами є:

- зручність формулювання математичних моделей у вигляді зрозумілого коду;
- наявність вбудованих функцій для роботи з матрицями, сигналами, системами управління, оптимізацією;
- висока точність чисельних методів та обчислень;
- багаті засоби для побудови графіків, аналізу результатів та візуалізації процесів;
- підтримка інтеграції з іншими мовами програмування та середовищами.

У контексті цієї системи MATLAB використовується як науково-інженерний модуль, а його обчислювальні функції розгорнуті за допомогою MATLAB Production Server — спеціальній серверній платформі, яка дозволяє

виконувати код функцій MATLAB у відповідь на зовнішні запити користувача. Таким чином, кожна симуляція, ініційована користувачем з вебінтерфейсу, спричиняє запуск MATLAB функції з відповідними параметрами, обчислення результатів і передачу їх назад у браузер у вигляді JSON.

Застосування MATLAB у складі даної вебсистеми дозволяє досягти поєднання гнучкості вебтехнологій та обчислювальної потужності інженерного програмного забезпечення, що є надзвичайно актуальним напрямом у сучасній інженерії програмного забезпечення. Адже більшість класичних вебзастосунків не здатні виконувати складні наукові розрахунки, а більшість інженерних систем не мають доступного інтерфейсу або інтеграції з користувачем у режимі онлайн. Запропонований підхід якраз і об'єднує ці можливості в одній системі

З погляду галузі інженерії програмного забезпечення в енергетиці, це рішення демонструє як можна розробити системи, які не лише збирають і виводять дані, але й реально моделюють поведінку фізичних процесів та як сучасні технології цифрової трансформації реалізуються у практичному інженерному проєкті.

Отже, створення власної вебсистеми з використанням MATLAB Production Server є не лише технічно обґрунтованим, а й доволі актуальним у контексті розвитку цифрових рішень для енергетики, автоматизації та освітньої сфери. Це рішення здатне не лише відтворювати моделі, а й слугувати основою для подальших досліджень, оптимізації енергетичних систем, викладання технічних дисциплін та впровадження розумних енергомереж.

2.3 Модель гібридної енергосистеми

У рамках даної роботи реалізується модель гібридної енергосистеми, яка складається з трьох основних компонентів: сонячних панелей,

акумуляторної батареї та дизельного генератора. Така система дозволяє забезпечити стабільне живлення споживача за змінних погодних умов і коливань навантаження, поєднуючи переваги відновлюваних та традиційних джерел енергії.

Сонячні панелі в даному випадку служать основним джерелом електроенергії. Потужність яка генерується цими панелями залежить від рівня сонячного випромінювання, площі їх поверхні та ефективності модулів. У даній моделі враховується змінність сонячної радіації, наприклад, через погодні умови або час доби, що дозволяє відтворити динаміку генерації протягом дня. Якщо виробленої енергії достатньо, вона безпосередньо покриває поточне навантаження, а надлишок може бути збережений у батареї.

Акумуляторна батарея використовується для накопичення надлишкової енергії в години активної генерації та для покриття навантаження у періоди коли сонячної енергії не вистачає. Модель передбачає врахування реального рівня заряду, ефективності заряду та розряду, обмежень за потужністю та ємністю. Це дозволяє точно визначити, скільки енергії може бути збережено чи віддано в певний момент, а також наскільки ефективно батарея працює у складі всієї системи.

Дизельний генератор виконує роль резервного джерела. Він вмикається лише тоді, коли і сонячна генерація, і батарея не можуть покрити потреби споживача. Такий підхід дозволяє знизити витрати пального та використовувати дизель лише в крайніх випадках. У розрахунках враховується максимальна потужність генератора, споживання пального, а також вплив його роботи на загальну ефективність системи. Розраховується також кількість енергії, яку вдалося подати, і обсяг непокритого навантаження у разі дефіциту.

Основні переваги даної моделі полягають в тому, що поєднання сонячних панелей, акумуляторної батареї та дизельного генератора дозволяє досягти балансу між екологічністю, автономністю та стабільністю. У сонячні

дні основну частину навантаження покривають панелі, батарея вирівнює піки та дефіцити, а дизель гарантує енергопостачання в критичних ситуаціях. Така модель дуже добре підходить для віддалених об'єктів, де немає доступу до централізованої мережі, а потреба в енергії постійно змінюється протягом доби (рисунок 2.1).

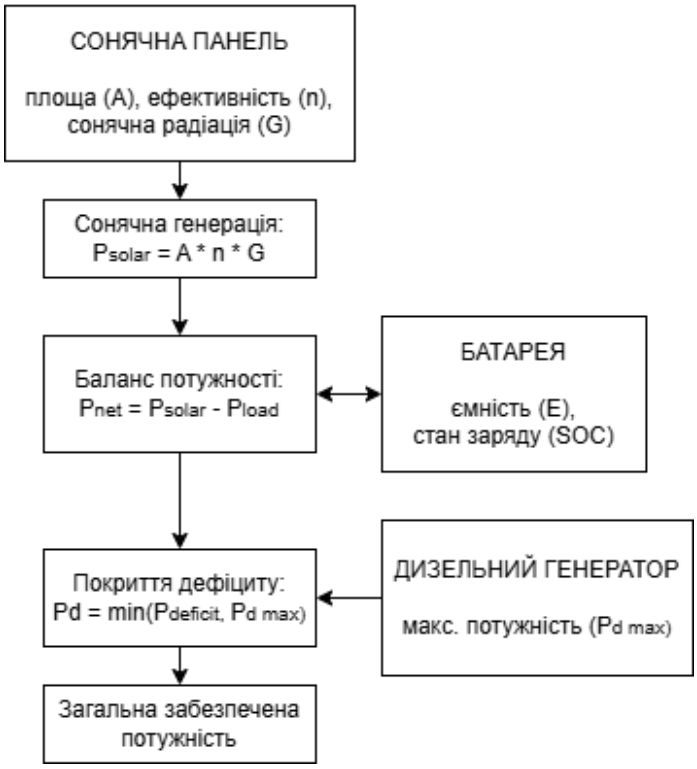


Рис. 2.1 – Модель гібридної енергосистеми

З технічного боку, взаємодія даних трьох джерел створює складну логіку пріоритетів та залежностей. Це робить систему цікавою з точки зору моделювання: можна аналізувати різні сценарії, тестувати зміну параметрів генерації, ємності батареї або навантаження, визначати, коли саме варто запускати дизель, і як змінюється ефективність даної системи при різних погодних умовах.

Таким чином, розглянута модель є досить реалістичною, практично орієнтованою та добре придатною для реалізації у вигляді веборієнтованої симуляційної системи. Вона дозволяє не тільки демонструвати роботу гібридної енергосистеми, а й аналізувати її поведінку в різних умовах, що особливо корисно для інженерів, дослідників та проєктувальників.

Висновки до розділу 2

У другому розділі, було проведено аналіз предметної області, який показав, що більшість готових рішень або є закритими та дорогими, або не мають потрібної гнучкості й інтерактивності необхідної для забезпечення оптимального моделювання гібридної енергосистеми для середнього користувача. Це обґрунтовує доцільність створення власної вебсистеми, орієнтованої на моделювання конкретної гібридної енергосистеми з точним математичним обчисленням, реалізованим у середовищі MATLAB.

РОЗДІЛ 3 ЗАСОБИ РОЗРОБКИ СИСТЕМИ

У цьому розділі описано архітектуру створеної системи, принцип її побудови на основі клієнт–серверної моделі, а також інструменти та технології, використані для реалізації кожного з компонентів. Детально розглянуто структуру проєкту та обґрунтовано вибір програмних засобів для кожної частини системи.

3.1 Архітектура системи

Архітектура розробленої системи базується на класичному клієнт–серверному підході, що дозволяє розділити функціональні обов'язки між різними частинами застосунку та спростити процес розробки даної вебсистеми.

Загальна схема роботи системи передбачає наявність трьох основних компонентів: клієнтської частини, серверної частини та обчислювального модуля, який забезпеує моделювання гібридної енергосистеми за допомогою MATLAB Production Server (рисунок 3.1).

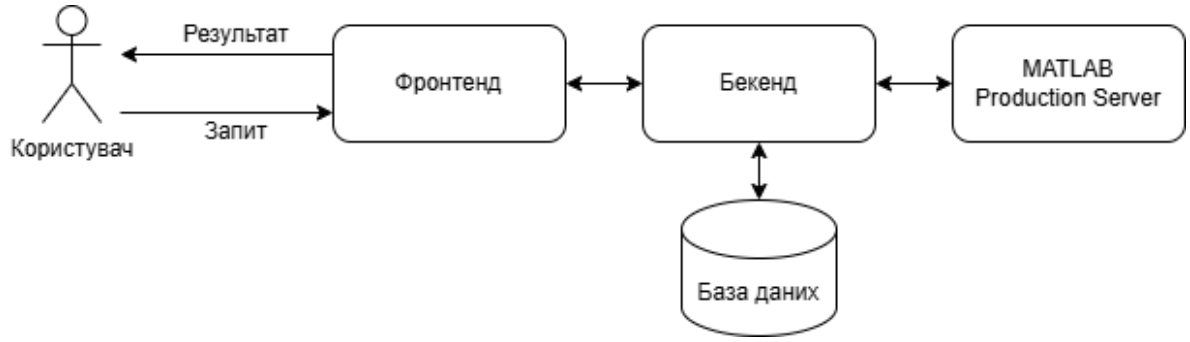


Рис. 3.1 – Архітектура системи

Фронтенд відповідає за взаємодію з користувачем. Користувач має можливість зареєструватися в системі, увійти до створенного особистого облікового запису, заповнити форму з вхідними параметрами для симуляції, а також переглянути результати. Усі ці дії виконуються в браузері, при цьому дані передаються на сервер через HTTP-запити у форматі JSON. Реалізація

клієнтської частини здійснена з використанням фреймворку React, який забезпечує побудову зручного, адаптивного та швидкого інтерфейсу користувача. У якості мови програмування використано TypeScript, що додає статичну типізацію та полегшує підтримку даного проєкту.

Серверна частина реалізована на мові програмування Python із використанням вебфреймворку Flask. Він обробляє запити від клієнта, виконує авторизацію та автентифікацію користувачів, зберігає облікові дані в базі даних SQLite, а також формує запити до MATLAB Production Server. Крім того, бекенд відповідає за логування подій, обробку помилок та перевірку вхідних даних.

Обчислювальний модуль, що є окремим компонентом системи, реалізований у середовищі MATLAB. Він запускається на MATLAB Production Server і виконує моделювання гібридної енергосистеми на основі вхідних параметрів. Модель враховує генерацію електроенергії сонячними панелями, заряд та розряд акумуляторної батареї, а також роботу дизельного генератора. Усі розрахунки виконуються на стороні MATLAB, а отримані результати передаються назад через сервер до клієнта.

Обрана архітектура має низку переваг. По перше, вона забезпечує модульність, тобто кожен компонент системи виконує чітко визначені функції. По друге, використання клієнт–серверної моделі дозволяє відокремити обчислювальну логіку від інтерфейсу користувача, що сприяє кращій масштабованості та зручнішій підтримці. По третє, використання MATLAB Production Server дозволяє інтегрувати математичні алгоритми безпосередньо у вебзастосунок.

Завдяки такій архітектурі система працює ефективно, надійно та дозволяє легко оновлювати окремі її частини без потреби змінювати всю структуру (рисунок 3.2).

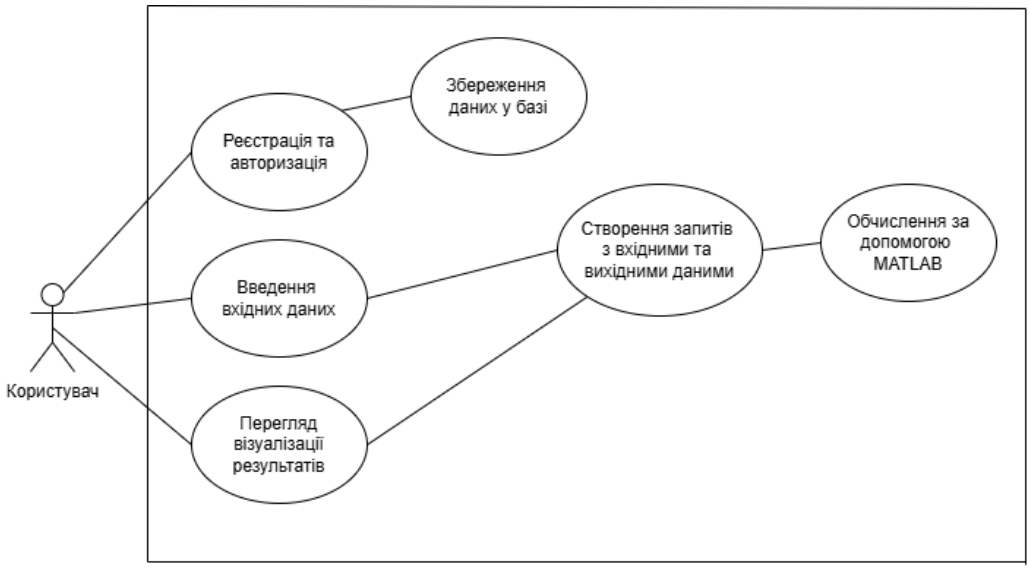


Рис. 3.2 – Діаграма прецедентів

Щоб краще зрозуміти реалізацію архітектурного підходу на практиці, варто детально розглянути файлову та логічну структуру проєкту, яка відображає розділення відповідальностей між компонентами системи (рисунок 3.3).

