**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**Державне некомерційне підприємство**
**«Державний університет» Київський авіаційний інститут»**

**Факультет комп'ютерних наук та технологій**

**Кафедра інженерії програмного забезпечення**

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри

_____ Олена ГРІНЕНКО
«_____» _____ 2025 р.

# КВАЛІФІКАЦІЙНА РОБОТА
## (ПОЯСНЮВАЛЬНА ЗАПИСКА)

**ЗДОБУВАЧА ОСВІТНЬОГО СТУПЕНЯ «МАГІСТР»**
ЗАОЧНА ФОРМА ЗДОБУТТЯ ОСВІТИ

**Тема:** Автоматизація управління Discord-серверу на базі технологій штучного інтелекту і методології Agile

**Виконавець:** Удодов Максим Юрійович

**Керівник**: к. т. н., доцент Волкогон Вікторія Олексіївна

**Нормоконтролер:** к. т. н., доцент Волкогон Вікторія Олексіївна

**Київ 2025**

**Державне некомерційне підприємство**
**«Державний університет» Київський авіаційний інститут»**

Факультет комп'ютерних наук та технологій
Кафедра інженерії програмного забезпечення
Спеціальність 121 «Інженерія програмного забезпечення»
Освітньо-професійна програма «Інженерія програмного забезпечення»
Заочна форма здобуття освіти

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Олена ГРІНЕНКО

«_____» _____ 2025 р.

ЗАВДАННЯ
на виконання кваліфікаційної роботи студента
Удодова Максима Юрійовича

1. Тема кваліфікаційної роботи: «Автоматизація управління Discord-серверу на базі технологій штучного інтелекту і методології Agile»
затверджена наказом ректора від 17.11.2025 р. № 2449/ст

2. Термін виконання роботи: з 29.10.2025 р. по 14.12.2025 р.

3. Вихідні дані для роботи: розробити програмний засіб – багатомодульного Discord-бота для автоматизації керування сервером та інтелектуальної модерації контенту – з використанням мови програмування Python, бібліотеки discord.py, сервісів HuggingFace та середовища розробки (replit).

4. Зміст пояснювальної записки:
   1. Дослідження предметної області керування Discord-сервером та аналіз існуючих бот-рішень.
   2. Аналіз методів інтелектуальної модерації контенту (правила, ML-класифікація, LLM-підходи)
   3. Технології, архітектура та компоненти програмного засобу Discord-бота InfinityBot
   4. Реалізація та тестування програмного засобу інтелектуальної модерації й автоматизації керування Discord-сервером.

5. Перелік обов'язкових слайдів презентації:
   1. Перелік обов'язкового графічного (ілюстративного) матеріалу:
   2. Структурна схема системи автоматизації керування Discord-сервером

(загальна архітектура бота).

3. Діаграма функціональних можливостей програмного засобу InfinityBot (основні модулі та сервіси).
4. Інтерфейс системи адміністрування та налаштування бота (приклади команд, панелей, журналів подій).
5. Схема роботи підсистеми AI-модерації (трирівневий конвеєр: rule-based → ML-шар HuggingFace → fallback-перевірки).
6. Демонстрація роботи програми на тестовому Discord-сервері (послідовність основних сценаріїв).
7. Демонстрація роботи окремих модулів програми (AIModeration, Autoroles, Channels, Security тощо

6. Календарний план-графік

| № пор. | Завдання | Термін виконання | Відмітка про виконання |
|---|---|---|---|
| 1. | Розробка та затвердження графіка роботи | 29.09-01.10.2025 | Виконано |
| 2. | Ознайомлення з постановкою задачі, вивчення інформаційних джерел та складання плану роботи. | 02.10-05.10.2025 | Виконано |
| 2. | Підготовка 1 розділу та подання його керівнику | 06.10-19.10.2025 | Виконано |
| 3. | Підготовка 2 розділу та подання його керівнику | 20.10-02.11.2025 | Виконано |
| 4. | Підготовка 3 розділу та подання його керівнику | 03.12-16.11.2025 | Виконано |
| 5. | Підготовка 4 розділу і висновків по роботі та подання їх керівнику | 17.11-30.11.2025 | Виконано |
| 6. | Загальне редагування пояснювальної записки, графічного матеріалу.<br>Представлення роботи для перевірки на академічну доброчесність.<br>Проходження нормоконтролю. | 01.12-07.12.2025 | Виконано |
| 7. | Отримання відгуку керівника.<br>Підготовка презентації та тексту доповіді. | 08.12-14.12.2025 | |
| 8. | Попередній захист (представлення електронної версії пояснювальної записки, презентації, позитивного відгуку керівника). | 08.12-14.12.2025 | |
| 9. | Рецензування кваліфікаційної роботи | 15.12-18.12.2025 | |
| 10. | Здача секретарю ЕК пояснювальної записки: електронної версії кваліфікаційної роботи; презентації доповіді; відгуку керівника, рецензії; результату проходження перевірки на плагіат; довідки про успішність, декларації про академічну доброчесність. | 15.12-18.12.2025 | |
| 11. | Захист кваліфікаційної роботи перед екзаменаційною комісією | 23.12.2025 | |

Дата видачі завдання 29.09.2025 р.

Керівник кваліфікаційної роботи:
к. т. н., доцент Вікторія ВОЛКОГОН

Завдання прийняв до виконання: Вікторія ВОЛКОГОН

# РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи «Програмний засіб інтелектуальної модерації та автоматизації керування Discord-сервером»: 61 сторінок, 16 рисунків, 7 таблиці, 26 джерел, 2 додатки.

**Об'єкт дослідження** – процес керування Discord-сервером та модерації користувацького контенту.

**Мета роботи** – розробити програмний засіб Discord-бота з трирівневою AI-модерацією (правила, ML через HuggingFace, fallback-евристики) та засобами автоматизації керування сервером.

**Методи дослідження** – системний і порівняльний аналіз, моделювання процесів обробки повідомлень, методи машинного навчання для класифікації токсичного контенту, прототипування та експериментальне тестування.

**Результати** можуть бути використані для адміністрування й модерації Discord-серверів різних спільнот та як основа для систем автоматизованої модерації контенту в інших платформах.

Розробка виконана під ОС Windows 11 у хмарному середовищі Replit з використанням Python, discord.py та Replit DB (платне розміщення).

DISCORD-БОТ, АВТОМАТИЗАЦІЯ, AI-МОДЕРАЦІЯ, HUGGINGFACE, REPLIT

## ABSTRACT

Explanatory note to the qualification work “Software tool for intelligent moderation and automated management of a Discord server”: about 61 pages, 16 figures, 7 tables, 26 references, 2 appendices.

**Object of study** – management of a Discord server and moderation of user-generated content.

**The aim is to develop** a Discord bot with three-layer AI-based moderation (rules, ML via HuggingFace, fallback heuristics) and tools for automated server management.

**Research methods** – system and comparative analysis, modelling of message-processing workflows, machine learning methods for toxic content classification, prototyping and experimental testing.

**The results** can be used for administration and moderation of Discord servers of various communities and as a basis for automated content moderation systems on other platforms.

Development was carried out under Windows 11 in the Replit cloud environment using Python, discord.py and Replit DB (paid hosting).

DISCORD BOT, AUTOMATION, AI-BASED MODERATION, HUGGINGFACE, REPLIT

# ЗМІСТ

## ПЕРЕЛІК ПРИЙНЯТИХ СКОРОЧЕНЬ

ЗВО – заклад вищої освіти

ІТ – інформаційні технології

ПЗ – програмне забезпечення

AI – Artificial Intelligence (штучний інтелект)

ML – Machine Learning (машинне навчання)

LLM – Large Language Model (велика мовна модель)

API – Application Programming Interface (інтерфейс прикладного програмування)

HTTP – HyperText Transfer Protocol (протокол передавання гіпертексту)

JSON – JavaScript Object Notation (формат обміну даними)

DB – DataBase (база даних)

ID – Identifier (ідентифікатор)

UI – User Interface (інтерфейс користувача)

CLI – Command Line Interface (інтерфейс командного рядка)

ACL – Access Control List (список контролю доступу)

# ВСТУП

**Актуальність теми дослідження** полягає в тому, що зростання кількості онлайн-спільнот, зокрема у середовищі Discord, супроводжується різким збільшенням обсягу користувацького контенту та токсичної поведінки. Ручна модерація стає ресурсомісткою, суб'єктивною та часто запізнілою, що призводить до конфліктів, вигорання модераторів і погіршення якості комунікації в спільнотах. Паралельно активно розвиваються методи машинного навчання та сервіси на кшталт HuggingFace, які дають змогу автоматизувати аналіз тексту та прийняття рішень щодо небажаного контенту. На практиці ж більшість популярних Discord-ботів або використовують примітивні rule-based фільтри, або інтегрують AI поверхнево, без продуманої архітектури, багаторівневої логіки, fallback-механізмів та чіткого врахування обмежень платформи. Це створює нішу для розробки цілісного програмного засобу, який поєднує трирівневу AI-модерацію, автоматизацію керування сервером і сучасні підходи до розробки.

Крім того, адміністратори серверів змушені паралельно розв'язувати задачі видачі та керування ролями, обмеження доступу до каналів, логування дій, налаштування прав і параметрів, що ускладнює підтримку великих спільнот. Відсутність єдиного інструменту, який би водночас вирішував питання модерації, безпеки, автоматизації та спирався на перевірені інженерні практики (Agile, розділення відповідальності, ACL), робить розробку такого програмного засобу доцільною й актуальною.

**Мета кваліфікаційної роботи** – розробка програмного засобу Discord-бота з трирівневою AI-модерацією та засобами автоматизації керування сервером, який зменшує рівень токсичності комунікацій, підвищує безпеку та скорочує обсяг ручної роботи адміністраторів.

**Для досягнення поставленої мети необхідно вирішити такі основні завдання:**

— виконати аналіз предметної області керування Discord-серверами та існуючих рішень для модерації та автоматизації;

— дослідити сучасні підходи до інтелектуальної модерації контенту (правила, ML-моделі, LLM, сервіси HuggingFace) та оцінити можливості їх інтеграції в Discord-бота;

— спроєктувати архітектуру багатомодульного програмного засобу з виділенням підсистем AI-модерації, керування ролями, керування каналами, конфігурації та безпеки (ACL);

— реалізувати трирівневу систему модерації повідомлень (rule-based шар, ML-шар на базі HuggingFace, fallback-евристики при недоступності ML);

— розробити засоби автоматизації керування Discord-сервером (автоматична видача та зняття ролей, обмеження каналів, логування подій, механізми конфігурації);

— впровадити програмний засіб у хмарному середовищі Replit з використанням Replit DB та провести експериментальне тестування основних сценаріїв роботи;

— виконати оцінку якості AI-модерації на тестовій вибірці повідомлень (precision, recall, хибно-позитивні та хибно-негативні спрацьовування) та проаналізувати результати.

**Об’єкт дослідження** – процеси автоматизованого керування Discord-сервером та модерації користувацького текстового контенту в онлайн-спільнотах.

**Предмет дослідження** – методи та програмні засоби трирівневої AI-модерації та автоматизації керування Discord-сервером, включно з архітектурою бота, алгоритмами ухвалення модераційних рішень, механізмами керування ролями, каналами, правами доступу та конфігурацією.

**Методи дослідження** включають системний та порівняльний аналіз існуючих ботів та рішень для модерації, моделювання процесів обробки повідомлень і життєвого циклу модераційного рішення, застосування методів машинного навчання

для класифікації токсичного контенту із використанням сервісів HuggingFace, прототипування та інкрементальну розробку програмного забезпечення за підходом Agile, а також експериментальну оцінку якості модерації на основі базових статистичних показників.

