1. Krótki opis projektu

W naszej grupie z sukcesem zrealizowaliśmy projekt stworzenia uproszczonej, jednak funkcjonalnej wersji aplikacji społecznościowej. Aplikacja ta, choć pozbawiona skomplikowanych funkcji, oferuje podstawowe, ale kluczowe możliwości interakcji między użytkownikami.

Użytkownicy mają możliwość publikowania własnych postów, co umożliwia dzielenie się przemyśleniami, doświadczeniami czy ciekawostkami. Ponadto, nasza aplikacja umożliwia komentowanie postów przez innych użytkowników, co sprzyja dynamicznym dyskusjom i wymianie opinii. Dodatkowo, istnieje funkcja 'lajkowania' postów, co pozwala użytkownikom na szybkie wyrażanie aprobaty lub zainteresowania danymi treściami.

2. Architektura

Jaka architektura została wykorzystana do realizacji projektu

W naszym projekcie zdecydowaliśmy się wykorzystać technologię Django. Django to otwarto-źródłowy framework webowy napisany w języku Python, który umożliwia szybkie tworzenie aplikacji internetowych. Jest oparty na architekturze Model-View-Controller (MVC) lub bardziej dokładnie Model-Template-View (MTV), która pomaga w organizacji kodu i separacji warstw aplikacji.

Architektura Model-Template-View w Django składa się z trzech kluczowych komponentów:

1. **Model**: Stanowi fundament architektury, reprezentując strukturę danych aplikacji oraz mechanizmy interakcji z bazą danych. Modele w Django są definiowane w sposób deklaratywny, co pozwala na łatwą konfigurację relacji między danymi, a także ich walidację. Dzięki ORM (Object-Relational Mapping) zapewnionemu przez Django, operacje na bazie danych stają się intuicyjne i bezpieczne, minimalizując ryzyko błędów.
2. **Widok (View)**: Jest kluczowym elementem obsługującym logikę aplikacji. Widoki Django są odpowiedzialne za przetwarzanie żądań od użytkowników, interakcję z modelami w celu pozyskania potrzebnych danych, a następnie przekazywanie ich do szablonów. W Django widoki mogą przyjmować różne formy – od prostych funkcji po bardziej złożone klasy – co pozwala na elastyczne dopasowanie do wymagań projektu.
3. **Szablony (Templates)**: Odpowiadają za warstwę prezentacji aplikacji. Szablony w Django są wyjątkowo elastyczne – pozwalają na tworzenie rozbudowanych interfejsów użytkownika z wykorzystaniem HTML, CSS oraz JavaScript. System tagów i filtrów Django oferuje potężne narzędzia do dynamicznego

renderowania treści, umożliwiając tworzenie interaktywnych i responsywnych interfejsów, które są zarówno estetyczne, jak i funkcjonalne.

3. Cechy Jakościowe

Opis głównych cech jakościowych wykorzystanych podczas realizacji projektu

W trakcie tworzenia naszego projektu identyfikujemy trzy kluczowe atrybuty jakościowe, które są istotne dla jego sukcesu. Te atrybuty, w kolejności priorytetów, to:

1. Użyteczność

Co to jest użyteczność?

Użyteczność, jako atrybut jakościowy, odnosi się do łatwości użytkowania danego produktu oraz komfortu, jakiego doświadcza użytkownik podczas interakcji z nim. Dotyczy ona stopnia wygody, z jakim użytkownicy mogą korzystać z oprogramowania, strony internetowej, aplikacji lub urządzenia.

Użyteczność opiera się na kilku kluczowych elementach. Po pierwsze, czy użytkownik może łatwo zrozumieć, jak korzystać z produktu bez większych trudności czy konieczności długiej nauki. Po drugie, czy produkt jest intuicyjny, czyli jego funkcje są jasne i naturalne dla użytkownika. Po trzecie, czy korzystanie z produktu jest wygodne i przyjemne, czy może sprawiać frustrację, być uciążliwe czy czasochłonne.

Dlaczego jest to istotne dla naszego projektu?

Użyteczność jest kluczowa dla aplikacji typu social media z kilku powodów:

1. **Przyciąganie i zatrzymywanie użytkowników:** Wysoka użyteczność sprawia, że aplikacja jest bardziej przystępna i łatwiejsza w obsłudze, co może przyciągać i zatrzymywać

użytkowników, ponieważ chętniej korzystają z platform, które są intuicyjne i proste w użyciu.

