Warszawa 01.02.2024

**Programmer Challenge – projekt na zaliczenie Lab z przedmiotu – Wprowadzenie do programowania**

Temat: Komputerowe przetwarzanie języka naturalnego (NLP): Zadania związane z analizą i przetwarzaniem tekstu, np.tworzenie chatbotów, analiza sentymentu. Przykład: Zaprojektuj system analizy sentymentu recenzji filmowych.

Projekt pt. „Konwersacyjny ChatBot oparty na algorytmach uczenia maszynowego”

Wykonali: Zespół R – Konrad Stachurski, Daniel Domalewski (grupa 4)

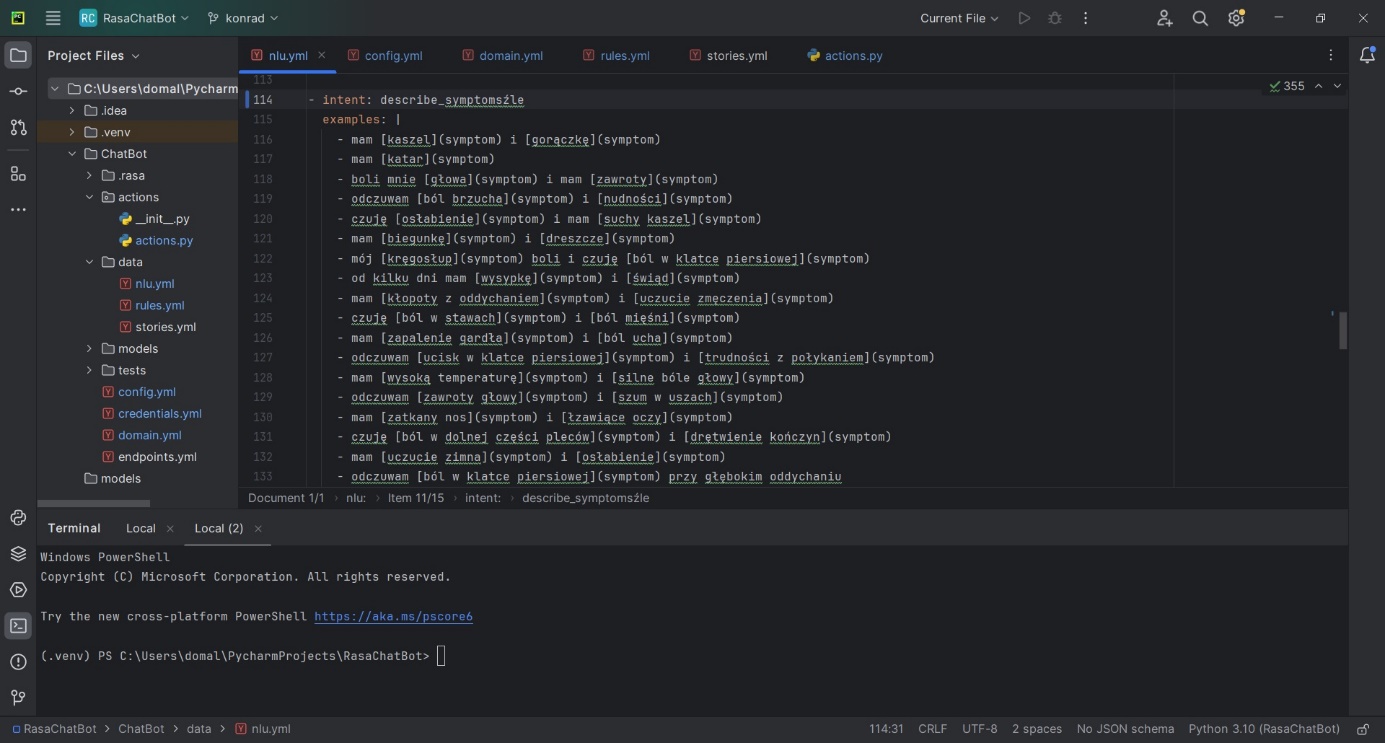
Prowadzący: Jacek Mochyła

Warszawa 2024

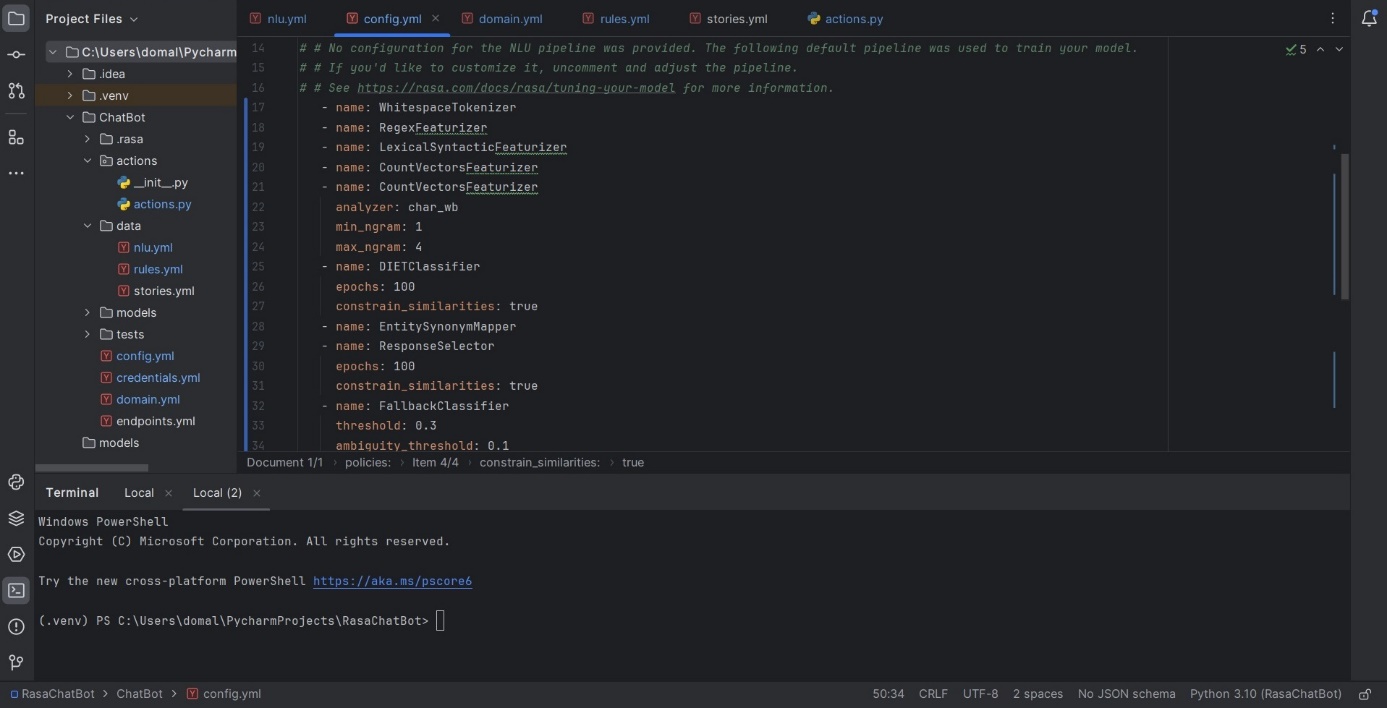
# **Kilka założeń projektowych:**

1. Użyliśmy frameworka RASA opartego na licencji open source
2. Do oprogramowania go używamy języka Python w wersji 3.9
3. W frameworku używamy algorytmów uczenia maszynowego:
4. Główne Rodzaje Komponentów w Pipeline RASA:
   1. Tokenizery (Tokenizacja): Rozdzielają tekst na tokeny (słowa).
   2. Featuryzery (Ekstrakcja Cech): Przekształcają tokeny w cechy numeryczne, które mogą być wykorzystane przez modele uczenia maszynowego.
   3. Komponenty Wyodrębniania Encji: Identifikują i klasyfikują encje w tekście (np. nazwy miejsc, daty).
   4. Komponenty Klasyfikacji Intencji: Określają, jaka intencja stoi za daną wiadomością użytkownika.
