Файл deploy.bat, який ви навели, перевіряє наявність папки venv та активує віртуальне середовище, щоб забезпечити ізоляцію від інших проектів та встановлених залежностей в системі.

Основні дії включають перевірку існування віртуального середовища (venv), активацію його, налаштування змінних, ініціалізацію файлу журналу, запуск сервера та вузлів (nodes) у різних командних вікнах, і запис виконаних команд у журнал.

Base.metadata.create\_all(engine) викликається для створення всіх таблиць, які були описані в об'єкті Base (який є частиною SQLAlchemy), у базі даних, яку визначено за допомогою об'єкта engine.

**SQLAlchemy:**

* **Основна мета:** SQLAlchemy - це бібліотека для взаємодії з базами даних. Вона надає інструменти для роботи з реляційними базами даних (такими як PostgreSQL, MySQL, SQLite тощо).
* **Функції:**
  + ORM (Object-Relational Mapping): SQLAlchemy дозволяє представляти дані в базі даних у вигляді об'єктів Python, що полегшує взаємодію з базою даних.
  + Створення SQL-виразів: Вона надає можливість виразити складні SQL-запити в Python-подібному синтаксисі.
  + Робота з транзакціями, об'єднаннями, індексами та іншими аспектами баз даних.

**declarative\_base()**: Це функція в SQLAlchemy, яка створює базовий клас для визначення моделей. Модель у контексті SQLAlchemy - це об'єкт, який представляє запис у таблиці бази даних.

FastAPI - це високопродуктивний фреймворк для створення веб-додатків та API на мові програмування Python.

1. **server\_stats = {server.id: 0 for server in session.query(db.models.Node).filter\_by(status="active").all()}**: Створення словника **server\_stats**, де ключами є ідентифікатори (id) активних серверів, а значеннями є початкова кількість завдань, яка дорівнює нулю.
2. **for task in session.query(db.models.Task).filter\_by(status="pending").all():**: Цикл, який перебирає всі невиконані завдання зі статусом "pending".
3. **server\_stats[task.node\_id] += 1**: Збільшення лічильника завдань для сервера, на якому знаходиться поточне завдання (**task**). **task.node\_id** - ідентифікатор сервера, на якому призначено це завдання.
4. **min\_loaded\_server\_id = sorted(server\_stats.items(), key=lambda pair: pair[1])[0][0]**: Знаходження ідентифікатора сервера з найменшою кількістю завдань, використовуючи сортування словника **server\_stats** за значенням лічильника завдань. В результаті ми отримуємо пари **(server\_id, task\_count)**, відсортовані за зростанням кількості завдань, і беремо перший елемент, тобто сервер з найменшим завантаженням.
5. **min\_loaded\_server = session.query(db.models.Node).filter\_by(id=min\_loaded\_server\_id).first()**: Отримання об'єкта сервера з бази даних, використовуючи ідентифікатор сервера, який має найменшу кількість завдань.

Отже, в кінці ми отримуємо об'єкт сервера (**min\_loaded\_server**), який має найменше невиконаних завдань серед усіх активних серверів.

Спочатку функція getUserData (formId) яка перевіряє чи реєстрацію чи логін ти обрав, а потім вже відповідно до вибору вибирає async function registerUser() або async function loginUser()

Отже, загалом, цей код відображає інформаційні повідомлення користувачеві після виконання операції (логін або реєстрація), очищає форму та виконує інші дії, такі як приховання модального вікна чи перехід на головну сторінку.

**setAttribute** - це метод JavaScript, який встановлює атрибут для вказаного елемента HTML. В основному, він використовується для динамічної зміни атрибутів елементів під час виконання скриптів.

**ancelbtn.setAttribute("class", "cancelbtn");**: Додаємо атрибут класу до кнопки, щоб застосувати стилізацію (стилі CSS) до кнопки. Клас "cancelbtn" вказує на конкретний стиль для кнопки скасування.

**res = cancelTask(task["id"]);**: Викликаємо функцію **cancelTask(task["id"])**, яка відправляє запит на сервер для скасування (відміни) завдання з вказаним ідентифікатором (**task["id"]**