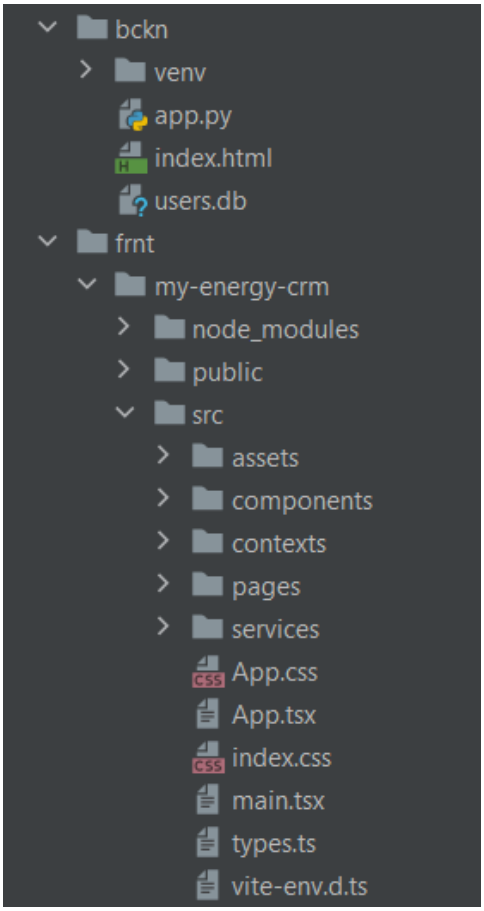


Рис. 3.3 – Файлова структура проєкту (фронтенд та бекенд)

Усі елементи що відповідають за серверну логіку, зберігаються в директорії bckn/. Це незалежний Python-додаток, побудований на Flask, який обробляє API-запити, здійснює автентифікацію, взаємодіє з MATLAB Production Server і зберігає дані.

- Зокрема app.py — головний файл серверної частини. У ньому оголошені маршрути Flask, реалізовано логіку обробки HTTP-запитів, перевірка токенів, виклики до MATLAB Production Server, формування відповіді. Саме і з нього запускається бекенд сервер.
- index.html — базова сторінка HTML, яка в цьому проєкті слугувала для перевірки доступності сервера, демонстрації простого інтерфейсу та загального тестування додатку.
- users.db — локальна база даних SQLite, у якій зберігаються облікові записи користувачів. Вона інтегрується із Flask через бібліотеку SQLAlchemy.

Весь вебінтерфейс користувача розташований відповідно у директорії frnt/my-energy-crm/. Це повноцінний клієнтський застосунок, розроблений на основі фреймворку React з використанням TypeScript, зібраний за допомогою Vite — сучасного інструменту для швидкого побудування та запуску вебдодатку.

Найважливіші файли розташовані у директорії src/, яка є центром розробки основного функціоналу (рисунок 3.4):

- App.tsx — головний компонент програми, де визначається структура маршрутизації, компонування інтерфейсу.
- main.tsx — точка входу React-додатку. Тут ReactDOM монтує кореневий компонент у DOM.
- types.ts — опис структур даних і типів, які використовуються в усій системі.

- AuthContext.tsx (у папці contexts/) — контекст авторизації, що централізовано зберігає стан користувача, токен, логіку логіну, реєстрації та виходу.
- authService.ts (у services/) — файл, який містить логіку для взаємодії з API авторизації: запити на реєстрацію, логін, перевірку токена.

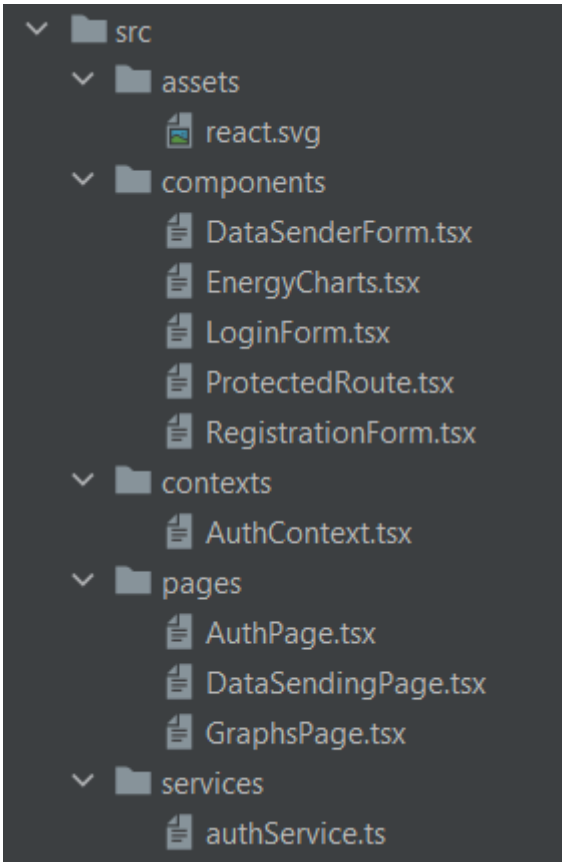


Рис. 3.4 – Вміст папок в основній директорії фронтенду

Компоненти, що формують інтерфейс, згруповані у директорії components/:

- RegistrationForm.tsx та LoginForm.tsx — реалізують форми для входу і реєстрації.
- ProtectedRoute.tsx — обгортка для захищених маршрутів доступних лише після авторизації.
- DataSenderForm.tsx — форма для введення параметрів моделювання вручну.

- EnergyCharts.tsx — компонент для відображення результатів симуляції у вигляді JSON, таблиць та графіків.

Навігація по сторінках реалізована через React Router і згрупована в директорії pages/:

- AuthPage.tsx — сторінка з логіном і реєстрацією.
- DataSendingPage.tsx — сторінка для запуску моделювання вручну або в автоматичному режимі.
- GraphsPage.tsx — сторінка для візуалізації результатів у вигляді графіків.

Також окремо від фронтенду та бекенду зберігається і модель, реалізована у MATLAB (рисунок 3.5).

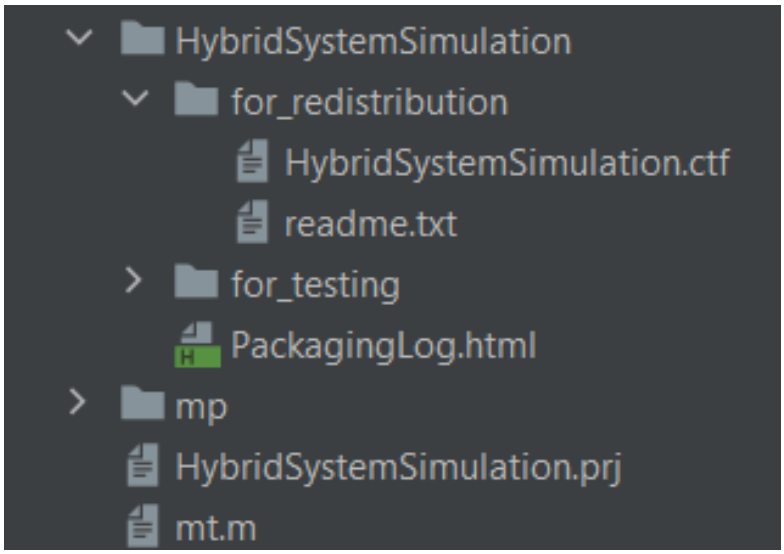


Рис. 3.5 – Обчислювальний модуль у проєкті

mt.m — файл, що містить логіку основної функції MATLAB, яка називається simulateHybridSystem та відповідає за моделювання гібридної енергосистеми. Вона отримує вхідні параметри, та розраховує сонячну генерацію, стан заряду батареї, витрати палива, а потім передає отримані значення назад на бекенд

mp/ — робоча директорія інстанції MATLAB Production Server. Тут зберігаються всі компоненти, необхідні для обслуговування запитів: опубліковані функції, конфігураційні файли, логіка взаємодії з API. Зокрема

у відповідній для цього папці auto_deploy маємо скомпільований готовий архів HybridSystemSimulation.ctf що й містить в собі функцію simulateHybridSystem (рисунок 3.6).

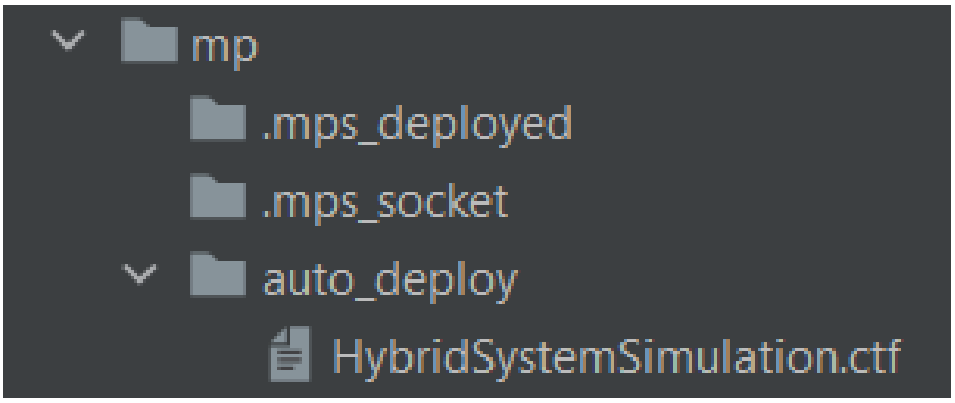


Рис. 3.6 – Робоча директорія ісінтанції MATLAB Production Server

MATLAB Production Server взаємодіє із Flask-сервером через вбудовані клієнтські бібліотеки MATLAB Engine API for Python. Це дозволяє викликати mt.m з бекенду як зовнішню функцію, передавши аргументи через Python-структури.

Таким чином, така структура проєкту реалізує класичну трирівневу архітектуру: клієнт, сервер, обчислювальний модуль. Це дозволяє підтримувати чіткий розподіл відповідальності, швидко вносити зміни до окремих частин, повторно використовувати компоненти й масштабувати систему в майбутньому.

3.2 Інструменти та технології

Для реалізації кожного з компонентів системи було обрано відповідні інструменти та технології, що забезпечили ефективну розробку та якісне виконання всіх вимог проєкту. Усі обрані інструменти є відкритими, мають популярність серед сучасних розробників та хорошу документацію, що дозволило швидко інтегрувати їх у проєкт (рисунок 3.7).

Рис. 3.7 – Основні використані інструменти

Фронтенд реалізований з використанням React — одного з найпопулярніших JavaScript фреймворків для розробки односторінкових додатків. React дозволяє розробляти інтерфейс у вигляді незалежних компонентів, які можна багаторазово використовувати, що значно спрощує структурування коду. Для типізації застосовано мову TypeScript, яка дозволяє виявляти помилки ще на етапі розробки й робить код більш надійним. Для зборки та розгортання клієнтської частини обрано інструмент Vite, який забезпечує швидке перезбирання проєкту, оновлення сторінок і мінімізацію коду.

Бекенд побудовано на Python з використанням вебфреймворку Flask, що дозволяє швидко створити REST API з мінімальними витратами часу. Flask є гнучким і добре підходить для проєктів такого обсягу, оскільки даний фреймворк має модульну структуру та легко налаштовується, що й робить його ідеальним вибором для швидкої розробки серверної логіки. Однією з головних переваг Flask є його простота, оскільки для створення повноцінного серверу не вимагається написання занадто комплексного коду.

Для зберігання облікових даних користувачів використовується легка реляційна база даних SQLite, яка інтегрується у Flask за допомогою бібліотеки SQLAlchemy. SQLite не потребує окремого сервера тому вона є

дуже зручною та підходить для проєктів невеликого масштабу або їх прототипів.

Для реалізації безпечної авторизації та доступу до закритих частин API використовується бібліотека flask-jwt-extended, яка підтримує всі основні операції з JSON Web Token: створення, перевірку та оновлення відповідних облікових даних користувача. JWT дозволяє зберігати всю необхідну інформацію про користувача в зашифрованому вигляді та використовувати спеціальний токен для доступу до захищених маршрутів без необхідності щоразу вводити пароль.