   5. Komponenty Poprawiania Tekstu: Poprawiają literówki lub błędy w tekście.
   6. Response Selectors (Selektory Odpowiedzi): Służą do wybierania odpowiednich odpowiedzi w przypadkach, gdy konieczne jest pobranie odpowiedzi z zewnętrznych źródeł.
   7. Tokenizery (Tokenizacja): Rozdzielają tekst na tokeny (słowa).
   8. Featuryzery (Ekstrakcja Cech): Przekształcają tokeny w cechy numeryczne, które mogą być wykorzystane przez modele uczenia maszynowego.
   9. Komponenty Wyodrębniania Encji: Identifikują i klasyfikują encje w tekście (np. nazwy miejsc, daty).
   10. Komponenty Klasyfikacji Intencji: Określają, jaka intencja stoi za daną wiadomością użytkownika.
   11. Komponenty Poprawiania Tekstu: Poprawiają literówki lub błędy w tekście.
   12. Response Selectors (Selektory Odpowiedzi): Służą do wybierania odpowiednich odpowiedzi w przypadkach, gdy konieczne jest pobranie odpowiedzi z zewnętrznych źródeł.
5. Dane treningowe zawarte są w plikach .yml
   1. nlu.yml (przykłady intencji i encji),
   2. domain.yml (definiuje intencje, encje, odpowiedzi),
   3. config.yml (konfiguracja modelu języka),
   4. stories.yml (przykłady konwersacji) i
   5. rules.yml (reguły dla określonych zachowań).