2. **Zwiększenie zaangażowania użytkowników:** Dobra użyteczność skłania do częstszego i dłuższego korzystania z aplikacji. Użytkownicy, którzy łatwo odnajdują funkcje i interakcje, są bardziej zaangażowani, co podnosi aktywność na platformie.
3. **Konkurencyjność na rynku:** Wśród aplikacji social media rywalizacja o uwagę użytkowników jest ogromna. Platformy o lepszej użyteczności mogą zyskać przewagę, przyciągając użytkowników dzięki łatwiejszej obsłudze w porównaniu z mniej intuicyjnymi rozwiązaniami.
4. **Wizerunek marki:** Użyteczność wpływa na doświadczenie użytkownika. Pozytywne doświadczenia związane z łatwością obsługi mogą budować pozytywną reputację marki, podczas gdy trudności w korzystaniu mogą prowadzić do negatywnych opinii.

2. Dostępność

Co to jest dostępność?

Dostępność, jako atrybut jakościowy, odnosi się do zdolności systemu do pozostawania dostępnym i funkcjonalnym dla użytkowników w określonym czasie. Oznacza to, że system jest łatwo dostępny wtedy, gdy jest potrzebny, niezależnie od ewentualnych awarii, problemów technicznych czy planowanych przerw konserwacyjnych. W przypadku platform internetowych ważne jest zapewnienie dostępności 24/7, umożliwiającej użytkownikom korzystanie z nich o dowolnej porze.

Dlaczego jest to istotne dla naszego projektu?

Dostępność jest niezwykle istotna dla aplikacji typu social media z kilku powodów:

1. **Ciągła dostępność usługi:** Użytkownicy korzystają z platform społecznościowych o różnych porach dnia i nocy. Wysoka dostępność gwarantuje, że użytkownicy z różnych stref

czasowych czy harmonogramów mogą korzystać z aplikacji w dogodnym dla siebie momencie.

2. **Zachowanie zaufania i lojalności użytkowników:** Niemożność dostępu lub częste problemy techniczne mogą zniechęcić użytkowników do korzystania z platformy, co prowadzi do utraty lojalności na rzecz konkurencyjnych usług.
3. **Ciągła interakcja i komunikacja:** Społeczności online opierają się na ciągłej interakcji. Dostępność jest kluczowa, aby użytkownicy mogli komunikować się, udostępniać treści czy komentować posty. Problemy z dostępnością mogą zakłócić te interakcje.
4. **Wizerunek niezawodności:** Stabilna dostępność platformy społecznościowej buduje reputację niezawodności, co jest istotne dla długoterminowego sukcesu aplikacji.

3. Bezpieczeństwo

Co to jest bezpieczeństwo?

Bezpieczeństwo, jako atrybut jakościowy w kontekście oprogramowania, dotyczy ochrony danych, systemu oraz użytkowników przed nieuprawnionym dostępem, atakami lub zagrożeniami. Polega na zabezpieczeniu programu przed próbami nieautoryzowanego dostępu, manipulacji danymi lub uszkodzeniem przez osoby trzecie. Zapewnienie bezpieczeństwa danych obejmuje zapewnienie poufności, integralności i dostępności informacji.

Dlaczego jest to istotne dla naszego projektu?

Bezpieczeństwo jest kluczowe dla aplikacji social media z kilku powodów:

1. **Prywatność użytkowników:** Platformy przechowują ogromne ilości danych osobowych. Bezpieczeństwo jest istotne dla ochrony prywatności użytkowników przed nieautoryzowanym dostępem czy ujawnieniem danych.

2. **Zaufanie i reputacja platformy:** Bezpieczna aplikacja buduje zaufanie użytkowników. W przypadku naruszeń bezpieczeństwa, użytkownicy mogą stracić zaufanie do platformy, co negatywnie wpływa na jej reputację.
3. **Ochrona przed atakami:** Social media są celem ataków hakerskich. Zapewnienie bezpieczeństwa minimalizuje ryzyko takich incydentów.

Pozostałe atrybuty jakościowe

Modyfikowalność

Modyfikowalność to zdolność systemu do elastycznej modyfikacji lub zmiany bez naruszania jego stabilności czy struktury. Choć istotna, nie jest ona priorytetem w porównaniu do wcześniej wspomnianych atrybutów. Jednakże, dobra modyfikowalność jest kluczowa dla ewentualnych zmian w projekcie.