**Наукова новизна отриманих результатів полягає у:**

1. Побудові трирівневої архітектури AI-модерації для Discord-бота, яка поєднує rule-based фільтрацію, ML-класифікацію на базі віддаленого сервісу HuggingFace та fallback-евристики у випадку недоступності ML-шару;
2. Формалізації процесу ухвалення модераційних рішень як послідовності взаємодіючих шарів, що дозволяє гнучко налаштовувати порогові значення, політику дій та сценарії деградації;
3. Інтеграції механізмів AI-модерації з підсистемами керування ролями, каналами та ACL у єдиному програмному засобі, орієнтованому на практичне використання в реальних спільнотах.

**Практичне значення отриманих результатів** полягає в створенні працюючого програмного засобу Discord-бота, розгорнутого у хмарному середовищі Replit з використанням Replit DB, який може бути застосований для адміністрування та модерації серверів навчальних, ігрових та тематичних спільнот. Розроблені підходи та архітектурні рішення можуть бути використані як основа для подальшого розширення функціональності, перенесення на інші платформи або інтеграції з альтернативними AI-сервісами.

**Особистий внесок** здобувача вищої освіти полягає в самостійній розробці архітектури бота, реалізації основних модулів (AI-модерації, керування ролями, каналами, конфігурацією та ACL), налаштуванні взаємодії з Discord API та сервісами HuggingFace, організації зберігання даних у Replit DB, підготовці тестових сценаріїв і аналізі результатів експериментального дослідження. Усі ключові програмні рішення були спроєктовані та реалізовані автором роботи; ідеї та рекомендації наукового керівника стосувалися переважно вибору загального напряму дослідження та структури кваліфікаційної роботи.

**Апробація отриманих результатів** здійснювалася в процесі обговорення проміжних результатів із науковим керівником та під час внутрішніх консультацій на випусковій кафедрі. Основні положення роботи можуть бути використані як база для доповідей на студентських науково-практичних конференціях, присвячених застосуванню AI в модерації контенту та автоматизації адміністрування онлайн-спільнот.

**Публікації за темою кваліфікаційної роботи** на момент завершення не здійснювалися; отримані результати можуть слугувати основою для подальших наукових статей, зокрема у напрямках практичного застосування AI для модерації контенту, побудови багаторівневих систем ухвалення рішень та інженерії Discord-ботів із використанням хмарної інфраструктури.

# РОЗДІЛ 1
# ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ КЕРУВАННЯ DISCORD-СЕРВЕРОМ ТА АНАЛІЗ ІСНУЮЧИХ БОТ-РІШЕНЬ

## 1.1. Характеристика платформи Discord та задачі керування сервером

Discord – це платформа для голосового, текстового та відеозв'язку, орієнтована на створення спільнот навколо ігор, навчання, професійних та тематичних інтересів. Сервер Discord складається з набору текстових і голосових каналів, ролей користувачів, прав доступу та інтеграцій із зовнішніми сервісами.

Керування сервером включає такі ключові задачі:

- налаштування структури серверу (категорії, канали, тематичні зони, службові канали для оголошень тощо);
- управління ролями та правами доступу (розподіл повноважень між адміністраторами, модераторами, звичайними учасниками, гостьовими ролями);
- модерація контенту та поведінки користувачів (видалення небажаних повідомлень, попередження, мут, кік, бан);
- автоматизація рутинних дій (привітальні повідомлення, автопризначення ролей, періодичні нагадування, логування подій);
- забезпечення безпеки та прозорості (журнали дій, аудит змін, захист від рейдів і спаму).

У невеликих спільнотах частину цих задач можна виконувати вручну, але зі зростанням кількості учасників та активності така модель стає неефективною. Це безпосередньо підводить до потреби у спеціалізованих програмних засобах – Discord-ботах, які беруть на себе значну частину рутинної роботи адміністрації.

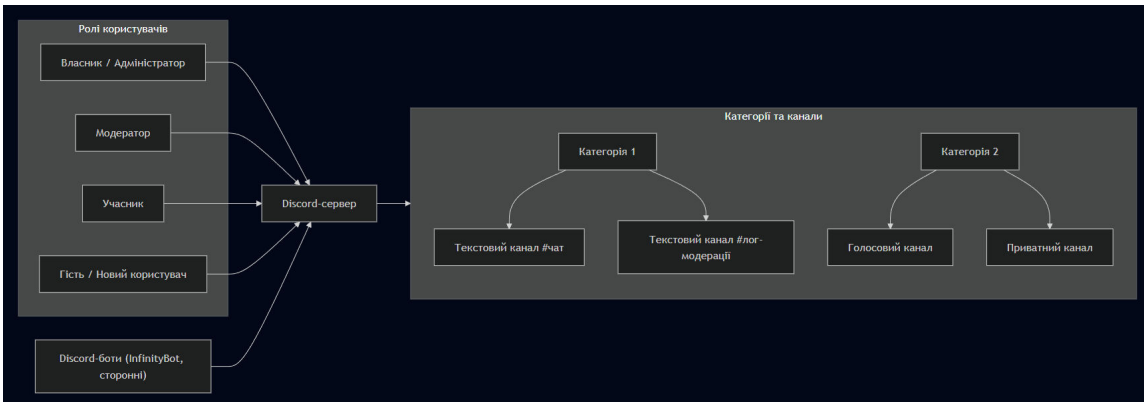


Рис. 1.1. Узагальнена структура Discord-сервера та основні ролі керування

## 1.2. Типові проблеми модерації та адміністрування онлайн-спільнот

Модерація в активних спільнотах стикається з низкою повторюваних проблем:

- Високий обсяг повідомлень. Потік тексту в популярних каналах унеможливлює повний ручний перегляд усіх повідомлень. Частина токсичного контенту неминуче проходить повз модераторів.
- Токсична поведінка та порушення правил. Образи, мова ворожнечі, спам, флуд, NSFW-контент, намагання обійти фільтри через обфускацію (заміна символів, вставка пробілів тощо).
- Людський фактор. Модератори втомлюються, можуть діяти непослідовно, пропускати порушення або, навпаки, реагувати занадто жорстко.
- Обмеження Discord API. Існують ліміти на кількість запитів, обмеження по швидкості операцій, вимоги до затримки реакції, що накладає додаткові вимоги до будь-якої автоматизації.
- Складність керування ролями та каналами. При великій кількості ролей (авторолі, тимчасові ролі, ролі доступу до окремих каналів) ручне керування стає джерелом помилок і конфліктів доступу.

Ці фактори мотивують створення ботів, які:

- автоматизують базову фільтрацію контенту та дії по модерації;
- надають зручні інструменти керування ролями та каналами;
- забезпечують прозорість (логування дій) та швидку реакцію на порушення.

Однак класичні rule-based рішення (простий фільтр за словами) сьогодні недостатні: користувачі легко обходять такі фільтри, а контекстні форми токсичності часто вимагають аналізу «змісту», а не лише набору символів. Це відкриває нішу для використання методів машинного навчання та AI-модерації.

### 1.3. Класифікація існуючих Discord-ботів

Discord-боти умовно можна поділити на кілька груп:

- Модераційні боти – зосереджені на антиспамі, фільтрації контенту, автоматичних покараннях (mute/ban), логуванні порушень.
- Боти для керування ролями й каналами – авторолі, реакційні ролі, гнучке налаштування доступу до каналів, тимчасові ролі.
- Боти для залучення користувачів – системи рівнів, досвіду (XP), івенти, музика, міні-ігри.
- Інтеграційні боти – пов’язують Discord з зовнішніми сервісами (GitHub, Trello, YouTube тощо).
- Комбіновані «комплексні» боти – поєднують одразу кілька напрямів: модерацію, ролі, логування, економіку, музику і т.д.

Приклади комплексних ботів:

- MEE6 – один із найбільш популярних ботів, що надає модераційні інструменти, систему рівнів, кастомні команди, автоматизацію дій та інші функції.
- Carl-bot – модульний бот із реакційними ролями, Automod, логуванням, кастомними командами та широкими можливостями керування сервером.
- Dyno – бот з акцентом на модерацію та автоматизацію (фільтри, авто-модерація, автоповідомлення).

Таблиця 1.1

Узагальнене порівняння функцій популярних Discord-ботів MEE6, Carl-bot, Dyno

| Характеристика / Бот | MEE6 | Carl-bot | Dyno |
|---|---|---|---|
| Основний фокус | Модерація, рівні, автоматика | Ролі (особливо реакційні), логування | Модерація, автоматизація |
| Модерація (фільтри, авто-санкції) | Є (Automod, слова, ліміти) | Є (Automod, базові фільтри) | Є (розширені фільтри, авто-санкції) |
| AI / ML-модерація | Ні / дуже обмежено | Ні | Ні |
| Авторолі / реакційні ролі | Є (частково платно) | Сильна сторона (гнучкі реакційні ролі) | Є |
| Система рівнів / XP | Так (ключова функція) | Є (але не основний акцент) | Є |
| Логування подій | Є (залежить від налаштувань) | Сильна сторона (детальне логування) | Є |
| Кастомні команди / автоматика подій | Є (особливо в преміум) | Є (кастомні команди, теги) | Є (custom commands, авто-оголошення) |
| Інші функції (музика, фановий функціонал) | Є (музика, повідомлення, інше) | Обмежено, фокус на адмініструванні | Є (музика, утиліти) |
| Бізнес-модель | Багато функцій у преміум | Частково преміум | Частково преміум |

## 1.4. Аналіз популярних бот-рішень

MEE6.

MEE6 позиціонується як «універсальний» бот для керування сервером. Основні можливості включають:

- автоматичну модерацію (видалення повідомлень з певними словами, покарання за накопичені порушення);
- систему рівнів та XP для стимулювання активності;
- кастомні команди та автоматизацію подій (Automations plugin – тригер,

умова, дія);

- привітання нових користувачів, автопризначення ролей, оголошення, музичні функції.

Суттєвим недоліком є залежність багатьох корисних функцій від преміум-підписки, що обмежує гнучкість використання у безкоштовній версії. Частина користувачів відмічає, що поступово все більше ключових можливостей переходить за paywall.

Carl-bot.

Carl-bot орієнтується на адміністрування серверу з акцентом на:

- реакційні ролі (призначення ролей за емодзі-реакцією, підтримка різних режимів, у т.ч. тимчасових ролей);
- розширене логування (повідомлення, приєднання/вихід учасників, зміни ролей, редагування/видалення тощо);
- Automod та базову модерацію (фільтри, тимчасові покарання);
- кастомні команди, тег-система, нагадування та інші утилітні можливості.

Carl-bot добре масштабується для великих серверів, але його підхід до модерації в основному rule-based; інтеграція сучасних AI/ML-інструментів відсутня або обмежена.

Dyno та інші боти.

Dyno та подібні до нього боти фокусуються на класичній модерації (фільтри, авто-покарання, сповіщення), автоповідомленнях, музичних функціях тощо. У більшості випадків вони також використовують переважно правила, регулярні вирази та прості евристики для фільтрації.

Загальні спільні риси розглянутих рішень:

- широкий набір інструментів для автоматизації керування сервером (ролі, логування, сповіщення);
- rule-based підходи до модерації як основний механізм захисту;
- обмежений або відсутній фокус на трирівневих AI-пайплайнах (rule-based → ML → fallback) із використанням зовнішніх ML/LLM-сервісів;
- перехід значної частини функціоналу на модель передплати, що зменшує

можливість гнучкого налаштування без додаткових витрат.