NLU

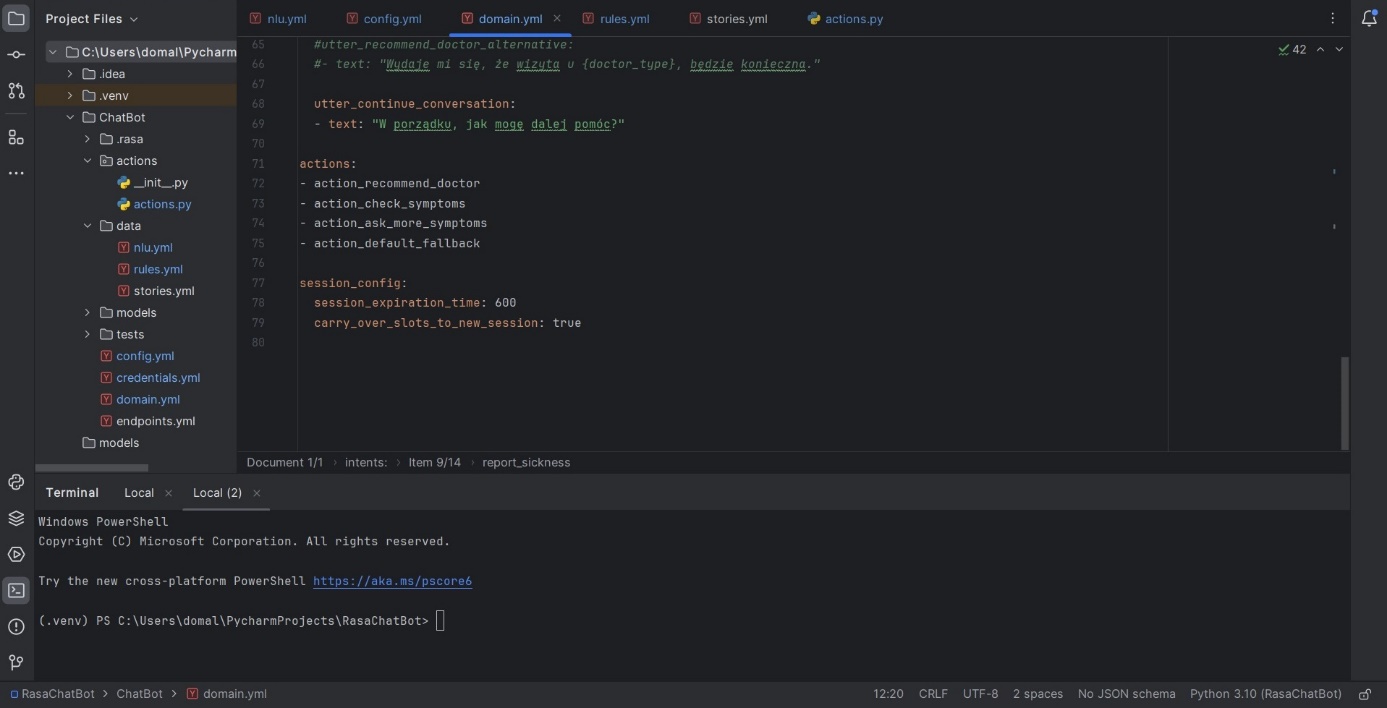


CONFIG

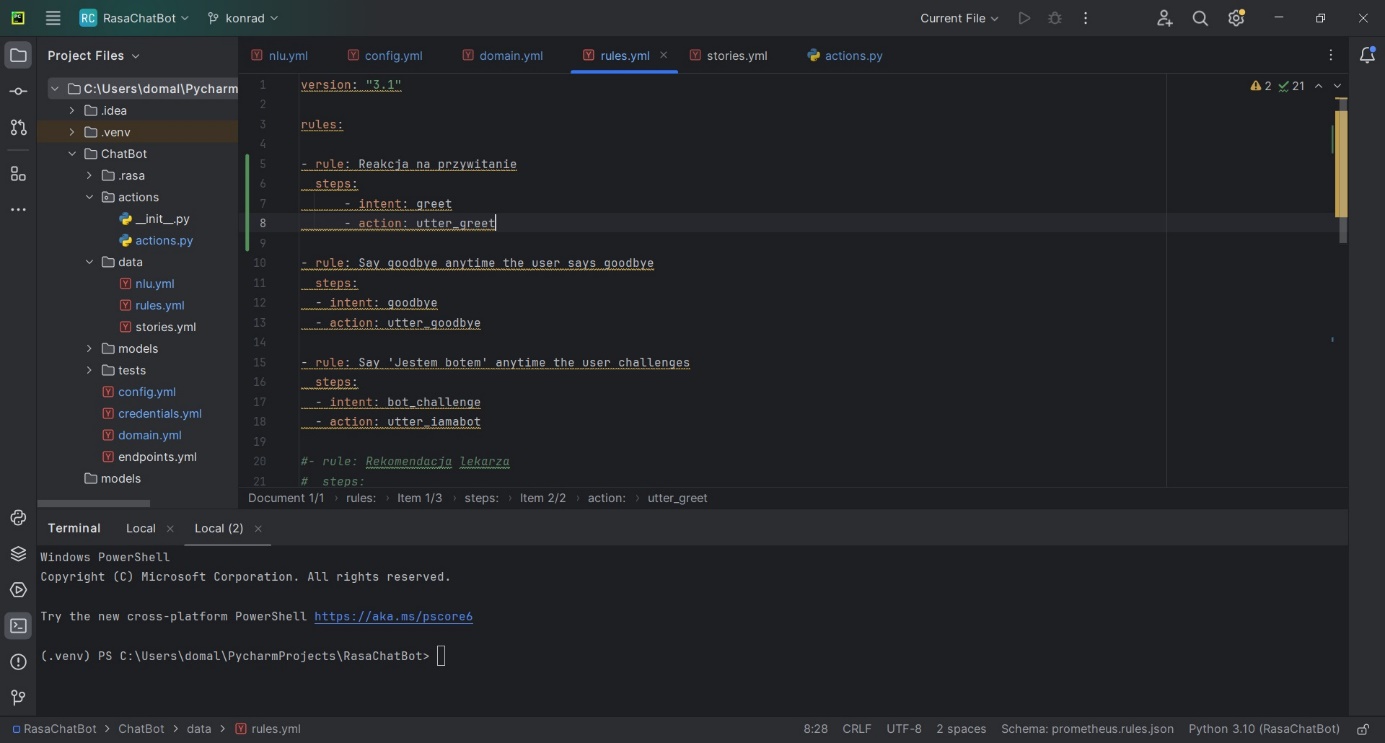


DOMAIN

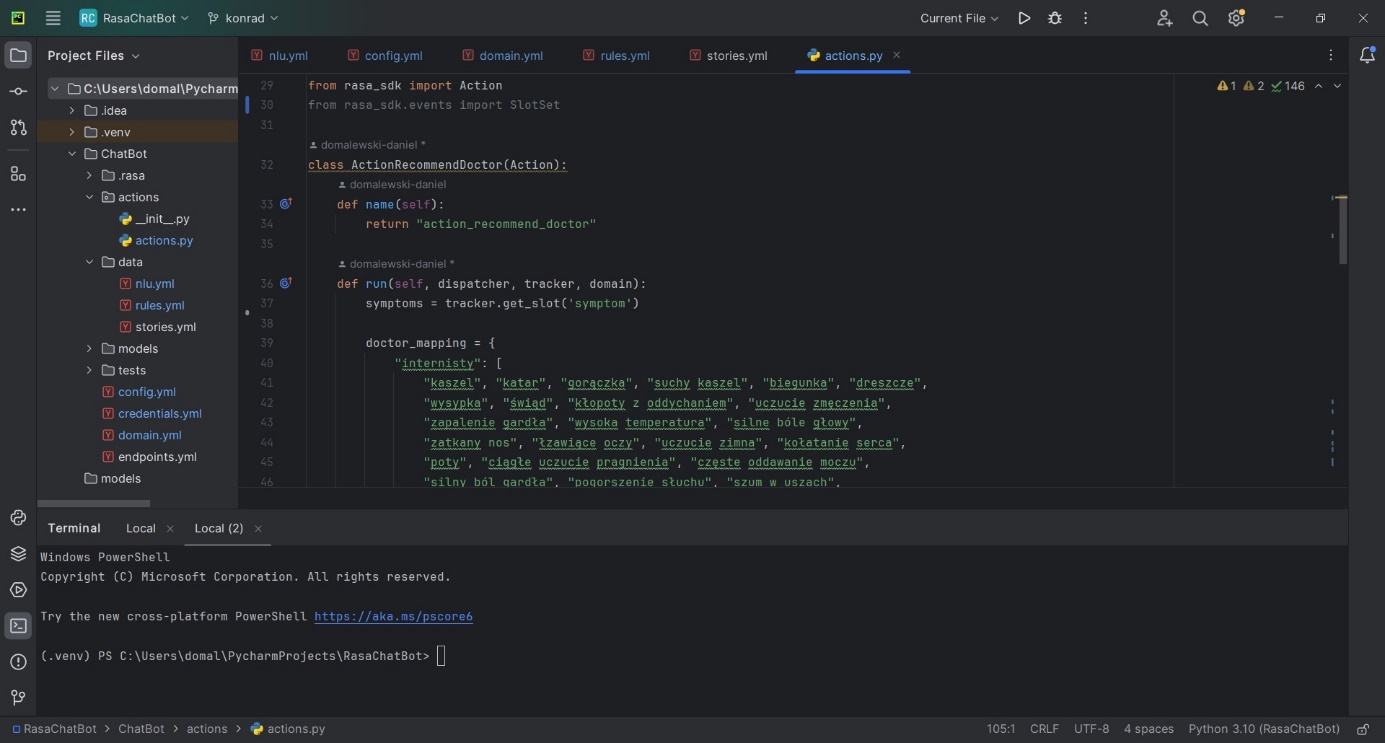
* slots: Definiuje sloty, które bot może używać do przechowywania informacji przez całą sesję rozmowy.
* symptom: Jest to nazwa slotu, który będzie używany.
* type: list: Określa, że slot symptom może przechowywać listę wartości. Jest to przydatne, gdy użytkownik może zgłaszać wiele objawów.
* influence\_conversation: false: Wskazuje, że wartość tego slotu nie wpłynie bezpośrednio na przepływ rozmowy. Oznacza to, że bot nie zmieni swojego zachowania w zależności od wartości przechowywanej w tym slocie.
* mappings: Definiuje, w jaki sposób slot ma być wypełniany.
* type: from\_entity: Określa, że slot symptom będzie wypełniany na podstawie encji o nazwie symptom rozpoznanej w wypowiedzi użytkownika.



