1. models.py

- **Co robi?**
  Plik zawiera definicje modeli, które reprezentują strukturę danych w bazie danych. Model opisuje, jakie dane będą przechowywane (np. tytuł, opis, status zadania) i w jaki sposób.
- **Przykład:**
  Jeśli tworzymy aplikację do zarządzania zadaniami, model będzie przechowywać informacje o tytule zadania, opisie i czy zadanie zostało zakończone.
- **Definicja:**
  Model w Django to Pythonowa klasa, która definiuje kolumny w tabeli bazy danych.

2. serializers.py

- **Co robi?**
  Serializator konwertuje dane z modeli na format JSON (czyli tekstowy format łatwy do przesyłania w aplikacjach internetowych) i odwrotnie. Dzięki temu dane mogą być wysyłane między backendem (Django) a frontendem (React).
- **Przykład:**
  Task o nazwie "Nauka Django" zostaje zamieniony na:

  {"id": 1, "title": "Nauka Django", "completed": false}
- **Definicja:**
  Serializacja to proces przekształcania danych na format możliwy do przesłania (np. JSON).

3. views.py

- **Co robi?**
  Ten plik zawiera logikę obsługującą żądania (requests) do API. Odpowiada za wysyłanie, edycję i usuwanie danych w odpowiedzi na zapytania od użytkownika.
- **Przykład:**
  - Jeśli frontend wyśle zapytanie: *"Pokaż mi wszystkie zadania"*, widok zwróci listę zadań.
  - Jeśli wyśle zapytanie: *"Dodaj nowe zadanie"*, widok zapisze je w bazie danych.
- **Definicja:**
  Widok (View) to funkcja, która przetwarza żądanie i zwraca odpowiedź.

4. urls.py

- **Co robi?**
  Plik definiuje, jakie adresy URL (np. /tasks/) będą dostępne w aplikacji i jakie widoki będą obsługiwać te adresy.
- **Przykład:**
  Adres URL /tasks/ zwraca listę wszystkich zadań, a /tasks/1/ zwraca szczegóły konkretnego zadania.
- **Definicja:**
  Routing to proces kierowania żądań użytkownika (adresów URL) do odpowiednich widoków.

5. settings.py

- **Co robi?**
  Plik konfiguracyjny projektu Django. Zawiera informacje o bazie danych, aplikacjach w projekcie, ustawieniach bezpieczeństwa i innych opcjach.
- **Przykład:**
  - Możesz określić, że używasz bazy danych SQLite lub PostgreSQL.
  - Możesz dodać aplikacje, takie jak rest_framework czy api.
- **Definicja:**
  Ustawienia to konfiguracje globalne, które określają, jak działa projekt Django.

1. App.js

- **Co robi?**
  Jest głównym plikiem frontendu. Tutaj definiujesz, które komponenty będą wyświetlane na stronie.
- **Przykład:**
  Wyświetlenie listy zadań:

  <TaskList />

2. TaskList.js

- **Co robi?**
  Wyświetla dane z backendu (listę zadań) na stronie. Obsługuje pobieranie danych z API za pomocą axios.

- **Przykład:**
  Wyświetla zadania w formie listy:

  Nauka Django

  React

