Fuzzy Car Controller - Dokumentacja Projektu

Informacje podstawowe

Nazwa projektu: Fuzzy Systems – Symulacja pojazdu z kontrolą prędkości (Cruise Control)
Autorzy:
- Mateusz Bocak
- Jakub Jakubowski
- Karol Dąbrowski

Repozytorium: *github.com/JustFiesta/fuzzy-systems*

Cel projektu

Celem projektu jest stworzenie symulacji jazdy pojazdu po torze owalnym (eliptycznym), w której logika rozmyta (fuzzy logic) steruje przepustnicą tak, aby utrzymać zadaną prędkość pojazdu – działanie zbliżone do samochodowego systemu cruise control.

System umożliwia:

- obserwację zachowania pojazdu na torze 2D,
- porównanie prędkości rzeczywistej z docelową,
- analizę działania kontrolera rozmytego,
- przełączenie między trybem FUZZY (automatycznym) a MANUAL (ręczne sterowanie przepustnicą).

Scenariusz: Przejazd po autostradowej obwodnicy

Wyobraźmy sobie kierowcę jadącego obwodnicą miasta o kształcie eliptycznym (typowa sytuacja w dużych miastach). Chce utrzymać prędkość **90 km/h** zgodną z ograniczeniem, ale:

- Na prostych odcinkach pojazd łatwo przyspiesza
- W łagodnych zakrętach rosną opory ruchu
- Wiatr i nachylenie jezdni zmieniają warunki

Faza 1: Aktywacja tempomatu

Sytuacja początkowa:

- Kierowca jedzie z prędkością **70 km/h**
- Naciska przycisk "SET" na tempomatcie i ustawia prędkość docelową: **90 km/h**
- System przejmuje kontrolę nad przepustnicą

Działanie systemu FUZZY:

System tempomat analizuje sytuację:

Prędkość aktualna: 70 km/h

Prędkość zadana: 90 km/h

Błąd: -20 km/h (jedzie za wolno)

Logika rozmyta rozumuje "po ludzku":

- "Jestem **znacznie** poniżej prędkości docelowej"
- "Nie zbliżam się jeszcze do celu"
- **Decyzja:** "Mocno przyspiesz – daj dużo gazu (75% przepustnicy)"

Efekt praktyczny: Pojazd płynnie przyspiesza. Kierowca czuje delikatny nacisk na fotel – system pracuje zdecydowanie, ale nie agresywnie. Po 5 sekundach prędkość wynosi **82 km/h**.

Obserwacja w symulacji:

- Na ekranie widzimy pojazd (ikona samochodu) przyspieszający na prostym odcinku toru
- Wykres pokazuje niebieską linię (prędkość rzeczywista) wspinającą się w kierunku czerwonej linii (prędkość docelowa 90 km/h)
- Wskaźnik przepustnicy przesuwa się w górę do 75%

Faza 2: Zbliżanie się do prędkości docelowej

Sytuacja:

Prędkość aktualna: 88 km/h

Prędkość zadana: 90 km/h

Błąd: -2 km/h (prawie idealnie)

Logika rozmyta:

- "Jestem **blisko** prędkości docelowej"
- "Przyspieszam, więc zbliżam się do celu"
- **Decyzja:** "Delikatnie przyspiesz – umiarkowany gaz (52% przepustnicy)"

Efekt praktyczny: System **automatycznie redukuje gaz**, zanim doszło do przeregulowania. W zwykłym samochodzie bez tempomatu kierowca musiałby teraz zwolnić nogę z gazu, by nie przekroczyć 90 km/h. Tempomat robi to za niego.