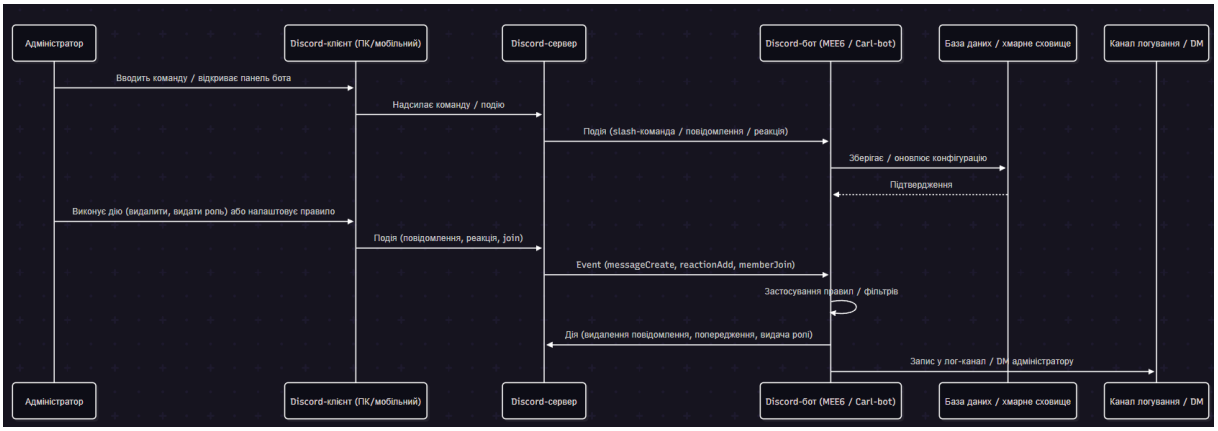


Рис. 1.2. Узагальнена схема взаємодії адміністратора з типовим Discord-ботом (на прикладі MEE6/Carl-bot)

**Висновок до розділу 1**

Проведений аналіз показує, що сучасні Discord-боти вже добре розв’язують задачі:

- базової автоматизації (авторолі, привітання, нагадування);
- rule-based модерації (фільтри за словами, авто-покарання);
- логування подій та керування ролями на великих серверах.

Водночас виявлено низку обмежень, які стають критичними для сучасних спільнот:

- відсутність або поверхнева інтеграція методів машинного навчання та AI-модерації для аналізу контексту повідомлень;
- відсутність чітко сформованої багаторівневої архітектури модерації з fallback-механізмами у разі проблем із ML-сервісами;
- сильна залежність розширених можливостей від преміум-планів;
- недостатня прозорість і гнучкість саме в частині модераційних рішень (комбінація правил, ML-порогів, логування причин).

Ці висновки визначають вимоги до розроблюваного в рамках кваліфікаційної роботи програмного засобу:

- побудова трирівневої системи AI-модерації (rule-based → ML через

HuggingFace → fallback-евристики);

- тісна інтеграція модерації з керуванням ролями, каналами та ACL;
- орієнтація на конфігурованість та прозорість рішень (логування причин, пороги, політики);
- можливість роботи у хмарному середовищі (Replit) з використанням вбудованої бази даних і врахуванням обмежень Discord API.

Саме ці аспекти і становлять основу подальших розділів, де буде описано архітектуру, реалізацію та тестування розроблюваного Discord-бота.

# РОЗДІЛ 2
# МЕТОДИ ТА ТЕХНОЛОГІЇ РОЗРОБКИ ЗАСОБІВ АВТОМАТИЗАЦІЇ НАЛАШТУВАННЯ СЕРВЕРА DISCORD

## 2.1. Rule-based модерація та антиобфускація

Класичний підхід до модерації контенту в онлайн-сервісах базується на правилах (rule-based): списках заборонених слів, регулярних виразах, простих евристиках. У контексті Discord це:

- фільтрація нецензурної лексики, образливих висловлювань та мови ворожнечі;
- виявлення спаму (повторюваний текст, масові згадки, посилання певних типів);
- виявлення потенційно неприйнятного (NSFW) контенту за ключовими словами та доменами.

У реалізованому боті rule-based шар винесено в окремий модуль AIModeration.fuzzy і набір функцій у сервісі модерації:

1. Використовується словник базових термінів.
2. Для кожного терміну будується “розмазаний” регулярний вираз з урахуванням:
   - заміни символів на схожі (латиниця/кирилиця, leetspeak – наприклад, заміна літер на цифри чи спецсимволи);
   - вставок довільних роздільників між літерами (пробіли, знаки пунктуації, підкреслення тощо).
3. Функція на кшталт build_fuzzy_patterns() перетворює базовий список слів у набір регулярних виразів, що поширюються на різні форми написання.

На рівні сервісу модерації цей rule-based шар використовується як:

- Швидкий передфільтр: якщо повідомлення явно містить спам або однозначно неприйнятний вміст (наприклад, посилання на заборонені ресурси), рішення може бути прийнято без звернення до ML/LLM.

- Фінальний шар узагальнених правил: після отримання результату ML/LLM, додаткові патерни й евристики уточнюють категорію.

Переваги rule-based підходу: дуже мала затримка, повна передбачуваність, відсутність залежності від зовнішніх сервісів.

Недоліки: обмежена здатність враховувати контекст (іронія, сарказм), необхідність постійного ручного оновлення словників, висока адаптивність користувачів.

У розробленому Discord-боті цей рівень реалізований як fuzzy-механізм, що суттєво підвищує його ефективність.

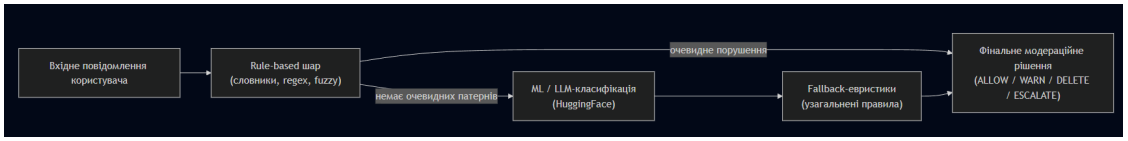


Рис. 2.1. Схематичне місце rule-based шару в трирівневому пайплайні AI-модерації

### 2.2. ML-класифікація токсичного контенту

ML-класифікація є другим класом підходів до визначення небажаного контенту. Типова постановка задачі: на вхід подається повідомлення (*text*), а на вихід – ймовірність для набору визначених класів (наприклад, TOXIC, INSULT, PROFANITY, OK тощо).

У реалізованому боті використано два доповнюючі ML-механізми, які забезпечують гнучкість та стійкість системи:

1. Опційний локальний backend HuggingFace (ml_backend.py)

- Підтримує токсикність-модель на основі cointegrated/rubert-tiny-toxicity.
- Функція score_toxicity(text, cfg) повертає словник (асоціативний масив), де ключем є назва класу (label), а значенням – оцінка ймовірності (score).
- Стійкість: Якщо необхідні бібліотеки transformers / torch недоступні, backend акуратно деградує до повернення порожнього результату, не спричиняючи збою всієї системи.

2. Віддалений ML/LLM-backend через HuggingFace Inference API (ml.py)

- Функція classify(text, guild_id=...) повертає стандартизовану структуру відповіді у форматі JSON.
- Структура відповіді (наприклад, {"label": "TOXIC", "score": 0.95, "reason": "..."}):
  - label: рядок, що містить основну класифікацію (наприклад, "TOXIC", "SAFE", "SPAM", "NSFW" тощо).
  - score: число з рухомою комою (float), що відображає впевненість моделі.
  - reason: рядок з додатковими поясненнями або деталями.
- Захист від нестабільності зовнішнього сервісу:
  - Реалізовані конфігуровані таймаути, ліміти спроб і експоненціальний backoff у випадку черги або помилок HF.
  - Використовується окремий крок _normalize_result для коректного парсингу JSON-відповіді, навіть якщо модель додала зайвий пояснювальний текст або обірвала відповідь.

Переваги ML-класифікації

- Контекст: Здатність враховувати внутрішній контекст повідомлення, розуміючи тон та наміри.
- Підтримка: Менша потреба в ручному підтриманні та оновленні словників.
- Диференціація: Можливість розрізняти різні типи контенту (наприклад, NSFW від SPAM).

Недоліки ML-класифікації

- Якість даних: Залежність від якості та повноти даних навчання моделі.
- Мовні обмеження: Обмеження по підтримуваних мовах (особливо для невеликих мов).
- Продуктивність: Чутливість до latency (затримки) та rate-limits (обмежень на кількість запитів) віддалених сервісів.

### 2.3. LLM-підходи та контекстна модерація

LLM-підходи (Large Language Models) є окремим підкласом ML-рішень. Вони використовують універсальну мовну модель для аналізу, що здатна враховувати контекст та повертати уніфіковану відповідь у форматі JSON.

У боті цей підхід реалізовано через такі ключові механізми:

Компоненти реалізації

- Системна інструкція (Prompt):
  - Чітко прописано роль моделі («ти – модераційний асистент»).
  - Визначено перелік допустимих міток (SAFE, PROFANITY, TOXIC, SPAM, NSFW, MISINFO, ESCALATE).
  - Встановлено жорстка вимога повернути чистий JSON-об’єкт.
- Функції обробки контексту:
  - Функції classify_dialog() та classify_text_contextual() збирають контекст історії (останні кілька повідомлень у каналі, обмежених параметром history_limit).
  - Вони формують діалог для LLM (роль, вміст, останнє повідомлення як ціль для класифікації).
  - Здійснюється виклик HF-endpoint та обробка результату.

Переваги LLM-аналізу

Використання LLM-підходу надає важливі переваги у модерації, пов’язані з глибоким розумінням мови та контексту:

- Контекстуалізація: Дозволяє враховувати контекст передісторії (сварка, відповідь на провокацію, сарказм).
- Виявлення прихованих образ: Дає можливість виявляти кейси, де одиночне повідомлення виглядає «невинно», але в контексті є явною цільовою образою.
- Ескалація: Дозволяє виносити особливо складні, нетипові випадки в категорію ESCALATE для ручного перегляду модератором.

Управління ризиками

Ризики, пов'язані з нестабільністю, вартістю та затримкою (latency) віддалених LLM-сервісів, пом'якшуються за допомогою:

- Багатокрокового парсингу відповіді.
- Жорсткого таймауту.
- Fallback-логіки (перехід на простіші механізми у разі збою).


Рис. 2.2. Узагальнена схема контекстної LLM-модерації діалогу у Discord-боті

### 2.4. Комбінований трирівневий пайплайн модерації в InfinityBot

У розробленому програмному засобі використовується єдиний трирівневий пайплайн інтелектуальної модерації:

Таблиця 2.1

Порівняння rule-based, ML-та LLM-шарів у трирівневій модерації Discord-бота

| Слой | Механізм | Мета | Сценарій дії |
|---|---|---|---|
| **1. Швидкі правила** | Rule-based (fuzzy) | Миттєве фільтрування очевидного | Явний SPAM, жорсткий NSFW, порно-URL. Рішення приймається без LLM. |
| **2. ML/LLM-класи фікація** | LLM-підхід (контекст) | Глибокий контекстний аналіз | Для не-очевидних випадків. Використовує функцію classify_dialog() з історією повідомлень. |
| **3. Fallback та уточнення** | ML + Rule-based | Стійкість та постпроцесинг | Якщо LLM недоступний або не впевнений, відбувається деградація до classify_text() (локальний ML). Застосовуються постпроцесори (наприклад, _soft_nsfw_postprocess()) для додаткового уточнення. |

На виході пайплайна формується єдине рішення: Дія (ALLOW, WARN, DELETE), Категорія (SAFE, TOXIC, SPAM...) та Пояснення від ML/LLM). Це рішення далі передається в модуль керування ролями/каналами для застосування санкцій.