Efektywność

Efektywność odnosi się do zdolności systemu do wykonywania operacji zminimalizowanym zużyciem zasobów. W początkowych etapach projektu może być mniej kluczowa, ale z czasem, w miarę wzrostu liczby użytkowników, staje się coraz bardziej istotna.

Testowalność

Testowalność dotyczy łatwości przeprowadzania testów na systemie. Choć istotna dla jakości kodu, użytkownicy końcowi zazwyczaj nie są zainteresowani tą kwestią.

4. Mechanizmy Napędzające

Opis mechanizmów napędzających

Mechanizmy napędzające, znane również jako "driving mechanisms", stanowią kluczowe elementy lub strategie generujące wzrost, zaangażowanie użytkowników, ich utrzymanie oraz zyski w aplikacjach. Szczególnie w aplikacjach social media istnieje kilka głównych mechanizmów napędzających, które są często wykorzystywane w procesie tworzenia i rozwoju tych platform:

1. Interakcja społecznościowa

Jest to centralny mechanizm, opierający się na interakcjach między użytkownikami. Dostarczanie funkcji, które ułatwiają komunikację, udostępnianie treści, komentowanie, lajkowanie i reagowanie na posty innych, jest kluczowe dla zaangażowania i budowania społeczności. Skupienie się na aspektach społecznych pomaga w utrzymaniu użytkowników na platformie przez dłuższy czas, co zwiększa potencjalne dochody z reklam.

2. Algorytmy personalizacji

Stosowanie algorytmów personalizacji treści wzmacnia zaangażowanie użytkownika, dostarczając mu treści dopasowane do jego zainteresowań i preferencji. To zwiększa czas spędzany na platformie, co bezpośrednio przekłada się na większe dochody z reklam.

3. Viralność i udostępnianie

Ułatwienie udostępniania treści przez użytkowników sprzyja rozprzestrzenianiu się treści i budowaniu społeczności. Mechanizmy te wspierają efekt viralności, przyczyniając się do szybszego wzrostu bazy użytkowników.

4. Kreowanie wartości użytkownika

Tworzenie wartościowych doświadczeń poprzez narzędzia umożliwiające wyrażanie siebie, tworzenie treści i budowanie relacji z innymi. Takie doświadczenia przyciągają użytkowników i zachęcają ich do spędzania więcej czasu na platformie.

5. Model ekonomiczny

Adaptacja modelu biznesowego, w tym reklamy, subskrypcje, funkcje premium i afiliacje, jest kluczowa dla generowania dochodów. Więcej czasu spędzonego przez użytkowników na platformie oznacza więcej możliwości do wyświetlania reklam i oferowania płatnych usług.