Основною особливістю цього проєкту є використання MATLAB Production Server як окремого компонента. MATLAB Production Server — спеціалізоване серверне середовище, розроблене компанією MathWorks. MATLAB Production Server дозволяє інтегрувати функції для математичних розрахунків написані у середовищі MATLAB, в окремі застосунки, виконуючи їх як хмарні або локальні обчислення. Зокрема і в нашому розробленому застосунку, сервер MATLAB отримує дані у форматі JSON, що дозволяє легко передавати їх з Python або будь-якої іншої мови. Він виконує відповідну MATLAB функцію яка реалізує повну модель гібридної енергосистеми. Після обробки даних та відповідних обрахунків, MATLAB Production Server повертає результат у вигляді об’єкта JSON, який потім використовується у відповіді вебсервера. Цей підхід дозволяє використовувати алгоритми MATLAB у складі вебзастосунку без необхідності переписувати їх іншими мовами.

Окрім основних інструментів, під час розробки проєкту також використовувались:

Git – система контролю версій, яка дозволила зберігати історію змін, працювати над проєктом поетапно та легко повертатися до попередніх версій коду.

PyCharm – основне середовище розробки;

npm – менеджер пакетів для керування залежностями у клієнтській частині.

pip – менеджер пакетів Python для встановлення всіх необхідних бібліотек на серверній частині.

Postman – інструмент для тестування REST API, який активно використовувався для перевірки коректності запитів, відлагодження маршрутів та перевірки форматів даних.

Загалом, обрані інструменти та технології є досить сучасними та зручними у використанні, і допомогли забезпечити високу ефективність при створенні вебсистеми з обчислювальним модулем на MATLAB.

Висновки до розділу 3

У третьому розділі було продемонстровано, що розроблена система має чітко структуровану архітектуру з поділом на фронтенд, бекенд і обчислювальний модуль. Для реалізації використано сучасні та перевірені технології — React, Flask, MATLAB Production Server, що забезпечило ефективність розробки, простоту масштабування та інтеграцію математичної моделі з вебінтерфейсом.

РОЗДІЛ 4 ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

Цей розділ присвячений практичній реалізації основних функцій системи: автентифікації користувача, передачі параметрів до обчислювального модуля та симуляції роботи гібридної енергосистеми. Наведено опис логіки взаємодії компонентів та алгоритму моделювання у MATLAB.

4.1 Авторизація користувачів

Для забезпечення доступу до функціоналу системи з можливістю персоналізації була реалізована повноцінна система авторизації та автентифікації користувачів. Її основою є використання REST API, розробленого на Flask, і технології JSON Web Token, що дозволяє реалізувати безпечну взаємодію між клієнтом і сервером.

Користувач має можливість зареєструватися, вказавши своє ім’я, електронну пошту та пароль. Після підтвердження правильності введених даних, вони надсилаються на сервер, де перевіряється унікальність email-адреси. Якщо така адреса ще не зареєстрована, новий користувач записується до бази даних, реалізованої на основі SQLite. З метою безпечного користування пароль попередньо хешується з використанням алгоритму SHA256.

Увійти до системи можна через відповідну форму входу. Після надсилання логіна та пароля на серверній частині проводиться перевірка — звіряється електронна пошта та хеш пароля з наявними у базі даними. Якщо дані коректні, система генерує JWT-токен, який надсилається у відповідь користувачу. Цей токен зберігається на клієнтській частині у локальному сховищі браузера і автоматично використовується при виконанні подальших запитів до захищених маршрутів API.

На стороні клієнта за логіку автентифікації відповідає компонент AuthContext, який побудований з використанням контекстного API React. Він забезпечує централізоване зберігання інформації про користувача, токен доступу, а також методи логіну, реєстрації, виходу з системи та обробки помилок. Компоненти RegistrationForm.tsx та LoginForm.tsx забезпечують графічний інтерфейс для відповідних дій.

Таким чином, реалізовано частину системи, що відповідає за авторизацію та дозволяє розрізняти користувачів та обмежувати доступ до функцій моделювання лише для авторизованих осіб. Це також може слугувати прототипом для подальшої реалізації індивідуального збереження історії розрахунків та налаштувань.

4.2 Передача та обробка даних

Після входу в систему користувач потрапляє на сторінку запуску симуляції, де він може заповнювати параметри гібридної енергосистеми. Серед них — площа сонячної панелі, її ефективність, інтенсивність сонячної радіації, ємність акумулятора, початковий рівень заряду, потужність дизельного генератора, навантаження та інші характеристики. Дані можна вводити вручну або згенерувати автоматично — для цього передбачений функціонал генерації випадкових, але фізично допустимих значень. Також реалізовано збереження попереднього рівня заряду акумулятора між ітераціями моделювання, що дозволяє симулювати безперервну роботу системи в динаміці.

Після формування вхідного набору параметрів дані передаються до серверної частини через маршрут /api/simulate_local методом POST у форматі JSON. Фреймворк Flask приймає ці дані та виконує валідацію значень, зокрема перевіряє їх на відповідність очікуваним типам і межам, а потім формує структуру даних для передачі до MATLAB Production Server та подальшої обробки даних.

Для взаємодії з MATLAB використовується вбудований клієнтський модуль MATLAB Python Integration, який дозволяє передавати вхідні аргументи у вигляді структури record до функції simulateHybridSystem, розміщеної на MATLAB Production Server. Ця функція виконує основн обчислення та повертає результати, які обробляються серверною частиною та надсилаються назад клієнту у форматі JSON. Запити до MATLAB Production Server виконуються з серверу за допомогою відповідного маршруту, враховуючи назви скомпільованого ctf-архіву, та основної функції, що він містить – відповідно до документації MATLAB (рисунок 4.1).

```
MPS_FUNCTION_URL = "http://localhost:9910/HybridSystemSimulation/simulateHybridSystem"
```

Рис. 4.1 – Запит до сервера MATLAB

Клієнтська частина отримує відповідь з сервера у форматі JSON та виводить отримані результати у зручному для користувача вигляді. У базовій реалізації дані виводяться у форматі структурованого тексту, що дозволяє переглядати числові значення усіх параметрів моделювання. Однак для кращого сприйняття та аналізу динаміки системи, в інтерфейс було додано графічну візуалізацію результатів із використанням бібліотеки Chart.js.

Chart.js — це сучасна, легка JavaScript-бібліотека для побудови інтерактивних графіків, яка має вбудовану підтримку різних типів діаграм: лінійних, стовпчикових, кругових, комбінованих тощо. Її інтеграція з React реалізується через обгортку react-chartjs-2, яка дозволяє вставляти графіки безпосередньо в JSX-структуру компонентів.

У компоненті EnergyCharts.tsx реалізовано логіку перетворення отриманих з сервера результатів у формат, придатний для візуалізації. Зокрема, дані форматуються у вигляді масивів значень, які подаються на вхід графіків як масиви labels і datasets. Користувач може спостерігати на побудованих графіках, як автоматично змінюються значення ключових параметріх в ході послідовних симуляцій. Це особливо зручно у випадку автоматичного оновлення моделювання, коли нові дані надходять регулярно,

наприклад, кожні 30 секунд, і графіки оновлюються відповідно до останніх обчислень.

Крім того, Chart.js підтримує інтерактивність: наведення курсору миші на точку або сектор діаграми дозволяє побачити точні значення параметрів. Це робить інтерфейс більш зручним та придатним для порівняльного аналізу роботи гібридної енергосистеми у різних умовах.

Таким чином, використання Chart.js для візуалізації результатів обчислень дозволяє значно покращити інформативність системи та її зручність для середнього користувача. Графіки не лише доповнюють числову інформацію, а й дають змогу швидко оцінити загальні тенденції, виявити граничні значення, та порівняти ефективність різних сценаріїв роботи енергосистеми.

Таким чином, забезпечено повноцінний алгоритм від заповнення параметрів користувачем, вручну чи за допомогою методів генерації даних, до отримання результатів обчислень у вебінтерфейсі. Комунікація між усіма частинами системи відбувається за допомогою стандартних технологій у вигляді HTTP та API запитів та форматів JSON, що робить систему гнучкою і масштабованою.

4.3 Алгоритм моделювання енергосистеми

Обчислювальний модуль системи реалізовано у вигляді функції simulateHybridSystem.m, яка виконується у середовищі MATLAB Production Server. Вона приймає набір параметрів, що характеризують компоненти гібридної енергосистеми, виконує розрахунки та формує результат у структурованому вигляді (рисунок 4.2).

```
if net_power_after_solar > 0 % якщо є надлишок сонячної енергії - спробувати зарядити батарею
    surplus_power_w = net_power_after_solar;

    % Скільки енергії ще можна зарядити до макс SOC (Вт*год)
    energy_to_max_soc_wh = battery_max_soc_wh - battery_energy_current_wh;

    % Максимальна потужність заряду, обмежена SOC (Вт)
    charge_power_limit_soc_w = energy_to_max_soc_wh / simulation_step_hours;


    % Потужність що піде на заряд батареї
    power_to_charge_gross_w = min([surplus_power_w, ...
                                   battery_max_charge_power_w, ...
                                   charge_power_limit_soc_w]);
    power_to_charge_gross_w = max(0, power_to_charge_gross_w);


    % Енергія що збережеться в батареї
    energy_charged_net_wh = power_to_charge_gross_w * battery_charge_eff * simulation_step_hours;

    battery_energy_current_wh = battery_energy_current_wh + energy_charged_net_wh;
    power_battery_flow_w = power_to_charge_gross_w; % Додатний потік - заряд


    % Невикористаний надлишок сонячної енергії
    solar_power_curtailed_w = surplus_power_w - power_to_charge_gross_w;
    solar_power_curtailed_w = max(0, solar_power_curtailed_w);

    % Дефіцит для ДГ = 0, оскільки був надлишок
    remaining_deficit_for_diesel_w = 0;
```

Рис. 4.2 – Фрагмент основної функції обчислювального модуля

Основна ідея моделювання полягає в тому, щоб визначити, яку частину електричного навантаження може покрити сонячна генерація, скільки енергії буде задіяно з батареї, яку частину навантаження доведеться компенсувати дизельним генератором, а також обчислити витрати пального, втрати генерації та кінцевий рівень заряду батареї.

Розрахунок у модулі simulateHybridSystem.m розпочинається з визначення потенційної потужності, яку може згенерувати сонячна панель у заданий момент часу. Для цього використовуються наступні вхідні параметри: площа панелі), ефективність фотомодуля n та поточний рівень сонячного випромінювання GHI. Потужність сонячної генерації розраховується за формулою P_solar = A * n * GHI

Це значення порівнюється з навантаженням системи, тобто кількістю електроенергії, яку необхідно подати до споживачів у цей момент часу. Далі виконуються умовні перевірки та розгалуження, які визначають, як саме буде

покрите навантаження: виключно за рахунок сонячної генерації, із залученням батареї, або з підключенням дизельного генератора.

Якщо сонячна генерація покриває все навантаження, система фіксує повне покриття навантаження з фотоелементів, з надлишком генерації. У такому випадку виконується перевірка: чи є в батареї вільна ємність для збереження надлишкової енергії. Якщо так — надлишок зберігається з урахуванням коефіцієнта ефективності заряду. Якщо ж батарея повністю заряджена або її максимальний стан заряду перевищено, надлишок вважається втраченим і цей обсяг також фіксується у вихідному об’єкті.

Якщо сонячної енергії недостатньо, спочатку покривається доступна частина навантаження за рахунок сонячної генерації. Різниця між потребою та сонячною генерацією вважається дефіцитом, який необхідно компенсувати з інших джерел.

У першу чергу задіюється акумуляторна батарея. Виконується перевірка поточного стану заряду і обчислюється, скільки енергії можна забрати без перевищення межі розряду. При цьому враховується ефективність розряду та допустимий мінімальний заряд. Якщо батарея має достатній заряд, вона таким чином покриває весь або частину дефіциту. Стан заряду батареї зменшується відповідно до витраченої енергії та втрат на розряд.

Якщо після використання батареї все ще залишається непокрите навантаження, запускається дизельний генератор. Його потужність обмежена заданим параметром P_diesel_max. Генератор подає енергію в межах своєї потужності — якщо і цього недостатньо, фіксується частина навантаження, яку не вдалося покрити жодним із джерел.

Крім цього, обчислюється обсяг спожитого пального дизельним генератором. Для цього використовується питомий коефіцієнт витрат пального fuel_rate, і кількість пального розраховується як Fuel = P_diesel * fuel_rate

Це значення додається до загального споживання ресурсу й фіксується у структурі результату.