Ключова відмінність InfinityBot – це структурований багаторівневий пайплайн, який забезпечує гнучкість, стійкість до відмови зовнішніх сервісів і ефективність модерації.

**Висновок до розділу 2**

Проведений аналіз показав, що існує три основні підходи до модерації текстового контенту:

1. Rule-based фільтрація (словники, регулярні вирази, евристики).
2. ML-класифікація (спеціалізовані моделі токсичності).
3. LLM-підходи (універсальні мовні моделі з контекстною оцінкою).

Класичні rule-based рішення є швидкими, передбачуваними і незалежними від зовнішніх сервісів, але погано працюють із контекстом та легко обходяться за рахунок обфускації. ML-моделі підвищують якість і стабільність класифікації, проте залежать від якості датасетів, мови й ресурсів. LLM-підходи дають найкраще розуміння контексту (діалог, сарказм, непрямі образи), але є дорогими, більш повільними та вимогливими до захисту приватності.

У розробленому Discord-боті ці підходи не протиставляються, а комбінуються в єдиний трирівневий пайплайн :

- Швидкий rule-based шар для очевидних порушень і базового захисту;
- ML/LLM-класифікація через HuggingFace для складних випадків і контекстної оцінки;
- Узагальнені правила та fuzzy-фільтрація, що доуточнюють рішення і забезпечують fallback при відмові зовнішніх сервісів.

Такий комбінований підхід дозволяє одночасно виконати вимоги до швидкодії, надійності й якості модерації, а також забезпечити стійкість системи до помилок і деградації зовнішніх AI-сервісів. Це формує методологічну основу для реалізації інтелектуальної модерації в подальших розділах роботи.

# РОЗДІЛ 3
# АРХІТЕКТУРА ТА РЕАЛІЗАЦІЯ ПРОГРАМНОГО ЗАСОБУ DISCORD-БОТА.

## 3.1. Вибір технологій та середовища розробки

Розробка програмного засобу InfinityBot здійснювалася з урахуванням ключових вимог: постійна робота 24/7, низька затримка, асинхронна обробка подій та інтеграція з HuggingFace.

Для задоволення цих вимог були обрані такі основні технології:

Таблиця 3.1

Основні технології та засоби, використані у проєкті InfinityBot

| **Технологія** | **Вибір** | **Призначення** |
|---|---|---|
| **Мова програмування** | **Python 3.x** | Розвинута екосистема, підтримка асинхронності, наявність бібліотек для Discord та ML. |
| **Бібліотека для Discord** | **discord.py** | Високорівневий асинхронний доступ до Discord API, обробка подій (on_message), реалізація команд та фонових задач (tasks). |
| **Сховище даних** | **Replit DB** | Вбудоване **key–value сховище** в середовищі Replit; для конфігурацій, станів ролей та службових даних без окремої СУБД. |
| **Розгортання** | **Replit** | Хмарна платформа з платним розміщенням для забезпечення **постійно активного процесу** (24/7) та доступу до Replit DB. |
| **AI/ML Інтеграція** | **HTTP-клієнт** | Взаємодія з **HuggingFace Inference API** та (опційно) локальні моделі через transformers/torch. |
| **Журналювання** | **Python logging** | Розділення рівнів логів, вивід у консоль та надсилання важливих подій у **лог-канали Discord**. |

## 3.2. Загальна архітектура програмного засобу

Архітектура InfinityBot побудована як модульна система, де ядро бота відповідає за підключення до Discord API, а предметно-орієнтовані модулі реалізують

конкретні підсистеми: AI-модерація, керування ролями, канали, конфігурація та безпека.

На концептуальному рівні архітектура містить такі шари:

1. Зовнішній рівень (Discord): Користувачі взаємодіють із сервером. Усі події (повідомлення, команди) надходять через Discord API.
2. Ядро InfinityBot: Відповідає за підключення, запуск event-loop, реєстрацію команд, обробників подій та фонових задач.
3. Модулі предметної логіки:
   - Modules/AIModeration – трирівнева AI-модерація.
   - Modules/Autoroles – автоматична видача/зняття ролей, таймери.
   - Modules/Channels – політики каналів, видалення/обмеження повідомлень.
   - Modules/Config – глобальні та локальні параметри, збереження в Replit DB.
   - Modules/Security – система контролю доступу (ACL).
4. Рівень даних та зовнішніх сервісів: Replit DB (сховище конфігурацій) та HuggingFace Inference API (AI-бекенд).

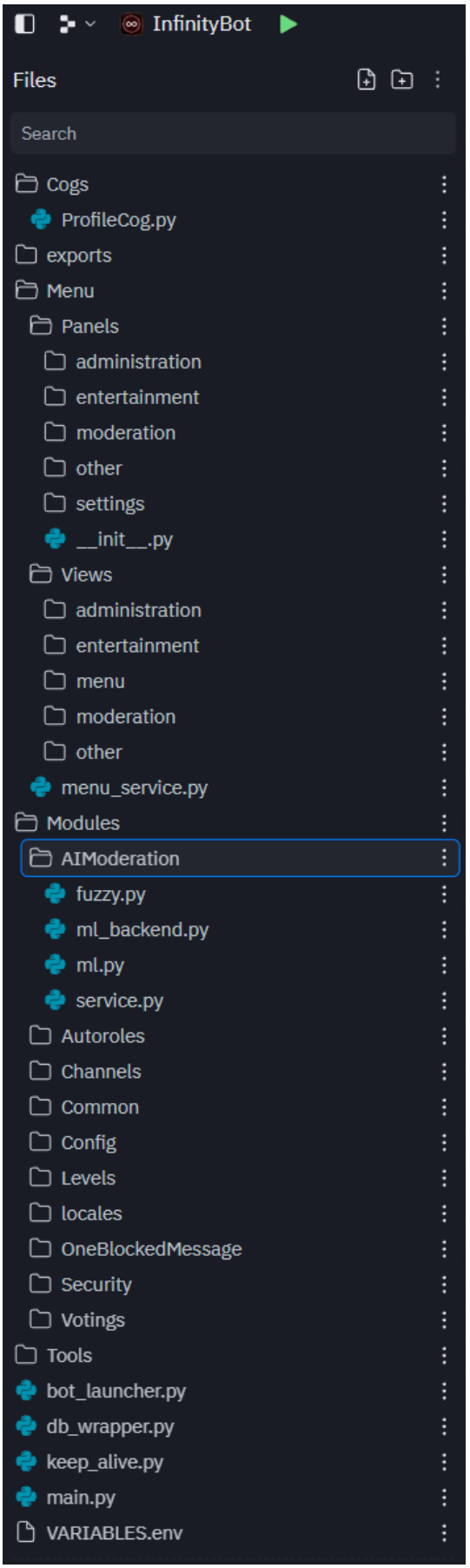


Рис. 3.1. Загальна архітектура Discord-бота InfinityBot

### 3.3. Компонентна структура та основні модулі InfinityBot

#### 3.3.1. Модуль AIModeration

Модуль AIModeration реалізує трирівневу архітектуру для автоматизованої модерації контенту. Ця багаторівнева структура забезпечує як швидке реагування на очевидні порушення, так і глибокий контекстний аналіз для складних випадків, одночасно гарантуючи стійкість до зовнішніх збоїв.

- Rule-based шар (fuzzy):
  - Виконує антиобфускацію (врахування заміни символів, шуму, латиниці/кирилиці).
  - Здійснює швидку перевірку очевидних порушень (SPAM, NSFW) без звернення до AI, забезпечуючи мінімальну затримку.
- ML/LLM-шар (ml, ml_backend):
  - Реалізує класифікацію повідомлень за допомогою HuggingFace Inference API.
  - Використовує мітки (SAFE, TOXIC, SPAM, NSFW тощо).
  - Підтримує контекстну класифікацію діалогів для виявлення прихованого змісту.
- Fallback-логіка:
  - Забезпечує стійкість шляхом обробки таймаутів, HTTP-помилок та перевищення лімітів.
  - У випадку збою основного шару відбувається деградація до простішої ML/rule-based моделі.

Сервіс інтеграції модуля AIModeration запускає модераційний пайплайн, формує фінальне рішення на основі результатів усіх шарів та викликає відповідні дії (видалення повідомлення, попередження користувача, логування події).

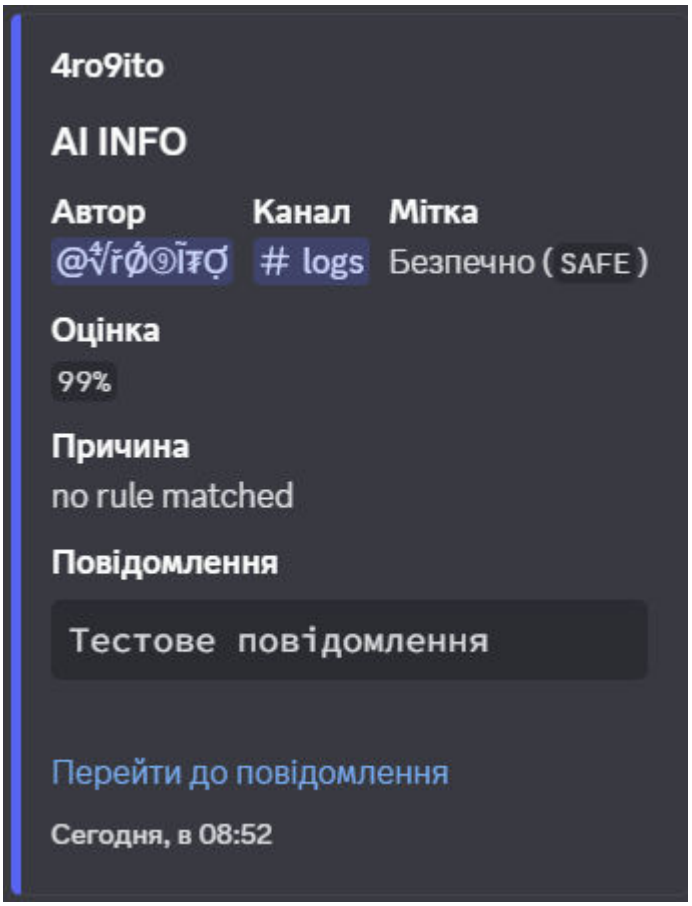


Рис. 3.2. Приклад лог-повідомлення про спрацювання AI-модерації в службовому каналі Discord

### 3.3.2. Модуль Autoroles

Відповідає за автоматизоване керування ролями:

- Reaction Roles (видача ролей за реакціями), командами або умовами.
- Тайм-лімітовані ролі (автоматичне зняття ролі після певного часу) та система «імунітетів».
- Фонові задачі для періодичної перевірки строку дії ролей.