Obserwacja w symulacji:

- Niebieska linia prędkości łagodnie dochodzi do czerwonej linii
- Przepustnica zmniejsza się z 75% do 52%
- **Brak gwałtownego "strzelenia"** powyżej 90 km/h – to zasługa logiki rozmytej

---

Faza 3: Wjazd w zakręt – test adaptacji

Sytuacja: Pojazd osiągnął prędkość **90 km/h** i wjeżdża w łagodny zakręt obwodnicy. W rzeczywistości:

- Opory aerodynamiczne rosną (siła dośrodkowa)
- Tarcie opon zwiększa się
- Prędkość zaczyna spadać do **88 km/h**

Prędkość aktualna: 88 km/h (spada!)

Prędkość zadana: 90 km/h

Błąd: -2 km/h

Trend: błąd się ZWIĘKSZA (prędkość dalej spada)

Logika rozmyta:

- "Jestem **trochę** poniżej prędkości"
- "Ale uwaga – **tracę** prędkość, błąd rośnie!"
- **Decyzja:** "Dodaj gazu wyprzedzająco (58% przepustnicy)"

Efekt praktyczny: System **sam dodaje gazu w zakręcie**, dokładnie tak jak zrobiłby doświadczony kierowca. Bez tego prędkość spadłaby do 85 km/h, zanim system zareagowałby. Dzięki analizie trendu (Δe) tempomat działa przewidująco.

Obserwacja w symulacji:

- Na wizualizacji 2D widzimy pojazd w zakręcie elipsy
- Przepustnica wzrasta z 48% do 58%
- Prędkość stabilizuje się na 89-90 km/h mimo trudnych warunków
- **To kluczowa przewaga logiki rozmytej** – zwykły regulator zareagowałby za późno

---

Faza 4: Wyjście z zakrętu – kompensacja

Sytuacja: Pojazd wychodzi z zakrętu na prostą. Opory maleją, pojazd naturalnie przyspiesza do **92 km/h**.

Prędkość aktualna: 92 km/h (za szybko!)

Prędkość zadana: 90 km/h

Błąd: +2 km/h

Trend: prędkość ROŚNIE

Logika rozmyta:

- "Jestem **trochę** powyżej prędkości"
- "I nadal przyspieszam!"
- **Decyzja:** "Zmniejsz gaz wyraźnie (35% przepustnicy)"

Efekt praktyczny: System delikatnie **hamuje silnikiem**, zmniejszając przepustnicę. Nie używa hamulców – to byłoby niekomfortowe i nieekonomiczne. Po 2 sekundach prędkość wraca do 90 km/h.

Obserwacja w symulacji:

- Przepustnica spada do 35%
- Niebieska linia prędkości łagodnie wraca do czerwonej linii
- Brak gwałtownych oscylacji – kierowca nie odczułby dyskomfortu

---

Faza 5: Stan ustabilizowany – tryb podróżny

Sytuacja: Pojazd jedzie już 2 minuty z tempomatem. Prędkość oscyluje wokół **90 km/h** (89.5 - 90.5 km/h).

Błąd: ±0.5 km/h (praktycznie zero)

Przepustnica: 45-48% (mikro-korekty)

Logika rozmyta:

- "Prędkość **idealna**"
- "Bez zmian sytuacji"
- **Decyzja:** "Utrzymuj obecny gaz (46-48%)"

Efekt praktyczny: Kierowca może zdjąć nogę z pedałów i skupić się na prowadzeniu samochodu. System wykonuje **mikroskopijne korekty** (±2% przepustnicy), których kierowca nie odczuwa. To właśnie moment, w którym tempomat pokazuje swoją wartość – zmniejsza zmęczenie i pozwala utrzymać stałą, bezpieczną prędkość.