Стан заряду SOC батареї оновлюється після кожної симуляції відповідно до отриманого заряду з надлишку сонячної генерації, витраченої енергії на покриття дефіциту та ефективності заряду чи розряду.

Всі зміни SOC обмежуються встановленими реалстичними межами — батарея не може зарядитися більше 100% і не повинна розряджатися нижче допустимого мінімуму.

Після завершення розрахунків функція формує структурований об’єкт із результатами, який містить такі поля:

- solarGeneration — вироблена сонячна енергія;
- batteryUsed — енергія, взята з батареї;
- batteryCharged — енергія, збережена в батареї;
- SOC — оновлений рівень заряду;
- dieselUsed — спожите дизельне пальне;
- unmetLoad — обсяг непокритого навантаження;
- curtailedSolar — втрачена сонячна енергія, яку не вдалося зберегти.

Цей об’єкт кодується у форматі JSON і повертається до серверної частини для подальшого відображення у вебінтерфейсі.

Таким чином, запропонований алгоритм моделювання є логічно завершеним, реалістичним і гнучким. Він охоплює всі ключові компоненти гібридної енергосистеми та дозволяє детально аналізувати її поведінку в умовах змінного навантаження, нестабільного сонячного випромінювання та обмежених енергетичних ресурсів. Крім того, структура функції simulateHybridSystem.m передбачає можливість подальшого розширення — наприклад, за рахунок додавання нового типу генерації, моделювання втрат або впровадження економічного блоку.

Цей модуль є основою для математичної моделі всієї розробленої системи та забезпечує ключовий функціонал моделювання в реальному часі з високою точністю та чітким алгоритмом (рисунок 4.3).

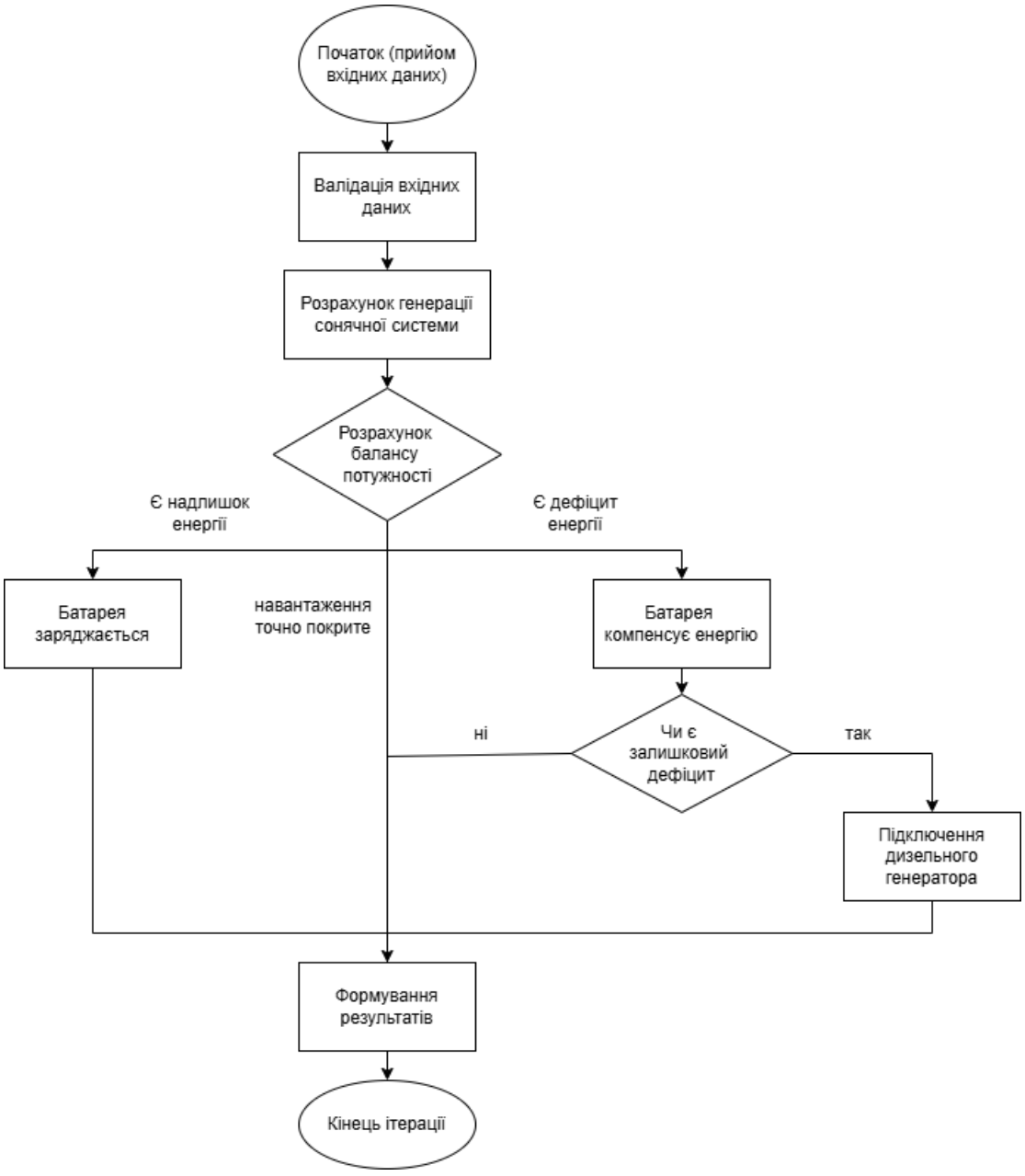


Рис. 4.3 – Алгоритм роботи обчислювального модуля

У результаті виконання цього алгоритму система отримує точну оцінку таких параметрів, як:

- обсяг згенерованої сонячної енергії;
- кількість енергії, отриманої з батареї або збереженої в ній;
- рівень залишкового заряду акумулятора;

- спожите дизельне паливо;
- величина непокритого навантаження;
- обсяг втраченої сонячної генерації.

Усі ці дані повертаються у вигляді структурованого об’єкта JSON та передаються назад до серверної частини системи.

Завдяки використанню MATLAB Production Server цей модуль може працювати у режимі віддаленого обчислення з мінімальними затримками, що дозволяє досягти ефекту моделювання в модельному або реальному часі. Алгоритм є універсальним і може бути доповнений новими елементами, та розгалуженнями.

Висновки до розділу 4

У четвертому розділі, у межах реалізації системи вдалося забезпечити повний алгоритм роботи від авторизації користувача до отримання результатів симуляції. Система підтримує як ручне, так і автоматичне введення параметрів, передає їх до MATLAB Production Server, де виконується точне моделювання процесів генерації, зберігання й споживання енергії. Отримані результати можуть бути використані для подальшого аналізу ефективності гібридної енергосистеми.

РОЗДІЛ 5 КЕРІВНИЦТВО КОРИСТУВАЧА

У цьому розділі наведено інструкцію щодо користування розробленою вебсистемою моделювання гібридної енергосистеми. Система створена з урахуванням зручності та інтуїтивності взаємодії, тому навіть користувачі без глибоких технічних знань можуть легко працювати з нею. Усі дії виконуються через вебінтерфейс за допомогою браузера.

5.1 Реєстрація та вхід

Для початку роботи користувачу необхідно відкрити вебінтерфейс системи у браузері, перейшовши за відповідним посиланням. Після завантаження відкривається головна сторінка авторизації. На початковому екрані користувач бачить форму входу та кнопку реєстрації.

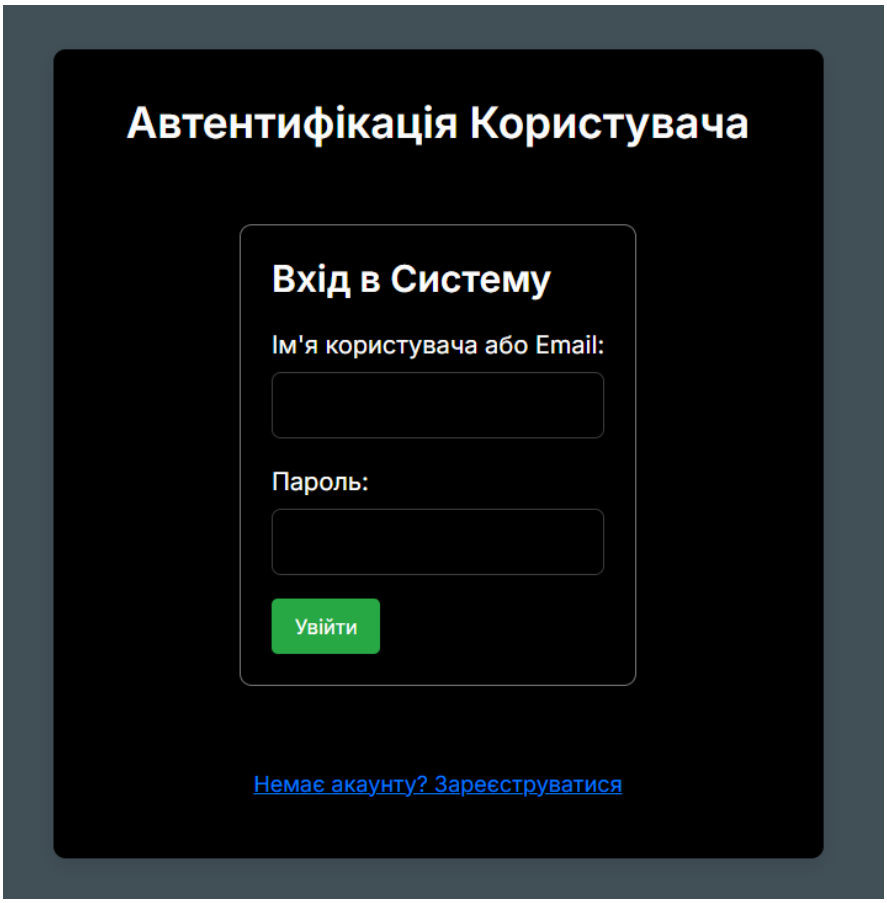


Рис. 5.1 – Форма для авторизаціїї

Якщо користувач вперше працює із системою, він натискає відповдну кнопку для реєстраці після чого відкривається форма створення нового облкового запису, в якій необхідно ввести ім’я, електронну адресу та пароль для нового користувача (рисунок 5.2).

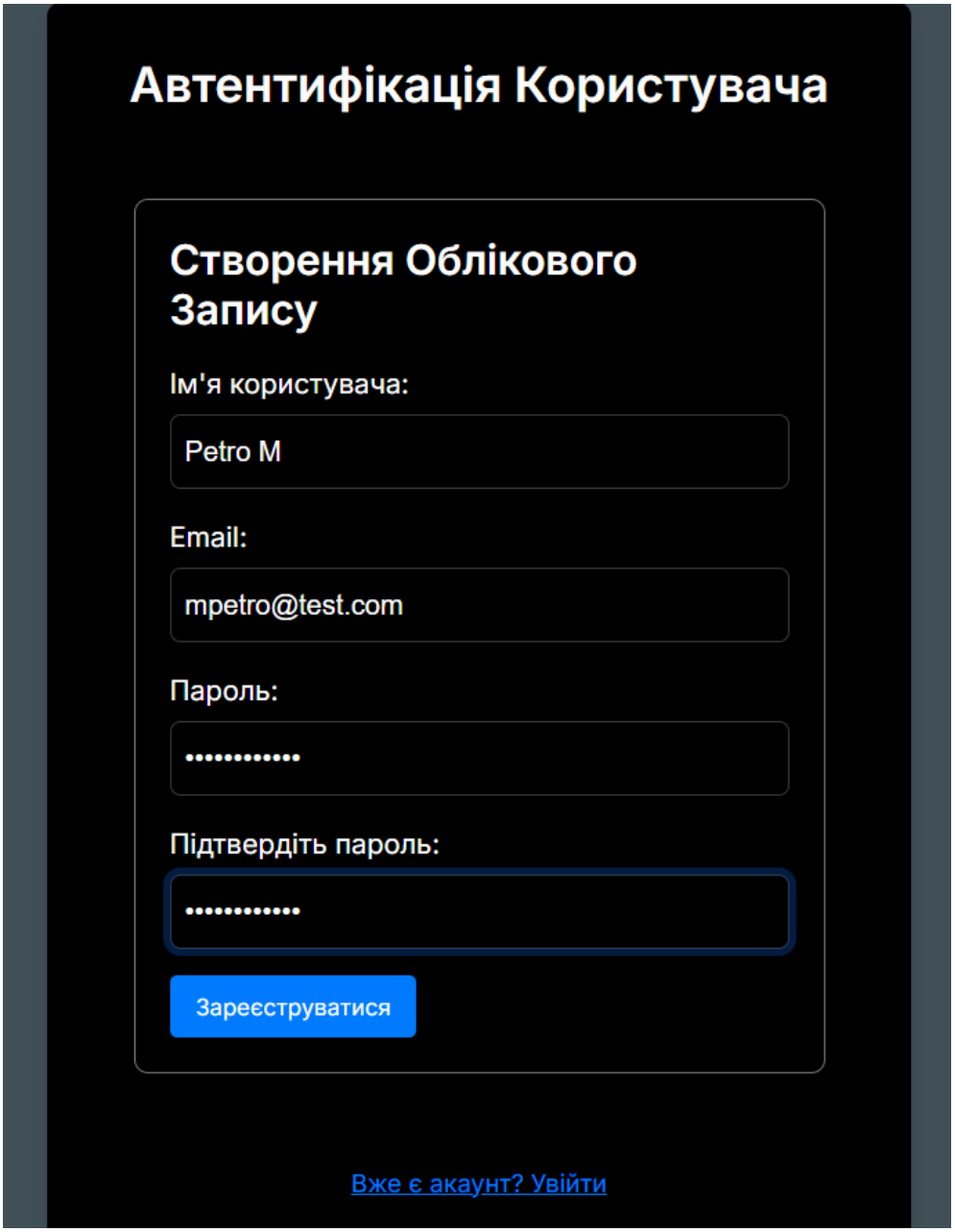


Рис. 5.2 – Форма для реєстрації

Після введення необхідних даних та натискання кпонки реєстрації, система перевіряє, чи вже існує такий користувач, і у разі успішної реєстрації, якщо користувач унікальний, то клієнту виводиться відповідне

повідомлення про успішну реєстрацію, та надається можливість увійти в систему для подальшого користування (рисунок 5.3).