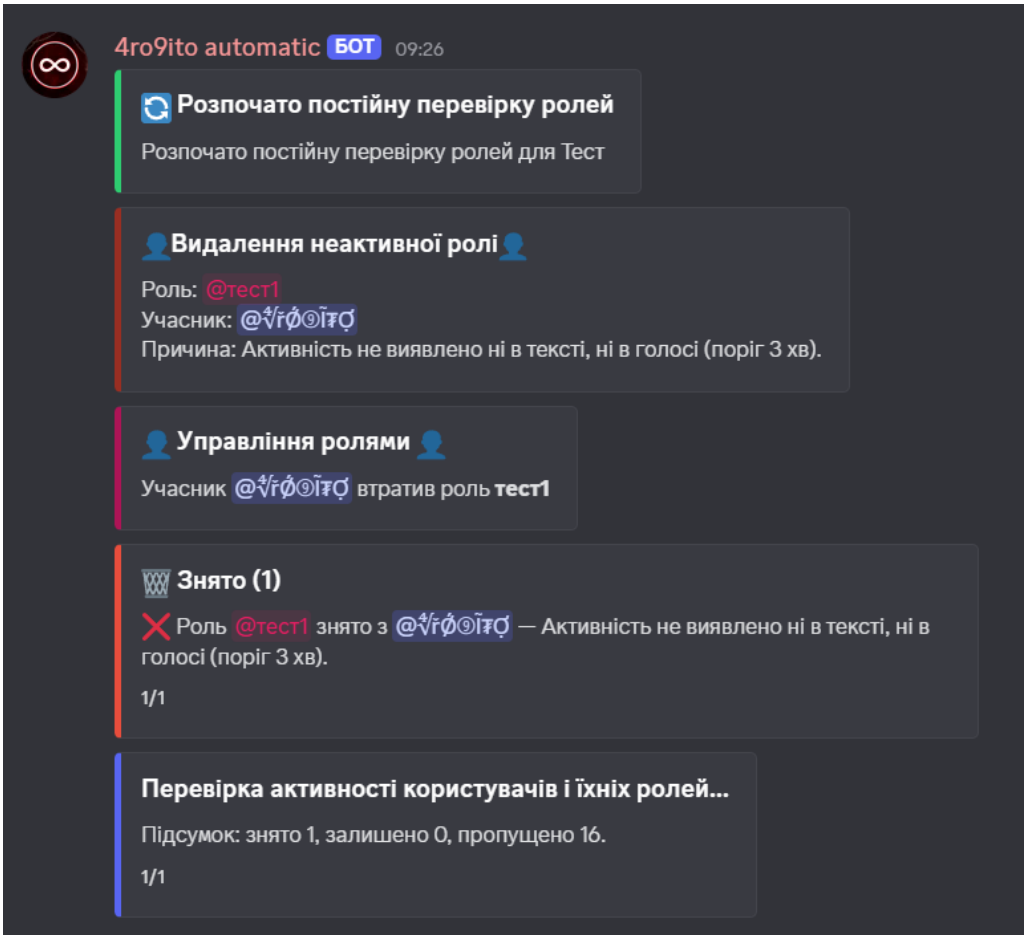


Рис. 3.3. Приклад повідомлення Discord-бота з панеллю авторолей

### 3.3.3. Модуль Channels

Реалізує політики та обмеження на рівні каналів і є «виконавцем рішень» AI-модерації:

- Режим обмежених каналів (лише медіа, лише команди).
- Допоміжні функції delete_message_and_notify: акуратне видалення повідомлення, надсилання користувачу пояснення (DM або у відповідь) та логування події.

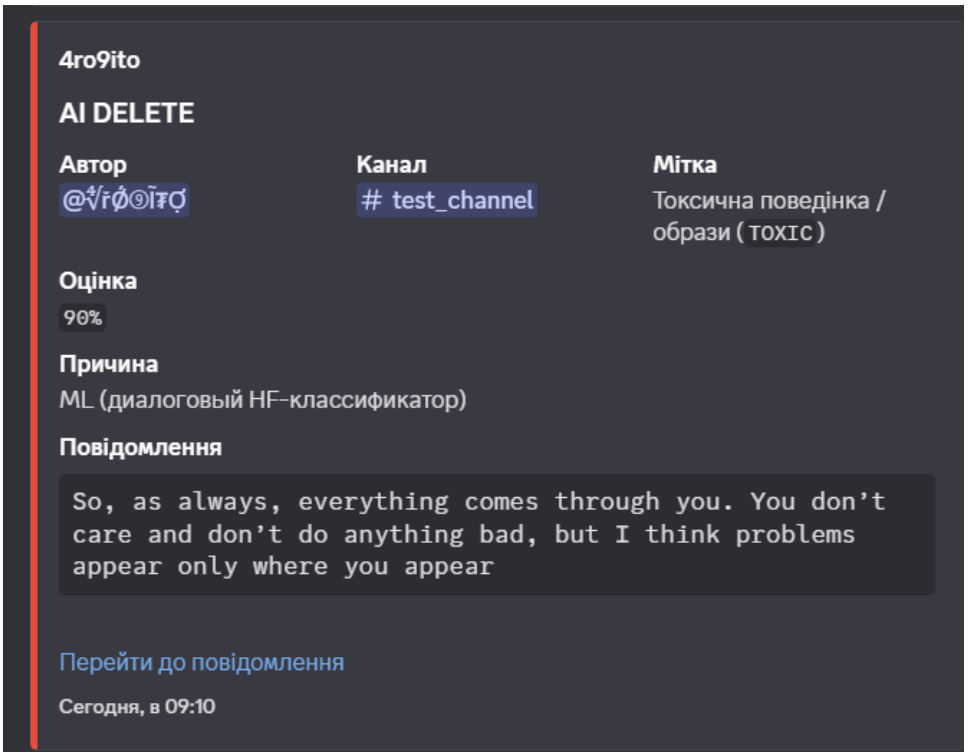


Рис. 3.4. Приклад сповіщення користувача в каналі

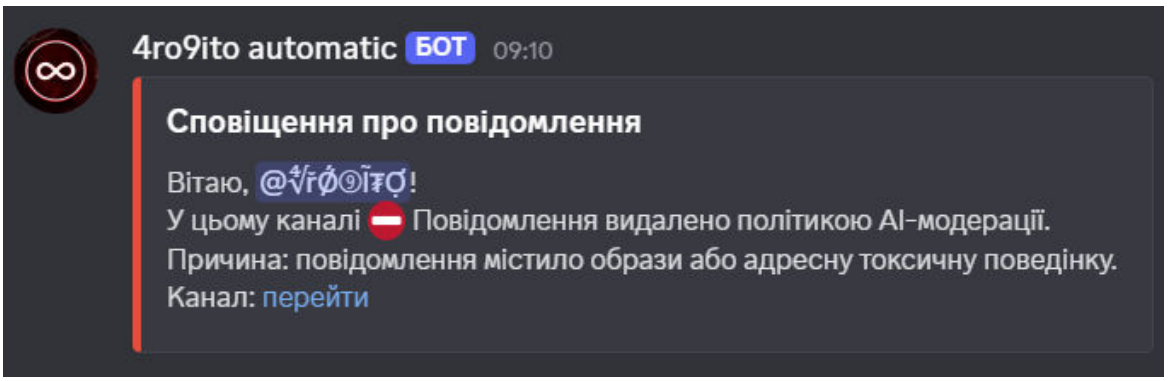


Рис. 3.5. Приклад автоматичного видалення повідомлення та сповіщення користувача в каналі

### 3.3.4. Модуль Config

Модуль Config відповідає за зберігання та доступ до конфігурацій у Replit DB і є «точкою входу» до всіх налаштувань бота.

Основні функції:

- Глобальні параметри – зберігання базових налаштувань, спільних для всіх серверів:

власник бота, службові/технічні канали, мова за замовчуванням, режим логування тощо.

- Конфігурації на рівні окремих серверів (guild-specific) – налаштування AI-модерації (режим strict/soft, пороги для категорій, дозволені дії), прив'язка лог-каналів, службових ролей, параметри роботи модулів Autoroles та Channels.
- Структура ключів у Replit DB побудована за узгодженим префіксним підходом:
  - config:global – глобальні параметри;
  - config:guild:{guild_id} – конфігурація конкретного Discord-сервера;
  - додаткові службові ключі (state:*, кеші) для технічних станів.

Таблиця 3.2

Основні групи конфігураційних параметрів InfinityBot

| Група параметрів | Приклади ключів / налаштувань | Короткий опис |
|---|---|---|
| Глобальні параметри | config:global.owner_id, config:global.locale, config:global.log_level | Загальні налаштування бота: власник, мова за замовчуванням, рівень логування, технічні режими. |
| Параметри гільдії (серверу) | config:guild:{id}.log_channel, config:guild:{id}.staff_role | Прив'язка службових каналів (логів, сповіщень), основних ролей персоналу, локальних опцій сервера. |
| Параметри AI-модерації | config:guild:{id}.ai.enabled, ai.mode, ai.thresholds, ai.nsfw_mode | Увімкнення/вимкнення AI, режим (strict/soft), пороги для категорій, політика обробки NSFW, ліміти. |
| Параметри авторолей | config:guild:{id}.autoroles.messages, autoroles.ttl, autoroles.immunity | Карта повідомлень з панелями авторолей, строки дії тимчасових ролей, параметри «імунітетів». |
| Параметри каналів | config:guild:{id}.channels.restricted, channels.modes, | Список «особливих» каналів, режими (тільки медіа, тільки команди), канали для |

| | | |
|---|---|---|
| | channels.log_channel | логів дій у каналах. |
| Параметри безпеки / ACL | config:guild:{id}.acl.roles, acl.levels, acl.overrides | Відповідність ролей рівням доступу (OWNER/ADMIN/MOD/USER), винятки та спеціальні дозволи/заборони. |
| Технічні / службові стани | state:guild:{id}.tasks, state:guild:{id}.last_checks | Стан фонових задач, час останніх перевірок, службові маркери для внутрішньої логіки бота. |

### 3.3.5. Модуль Security / ACL

Модуль Security / ACL (Access Control List) реалізує систему контролю доступу всередині бота. Це забезпечує безпеку та розмежування прав при виконанні адміністративних та модераційних функцій.

- Реалізація системи контролю доступу (ACL): Модуль визначає ієрархічні рівні доступу користувачів, які використовуються для внутрішньої логіки перевірки. До цих рівнів належать: OWNER (Власник), ADMIN (Адміністратор), MOD (Модератор) та USER (Звичайний користувач).
- Визначення рівня доступу: Модуль надає утиліту resolve_access_level. Ця функція використовується для визначення ефективного рівня доступу користувача на основі його ролей на сервері та поточної конфігурації.
- Призначення: Функції цього модуля використовуються для перевірки прав при виконанні критичних команд (таких як налаштування модерації, зміна політик безпеки або виконання адміністративних дій).

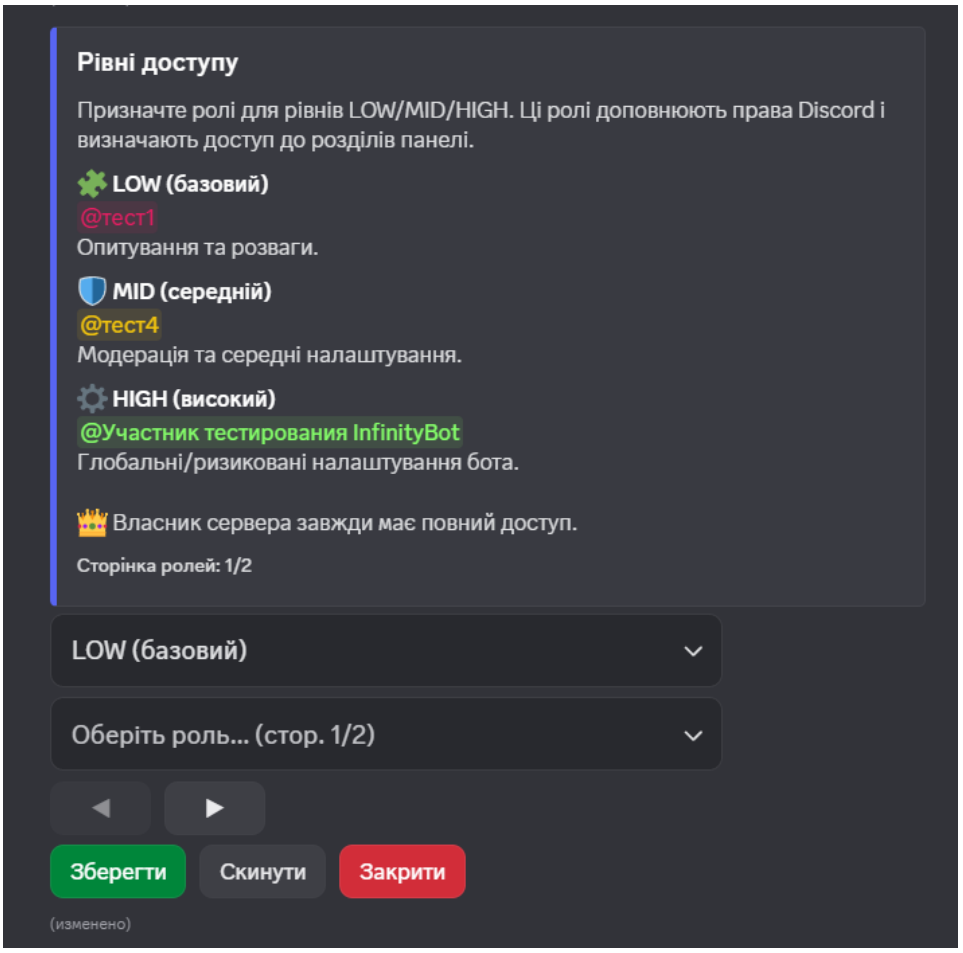


Рис. 3.6. Приклад панелі налаштування прав доступу (ACL) у Discord-боті InfinityBot