Obserwacja w symulacji:

- Pojazd płynnie okrąża tor
- Wykres prędkości to niemal idealna linia pozioma
- Wykres przepustnicy pokazuje minimalne drgania wokół 47%
- System działa **transparentnie** – prawie niewidoczny dla użytkownika

---

Przełączenie trybów: FUZZY vs MANUAL

Tryb MANUAL – eksperyment porównawczy

Kierowca decyduje się przetestować różnicę i **wyłącza tempomat**, przejmując ręczną kontrolę:

Sytuacja w zakręcie (ręczne sterowanie):

- Kierowca utrzymuje stałe wciśnięcie pedału gazu (50%)
- W zakręcie prędkość spada z 90 km/h do **85 km/h**
- Kierowca musi **ręcznie dodać gazu** do 60%
- Po wyjściu z zakrętu prędkość rośnie do **94 km/h**
- Kierowca musi **zmniejszyć gaz** do 45%

Rezultat:

- Duże oscylacje prędkości: 85-94 km/h
- Wymaga ciągłej uwagi kierowcy
- Mniej komfortowe i mniej ekonomiczne

Obserwacja w symulacji:

- Niebieska linia prędkości "faluje" znacznie bardziej
- Przepustnica zmienia się skokowo (ręczne korekty kierowcy)
- Widać wyraźną różnicę w stabilności systemu

Powrót do trybu FUZZY

Kierowca ponownie aktywuje tempomat:

- System natychmiast analizuje sytuację
- W ciągu 3 sekund stabilizuje prędkość z powrotem na 90 km/h
- Oscylacje znikają
- Komfort powraca

Analiza zachowania systemu w symulacji

Elementy wizualizacji 2D:

1. **Tor eliptyczny:**
    - Niebieski kontur elipsy (np. 200m × 100m)
    - Pojazd (ikona samochodu) porusza się po obwodzie
    - Widoczne zmiany krzywizny: ostre zakręty vs długie proste
2. **Wykres prędkości w czasie:**
    - **Czerwona linia pozioma:** prędkość docelowa (90 km/h)
    - **Niebieska linia:** prędkość rzeczywista
    - W trybie FUZZY: linia niebieska "przykleja się" do czerwonej
    - W trybie MANUAL: linia niebieska oscyluje wokół czerwonej
3. **Wykres przepustnicy:**
    - Pokazuje aktywność kontrolera (0-100%)
    - W trybie FUZZY: płynne zmiany (35%-65%)
    - W trybie MANUAL: skokowe zmiany wg użytkownika
4. **Wskaźniki numeryczne:**

Prędkość: 90.2 km/h

Docelowa: 90.0 km/h

Błąd: -0.2 km/h

Przepustnica: 47%

Tryb: FUZZY

Praktyczne wnioski z projektu

Dlaczego logika rozmyta działa lepiej niż proste podejścia?

1. "Myśli" jak człowiek: Zamiast matematycznych wzorów używa reguł typu:

- "Jeśli **mocno** za wolno i **szybko** przyspieszam → daj **dużo** gazu"
- "Jeśli **trochę** za szybko ale **powoli** zwalniasz → daj **trochę mniej** gazu"

2. Adaptuje się do warunków:

- Automatycznie kompensuje zakręty (dodaje gaz)
- Reaguje na proste (zmniejsza gaz)
- Nie wymaga ręcznej kalibracji dla różnych torów

3. Działa płynnie:

- Brak gwałtownych zmian przepustnicy
- Minimalne oscylacje prędkości (±0.5 km/h)
- Komfort jazdy porównywalny z ludzkim kierowcą

4. Jest przewidujący: Analizuje nie tylko błąd, ale i **tempo jego zmiany** (Δe):

- Widzi, że prędkość spada → dodaje gaz **przed** dużym błędem
- Widzi, że prędkość rośnie → zmniejsza gaz **przed** przekroczeniem limitu

Główne funkcjonalności:

Automatyczna kontrola prędkości

Fuzzy kontroler wyznacza sygnał przepustnicy `throttle` na podstawie błędu prędkości i przyspieszenia.

Symulacja fizyczna pojazdu

Jednowymiarowy model ruchu (wzdłuż toru) z uwzględnieniem siły napędowej i oporu ruchu.