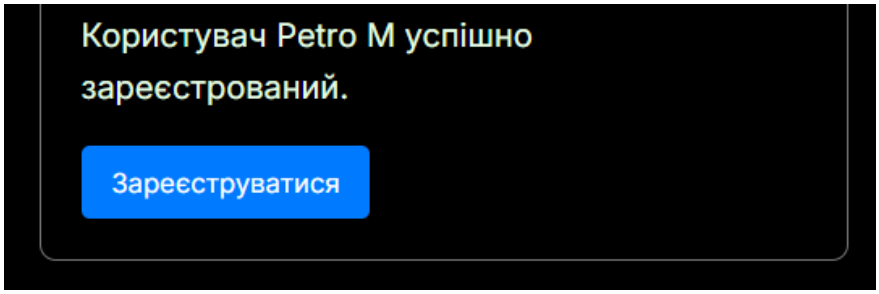


Рис. 5.3 – Повідомлення про успішну реєстрацію

Для входу до системи достатньо ввести логін, у вигляді імені користувача або електронної адреси, та пароль, після чого натиснути кнопку для входу. У разі правильних даних користувач отримує доступ до особистого кабінету. Якщо введено некоректні дані — система повідомляє про помилку. Після входу автентифікація зберігається за допомогою токена, і користувач може працювати із захищеними функціями без повторного входу протягом поточної сесії.

5.2 Задання вхідних параметрів та отримання результатів

Після успішної авторизації користувач потрапляє на сторінку особистого кабінету, звідки він може і перейти до основного функціоналу моделювання гібридної енергосистеми, де йому і представлено вибір між головними опціями, такими як ручне введення вхідних параметрів симуляції, перемикання до автоматичного режиму роботи системи та перегляд візуалізації результатів моделювання.

Інтерфейс моделювання енергосистеми має доволі просту структуру: у центрі сторінки розташована форма для введення даних, під нею знаходиться панель для виводу результатів обчислення, а вгорі — панель навігації та інформації про сесію користувача.

На головній сторінці моделювання користувач має можливість вручну ввести вхідні дані для обчислення (рисунок 5.4). Для цього передбачено окремі поля з відповідними підписами для наступних змінних:

- площа сонячної панелі (м²);
- ефективність сонячної панелі (%);
- рівень сонячної радіації (Вт/м²);
- загальне навантаження (Вт);
- потужність дизельного генератора (Вт);
- ємність батареї (Вт·год);
- поточний стан заряду SOC батареї (%).

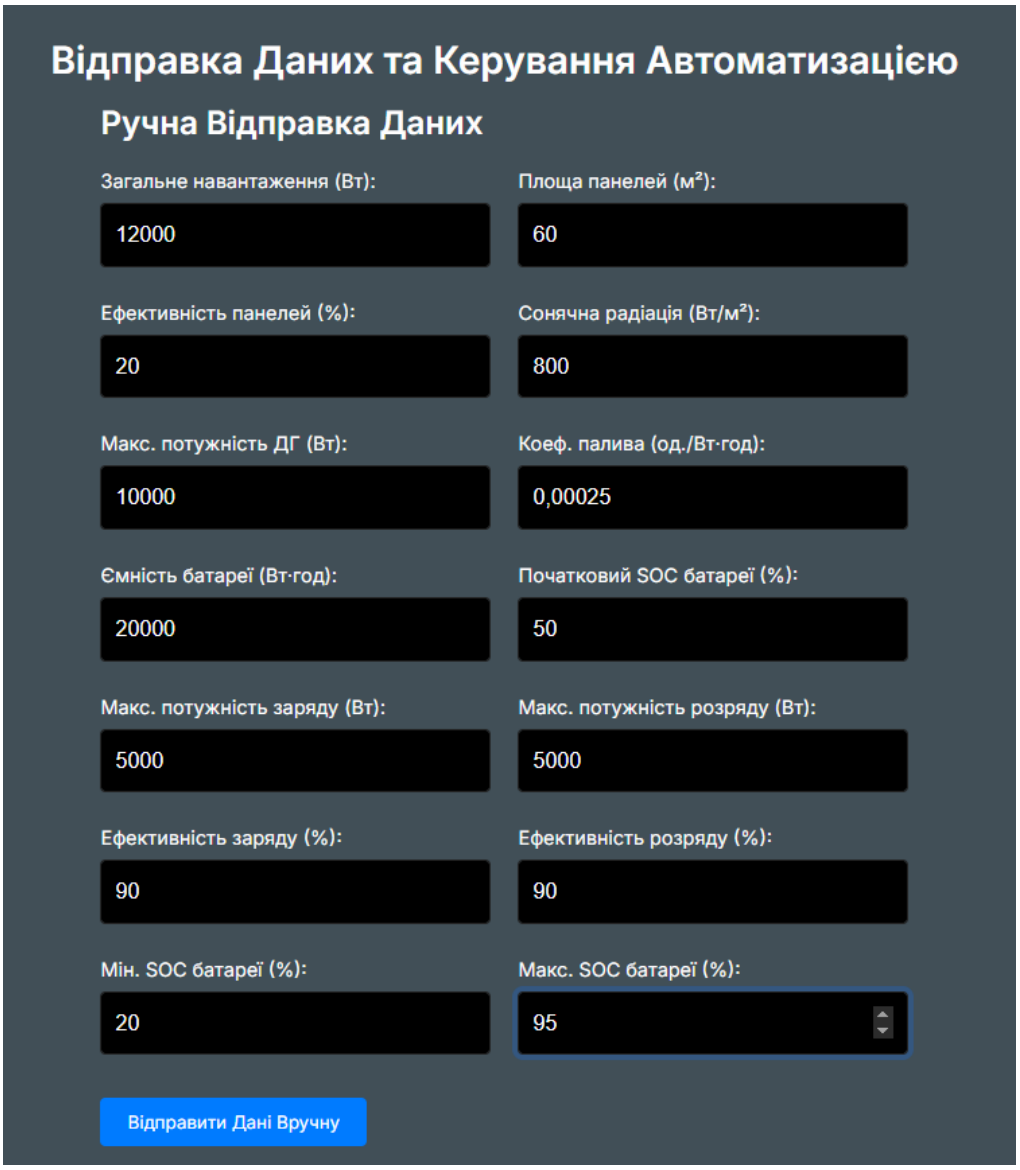


Рис. 5.4 – Форма для введення даних

Після заповнення даної форми користувач натискає на кнопку для запуску моделювання, яка надсилає всі вхідні параметри на сервер. У разі некоректного введення, наприклад від'ємних значень де це є недопустимим, або нечислових значень, система виводить повідомлення про помилку валідації даних.

Користувач також може активувати режим автоматичного моделювання, яке запускає симуляції кожні 30 секунд, використовує збережене значення стану заряду SOC батареї з попередньої симуляції та оновлює параметри генерації та навантаження (рисунок 5.5). Цей режим дозволяє імітувати зміну параметрів та спостерігати за поведінкою системи у тривалі періоди часу та виявляти періоди найбільшого перевантаження, дефіциту генерації чи надлишків енергії.

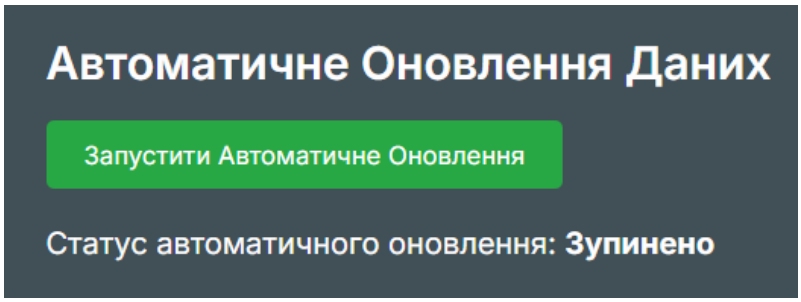


Рис. 5.5 – Кнопка для запуску автоматичного оновлення даних

Після виконання обчислень, результати виводяться у нижній частині сторінки. Користувач бачить наступні результати:

- згенеровану сонячну енергію;
- енергію, подану з батареї;
- енергію, збережену в батареї;
- залишковий рівень заряду (SOC);
- споживання дизельного пального;
- покрите та непокрите навантаження;
- втрати надлишкової енергії.

Результати виводяться у вигляді формату JSON одразу після кнопок запуску моделювання, відповідно для ручного вводу та автоматичного оновлення (рисунок 5.6).

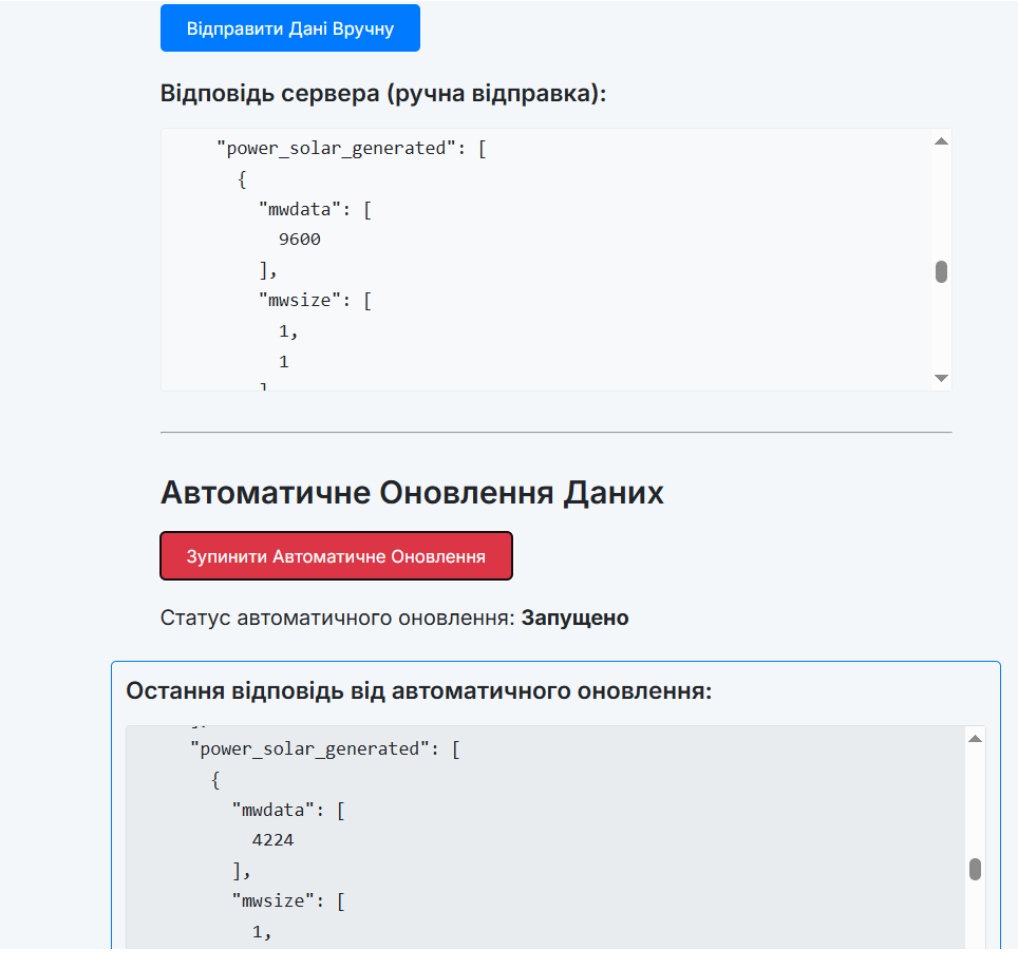


Рис. 5.6 – Вивід результатів у форматі JSON

Окрім текстового представлення результатів у форматі JSON, у системі також реалізована графічна візуалізація отриманих даних, що значно покращує зручність аналізу. Для побудови графіків використовується популярна бібліотека Chart.js, яка дозволяє швидко та гнучко створювати інтерактивні діаграми в межах React-компонентів.