6. Gamifikacja

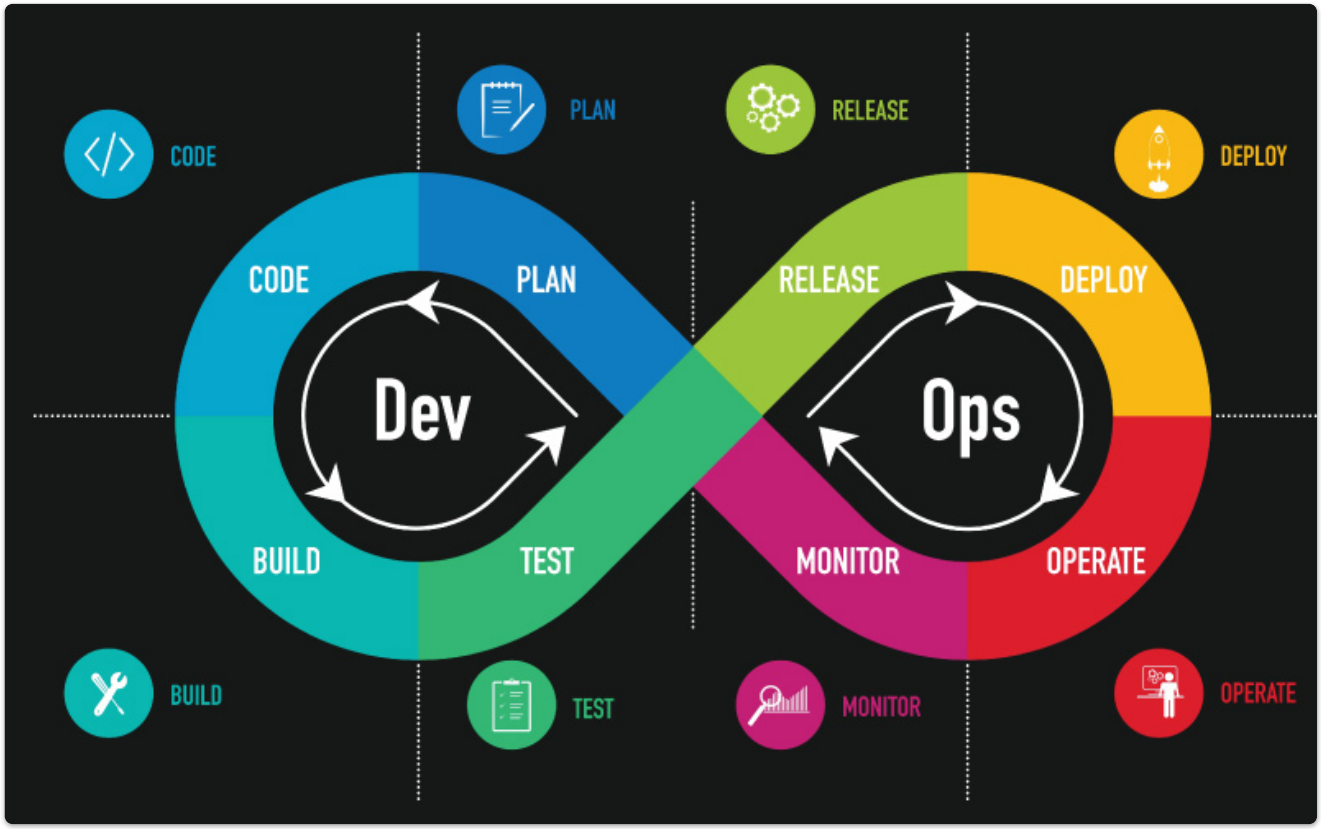
Wprowadzenie elementów gier, takich jak nagrody, poziomy, osiągnięcia, punkty i konkursy, zwiększa zaangażowanie użytkowników i zachęca ich do częstszego korzystania z aplikacji.

7. Analiza danych i dostosowanie

Wykorzystanie analizy danych do zrozumienia zachowań użytkowników i ich preferencji pozwala na dostosowywanie funkcji, treści i strategii marketingowej, co prowadzi do lepszego dopasowania oferty do potrzeb użytkowników i zwiększenia ich czasu spędzanego na platformie.

5. Cykl Biznesowy

Cykl biznesowy dla aplikacji typu social media obejmuje szereg etapów, które są istotne dla sukcesu takiego przedsięwzięcia. W istocie przypomina on cykl devops.



Nasz cykl będzie bardzo podobny. Oto główne fazy tego cyklu:

1. Badanie, Planowanie i Integracja

Integracja Analizy Rynku z Planowaniem Technicznym:

- Połączenie badań rynkowych i analizy konkurencji z technicznym planowaniem rozwoju. To obejmuje wybór narzędzi, platform i technologii zgodnych z potrzebami rynku.
- Stosowanie metod agile do planowania, umożliwiając elastyczne dostosowywanie się do zmieniających się wymagań i trendów rynkowych.

2. Projektowanie, Rozwój i Ciągłe Wdrażanie

Iteracyjne Projektowanie i Rozwój:

- Stosowanie iteracyjnych metodologii w procesie projektowania i rozwoju, umożliwiając szybką adaptację i zmiany.
- Wdrażanie systemów CI/CD (Continuous Integration/Continuous Deployment) do ciągłego testowania i wdrażania zmian w aplikacji.

Testowanie i Feedback:

- Automatyzacja testów w celu zapewnienia jakości i stabilności aplikacji.
- Wprowadzenie systemów do zbierania informacji zwrotnej od użytkowników na wczesnych etapach rozwoju, pozwalających na szybkie wprowadzanie poprawek.

3. Wdrożenie, Marketing i Ciągła Dostawa

Agile Wprowadzenie na Rynek i Marketing:

- Zastosowanie zwinnych metod wypuszczania aplikacji na rynek, pozwalających na szybkie reagowanie na opinie użytkowników i zmiany rynkowe.
- Integracja działań marketingowych z procesem wdrożenia, aby lepiej docierać do docelowych użytkowników.

