Zaawansowane Prompty dla Cursor AI

Prompty do Refaktoryzacji

1. Refaktoryzacja funkcji

```
Refaktoryzuj tę funkcję, aby była bardziej czytelna i wydajna. Zastosuj zasady SOLID i do
```

2. Optymalizacja kodu

```
Zoptymalizuj ten kod pod kątem wydajności. Zidentyfikuj potencjalne wąskie gardła i zapro
```

3. Modernizacja kodu

```
Przepisz ten kod używając nowoczesnych praktyk programistycznych dla [język programowania
```

Prompty do Debugowania

1. Analiza błędu

```
Przeanalizuj ten kod i zidentyfikuj potencjalne błędy. Wyjaśnij, co może powodować niepra
```

2. Rozwiązanie błędu

```
Napraw ten błąd i wyjaśnij, dlaczego wystąpił. Dodaj komentarze wyjaśniające rozwiązanie.
```

3. Profilowanie wydajności

```
Zidentyfikuj fragmenty kodu, które mogą powodować problemy z wydajnością. Zaproponuj opty
```

Prompty do Generowania Testów

1. Testy jednostkowe

```
Wygeneruj pełny zestaw testów jednostkowych dla tej funkcji. Uwzględnij przypadki granicz
```

2. Testy integracyjne

```
Stwórz testy integracyjne dla tego modułu. Uwzględnij różne scenariusze użycia.
```

3. Testy wydajnościowe

```
Wygeneruj testy wydajnościowe dla tej funkcji. Sprawdź zachowanie przy różnych rozmiarach
```

Prompty do Analizy Kodu

1. Analiza złożoności

```
Przeanalizuj złożoność czasową i przestrzenną tego algorytmu. Zaproponuj ulepszenia.
```

2. Analiza bezpieczeństwa

```
Sprawdź ten kod pod kątem potencjalnych luk w zabezpieczeniach. Zidentyfikuj zagrożenia.
```

3. Analiza zgodności ze standardami

```
Sprawdź zgodność tego kodu ze standardami kodowania dla [język/framework]. Zaproponuj pop
```

Prompty do Dokumentacji

1. Generowanie dokumentacji

```
Wygeneruj kompletną dokumentację dla tej funkcji/klasy. Uwzględnij parametry, wartości zw
```

2. Komentarze w kodzie

```
Dodaj odpowiednie komentarze do tego kodu. Wyjaśnij złożone fragmenty i logikę biznesową.
```

3. README dla projektu

```
Stwórz plik README dla tego projektu. Uwzględnij instalację, użycie i przykłady.
```

Prompty do Architektury

1. Wzorce projektowe

```
Zaproponuj odpowiedni wzorzec projektowy dla tego problemu. Zaimplementuj rozwiązanie.
```

2. Architektura modułowa

```
Przeprojektuj ten kod na modułową architekturę. Zwiększ możliwość ponownego użycia i łatw
```

3. Separacja odpowiedzialności

```
Zastosuj zasadę Single Responsibility Principle do tego kodu. Podziel na mniejsze, wyspec
```

Prompty do LLMOps

1. Monitoring promptów

```
Wygeneruj system monitorowania dla tego promptu. Uwzględnij metryki wydajności i jakości.
```

2. Wersjonowanie promptów

```
Stwórz system wersjonowania dla promptów. Umożliw łatwe rollback i porównywanie wersji.
```

3. Testy A/B promptów

```
Zaprojektuj framework do testów A/B różnych wersji promptów. Zdefiniuj metryki sukcesu.
```

Zaawansowane Techniki

1. Kontekstowe prompty

```
@codebase Użyj kontekstu całego projektu, aby zaproponować rozwiązanie tego problemu.
```

2. Prompty z dokumentacją

```
@docs Wykorzystaj dokumentację [biblioteka/framework] do implementacji tej funkcjonalnośc
```

3. Prompty z wyszukiwaniem

```
@web Znajdź najnowsze best practices dla tego problemu i zastosuj je w kodzie.
```

Przykłady Użycia

Case Study 1: Refaktoryzacja Legacy Code

```
Przeanalizuj ten legacy kod i wykonaj następujące kroki:
1. Zidentyfikuj code smells
2. Zaproponuj plan refaktoryzacji
3. Zaimplementuj zmiany krok po kroku
4. Dodaj testy dla zrefaktoryzowanego kodu
5. Udokumentuj zmiany
```

Case Study 2: Optymalizacja Wydajności

```
Zoptymalizuj tę funkcję pod kątem wydajności:
1. Zprofiluj obecne działanie
2. Zidentyfikuj wąskie gardła
3. Zaproponuj optymalizacje
4. Zaimplementuj zmiany
5. Zweryfikuj poprawę wydajności
```

Case Study 3: Implementacja Nowej Funkcjonalności

```
Zaimplementuj nową funkcjonalność zgodnie z wymaganiami:
1. Przeanalizuj wymagania
2. Zaprojektuj architekturę
3. Zaimplementuj kod
4. Dodaj testy
5. Udokumentuj rozwiązanie
```

Wskazówki

1. **Bądź precyzyjny** - Im bardziej szczegółowy prompt, tym lepsze wyniki
2. **Używaj kontekstu** - Wykorzystuj funkcje @codebase, @docs, @web
3. **Iteruj** - Rozwijaj prompty na podstawie otrzymanych wyników
4. **Testuj** - Sprawdzaj wygenerowany kod przed użyciem
5. **Dokumentuj** - Zachowaj udane prompty do ponownego użycia

Najlepsze Praktyki

1. Zawsze sprawdzaj wygenerowany kod
2. Używaj odpowiednich modeli AI dla różnych zadań
3. Łącz prompty z narzędziami deweloperskimi
4. Regularnie aktualizuj prompty
5. Monitoruj jakość generowanych rozwiązań