Wizualizacja toru owalnego

Widok z góry na owalny tor (elipsa), marker pojazdu, strzałka kierunku ruchu, ślad przebytej trasy.

Wykresy czasowe

prędkość rzeczywista vs. prędkość docelowa,

procent otwarcia przepustnicy (throttle)

Interfejs użytkownika (PyQt5 + PyQtGraph)

suwaki do zmiany parametrów pojazdu (masa, opór, maks. siła),

suwak prędkości docelowej,

suwak ręcznej przepustnicy,

przyciski: Start/Pause, Reset, przełącznik trybu FUZZY/MANUAL.

System rozmyty (Fuzzy Logic)

Zmienne systemu – Etap 1

Typ zmiennej	Nazwa	Opis	Jednostka	Zakres
Wejście 1	`speed_error`	Różnica między prędkością zadaną a aktualną	km/h	–30 … +30
Wejście 2	`acceleration`	Aktualne przyspieszenie p	m/s²	–10 … +10
Wyjście	`throttle`	Sygnał sterujący mocą silnika (pedał gazu)	%	0 … 100

Koncepcja lingwistyczna

Zmienna `speed_error` (błąd prędkości)

Stopnie:

- negative_large – pojazd jedzie znacznie za szybko,

- negative_small – pojazd nieco przekracza prędkość zadaną,
- zero – prędkość bliska zadanej,
- positive_small – pojazd jedzie lekko za wolno,
- positive_large – pojazd jedzie znacznie za wolno.

Zmienna `acceleration` (przyspieszenie)

Reprezentuje aktualną dynamikę pojazdu:

- **negative** – pojazd hamuje / zwalnia
- **zero** – prędkość stała, brak zmian
- **positive** – pojazd przyspiesza

Zmienna wyjściowa `throttle` (przepustnica)

Stopnie:

- **very_low** – bardzo niska moc,
- **low** – niska moc,
- **medium** – średnia moc,
- **high** – wysoka moc,
- **very_high** – bardzo wysoka moc.

Funkcje przynależności

Speed Error

- negative_large: trapezoidalna [-30, -30, -20, -10]
- negative_small: trójkątna [-15, -5, 0]
- zero: trójkątna [-5, 0, 5]
- positive_small: trójkątna [0, 5, 15]
- positive_large: trapezoidalna [10, 20, 30, 30]

Acceleration

- negative: trapezoidalna [-10, -10, -5, 0]
- zero: trójkątna [-3, 0, 3]
- positive: trapezoidalna [0, 5, 10, 10]

Throttle

- very_low: trapezoidalna [0, 0, 10, 20]
- low: trójkątna [10, 25, 40]
- medium: trójkątna [30, 50, 70]
- high: trójkątna [60, 75, 90]
- very_high: trapezoidalna [80, 90, 100, 100]

Baza reguł (Rule Base)

System wykorzystuje 12 reguł pokrywających wszystkie kombinacje stanów wejściowych:

Wybrane reguły (w notacji IF–THEN)

1. Gdy jedziemy dużo za szybko → bardzo mała przepustnica:

IF speed_error IS negative_large THEN throttle IS very_low

2. Gdy jedziemy trochę za szybko i nadal zwalniamy:

IF speed_error IS negative_small AND acceleration IS negative THEN throttle IS very_low

3. Gdy jedziemy trochę za szybko, ale prędkość się stabilizuje:

IF speed_error IS negative_small AND acceleration IS zero THEN throttle IS low

4. Gdy jedziemy trochę za szybko, ale jeszcze przyspieszamy:

IF speed_error IS negative_small AND acceleration IS positive THEN throttle IS medium