Ciągła Dostawa i Wdrażanie Zmian:

- Utrzymywanie ciągłego procesu wdrażania zmian i aktualizacji w aplikacji.
- Wykorzystanie analityki i danych rynkowych do informowania o decyzjach dotyczących wdrażania nowych funkcji.

4. Utrzymanie, Wsparcie i Ciągła Optymalizacja

Ciągłe Wsparcie i Reagowanie na Feedback:

- Zapewnienie ciągłego wsparcia technicznego i obsługi klienta, szybko reagując na potrzeby użytkowników.

- Stosowanie metodologii lean i kaizen do ciągłego doskonalenia aplikacji na podstawie opinii użytkowników.

Analiza i Optymalizacja:

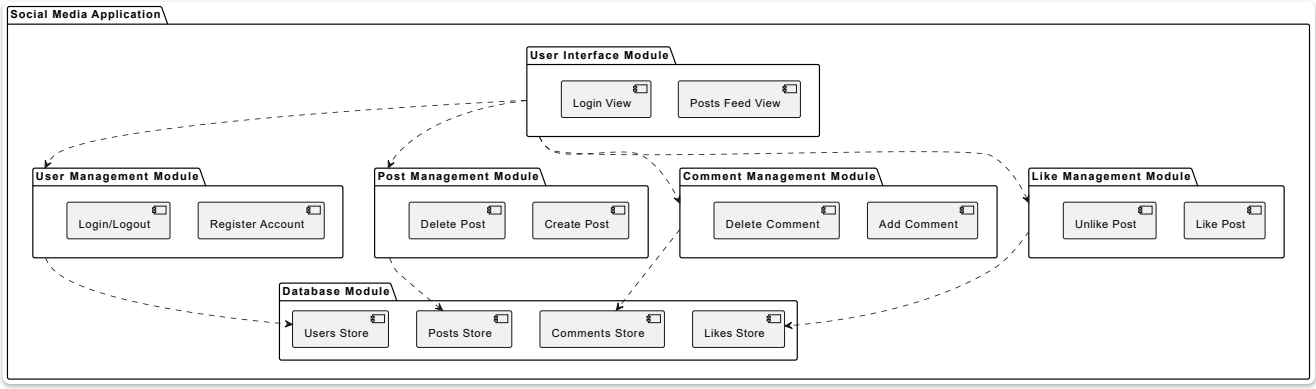
- Ciągła analiza danych użytkowania i zachowań użytkowników w celu optymalizacji aplikacji.
- Regularne aktualizacje i ulepszenia aplikacji, oparte na rzeczywistych danych i potrzebach użytkowników.

5. Monetyzacja i Ciągłe Dostosowywanie

Elastyczne Planowanie Modelu Biznesowego:

- Dynamiczne dostosowywanie strategii monetarnej w odpowiedzi na zmieniające się warunki rynkowe i preferencje użytkowników.
- Integracja modelu biznesowego z procesem rozwoju, aby zapewnić zgodność z potrzebami i oczekiwaniami użytkowników.

6. Dekompozycja Modułowa



Aplikacja Social Media we frameworku django składać się będzie z kilku modułów, które są ze sobą powiązane:

1. **Moduł Interfejsu Użytkownika (User Interface Module)**: Ten moduł obejmuje różne aspekty interfejsu użytkownika aplikacji, takie jak "Posts Feed View" (Widok Postów) i "Login View" (Widok Logowania). Ten moduł działa jako punkt interakcji z użytkownikami i jest powiązany z innymi modułami zarządzającymi działaniami użytkownika.
2. **Moduł Zarządzania Użytkownikami (User Management Module)**: Zawiera funkcje takie jak "Register Account" (Rejestracja Konta) i "Login/Logout" (Logowanie/Wylogowanie). Ten moduł jest powiązany z "User Interface Module" oraz z "Users Store" w module bazy danych, co oznacza, że zarządza danymi użytkowników i ich autentykacją.
3. **Moduł Zarządzania Postami (Post Management Module)**: Skupia się na zarządzaniu postami użytkowników, w tym na "Create Post" (Tworzenie Posta) i "Delete Post" (Usuwanie Posta). Jest połączony zarówno z interfejsem użytkownika, jak i z "Posts Store" w module bazy danych.
4. **Moduł Zarządzania Komentarzami (Comment Management Module)**: Umożliwia użytkownikom dodawanie ("Add Comment") i usuwanie komentarzy ("Delete Comment"). Jest powiązany z

interfejsem użytkownika oraz z "Comments Store" w module bazy danych.