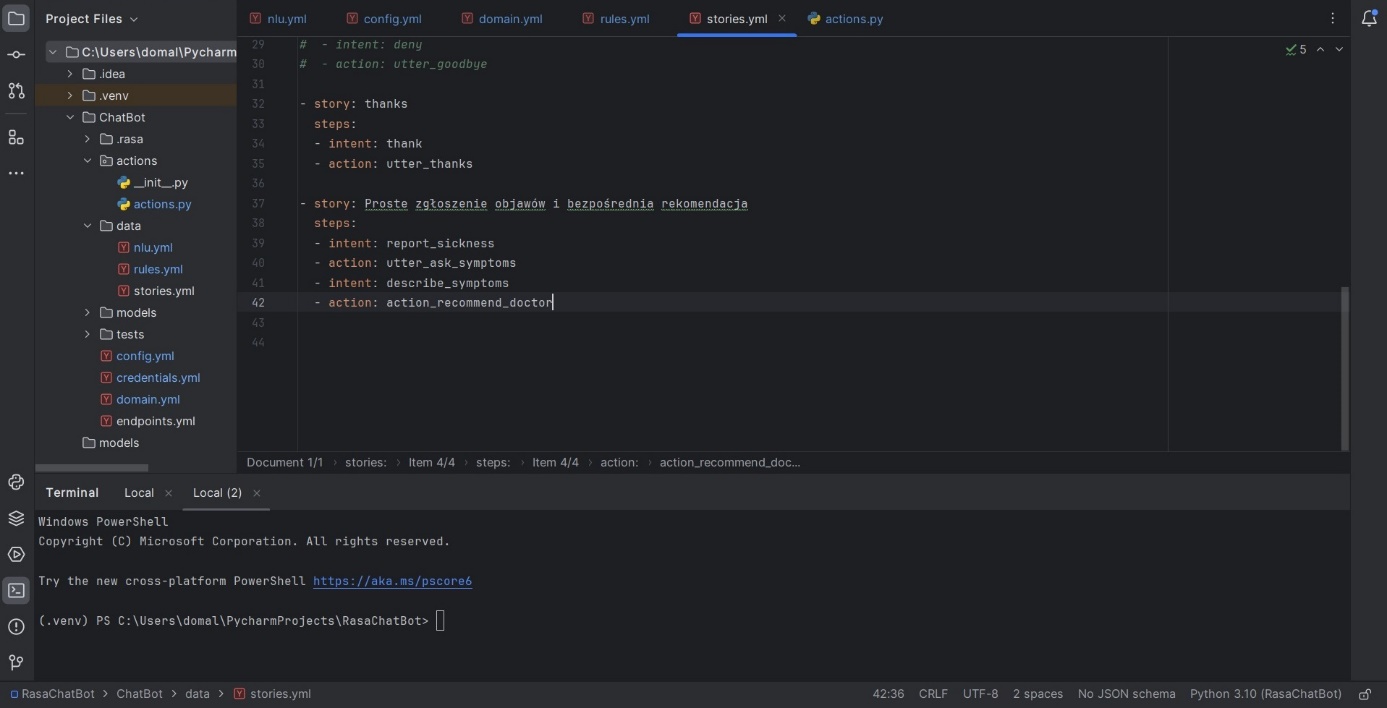
**RULES**



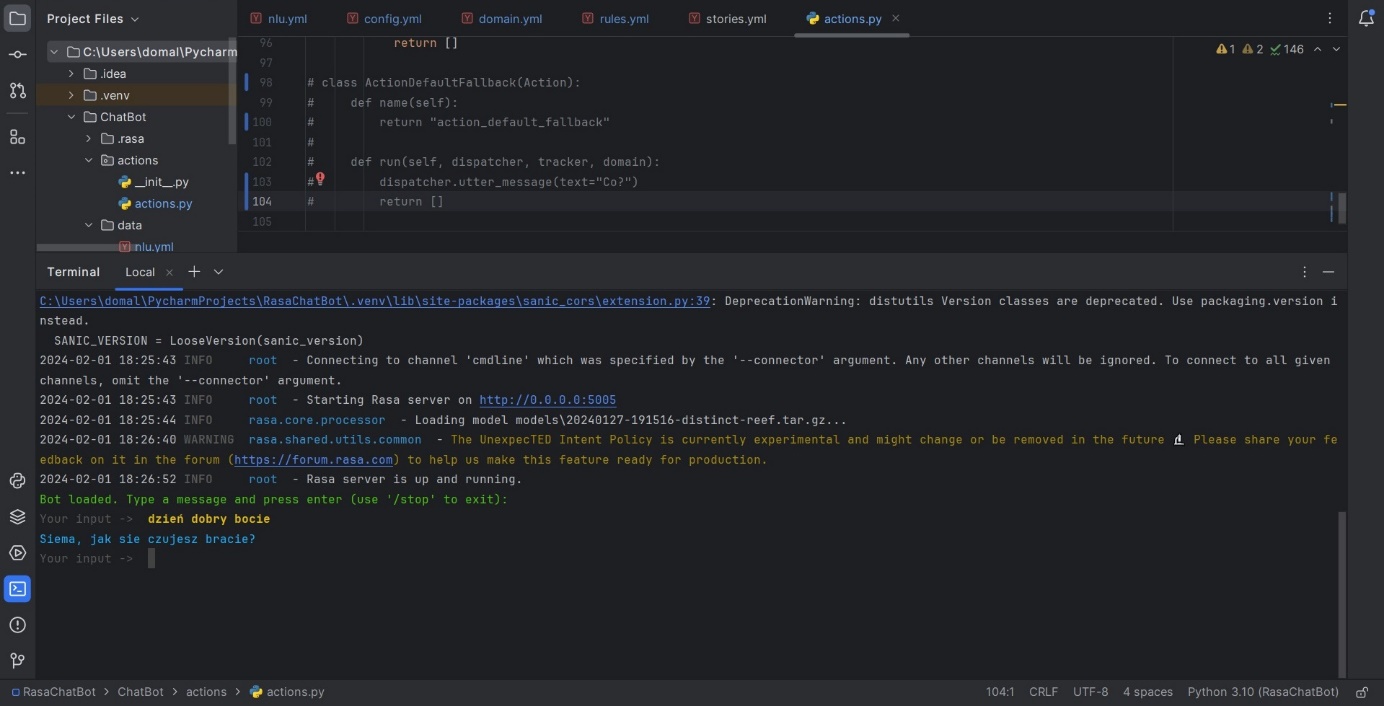