5. Prędkość bliska zadanej, ale zaczynamy zwalniać:

IF speed_error IS zero AND acceleration IS negative THEN throttle IS low

6. Prędkość bliska zadanej, stabilna:

IF speed_error IS zero AND acceleration IS zero THEN throttle IS medium

7. Prędkość bliska zadanej, ale przyspieszamy:

IF speed_error IS zero AND acceleration IS positive THEN throttle IS medium

8. Jedziemy trochę za wolno i zwalniamy (spadek prędkości):

IF speed_error IS positive_small AND acceleration IS negative THEN throttle IS medium

9. Jedziemy trochę za wolno i prędkość jest stabilna:

IF speed_error IS positive_small AND acceleration IS zero THEN throttle IS high

10. Jedziemy trochę za wolno, ale już przyspieszamy:

IF speed_error IS positive_small AND acceleration IS positive THEN throttle IS medium

11. Duży błąd dodatni i zwalniamy → maksymalne przyspieszenie:

IF speed_error IS positive_large AND acceleration IS negative THEN throttle IS very_high

12. Duży błąd dodatni i brak/pozytywne przyspieszenie:

IF speed_error IS positive_large AND (acceleration IS zero OR positive) THEN throttle IS very_high

#	Speed Error	Acceleration	Throttle	Uzasadnienie
1	PB	N	High	Bardzo wolno i hamuje → maksymalne dodanie gazu
2	PB	Z	High	Bardzo wolno, stała prędkość → duże przyśpieszenie
3	PB	P	Medium	Bardzo wolno, ale już przyspiesza → utrzymaj
4	PS	N	High	Lekko wolno i hamuje → dodaj gaz
5	PS	Z	Medium	Lekko wolno, stała → umiarkowane dodanie
6	PS	P	Low	Lekko wolno, ale przyspiesza → lekko zmniejsz
7	Z	N	Medium	OK, ale hamuje → lekkie dodanie gazu
8	Z	Z	Medium	Idealna prędkość → utrzymaj
9	Z	P	Low	OK, ale przyspiesza → lekko zmniejsz

#	Speed Error	Acceleration	Throttle	Uzasadnienie
10	NS	N	Low	Lekko szybko i hamuje → ok, niech zwalnia
11	NS	Z	Low	Lekko szybko → delikatnie zmniejsz
12	NS	P	Low	Lekko szybko i przyspiesza → zmniejsz mocno
13	NB	N	Low	Bardzo szybko i hamuje → minimum gazu
14	NB	Z	Low	Bardzo szybko → zmniejsz
15	NB	P	Low	Bardzo szybko i przyspiesza → minimum gazu

Podejście implementacyjne

Na początkowym etapie dane są **generowane syntetycznie** w Pythonie (biblioteka `numpy`), co pozwala na: - Szybkie testowanie logiki rozmytej - Niezależność od zewnętrznych źródeł danych - Pełną kontrolę nad parametrami symulacji - Łatwą modyfikację warunków testowych

Architektura techniczna

```
fuzzy-systems/
├── core/
│   ├── fuzzy_controller.py    # System rozmyty
│   ├── vehicle_model.py        # Model fizyczny pojazdu
│   └── simulation.py           # Główna pętla symulacji
├── ui/
│   ├── app.py                  # Główna aplikacja UI
│   └── components/             # Komponenty interfejsu
├── data/
│   └── parameters.json         # Konfiguracja parametrów
└── README.md
```

Model fizyczny pojazdu

Parametry:

- mass – masa pojazdu [kg],
- drag_coeff – liniowy współczynnik oporu [N·s/m],
- max_throttle – maksymalna siła napędowa [N],
- dt – krok czasowy symulacji [s].

Domyślne wartości używane w UI:

- mass = 1000.0 kg
- drag_coeff = 50.0 N·s/m
- max_throttle = 5000.0 N
- dt = 0.1 s

Stan pojazdu:

- position [m] – droga przebyta po torze (skalar, wykorzystywany do obliczeń pozycji na elipsie),
- speed [m/s] – prędkość liniowa,
- acceleration [m/s²] – przyspieszenie liniowe.