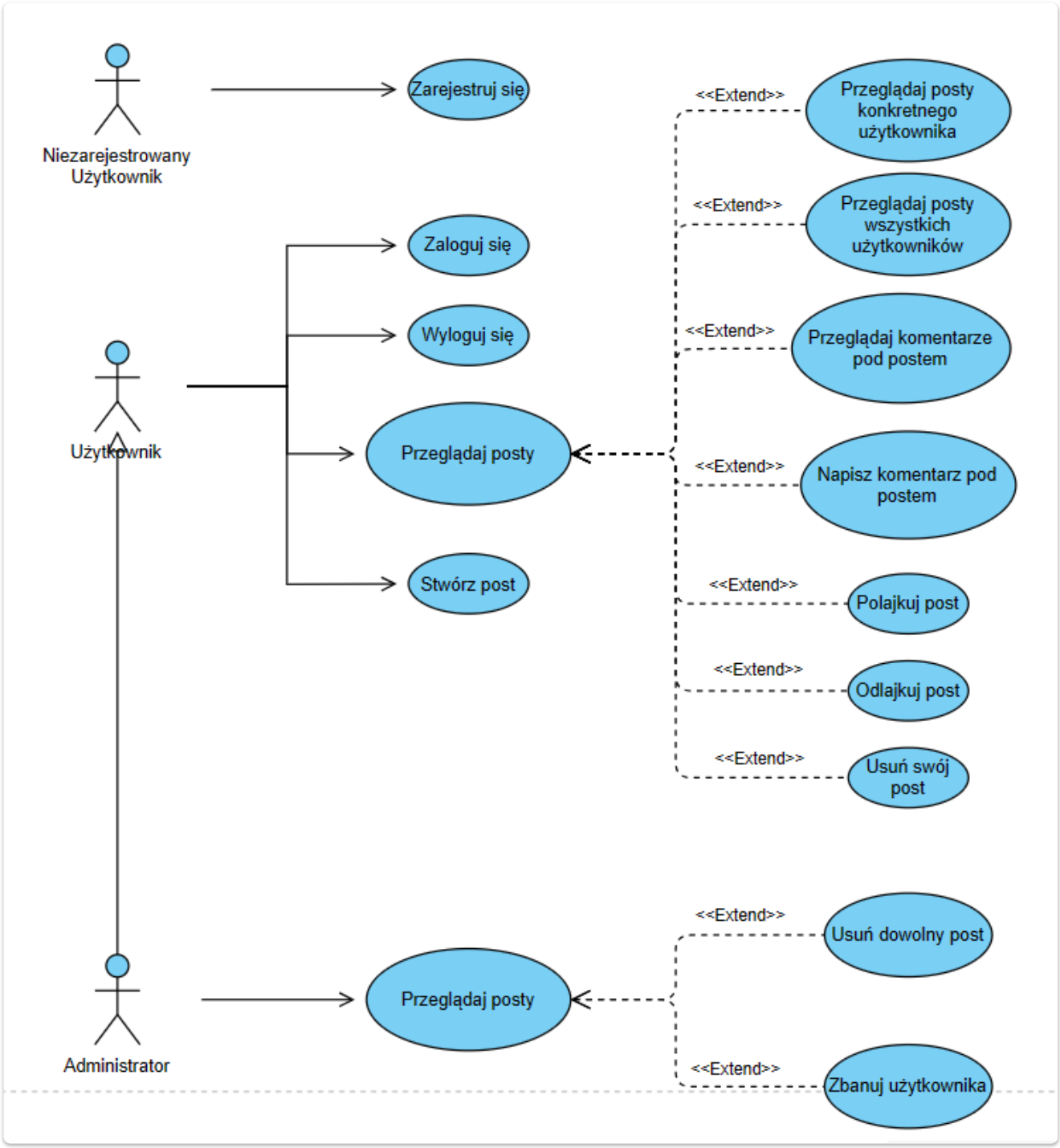
5. **Moduł Zarządzania Polubieniami (Like Management Module)**: Ten moduł pozwala na "Like Post" (Polubienie Posta) i "Unlike Post" (Odlubienie Posta). Podobnie jak inne moduły, łączy się z interfejsem użytkownika oraz z "Likes Store" w module bazy danych.
6. **Moduł Bazy Danych (Database Module)**: Zawiera różne składowe bazy danych, takie jak "Users Store", "Posts Store", "Comments Store", i "Likes Store". Każdy z nich przechowuje odpowiednie dane, które są wykorzystywane przez inne moduły.