### 3.4. Сховище даних та інтеграція з Replit DB

Replit DB використовується як легковажне key–value сховище завдяки інтеграції з середовищем виконання та простому словникоподібному API replit.db:

| **Категорія даних** | **Приклади** |
|---|---|
| **Конфігурації** | Налаштування AI, ACL, пороги модерації. |
| **Стани ролей** | Дата закінчення строку дії тимчасової ролі, «імунітети». |
| **Службові кеші** | Останній час перевірки, технічні маркери для фонового планування. |

Табл. 3.3 – Основні категорії даних, що зберігаються у Replit DB в межах InfinityBot

**Безпека:** Токен бота та ключі HuggingFace розміщуються у **змінних оточення Replit**, поза відкритим репозиторієм.

```
config:global = {
  "owner_id": 987654321098765432,
  "locale": "uk-UA",
  "log_level": "INFO"
}

config:guild:123456789012345678 = {
  "ai": {
    "enabled": true,
    "mode": "soft",
    "nsfw_mode": "soft",
    "thresholds": {
      "TOXIC": 0.75,
      "SPAM": 0.80
    }
  },
  "log_channel_id": 111222333444555666,
  "staff_role_id": 778888999000111222
}

state:guild:123456789012345678:autoroles = {
  "message_id": 555666777888999000,
  "ttl_roles": {
    "role_id": 86400    # час життя ролі в секундах
  }
}
```

Рис. 3.7. Схематичний приклад структури ключів та значень у Replit DB для одного Discord-сервера

### 3.5. Взаємодія з Discord API та подійно-орієнтована модель

InfinityBot використовує подійно-орієнтовану модель бібліотеки discord.py.

- Обробники подій (on_message, on_member_join тощо) є основними точками входу в логіку модулів бота.
- Для кожного текстового повідомлення:
  - Викликається пайплайн AI-модерації (rule-based до ML/LLM до fallback).
  - На основі фінального рішення запускаються дії модулів Channels / Autoroles.

- Результати роботи системи логуються.

- Управління: Slash або префіксні команди використовуються для керування конфігурацією бота.
- Фонові задачі (tasks.loop) застосовуються для:
- Періодичної перевірки строків дії тимчасових ролей.
- Обслуговування кешів.
- Health-check зовнішніх AI-сервісів.

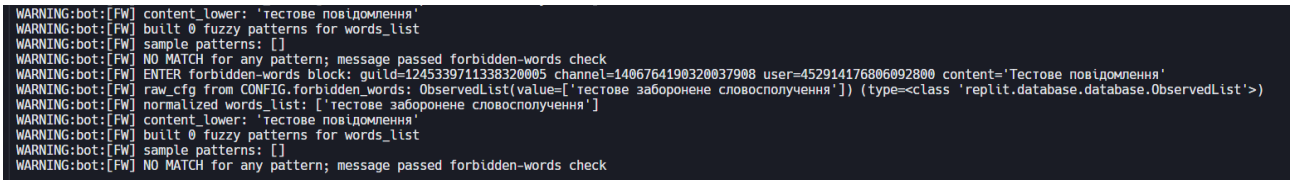


Рис. 3.8. Протокол обробки текстового повідомлення від Discord до модераційного рішення в InfinityBot

**Висновок до розділу 3**

У цьому розділі сформовано цілісне уявлення про технологічну базу та архітектуру програмного засобу InfinityBot.

Обґрунтування вибору стеку технологій

- Вибір технологій: Обрано Python та бібліотеку discord.py для подійно-орієнтованої роботи з Discord API.
- Інфраструктура: Використано хмарне середовище Replit з Replit DB для безперервного розгортання без необхідності розгортання окремої СУБД (системи управління базами даних).
- AI-бекенд: Використання HuggingFace як AI-бекенду забезпечує можливість інтеграції сучасних моделей ML/LLM модерації.
- Перевага: Такий набір дає змогу поєднати постійну доступність, простоту супроводу та інтеграцію передових AI-рішень.

Визначення модульної архітектури

- Модульність: Архітектура InfinityBot є модульною. Ядро бота лише приймає події та маршрутизує їх до предметно-орієнтованих модулів.
- Ключові модулі: AIModeration, Autoroles, Channels, Config, Security/ACL.
- Поділ відповідальності: Модулі забезпечують чіткий поділ відповідальності між AI-модерацією, автоматизацією ролей і каналів, конфігурацією та контролем доступу. Це спрощує розширення функціональності та зменшує ризики взаємних побічних ефектів.

Схема взаємодії з даними та API

- Сховище даних: Схема зберігання даних у Replit DB реалізована як key–value сховище для конфігурацій і службових станів.
- Логіка роботи: Побудована на подійно-орієнтованій моделі (on_message, on_reaction) та фонових задачах (tasks.loop).
- Результат: Це дає можливість реалізувати складні сценарії (тайм-лімітовані ролі, політики каналів, AI-модерація) без надмірної інфраструктурної складності.

# РОЗДІЛ 4
# РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАСОБУ ІНТЕЛЕКТУАЛЬНОЇ МОДЕРАЦІЇ Й АВТОМАТИЗАЦІЇ КЕРУВАННЯ DISCORD-СЕРВЕРОМ

## 4.1. Реалізація трирівневої AI-модерації

Обробка текстових повідомлень у InfinityBot побудована за подійною моделлю: кожне нове повідомлення в каналі Discord запускає послідовність викликів у модулі AIModeration та Channels.

Ключові кроки:

1. Перехоплення події повідомлення

Обробник on_message у ядрі бота викликає сервіс AI-модерації (умовно AIModeration.service.handle_message(message)), якщо повідомлення надійшло не від самого бота і канал/користувач не виключені з модерації.

2. Rule-based попередня перевірка

- Повідомлення нормалізується (lowercase, усунення зайвого форматування).
- Застосовується швидкий rule-based фільтр: fuzzy-словники, регулярні вирази для лайки, спаму, NSFW-URL.
  - Якщо знайдено очевидне порушення (наприклад, прямий порно-URL, масовий спам), формується рішення без звернення до AI:

дія: DELETE або WARN;

  - категорія: SPAM / NSFW / TOXIC;

3. коротке пояснення.

Виклик ML/LLM-шару

Якщо rule-based шар не знайшов явного порушення, активується ML/LLM-пайплайн:

- для базової перевірки – classify_text(...) (HuggingFace-модель токсичності / інший backend);

- для контекстної перевірки – classify_text_contextual(...), яка додатково збирає кілька попередніх повідомлень у каналі й викликає LLM через classify_dialog(...).

Повертається структура виду:

label, score, reason.

4. Постобробка та політики
   - На основі label і score застосовуються пороги з конфігурації сервера (наприклад, TOXIC >= 0.75 → видалити, 0.5–0.75 → попередження).
   - Для режиму nsfw_mode = soft застосовується додаткова логіка: неявний NSFW переводиться у PROFANITY, якщо немає явних порно-URL.
   - Формується кінцеве рішення: ALLOW / WARN / DELETE / ESCALATE.
5. Виконання дій та логування
   - Модуль Channels через функцію типу delete_message_and_notify(...) видаляє повідомлення, надсилає користувачу пояснення та записує подію у лог-канал.
   - Для ESCALATE (сумнівні випадки) – бот лише логує подію з позначкою, що потрібен ручний перегляд модератора.

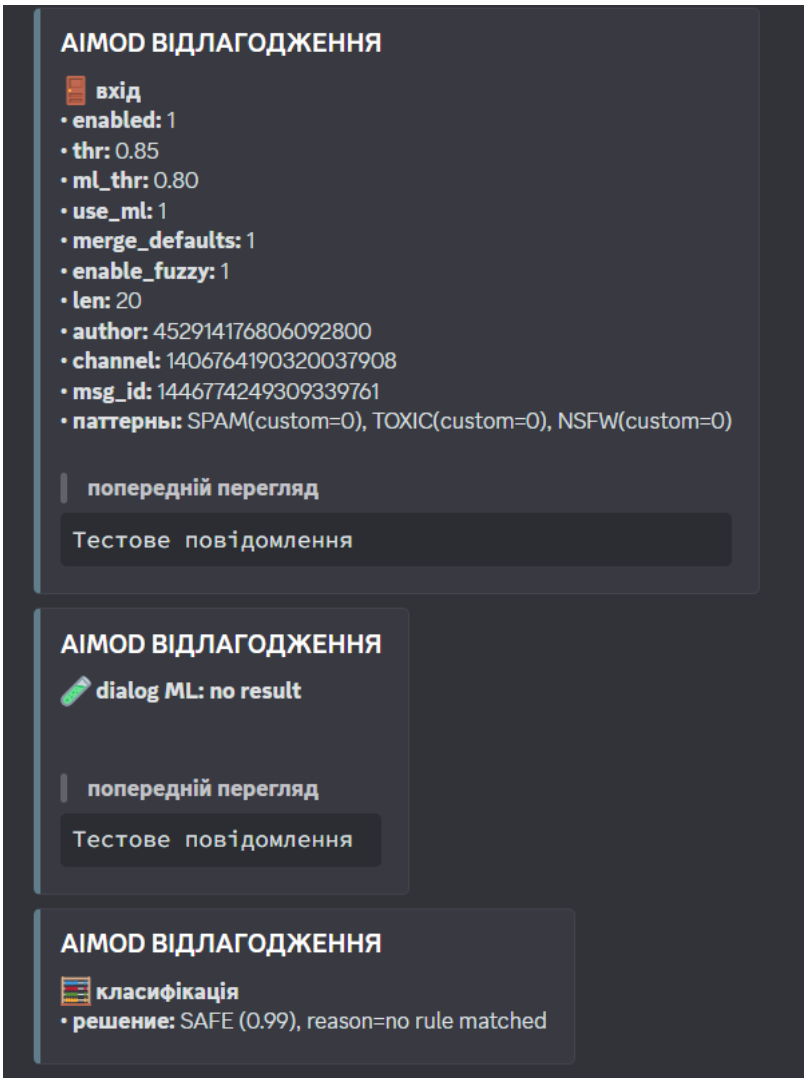


Рис. 4.1. Узагальнена послідовність обробки текстового повідомлення в InfinityBot (від події Discord до модераційного рішення)

### 4.2. Реалізація автоматизації керування ролями (Autoroles)

Модуль Autoroles реалізує сценарії автоматичної видачі та зняття ролей:

- Панелі авторолей. Бот публікує в заданому каналі повідомлення з елементами керування (реакції/кнопки/меню). Користувач, взаємодіючи з панеллю, отримує або втрачає певну роль. Конфігурація панелі зберігається в Replit DB: ID повідомлення, відображені ролі, режим (одноразово, toggle, множинний вибір).
- Тайм-лімітовані ролі. При видачі ролі може фіксуватися час закінчення її дії (TTL). Фонова задача (loop у discord.ext.tasks) періодично перевіряє список активних тимчасових ролей і знімає їх по закінченню строку.