Równania modelu:

W każdej iteracji (metoda update):

1. Ograniczenie throttle do zakresu [0, 100] [%].
2. Przeliczenie przepustnicy na siłę napędową:

$$F_{\text{throttle}} = \textit{throttle} \,/\, 100 \cdot \textit{max\_throttle}$$

3. Siła oporu:

$$F_{\text{drag}} = \textit{drag\_coeff} \cdot v$$

4. Siła wypadkowa i przyspieszenie:

$$F_{\text{net}} = F_{\text{throttle}} - F_{\text{drag}}, \quad a = F_{\text{net}} \,/\, \textit{mass}$$

5. Aktualizacja prędkości i pozycji:

$$v_{k+1} = \max(0, v_k + a \cdot dt), \quad s_{k+1} = s_k + v_{k+1} \cdot dt$$

Prędkość nigdy nie jest ujemna (symulujemy tylko jazdę do przodu).

Parametry konfiguracyjne UI

W pliku ui_app.py parametry sterowane przez użytkownika są zdefiniowane za pomocą suwaków:

1. Parametry pojazdu

- **Masa [kg]:**
  - suwak: od 500 do 2000 kg, krok 50 kg,
  - domyślnie: 1000 kg.
- **Opór (współczynnik oporu drag_coeff):**
  - suwak: od 20 do 100 N·s/m, krok 5,
  - domyślnie: 50.
- **Moc [kN] / maks. siła max_throttle:**
  - suwak: od 2 do 10 (kN), skalowany przez 1000 → 2000–10000 N,
  - domyślnie: 5 kN → 5000 N.

2. Sterowanie

- **Prędkość docelowa [m/s]:**
  - suwak: od 10 do 35 m/s (ok. 36–126 km/h),
  - domyślnie: 20 m/s (ok. 72 km/h).
- **Manual Throttle [%]:**
  - suwak: 0–100 %,
  - domyślnie: 50 %.

3. Przyciski

- ▶ Start / Pause – uruchomienie/zatrzymanie pętli symulacji,
- Reset – reset stanu pojazdu, czasu, historii oraz wyczyszczenie wykresów,

- Tryb: FUZZY / Tryb: MANUAL – przełączenie między trybem sterowania rozmytego a ręcznym.

Metryki i wizualizacje

Dane wyświetlane w czasie rzeczywistym:

Widok toru (2D):

- eliptyczny tor,
- punkt reprezentujący pojazd,
- strzałka wskazująca kierunek ruchu,
- ślad przebytej trasy (kilkadziesiąt ostatnich punktów).

Wykres prędkości:

- prędkość rzeczywista [m/s],
- prędkość docelowa [m/s] jako linia referencyjna,
- zakres Y: 0–35 m/s.

Wykres przepustnicy:

- throttle [%] w funkcji czasu,
- zakres Y: 0–100 %.

Pasek informacyjny (górny):

- czas symulacji,
- pozycja [m],
- prędkość [m/s i km/h],
- prędkość docelowa [m/s i km/h],
- ostatnia wartość throttle,
- ostatni błąd prędkości [km/h].

Model symulacji – pętla główna
Logika pętli symulacji jest zaimplementowana w metodzie _simulation_step klasy FuzzyCarUI.

W każdym kroku:

Pobranie pozycji na torze

- Na podstawie car.position (droga [m]) wyznacza się (x, y, angle) na eliptycznym torze (OvalTrack.get_position).

Wyznaczenie prędkości docelowej

- target_speed pochodzi z suwaka w UI i jest podawana w **m/s**.

Obliczenie błędu prędkości

- speed_error = (target_speed - self.car.speed) * 3.6 → [km/h].

Wyznaczenie przepustnicy throttle

- tryb **MANUAL**: wartość z suwaka manual_throttle [0–100 %],
- tryb **FUZZY**: throttle = controller.compute_throttle(speed_error, car.acceleration).