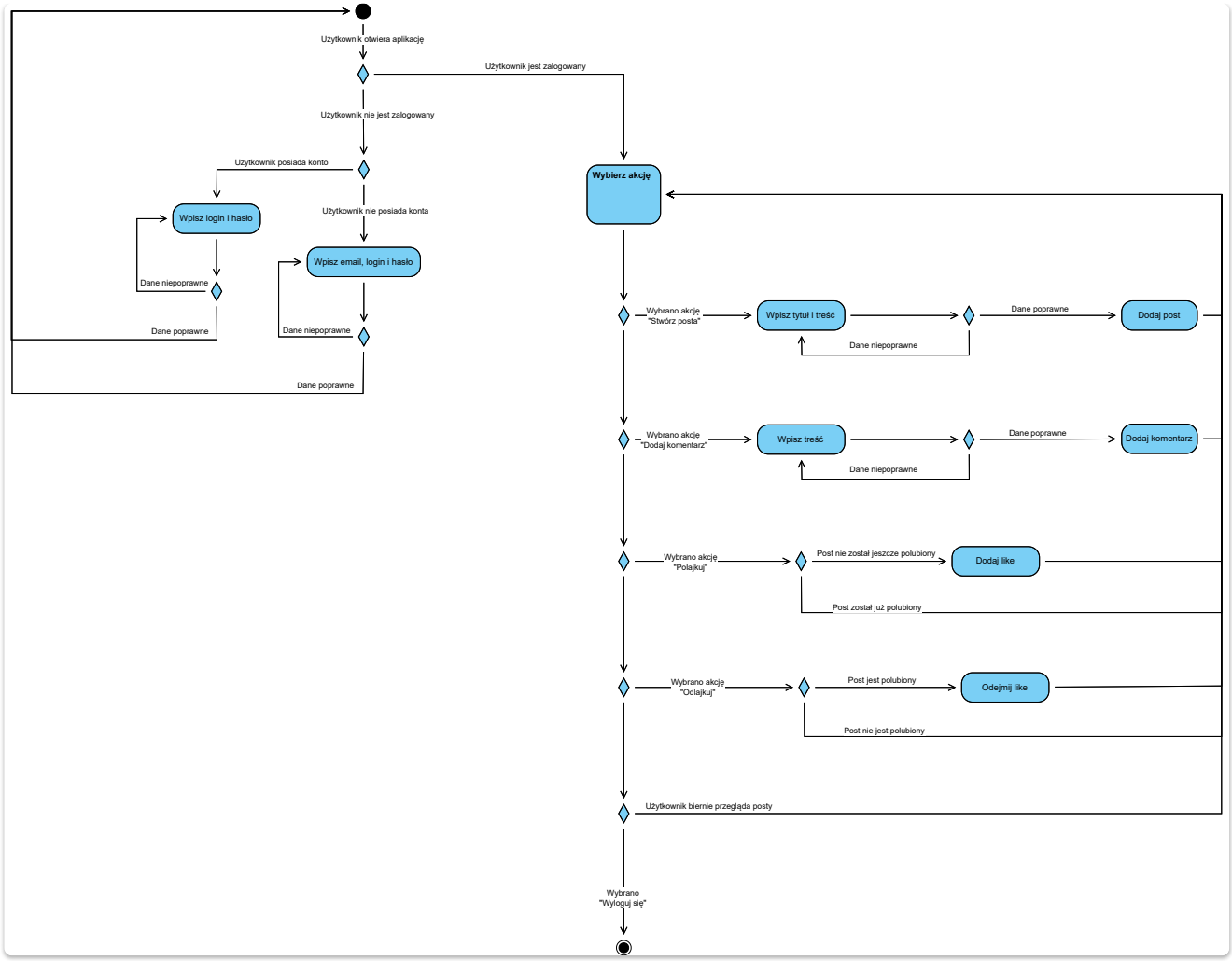
Strzałki w diagramie ("..>") wskazują na zależności między różnymi modułami, sugerując, że moduły interfejsu użytkownika korzystają z funkcji zapewnianych przez moduły zarządzania (Użytkownikami, Postami, Komentarzami, Polubieniami) oraz że te moduły zarządzania są z kolei zależne od odpowiednich składowych modułu bazy danych. To ukazuje modułową i warstwową naturę systemu, gdzie każdy moduł jest specjalizowany w określonych zadaniach i współpracuje z innymi modułami.

```
@startuml

package "Social Media Application" {
    package "User Interface Module" {
        [Posts Feed View]
        [Login View]
    }

    package "User Management Module" {
        [Register Account]
        [Login/Logout]
    }

    package "Post Management Module" {
```

```
        [Create Post]
        [Delete Post]
    }

    package "Comment Management Module" {
        [Add Comment]
        [Delete Comment]
    }

    package "Like Management Module" {
        [Like Post]
        [Unlike Post]
    }

    package "Database Module" {
        [Users Store]
        [Posts Store]
        [Comments Store]
        [Likes Store]
    }

    "User Interface Module" ..> "User Management Module"
    "User Interface Module" ..> "Post Management Module"
    "User Interface Module" ..> "Comment Management Module"
    "User Interface Module" ..> "Like Management Module"

    "User Management Module" ..> "Users Store"
    "Post Management Module" ..> "Posts Store"
    "Comment Management Module" ..> "Comments Store"
    "Like Management Module" ..> "Likes Store"
}

@enduml
```