**ACTIONS (funkcje – python)**



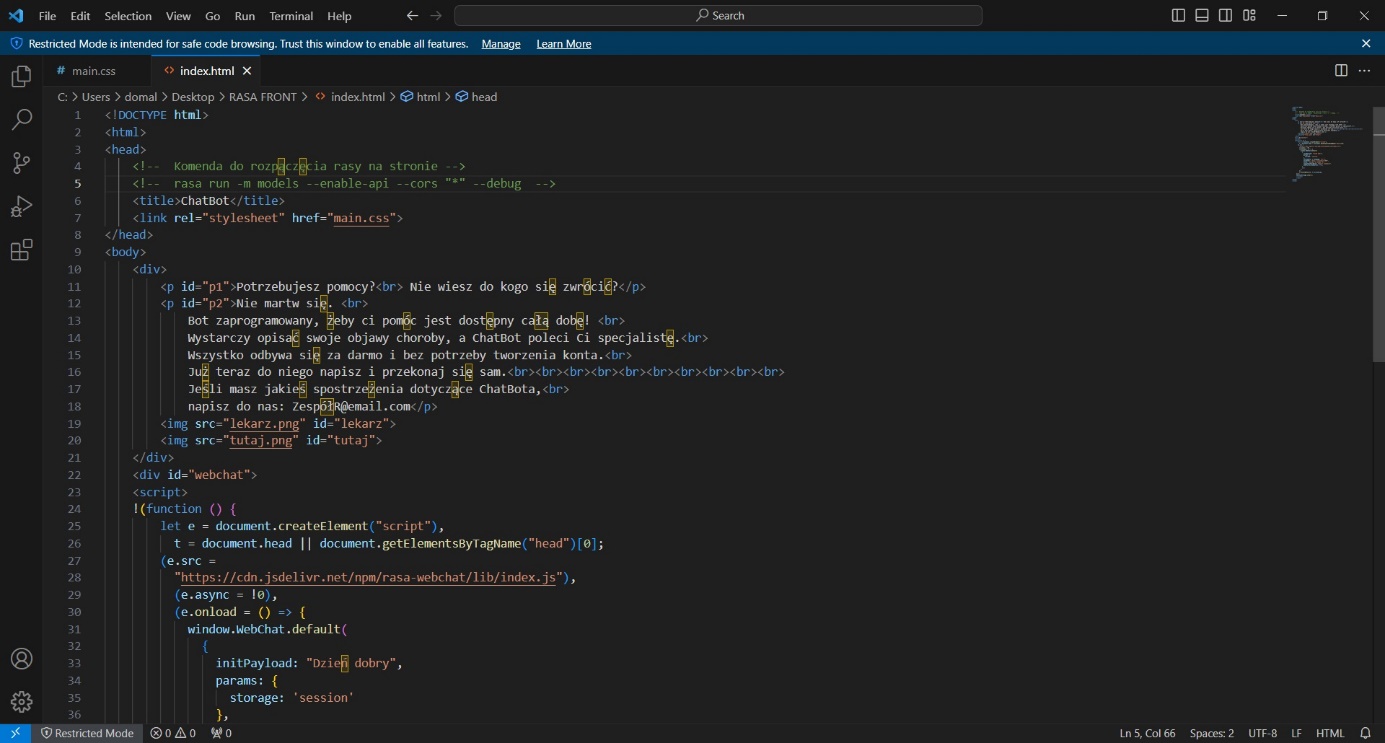
**STORIES**