Aktualizacja modelu pojazdu

- car.update(throttle) – przeliczenie pozycji, prędkości i przyspieszenia.

Zapisanie historii

- Czas, prędkość, prędkość docelowa, throttle, błąd prędkości oraz pozycja (x, y) są dodawane do bufora historii o ograniczonej długości

Aktualizacja wizualizacji

- pozycja pojazdu na torze,
- strzałka kierunku jazdy,
- ślad ostatnich punktów,
- wykres prędkości vs cel,
- wykres throttle.

Zasady projektowe

KISS (Keep It Simple, Stupid)

- Proste, zrozumiałe rozwiązania zamiast skomplikowanych
- Kod łatwy do debugowania i rozszerzania

Modularność

- Każdy komponent w osobnym pliku
- Jasno zdefiniowane interfejsy między modułami
- Łatwa wymiana implementacji (np. UI: Streamlit → Pygame)

Single Responsibility Principle

- Jedna funkcja/klasa = jedno zadanie
- `FuzzyController` → tylko logika rozmyta
- `Vehicle` → tylko model fizyczny
- `UI` → tylko prezentacja

DRY (Don't Repeat Yourself)

- Unikanie duplikacji kodu
- Parametry w plikach konfiguracyjnych
- Współdzielone utility functions

Prompty

1. Core fuzzy logic (core_logic.txt)

Prompt definiujący **serce projektu**: moduł logiki rozmytej sterujący mocą silnika. Opisuje zmienne lingwistyczne, funkcje przynależności, reguły IF–THEN, wnioskowanie i defuzyfikację oraz prosty interfejs do użycia w symulacji
https://github.com/JustFiesta/fuzzy-systems/tree/main/src/prompts/2

---

2. Simulation Engineer (simulation.txt)

Prompt odpowiedzialny za **model fizyczny pojazdu**. Skupia się na symulacji ruchu (prędkość, przyspieszenie, pozycja) w czasie dyskretnym i **działa całkowicie niezależnie od fuzzy logic**
https://github.com/JustFiesta/fuzzy-systems/tree/main/src/prompts/2

---

3. UI integration / problem architektoniczny (ui.txt)

Prompt opisujący, **jak sensownie połączyć backend (fuzzy + symulacja) z UI**. Koncentruje się na płynności, separacji odpowiedzialności i pracy w czasie rzeczywistym

https://github.com/JustFiesta/fuzzy-systems/tree/main/src/prompts/2

---

4. Podział pracy w zespole (work-split.txt)

Prompt organizacyjny — dzieli projekt na **3 niezależne role**, tak aby prace mogły być wykonywane równolegle. Wymusza modularność, zasadę KISS i jedną odpowiedzialność na moduł

https://github.com/JustFiesta/fuzzy-systems/tree/main/src/prompts/2

---

5. Starter / mentor fuzzy logic (starter-context.txt)

Prompt-mentor, który ustawia chatbota jako **eksperta od logiki rozmytej**. Prowadzi od koncepcji, przez implementację, aż po analizę i debugowanie — typowo edukacyjny i projektowy

https://github.com/JustFiesta/fuzzy-systems/blob/main/src/prompts/1/starter-context.txt

---

6. UI enhancements / optymalizacja (ui_enhancements.txt)

Prompt stricte **teoretyczny**: dotyczy poprawy responsywności UI oraz sposobów dystrybucji aplikacji (desktop, docker, alternatywy). Bez kodu — same koncepcje do wyboru

https://github.com/JustFiesta/fuzzy-systems/tree/main/src/prompts/3

---

7. Notes / README pod obronę (notes-docs.txt)

Prompt przygotowujący **notatki do obrony projektu**. Skupia się na mechanizmach, decyzjach projektowych i tym, *co i dlaczego działa*, a nie tylko *że działa*

https://github.com/JustFiesta/fuzzy-systems/blob/main/src/prompts/4/notes-docs.txt