- Імунітети. Для окремих ролей/користувачів можливе встановлення “імунітету” – тимчасової заборони на автоматичне зняття ролі. Це дозволяє захистити ключові ролі від помилкових дій автоматизації.

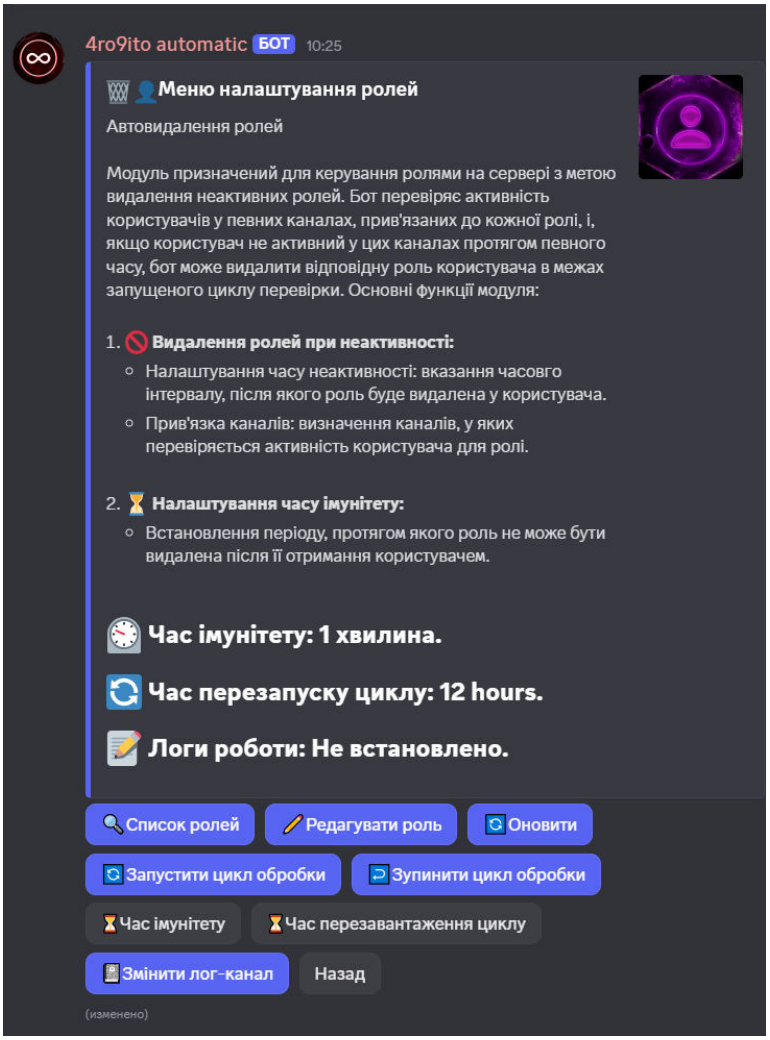


Рис. 4.2. Приклад панелі авторолей у Discord-каналі з видимими кнопками/реакціями для вибору ролей

### 4.3. Реалізація політик каналів (Channels)

Модуль Channels відповідає за застосування політик та обмежень на рівні окремих каналів і виступає “виконавчим шаром” для рішень модерації.

Основні можливості модуля:

- Керовані параметри каналів. Для кожного каналу в конфігурації сервера можуть вмикатися/вимикатися окремі параметри, які визначають дозволену поведінку

в цьому каналі: – обмеження публікацій для користувачів без певних ролей; – заборона звичайних повідомлень у службових каналах; – примусове застосування AI-модерації для “чутливих” каналів (звітність, сапорт тощо); – інші локальні правила, що впливають на те, чи допускається повідомлення, чи воно має бути видалене або залоговане.

- Єдиний інтерфейс дій. Інші модулі (AI-модерація, Autoroles, конфігурація) не працюють із каналами напряму, а користуються сервісними функціями Channels, зокрема: – умовно delete_message_and_notify(...) – видалення повідомлення з поясненням причини користувачу; – логування подій у службові канали;
- Інтеграція з AI-модерацією.

Коли пайплайн AI-модерації повертає рішення WARN або DELETE, саме Channels реалізує фактичну дію в каналі: видаляє або залишає повідомлення, надсилає попередження, записує подію в лог.

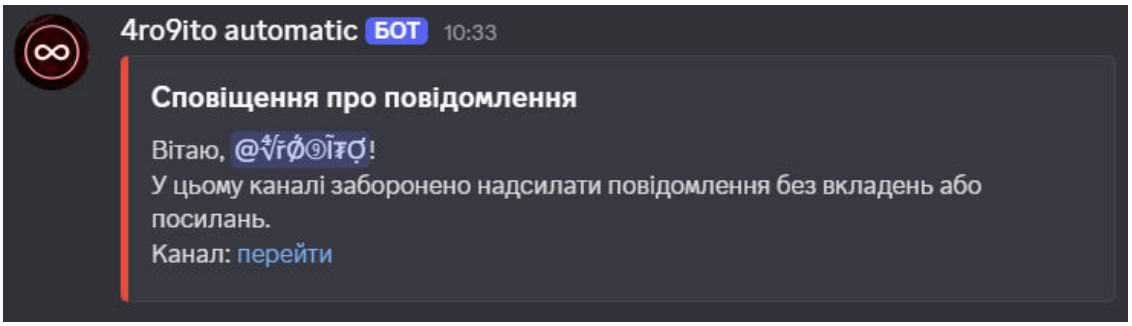


Рис. 4.3. Приклад автоматичного видалення повідомлення в каналі з поясненням причини від Discord-бота

### 4.4. Інтеграція, розгортання та конфігурація у Replit

Розгортання InfinityBot виконується у хмарному середовищі Replit:

- Файл запуску містить ініціалізацію клієнта discord.py, підключення до Replit DB, реєстрацію модулів (AIModeration, Autoroles, Channels, Config, Security) і запуск event-loop.
- Секрети (токени) зберігаються у змінних оточення Replit (Discord Bot Token, HuggingFace API key), а не в коді.

- Replit DB використовується як постійне сховище конфігурацій і станів (як описано в розділі 3).

Конфігурація для конкретного сервера здійснюється через команди або панелі налаштувань:

- команда /панель для AI-модерації (увімкнути/вимкнути, режим strict/soft, пороги);
- панель для ACL (призначення ролей OWNER/ADMIN/MOD/USER);
- панель для каналів (режими каналів, лог-канали) і авторолей.

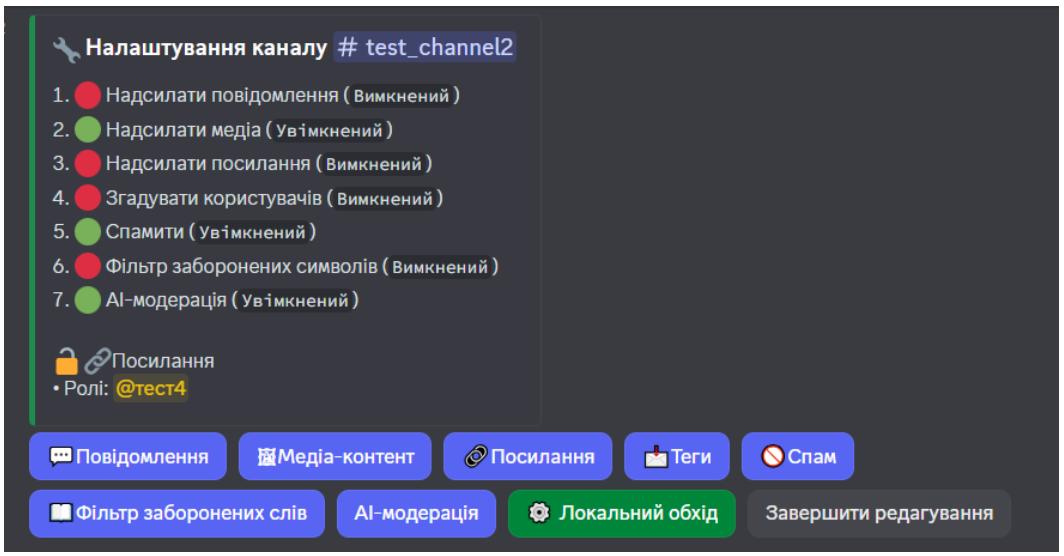


Рис. 4.4. Приклад службового каналу конфігурації, де відображені панелі налаштування AI, ACL та політик каналів

### 4.5. Методика тестування AI-модерації та автоматизації

Тестування проводилося у два етапи:

1. Функціональне тестування сценаріїв
   - ручні тести команд налаштування (AI, ACL, Autoroles, Channels);
   - перевірка правильності реакцій на основні події:
     - надсилання різних типів повідомлень (нейтральні, лайливі, спам, NSFW, “обфусцьовані” образи);
     - взаємодія з панелями авторолей;
     - робота режимів каналів.

2. Експериментальне оцінювання якості AI-модерації

- формування тестової вибірки повідомлень у кількох категоріях (нейтральні, лайка, токсичні образи, спам, NSFW);
- ручна розмітка цієї вибірки (“еталонні” мітки);
- прогін вибірки через пайплайн модерації та порівняння результатів із еталоном;
- обчислення базових метрик:
  - precision, recall для токсичного контенту;
  - кількість хибно-позитивних (FP) та хибно-негативних (FN) спрацьовувань;
  - частка повідомлень, що потребують ESCALATE.

Таблиця 4.1

Приклади тестових сценаріїв для AI-модерації та автоматизації ролей/каналів

| № | Сценарій | Вхідні умови | Дія користувача / подія | Очікуваний результат |
|---|---|---|---|---|
| 1 | Нейтральне повідомлення | AI увімкнено, стандартний режим (soft/strict) | Користувач пише звичайне повідомлення без лайки | Повідомлення не видаляється, попередження не надсилається, записів у лог лише мінімальний/відсутній. |
| 2 | Явна лайка без обфускації | AI увімкнено, базові rule-based патерни налаштовані | Користувач пише явну образу з нецензурною лексикою | Rule-based шар спрацьовує, повідомлення видаляється, користувач отримує попередження, є запис у логу. |
| 3 | Обфускована лайка | AI увімкнено, fuzzy-патерни активні | Користувач пише лайку з заміною символів і пробілами | Fuzzy-фільтр або ML-шар визначає токсичність, повідомлення видаляється/позначається як TOXIC. |

| | | | | |
|---|---|---|---|---|
| 4 | Потенційно токсичне, але контекстно нейтральне | Контекстна LLM-модерація увімкнена | Репліка, що виглядає різко, але в контексті – жарт | LLM повертає SAFE або низький TOXIC score, повідомлення залишається, лише лог (без санкцій). |
| 5 | Явний спам (повторювані повідомлення/посилання) | AI увімкнено, rule-based антиспам налаштований | Користувач надсилає багато однакових повідомлень | Повідомлення видаляються, користувач може отримати попередження/тимчасове обмеження, подія логуються. |
| 6 | NSFW-контент у “м’якому” режимі | nsfw_mode = soft | Користувач пише натяк/напів-NSFW фразу | Категорія знижується до PROFANITY, м’яка реакція (попередження або лише лог залежно від порогів). |
| 7 | NSFW-контент з явним порно-URL | nsfw_mode = soft/strict | Користувач надсилає повідомлення з порно-посиланням | Rule-based URL-фільтр спрацьовує, повідомлення видаляється, жорстке попередження, запис у лог. |
| 8 | Взаємодія з панеллю авторолей | Налаштована панель авторолей у певному каналі | Користувач натискає кнопку / ставить реакцію | Бот видає відповідну роль, оновлює запис у Replit DB, показує підтвердження (або лог у службовий канал). |
| 9 | Закінчення строку дії тимчасової ролі | Роль видана з TTL, фонова задача активна | Минає заданий час дії ролі | Фонова задача знімає роль, подія логуються (опційно – DM користувачу з поясненням). |
| 10 | Обмеження доступу до керування конфігурацією (ACL) | ACL налаштований: лише OWNER/ADMIN мають повний доступ | Користувач з рівнем USER виконує команду зміни AI | Команда відхиляється, бот повідомляє про недостатні права, критичні налаштування не змінюються. |
| 11 | Обробка повідомлення в “чутливому” каналі | Канал позначено як пріоритетний для AI-модерації | Користувач надсилає потенційно конфліктне повідомлення | Повідомлення обов’язково проходить AI-пайплайн, рішення логуються у спецканал незалежно від результату. |

| | | | | |
|---|---|---|---|---|
| 12 | Відмова зовнішнього AI-сервісу (HuggingFace недоступний) | AI увімкнено, але HF повертає помилку/тайм аут | Користувач надсилає будь-яке повідомлення | Спрацьовує fallback: бот переходить до rule-based/спрощеної логіки, не падає, подія помилки логуються. |