На сторінці з результатами моделювання користувач може переглянути графіки, які відображають:

- зміну згенерованої сонячної енергії;
- рівень заряду батареї в часі;
- обсяг спожитого дизельного пального;

- значення покритого та непокритого навантаження;
- втрати надлишкової енергії.

Графіки автоматично оновлюються після кожного нового моделювання або при надходженні нових даних у режимі автоматичного оновлення. Вони відображають динаміку параметрів у доволі зручному вигляді — у форматі лінійних діаграм, де осі абсцис відповідають за періоди часу, а осі ординат – за значення потужності у Вт. Така графічна візуалізація дозволяє користувачеві наглядно оцінити ефективність роботи енергосистеми та виявити критичні моменти, такі як перевищення навантаження чи недостатній рівень заряду батареї, особливо у періоди найвищого споживання або генерації енергії (рисунок 5.7). Візуалізація результатів із саме такими граничними значеннями є найбільш важливою для подібного моделювання гібридної енергосистеми. Передбачено побудування як і загального графіку, так і окремих графіків для значень сонячної генерації, витрат енергії батереєю чи дизельним генератором, тощо.

Рис. 5.7 – Графічна візуалізація результатів моделювання

Користувач може вийти із системи, натиснувши відповідну кнопку, яка очищує токен авторизації та перенаправляє на сторінку входу (рисунок 5.8). Усі сесійні дані завершуються, і подальший доступ до функціоналу вимагає повторної авторизації.

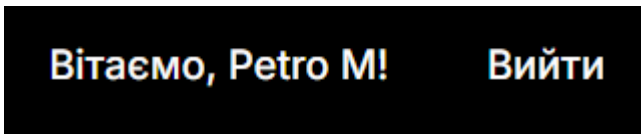


Рис. 5.8 – Кнопка виходу із системи

Висновки до розділу 5

У п’ятому розділі, було продемонстровано основну логіку взаємодії користувача з системою. Розроблена система має гнучкий інтерфейс, що дозволяє забезпечити процесс моделювання гібридної енергосистеми для середнього користувача. Усі функції згруповані логічно, а сценарії роботи — прості та послідовні. Система забезпечує повний функціонал взаємодії, від авторизації та формування параметрів до запуску моделювання й перегляду результатів. Це дозволяє ефективно використовувати вебзастосунок як у навчальних цілях, так і в практичному застосуванні для аналізу поведінки гібридної енергосистеми в різних умовах.

ВИСНОВКИ

У результаті виконання даної дипломної роботи було реалізовано повнофункціональну вебсистему для моделювання гібридної енергосистеми, яка поєднує в собі сонячні панелі, акумуляторні батареї для зберігання енергії та резервне джерело у вигляді дизельного генератора. Система дозволяє користувачеві вводити вхідні параметри, запускати автоматичне моделіювання та отримувати результати у зручному форматі з графічною візуалізацією.

Було реалізовано архітектуру типу клієнт–сервер із трирівневою логікою. Клієнтська частина, реалізована з використанням React і TypeScript, забезпечує інтерфейс для взаємодії користувача з системою. Серверна частина на Flask виконує роль API-сервера та логіки авторизації, перевірки даних і передачі запитів до обчислювального модуля. Для збереження облікових записів використовується локальна база даних SQLite із захистом через JWT-токени.

Особливу увагу приділено математичній моделі, реалізованій у середовищі MATLAB. Завдяки MATLAB Production Server вдалося об’єднати переваги інженерних обчислень із можливістю інтеграції до вебінтерфейсу. Модель враховує реалістичні параметри: генерацію сонячної енергії з урахуванням площі та ефективності панелей, роботу акумулятора з відсотковим рівнем заряду, втрати при заряді/розряді та запуск дизельного генератора у випадку дефіциту. Результати розрахунків містять інформацію про покриття навантаження, витрати пального, втрати енергії, залишкову ємність батареї та інші ключові метрики.

З боку користувача забезпечені реєстрація, вхід, введення параметрів, запуск симуляції, перегляд результатів. Реалізовано як ручний режим запуску, так і автоматичний, з можливістю запуску серії обчислень із динамічними вхідними параметрами.

Таким чином, поставлені завдання були повністю виконані - створено вебсистему моделювання гібридної енергосистеми, реалізовано повноцінну авторизацію, API-зв'язок, обробку даних та інтеграцію з MATLAB Production Server, де забезпечено точні інженерні обчислення, та розроблено графічне відображення результатів у вебінтерфейсі.

Розроблена система має практичну цінність у навчальних, інженерних та дослідницькихі напрямках, може ще бути удосконалена та розширена для роботи з новими джерелами енергї.

Отже, дана дипломна робота довела ефективнсть створення адаптивного вебінструменту для моделювання гібридної енергосистеми з використанням сучасних засобів розробки та інтеграції обчислювальної платформи MATLAB.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Stormy Attaway — Matlab: A Practical Introduction to Programming and Problem Solving — 592 c.

2. Cesar Perez Lopez — Matlab Control Systems Engineering — 180 c.

3. Brian R. Hunt — A Guide to MATLAB: For Beginners and Experienced Users — 346 c.

4. Asmae Berrada & Rachid El Mrabet — Hybrid Energy System Models — 382 c.

5. William H. Kersting, Robert J. Kerestes — Distribution System Modeling and Analysis with MATLAB and WindMil — 476 c.

ДОДАТОК А Лістинг розробленої системи

mt.m :

```
function results = simulateHybridSystem(params)

  % Логування
  fprintf('-------------------------------------------------\n');
  fprintf('[%s] Запуск simulateHybridSystem.\n', datestr(now, 'yyyy-mm-dd HH:MM:SS'));
  fprintf('Вхідні параметри:\n');
  disp(params); % Вивід вхідних параметрів


  % Перевірка вхідних параметрів
  requiredFields = {'load_demand', 'solar_area', 'solar_efficiency', ...
                    'solar_radiation', 'diesel_max_power', 'diesel_fuel_coeff', ...
                    'battery_capacity_wh', 'battery_soc_initial_percent', ...
                    'battery_max_charge_power_w', 'battery_max_discharge_power_w', ...
                    'battery_charge_efficiency', 'battery_discharge_efficiency', ...
                    'battery_min_soc_percent', 'battery_max_soc_percent'};

  for i = 1:length(requiredFields)
    if ~isfield(params, requiredFields{i})
       error('simulateHybridSystem:MissingInput', ...
           ['Відсутнє необхідне поле: ', requiredFields{i}]);
    end
    if ~isnumeric(params.(requiredFields{i})) || ~isscalar(params.(requiredFields{i}))
        error('simulateHybridSystem:InvalidInputType', ...
            ['Параметр ', requiredFields{i}, ' повинен бути числовим скаляром.']);
    end
    if params.(requiredFields{i}) < 0 && ...
      ~strcmp(requiredFields{i}, 'power_battery_flow_w')
        error('simulateHybridSystem:NegativeInput', ...
            ['Параметр ', requiredFields{i}, ' не може бути відємним.']);
    end
  end

  percentage_fields = {'solar_efficiency', 'battery_soc_initial_percent', ...
                       'battery_charge_efficiency', 'battery_discharge_efficiency', ...
                       'battery_min_soc_percent', 'battery_max_soc_percent'};
  for i = 1:length(percentage_fields)
    if params.(percentage_fields{i}) < 0 || params.(percentage_fields{i}) > 100
```

```
            error('simulateHybridSystem:InvalidPercentage', ...
                ['Параметр ', percentage_fields{i}, ' повинен бути в діапазоні 0-100%.']);
        end
    end
    if params.battery_min_soc_percent >= params.battery_max_soc_percent
        error('simulateHybridSystem:InvalidSOClimits', 'Мін. SOC батареї повинен бути меншим за Макс. SOC.');
    end



    load_demand               = params.load_demand;
    solar_area                = params.solar_area;
    solar_efficiency          = params.solar_efficiency / 100;
    solar_radiation           = params.solar_radiation;
    diesel_max_power          = params.diesel_max_power;
    diesel_fuel_coeff         = params.diesel_fuel_coeff;

    battery_capacity_wh       = params.battery_capacity_wh;
    battery_soc_current_percent  = params.battery_soc_initial_percent;
    battery_max_charge_power_w   = params.battery_max_charge_power_w;
    battery_max_discharge_power_w= params.battery_max_discharge_power_w;
    battery_charge_eff        = params.battery_charge_efficiency / 100;
    battery_discharge_eff     = params.battery_discharge_efficiency / 100;
    battery_min_soc_wh        = (params.battery_min_soc_percent / 100) * battery_capacity_wh;
    battery_max_soc_wh        = (params.battery_max_soc_percent / 100) * battery_capacity_wh;



    % Поточна енергія в батареї (Вт*год)
    battery_energy_current_wh = (battery_soc_current_percent / 100) * battery_capacity_wh;

    % Вихідні змінні
    power_battery_flow_w = 0;
    solar_power_curtailed_w = 0;


    % 1. Розрахунок генерації сонячної енергії
    power_solar_generated = solar_area * solar_efficiency * solar_radiation;
    power_solar_generated = max(0, power_solar_generated);

    % 2. Баланс потужності: сонячна генерація та навантаження
```

```
net_power_after_solar = power_solar_generated - load_demand;

% 3. Логіка роботи батареї
simulation_step_hours = 1;

if net_power_after_solar > 0 % якщо є надлишок сонячної енергії - спробувати зарядити батарею
   surplus_power_w = net_power_after_solar;

   % Скільки енергії ще можна зарядити до макс SOC (Вт*год)
   energy_to_max_soc_wh = battery_max_soc_wh - battery_energy_current_wh;

   % Максимальна потужність заряду, обмежена SOC (Вт)
   charge_power_limit_soc_w = energy_to_max_soc_wh / simulation_step_hours;


   % Потужність що піде на заряд батареї
   power_to_charge_gross_w = min([surplus_power_w, ...
                              battery_max_charge_power_w, ...
                              charge_power_limit_soc_w]);
   power_to_charge_gross_w = max(0, power_to_charge_gross_w);


   % Енергія що збережеться в батареї
   energy_charged_net_wh = power_to_charge_gross_w * battery_charge_eff * simulation_step_hours;

   battery_energy_current_wh = battery_energy_current_wh + energy_charged_net_wh;
   power_battery_flow_w = power_to_charge_gross_w; % Додатний потік - заряд


   % Невикористаний надлишок сонячної енергії
   solar_power_curtailed_w = surplus_power_w - power_to_charge_gross_w;
   solar_power_curtailed_w = max(0, solar_power_curtailed_w);

   % Дефіцит для ДГ = 0, оскільки був надлишок
   remaining_deficit_for_diesel_w = 0;

elseif net_power_after_solar < 0 % якщо є дефіцит - спробувати розрядити батарею
   deficit_power_w = -net_power_after_solar;

   % Скільки енергії можна розрядити до мін SOC (Вт*год)
   energy_to_min_soc_wh = battery_energy_current_wh - battery_min_soc_wh;
```

```
        % Максимальна потужність розряду, обмежена SOC (Вт)
        discharge_power_limit_soc_w = energy_to_min_soc_wh / simulation_step_hours;


        % Потужність яку батарея може віддати на навантаження
        power_from_battery_gross_w = min([deficit_power_w / battery_discharge_eff, ...
                                          battery_max_discharge_power_w, ...
                                          discharge_power_limit_soc_w]);
        power_from_battery_gross_w = max(0, power_from_battery_gross_w);


        % Енергія, що фактично забирається з батареї
        energy_discharged_gross_wh = power_from_battery_gross_w * simulation_step_hours;

        battery_energy_current_wh = battery_energy_current_wh - energy_discharged_gross_wh;
        power_battery_flow_w = -power_from_battery_gross_w; % Відємний потік - розряд


        % Потужність що доставлена на навантаження від батареї
        power_supplied_by_battery_net_w = power_from_battery_gross_w * battery_discharge_eff;


        % Залишковий дефіцит для покриття дизелем
        remaining_deficit_for_diesel_w = deficit_power_w - power_supplied_by_battery_net_w;
        remaining_deficit_for_diesel_w = max(0, remaining_deficit_for_diesel_w);

    else % сонячна енергія точно покриває навантаження
        power_battery_flow_w = 0;
        remaining_deficit_for_diesel_w = 0;
        solar_power_curtailed_w = 0;
    end


    % Оновлення SOC у відсотках
    battery_soc_final_percent = (battery_energy_current_wh / battery_capacity_wh) * 100;
    battery_soc_final_percent = max(params.battery_min_soc_percent,
min(params.battery_max_soc_percent, battery_soc_final_percent));
    battery_soc_final_percent = max(0, min(100, battery_soc_final_percent));


    % 4. Розрахунок роботи дизельного генератора на залишковий дефіцит
    power_diesel_generated = 0;
```

```
    fuel_consumed = 0;

    if remaining_deficit_for_diesel_w > 0
        power_diesel_generated = min(remaining_deficit_for_diesel_w, diesel_max_power);
        if power_diesel_generated > 0
            fuel_consumed = power_diesel_generated * diesel_fuel_coeff * simulation_step_hours;
        end
    end


    % 5. Розрахунок загальної забезпеченої потужності та непокритого навантаження
    power_supplied_by_solar_direct_to_load = power_solar_generated - max(0, net_power_after_solar - solar_power_curtailed_w);
    if net_power_after_solar > 0 % був надлишок
        power_supplied_by_battery_net_w = 0; % батарея заряджалася, не віддавала на навантаження
    else % був дефіцит або баланс
        % power_supplied_by_battery_net_w вже розрахована
    end

    total_power_supplied = power_solar_generated + (power_battery_flow_w < 0) * (abs(power_battery_flow_w) * battery_discharge_eff) + power_diesel_generated - solar_power_curtailed_w;

    power_supplied_by_battery_effective = 0;
    if power_battery_flow_w < 0 % батарея розряджалася
        power_supplied_by_battery_effective = abs(power_battery_flow_w) * battery_discharge_eff;
    end

    total_power_supplied = (power_solar_generated - solar_power_curtailed_w - (power_battery_flow_w > 0)*power_battery_flow_w) + ...
                           power_supplied_by_battery_effective + ...
                           power_diesel_generated;

    unmet_load = load_demand - total_power_supplied;
    unmet_load = max(0, unmet_load);



    % Формування результатів
    results.power_solar_generated = round(power_solar_generated, 2);
    results.power_diesel_generated = round(power_diesel_generated, 2);
    results.power_battery_flow_w = round(power_battery_flow_w, 2);
```

```
    results.battery_soc_final_percent = round(battery_soc_final_percent, 2);
    results.fuel_consumed = round(fuel_consumed, 4);
    results.load_demand_actual = round(load_demand, 2);
    results.total_power_supplied = round(total_power_supplied, 2);
    results.unmet_load = round(unmet_load, 2);
    results.solar_power_curtailed_w = round(solar_power_curtailed_w, 2);


    % Логування результатів
    fprintf('Результати симуляції:\n');
    disp(results);
    fprintf('[%s] Завершення simulateHybridSystem.\n', datestr(now, 'yyyy-mm-dd HH:MM:SS'));
    fprintf('--------------------------------------------------\n');
end
```