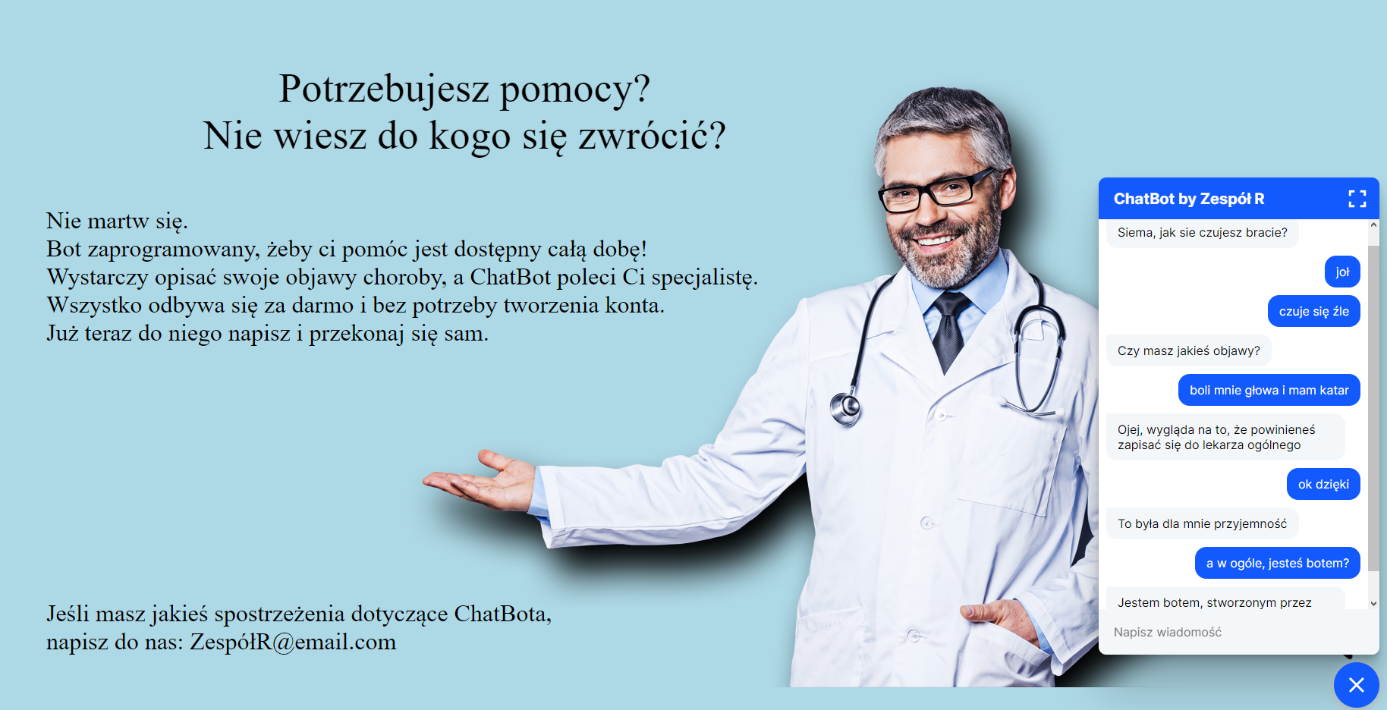
**RASA SHELL**

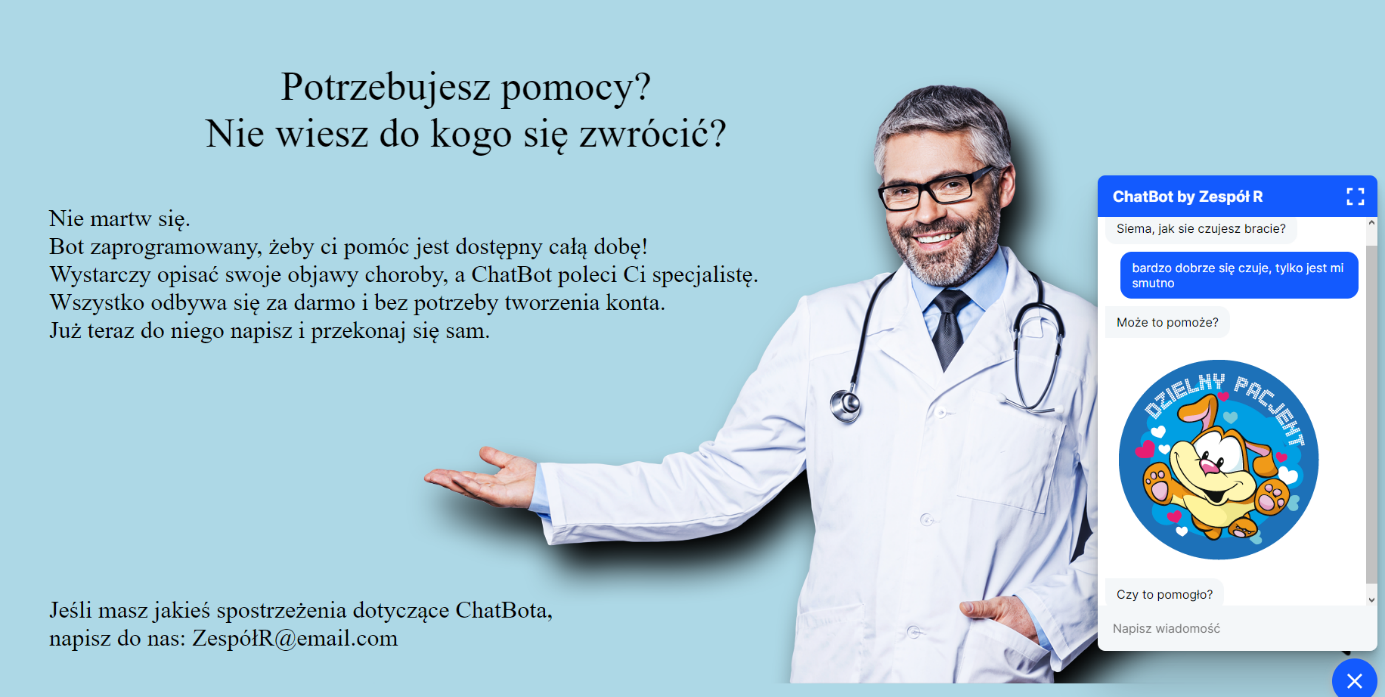


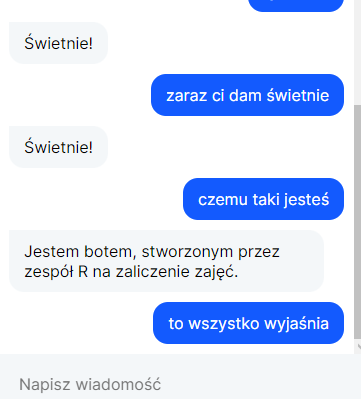
# **FRONTEND**



# **PRZYKŁADY DZIAŁAŃ**







# **DEBUG**