7. Diagram czynności i przypadków użycia




Użytkownik otwiera aplikację
Użytkownik jest zalogowany
Użytkownik nie jest zalogowany
Użytkownik posiada konto
Użytkownik nie posiada konta
Wpisz login i hasło
Wpisz email, login i hasło
Dane niepoprawne
Dane poprawne
Dane niepoprawne
Dane poprawne
Wybierz akcję
Wybrano akcję "Stwórz posta"
Wpisz tytuł i treść
Dane poprawne
Dodaj post
Dane niepoprawne
Wybrano akcję "Dodaj komentarz"
Wpisz treść
Dane poprawne
Dodaj komentarz
Dane niepoprawne
Wybrano akcję "Polajkuj"
Post nie został jeszcze polubiony
Dodaj like
Post został już polubiony
Wybrano akcję "Odlajkuj"
Post jest polubiony
Odejmij like
Post nie jest polubiony
Użytkownik biernie przegląda posty
Wybrano "Wyloguj się"

8. Możliwości rozwoju

Czy i jak zadbano o możliwość modyfikacji oprogramowania po jego wdrożeniu

Możliwość modyfikacji oprogramowania jest ściśle związana z zastosowaną architekturą, która ma dużo narzędzi pozwalających radzić sobie z tym problemem. Architektura Model-Template-View (MTV) w Django, podobnie jak wiele innych architektur opartych na separacji warstw, stara się zapewnić elastyczność i łatwość w modyfikowaniu oprogramowania po jego wdrożeniu. Kilka kluczowych punktów dotyczących tego:

1. **Oddzielenie warstw**: MTV w Django zapewnia jasne rozdzielenie warstw modelu danych, logiki biznesowej (widoków) i warstwy prezentacji (szablonów). To ułatwia modyfikację jednej części aplikacji bez wpływu na pozostałe. Gdy konieczne są zmiany w logice biznesowej, można to zrobić w widokach, a zmiany w interfejsie użytkownika w szablonach, bez konieczności ingerencji w modele danych.
2. **Hermetyzacja**: Moduły i komponenty w Django są często zaprojektowane z myślą o hermetyzacji, co oznacza, że poszczególne części aplikacji są izolowane i mają zdefiniowane interfejsy, co ułatwia zmiany w jednej części bez wpływu na inne.
3. **Skalowalność**: Dzięki wykorzystaniu takiej architektury możliwe jest łatwiejsze dostosowanie aplikacji do nowych wymagań czy potrzeb, ponieważ każda warstwa może być modyfikowana niezależnie. To umożliwia szybkie reagowanie na zmiany i rozszerzanie funkcjonalności.
4. **Testowalność**: Clear separation of concerns (CSoC) pomaga w testowaniu poszczególnych warstw osobno. Dzięki temu zmiany

w kodzie można weryfikować poprzez testy jednostkowe, co zwiększa pewność co do funkcjonalności modyfikacji.

5. **Zastosowanie wzorców projektowych**: Django promuje stosowanie wzorców projektowych, co ułatwia rozpoznanie struktury aplikacji. Korzystanie z takich wzorców jak np. wzorzec projektowy MVC (lub MTV w przypadku Django) pomaga w zrozumieniu struktury aplikacji i ułatwia wprowadzanie zmian.