bckn/app.py :

```
import os
from flask import Flask, request, jsonify, Blueprint
import logging
from flask_cors import CORS
import requests
from werkzeug.security import generate_password_hash, check_password_hash
from flask_sqlalchemy import SQLAlchemy
from flask_jwt_extended import create_access_token, jwt_required, JWTManager, get_jwt_identity


app = Flask(__name__)
CORS(app)

logging.basicConfig(level=logging.INFO, format='%(asctime)s %(levelname)s:%(module)s:%(lineno)d
%(message)s')


basedir = os.path.abspath(os.path.dirname(__file__))
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///' + os.path.join(basedir, 'users.db')
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False


app.config["JWT_SECRET_KEY"] = "e805b18d24dd60d9a358a5f0b8e7f0dee1d2"
```

```
jwt = JWTManager(app)

db = SQLAlchemy(app)

MPS_FUNCTION_URL = "http://localhost:9910/HybridSystemSimulation/simulateHybridSystem"



class User(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(80), unique=True, nullable=False)
    email = db.Column(db.String(120), unique=True, nullable=False)
    password_hash = db.Column(db.String(200), nullable=False)

    def __repr__(self):
        return f'<User {self.username}>'



auth_bp = Blueprint('auth_bp', __name__, url_prefix='/api/auth')


@auth_bp.route('/register', methods=['POST'])
def register_user():
    data = request.get_json()
    if not data:
        return jsonify({"error": "Запит повинен містити JSON дані"}), 400

    username = data.get('username')
    email = data.get('email')
    password = data.get('password_hash')

    app.logger.info(f"Спроба реєстрації: username='{username}', email='{email}'")

    if not username or not email or not password:
        return jsonify({"error": "Будь ласка, заповніть усі поля: ім'я користувача, email та пароль"}), 400

    if len(password) < 6:
        return jsonify({"error": "Пароль має містити щонайменше 6 символів"}), 400

    if User.query.filter_by(email=email).first():
        app.logger.warning(f"Спроба реєстрації з існуючим email: {email}")
```

```
        return jsonify({"error": "Користувач з таким email вже існує"}), 409
    if User.query.filter_by(username=username).first():
        app.logger.warning(f"Спроба реєстрації з існуючим username: {username}")
        return jsonify({"error": "Ім'я користувача вже зайняте"}), 409

    hashed_password = generate_password_hash(password, method='pbkdf2:sha256')
    new_user = User(username=username, email=email, password_hash=hashed_password)

    try:
        db.session.add(new_user)
        db.session.commit()
        app.logger.info(f"Користувач '{username}' успішно зареєстрований. ID: {new_user.id}")
        return jsonify({
            "message": f"Користувач {username} успішно зареєстрований.",
            "user": {"id": new_user.id, "username": new_user.username, "email": new_user.email}
        }), 201
    except Exception as e:
        db.session.rollback()
        app.logger.error(f"Помилка при збереженні користувача '{username}' в БД: {str(e)}")
        return jsonify({"error": "Помилка на сервері при створенні користувача"}), 500


@auth_bp.route('/login', methods=['POST'])
def login_user():
    data = request.get_json()
    if not data:
        return jsonify({"error": "Запит повинен містити JSON дані"}), 400

    identifier = data.get('identifier')
    password = data.get('password_hash')

    if not identifier or not password:
        return jsonify({"error": "Будь ласка, вкажіть ім'я користувача/email та пароль"}), 400

    user_by_email = User.query.filter_by(email=identifier).first()
    user_by_username = User.query.filter_by(username=identifier).first()
    user = user_by_email if user_by_email else user_by_username

    if user and check_password_hash(user.password_hash, password):
        access_token = create_access_token(identity={"id": user.id, "username": user.username, "email":
user.email})
        app.logger.info(f"Користувач '{user.username}' успішно увійшов в систему.")
```

```
        return jsonify({
            "message": "Вхід успішний!",
            "access_token": access_token,
            "user": {"id": user.id, "username": user.username, "email": user.email}
        }), 200
    else:
        app.logger.warning(f"Невдала спроба входу для ідентифікатора: {identifier}")
        return jsonify({"error": "Невірне ім'я користувача/email або пароль"}), 401




@app.route('/api/simulate_local', methods=['POST'])
def handle_simulation_data():
    try:
        if not request.is_json:
            return jsonify({"error": "Запит повинен бути у форматі JSON"}), 400

        data_from_frontend = request.get_json()
        app.logger.info(f"Отримано дані для симуляції: {data_from_frontend}")

        required_keys = [
            'load_demand', 'solar_area', 'solar_efficiency', 'solar_radiation',
            'diesel_max_power', 'diesel_fuel_coeff',
            'battery_capacity_wh', 'battery_soc_initial_percent',
            'battery_max_charge_power_w', 'battery_max_discharge_power_w',
            'battery_charge_efficiency', 'battery_discharge_efficiency',
            'battery_min_soc_percent', 'battery_max_soc_percent'
        ]
        missing_keys = [key for key in required_keys if key not in data_from_frontend]
        if missing_keys:
            app.logger.error(f"Відсутні необхідні параметри: {', '.join(missing_keys)}")
            return jsonify({"error": f"Відсутні необхідні параметри: {', '.join(missing_keys)}"}), 400

        mps_payload = { "nargout": 1, "rhs": [data_from_frontend] }
        app.logger.info(f"Надсилання на MPS ({MPS_FUNCTION_URL}) з тілом: {mps_payload}")

        try:
            response_mps = requests.post(MPS_FUNCTION_URL, json=mps_payload, timeout=60)
            response_mps.raise_for_status()
            mps_results_raw = response_mps.json()
            app.logger.info(f"Отримано відповідь від MPS: {mps_results_raw}")
```

```
            if mps_results_raw and 'lhs' in mps_results_raw and len(mps_results_raw['lhs']) > 0:
                results_from_mps = mps_results_raw['lhs'][0]
                results_from_mps['message'] = "Дані успішно оброблені на Matlab Production Server"
                results_from_mps['processed_by'] = "MPS"
            else:
                app.logger.error(f"Неочікуваний формат відповіді від MPS: {mps_results_raw}")
                return jsonify({"error": "Не вдалося отримати коректні результати від MPS", "details": mps_results_raw}), 500

        except requests.exceptions.Timeout:
            app.logger.error(f"Timeout при запиті до MPS: {MPS_FUNCTION_URL}")
            return jsonify({"error": "Сервер моделювання (MPS) не відповів вчасно (timeout)"}), 504
        except requests.exceptions.ConnectionError:
            app.logger.error(f"Помилка з'єднання з MPS: {MPS_FUNCTION_URL}")
            return jsonify({"error": "Не вдалося з'єднатися з сервером моделювання (MPS)"}), 503
        except requests.exceptions.HTTPError as e_http:
            app.logger.error(f"HTTP помилка від MPS: Статус {e_http.response.status_code}, Відповідь: {e_http.response.text}")
            try:
                mps_error_details = e_http.response.json()
            except ValueError:
                mps_error_details = e_http.response.text
            return jsonify({
                "error": f"Сервер моделювання (MPS) повернув помилку {e_http.response.status_code}",
                "mps_details": mps_error_details
            }), e_http.response.status_code
        except Exception as e_req:
            app.logger.error(f"Непередбачена помилка при взаємодії з MPS: {str(e_req)}")
            return jsonify({"error": "Непередбачена помилка при обробці запиту сервером моделювання"}), 500

        app.logger.info(f"Результати (від MPS) для надсилання на frontend: {results_from_mps}")
        return jsonify(results_from_mps)

    except ValueError as ve: # Для помилок валідації JSON або ключів
        app.logger.error(f"Помилка валідації вхідних даних від frontend: {str(ve)}")
        return jsonify({"error": "Помилка вхідних даних", "details": str(ve)}), 400
    except Exception as e:
        app.logger.error(f"Загальна помилка в handle_simulation_data: {str(e)}")
        return jsonify({"error": "Внутрішня помилка сервера Python", "details": str(e)}), 500
```

```
app.register_blueprint(auth_bp)


def init_db():
    with app.app_context():
        app.logger.info("Ініціалізація та створення таблиць бази даних (якщо потрібно)...")
        db.create_all()
        app.logger.info("База даних готова.")



if __name__ == '__main__':
    init_db()
app.run(host='0.0.0.0', port=5000, debug=True)
```

frnt/my-energy-crm/DataSenderForm.tsx:

```
import React, { useState } from "react";
import type { InputParams } from "./EnergyCharts";

//Початкові значення для форми
const initialFormData: InputParams = {
  load_demand: 12000,
  solar_area: 60,
  solar_efficiency: 20,
  solar_radiation: 800,
  diesel_max_power: 10000,
  diesel_fuel_coeff: 0.00025,
  battery_capacity_wh: 20000,
  battery_soc_initial_percent: 50,
  battery_max_charge_power_w: 5000,
  battery_max_discharge_power_w: 5000,
  battery_charge_efficiency: 90,
  battery_discharge_efficiency: 90,
  battery_min_soc_percent: 20,
  battery_max_soc_percent: 95,
};
```

```
interface FormFieldProps {
  name: keyof InputParams;
  label: string;
  value: number;
  handleChange: (e: React.ChangeEvent<HTMLInputElement>) => void;
  type?: string;
  step?: string;
  min?: string;
  max?: string;
}

const FormField: React.FC<FormFieldProps> = ({
  name,
  label,
  value,
  handleChange,
  type = "number",
  step = "any",
  min,
  max,
}) => (
  <div style={{ marginBottom: "12px" }}>
    <label
      htmlFor={name}
      style={{ display: "block", marginBottom: "5px", fontSize: "0.9em" }}
    >
      {label}:
    </label>
    <input
      type={type}
      id={name}
      name={name}
      value={value}
      onChange={handleChange}
      step={step}
      min={min}
      max={max}
      style={{
        width: "100%",
        padding: "10px",
        boxSizing: "border-box",
```

```
        borderRadius: "4px",
        border: "1px solid #ccc",
      }}
    />
  </div>
);

interface DataSenderFormProps {
  onManualSubmit: (data: InputParams) => Promise<any>; //Функція для ручної відправки
  onToggleAutoUpdate: () => void; //Функція для перемикання авто-оновлення
  isAutoUpdateRunning: boolean; //Статус авто-оновлення
}

const DataSenderForm: React.FC<DataSenderFormProps> = ({
  onManualSubmit,
  onToggleAutoUpdate,
  isAutoUpdateRunning,
}) => {
  const [formData, setFormData] = useState<InputParams>(initialFormData);
  const [manualResponse, setManualResponse] = useState<any>(null);
  const [isSubmitting, setIsSubmitting] = useState<boolean>(false);

  const handleChange = (e: React.ChangeEvent<HTMLInputElement>) => {
    const { name, value } = e.target;
    setFormData((prev) => ({ ...prev, [name]: parseFloat(value) || 0 }));
  };

  const handleSubmit = async (e: React.FormEvent<HTMLFormElement>) => {
    e.preventDefault();
    setIsSubmitting(true);
    setManualResponse("Надсилання даних...");
    try {
      const result = await onManualSubmit(formData);
      setManualResponse(result);
    } catch (error: any) {
      setManualResponse({ error: error.message || "Невідома помилка" });
    } finally {
      setIsSubmitting(false);
    }
  };

  const formFieldsMeta: Array<Omit<FormFieldProps, "value" | "handleChange">> =
```

```
[
 { name: "load_demand", label: "Загальне навантаження (Вт)" },
 { name: "solar_area", label: "Площа панелей (м²)" },
 {
  name: "solar_efficiency",
  label: "Ефективність панелей (%)",
  min: "0",
  max: "100",
 },
 { name: "solar_radiation", label: "Сонячна радіація (Вт/м²)", min: "0" },
 { name: "diesel_max_power", label: "Макс. потужність ДГ (Вт)", min: "0" },
 {
  name: "diesel_fuel_coeff",
  label: "Коеф. палива (од./Вт·год)",
  step: "0.00001",
  min: "0",
 },
 {
  name: "battery_capacity_wh",
  label: "Ємність батареї (Вт·год)",
  min: "0",
 },
 {
  name: "battery_soc_initial_percent",
  label: "Початковий SOC батареї (%)",
  min: "0",
  max: "100",
 },
 {
  name: "battery_max_charge_power_w",
  label: "Макс. потужність заряду (Вт)",
  min: "0",
 },
 {
  name: "battery_max_discharge_power_w",
  label: "Макс. потужність розряду (Вт)",
  min: "0",
 },
 {
  name: "battery_charge_efficiency",
  label: "Ефективність заряду (%)",
  min: "0",
```

```
    max: "100",
  },
  {
    name: "battery_discharge_efficiency",
    label: "Ефективність розряду (%)",
    min: "0",
    max: "100",
  },
  {
    name: "battery_min_soc_percent",
    label: "Мін. SOC батареї (%)",
    min: "0",
    max: "100",
  },
  {
    name: "battery_max_soc_percent",
    label: "Макс. SOC батареї (%)",
    min: "0",
    max: "100",
  },
 ];

return (
  <div style={{ maxWidth: "600px", margin: "0 auto" }}>
    <h3>Ручна Відправка Даних</h3>
    <form onSubmit={handleSubmit}>
      <div
        style={{
          display: "grid",
          gridTemplateColumns: "1fr 1fr",
          gap: "10px 20px",
        }}
      >
        {formFieldsMeta.map((field) => (
          <FormField
            key={field.name}
            name={field.name as keyof InputParams}
            label={field.label}
            value={formData[field.name as keyof InputParams]}
            handleChange={handleChange}
            step={field.step}
            min={field.min}
```

```
        max={field.max}
      />
    ))}
  </div>
  <button
    type="submit"
    disabled={isSubmitting}
    style={{
      padding: "10px 20px",
      marginTop: "20px",
      backgroundColor: "#007bff",
      color: "white",
      border: "none",
      borderRadius: "4px",
      cursor: "pointer",
    }}
  >
    {isSubmitting ? "Надсилання..." : "Відправити Дані Вручну"}
  </button>
</form>
{manualResponse && (
  <div style={{ marginTop: "20px" }}>
    <h4>Відповідь сервера (ручна відправка):</h4>
    <pre
      style={{
        maxHeight: "200px",
        overflowY: "auto",
        backgroundColor: "#f8f9fa",
        padding: "10px",
        border: "1px solid #eee",
      }}
    >
      {JSON.stringify(manualResponse, null, 2)}
    </pre>
  </div>
)}

<hr style={{ margin: "30px 0" }} />

<h3>Автоматичне Оновлення Даних</h3>
<button
  onClick={onToggleAutoUpdate}
```

```
            style={{
              padding: "10px 20px",
              backgroundColor: isAutoUpdateRunning ? "#dc3545" : "#28a745",
              color: "white",
              border: "none",
              borderRadius: "4px",
              cursor: "pointer",
            }}
          >
            {isAutoUpdateRunning
              ? "Зупинити Автоматичне Оновлення"
              : "Запустити Автоматичне Оновлення"}
          </button>
          <p>
            Статус автоматичного оновлення:{" "}
            <strong>{isAutoUpdateRunning ? "Запущено" : "Зупинено"}</strong>
          </p>
        </div>
      );
    };

export default DataSenderForm;
```

ДОДАТОК Б Презентація

Національний технічний університет України
«Київський політехнічний інститутімені Ігоря Сікорського»

Навчально-науковий інститут атомної та теплової енергетики
Кафедра інженерії програмного забезпечення в енергетиці

ВЕБСИСТЕМА МОДЕЛЮВАННЯ ГІБРИДНОЇ ЕНЕРГОСИСТЕМИ НА ОСНОВІ MATLAB PRODUCTION SERVER

виконав: студент групи ТВ-12 Мельник Ігор Борисович

керівник: ас. Гейко Олег Олександрович

2025

Актуальність теми

Опис проблеми, яку вирішує робота: Змінні погодні умови та нерівномірне навантаження зменшують ефективність керування гібридною енергосистемою. Моделювання сприяє оптимізації використання сонячної енергії, батарей та дизельних генераторів.

Значення теми для галузі інженерії програмного забезпечення: Дана робота демонструє інтеграцію вебінтерфейсу з MATLAB, обробку та візуалізацію даних, що відповідає сучасним стандартам розробки систем моделювання.

Мета роботи: Розробити вебзастосунок для моделювання гібридної енергосистеми з використанням MATLAB Production Server.

Постановка завдання

При плануванні проєкту було поставлено такі основні завдання:

• Надати користувачу можливість вибору ручного вводу або автоматичного оновлення даних, наприклад моделювання з різними значеннями площі сонячних панелей, ємності батареї, і.т.д.

• Реалізувати валідацію вхідних параметрів (для уникнення введення від’ємних значень де це недопустимо, або нечислових даних)

• Підключити MATLAB Production Server для обчислення даних за відповідними формулами

• Розробити інтуітивний веб інтерфейс для користувача, запровадити графічну візуалізацію результатів

Аналіз предметної області

Гібридні енергосистеми поєднують відновлювані джерела (сонячні панелі) з традиційними (дизель-генератори) та накопичувачами енергії (батареї). Моделювання таких систем дозволяє тестувати їх роботу в різних сценаріях без навантаження для реального обладнання.

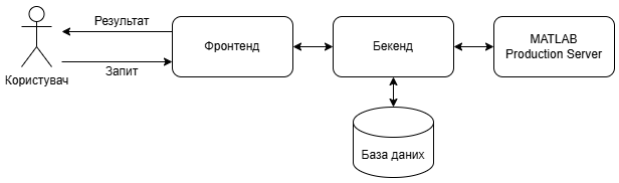
Основними недоліками що можуть зустрічатись при моделюванні аналогічних систем є:

- Відсутність вебінтерфейсу для віддаленого доступу.
- Обмежена інтеграція з погодними API або подібними сторонніми сервісами.
- Недостатня гнучкість для автоматизації та розширення функціональності.
- Висока складність для неінженерних користувачів.

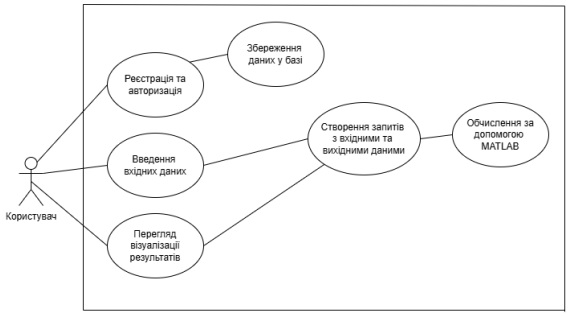
Архітектура системи

Архітектура системи передбачає наявність трьох основних компонентів:

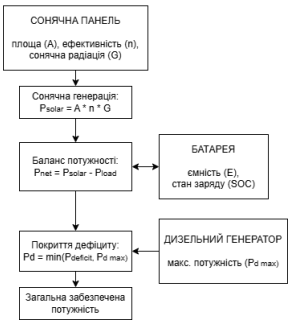
1. Клієнтська частина (фронтенд)
2. Серверна частина (бекенд)
3. Обчислювальний модуль MATLAB Production Server



Діаграма прецедентів



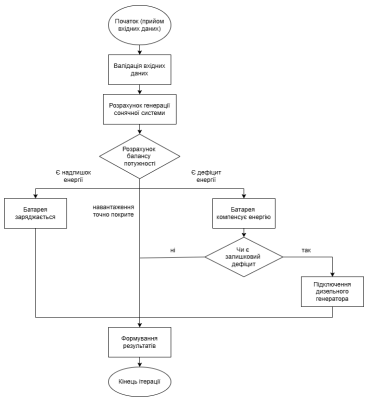
Модель гібридної енергосистеми



Модель гібридної енергосистеми реалізована як ctf-модель, де кожен компонент виконує свою роль у загальному енергетичному балансі, з забезпеченням передачі енергії між відповідними компонентами (наприклад, заряд/розряд батареї або покриття дефіциту дизелем)

У даному випадку модель описує взаємодію між сонячною генерацією, акумуляторною батареєю та дизельним генератором. Вона враховує енергетичний баланс та дозволяє оцінити покриття навантаження, втрати енергії та витрати палива.

Алгоритм роботи обчислювального модуля



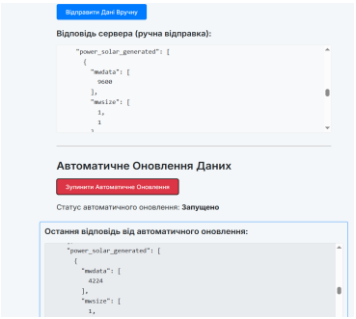
Основні етапи обчислення:

1. Валідація вхідних параметрів
2. Розрахунок сонячної генерації
3. Робота акумулятора (визначення надлишку чи дефіциту)
4. Підключення дизельного генератора
5. Формування результатів

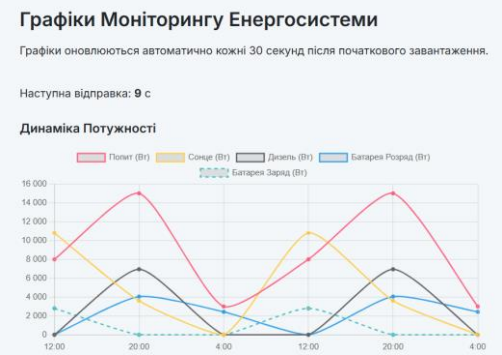
Використані інструменти

Клієнтський інтерфейс

Приклад ручного вводу або автоматичного оновлення даних та виводу результатів у JSON:



Клієнтський інтерфейс



Висновки

• Розроблено систему моделювання гібридної енергосистеми, що дозволяє здійснювати моделювання енергетичних процесів залежно від вироблення енергії від відновлюваних джерел.

• Створено зручний вебінтерфейс на React і TypeScript із підтримкою авторизації, вводу даних і графічної візуалізації результатів моделювання.

• Інтегровано обчислювальний модуль з MATLAB Production Server для виконання математичних розрахунків і обробки даних.

• Побудовано вебсистему що поєднує клієнтський інтерфейс з бекендом на Python і Flask та обчислювальним модулем MATLAB.