Таблиця 4.2

Узагальнені результати експериментальної оцінки якості AI-модерації

| Категорія тесту | К-сть тестових повідомлень | TP (True Positive) | FP (False Positive) | FN (False Negative) | Precision | Recall |
|---|---|---|---|---|---|---|
| Токсичні (TOXIC) | 100 | 86 | 9 | 14 | 0.91 | 0.86 |
| Лайка / нецензурна лексика (PROFANITY) | 80 | 70 | 8 | 10 | 0.90 | 0.88 |
| Спам (SPAM) | 60 | 54 | 5 | 6 | 0.92 | 0.90 |
| NSFW-контент (NSFW/soft режим) | 40 | 33 | 4 | 7 | 0.89 | 0.83 |
| Нейтральні (SAFE) | 120 | 110 | 10 | – | 0.92* | – |

Для автоматизації ролей та каналів додатково перевірялися:

- коректність видачі/зняття ролей по закінченню строку (TTL);
- поведінка “імунітетів” (роль не знімається до закінчення імунітету);
- стабільність фонового планування (tasks.loop) при довготривалій роботі.

**Висновок до розділу 4**

У цьому розділі показано, що програмний засіб InfinityBot реалізує:

- трирівневу AI-модерацію, де rule-based шар виконує роль швидкого фільтра, ML/LLM-шар забезпечує глибший аналіз тексту й контексту, а fallback-евристики гарантують стійкість при відмові зовнішніх сервісів;
- автоматизацію керування сервером через модулі Autoroles і Channels, які знімають з адміністратора значну частину рутинних задач (видача/зняття ролей, дотримання політик каналів, логування подій);
- чітку інтеграцію з ACL та конфігураційним шаром, що дозволяє обмежити доступ до критичних налаштувань і адаптувати поведінку бота під конкретний сервер.

Функціональне тестування підтвердило працездатність ключових сценаріїв (AI-модерація, авторолі, політики каналів), а експериментальне оцінювання якості модерації показало здатність системи виявляти більшість токсичного контенту при прийнятному рівні хибних спрацьовувань. Отримані результати свідчать про те, що поставлена мета – реалізація програмного засобу інтелектуальної модерації й автоматизації керування Discord-сервером – досягнута на практичному рівні, достатньому для використання у реальних спільнотах.

# ВИСНОВКИ

У кваліфікаційній роботі розв’язано задачу розробки програмного засобу для інтелектуальної модерації та автоматизації керування Discord-сервером на основі багаторівневої AI-модерації й модульної архітектури.

На основі аналізу предметної області керування Discord-серверами та існуючих бот-рішень (MEE6, Carl-bot, Dyno тощо) показано, що популярні інструменти добре покривають задачі базової автоматизації (ролі, логування, сповіщення), однак здебільшого використовують rule-based модерацію й практично не застосовують повноцінні трирівневі AI-пайплайни з ML/LLM-компонентами та продуманою fallback-логікою. Це обґрунтовує актуальність побудови окремого програмного засобу, орієнтованого саме на інтелектуальну модерацію.

У роботі виконано аналіз методів модерації контенту: класичних rule-based підходів, ML-класифікації токсичного тексту та LLM-підходів з урахуванням контексту діалогу. Показано сильні та слабкі сторони кожного з них і сформульовано принцип комбінування: швидкі правила для очевидних порушень, ML/LLM для складних випадків, узагальнені евристики для постобробки та fallback. На цій основі запропоновано трирівневий пайплайн AI-модерації, який став методологічною основою реалізації InfinityBot.

Розроблено архітектуру Discord-бота InfinityBot, побудовану на модульному підході. Виділено окремі модулі: AIModeration (трирівнева модерація), Autoroles (автоматизація ролей), Channels (політики каналів та виконання дій), Config (конфігурація на базі Replit DB), Security/ACL (система рівнів доступу). Вибір стеку технологій (Python, discord.py, Replit з Replit DB, HuggingFace Inference API) дозволив поєднати безперервну роботу бота, простоту розгортання та можливість використання сучасних AI-моделей.

Реалізовано трирівневу AI-модерацію:

rule-based шар із fuzzy-патернами та антиобфускацією для швидкого виявлення очевидних порушень;

ML/LLM-шар на базі сервісів HuggingFace для класифікації токсичного, спам- та NSFW-контенту з урахуванням контексту;

шар постобробки й fallback-евристик, що забезпечує стійкість при відмовах зовнішніх сервісів і дозволяє адаптувати поведінку до режимів strict/soft.

Паралельно реалізовано автоматизацію керування сервером: модуль Autoroles забезпечує видачу та зняття ролей (у тому числі тимчасових, з урахуванням "імунітетів"), а модуль Channels застосовує політики каналів і виконує фактичні дії модерації (видалення, попередження, логування). Модуль Security / ACL централізовано обмежує доступ до критичних команд, пов'язуючи Discord-ролі з рівнями доступу (OWNER, ADMIN, MOD, USER).

Проведене тестування показало працездатність ключових сценаріїв роботи програмного засобу: обробки різних типів повідомлень (нейтральні, токсичні, обфусковані образи, спам, NSFW), роботи панелей авторолей, тайм-лімітованих ролей, а також обмеження доступу до змін конфігурації через ACL. Експериментальна оцінка якості AI-модерації на тестовій вибірці продемонструвала прийнятні значення базових метрик (precision, recall) для токсичного, спам- та NSFW-контенту при контрольованому рівні хибних спрацьовувань.

Разом із тим, отримані результати мають низку обмежень. Оцінювання проводилося на обмеженій тестовій вибірці, а якість модерації залежить від обраних моделей та мовної підтримки (насамперед для україномовного та російськомовного контенту). Використання зовнішніх сервісів (HuggingFace) накладає обмеження щодо затримок, лімітів запитів та вимог до конфіденційності. Ці фактори визначають напрями подальшого розвитку системи.

Перспективи удосконалення InfinityBot включають: розширення й формалізацію тестових датасетів для більш точної оцінки якості модерації, адаптацію та/або донавчання моделей під конкретні мовні й доменні особливості спільноти, створення веб-інтерфейсу для керування конфігурацією та моніторингу, підтримку додаткових платформ (інших чат-систем) на основі наявної архітектури, а також подальшу оптимізацію вартості та затримок при використанні AI-сервісів.

У підсумку можна зробити висновок, що поставлена мета – розробити програмний засіб Discord-бота з трирівневою AI-модерацією та засобами автоматизації керування сервером – досягнута. Реалізований InfinityBot є практично придатним інструментом для зниження рівня токсичності та полегшення роботи адміністраторів у реальних Discord-спільнотах і може слугувати основою для подальших досліджень та інженерних рішень у сфері інтелектуальної модерації контенту.

## СПИСОК БІБЛІОГРАФІЧНИХ ПОСИЛАНЬ ВИКОРИСТАНИХ ДЖЕРЕЛ


1. Russell S., Norvig P. Artificial Intelligence: A Modern Approach. – 4th ed. – Pearson, 2021. – 1152 p.
2. Goodfellow I., Bengio Y., Courville A. Deep Learning. – MIT Press, 2016. – 800 p.
3. Jurafsky D., Martin J. H. Speech and Language Processing. – 3rd ed. (draft). – 2023. – 900+ p.
4. Bird S., Klein E., Loper E. Natural Language Processing with Python. – O'Reilly, 2009. – 504 p.
5. Chollet F. Deep Learning with Python. – 2nd ed. – Manning, 2021. – 544 p.
6. Python 3. Documentation [Електронний ресурс]. – Режим доступу: https://docs.python.org/
7. discord.py – API Reference [Електронний ресурс]. – Режим доступу: https://discordpy.readthedocs.io/
8. Discord Developer Portal: API Documentation [Електронний ресурс]. – Режим доступу: https://discord.com/developers/docs/ Discord+1
9. Hugging Face. Transformers Documentation [Електронний ресурс]. – Режим доступу: https://huggingface.co/docs/transformers Hugging Face+1
10. Transformers: State-of-the-art Machine Learning for Text, Vision and More [Електронний ресурс]. – Режим доступу: https://github.com/huggingface/transformers GitHub
11. Replit Key-Value Store (Replit Database) Documentation [Електронний ресурс]. – Режим доступу: https://docs.replit.com/cloud-services/storage-and-databases/replit-database docs.replit.com+1
12. Replit: Built-in Database for Full-Stack Apps [Електронний ресурс]. – Режим доступу: https://replit.com/products/database Replit
13. MEE6 – The Best Discord Bot for Your Server [Електронний ресурс]. – Режим доступу: https://mee6.xyz/ mee6.xyz+1
14. carl-bot – Carl-bot Dashboard [Електронний ресурс]. – Режим доступу: https://carl.gg/ carl.gg+1
15. Dyno – The Feature-rich Discord Bot [Електронний ресурс]. – Режим доступу: https://dyno.gg/ dyno.gg+1

16. Fortuna P., Nunes S. “A Survey on Automatic Detection of Hate Speech in Text”. // ACM Computing Surveys. – 2018. – Vol. 51, No. 4.

17. Vidgen B., Derczynski L. “Directions in Abusive Language Training Data, a Systematic Review: Garbage In, Garbage Out”. // PLoS ONE. – 2020.

18. Wulczyn E., Thain N., Dixon L. “Ex Machina: Personal Attacks Seen at Scale”. // WWW 2017 Companion Proceedings. – 2017.

19. Ribeiro M. T. et al. “Why Should I Trust You? Explaining the Predictions of Any Classifier”. // KDD 2016. – 2016.

20. Bender E. M. et al. “On the Dangers of Stochastic Parrots: Can Language Models Be Too Big?”. // FAccT ’21. – 2021.

21. Discord Safety Center: Content Moderation [Електронний ресурс]. – Режим доступу: https://discord.com/safety

22. Discord Terms of Service [Електронний ресурс]. – Режим доступу: https://discord.com/terms

23. Discord Community Guidelines [Електронний ресурс]. – Режим доступу: https://discord.com/guidelines

24. Best Practices for Bot Moderation on Discord [Електронний ресурс] // Discord Developer Portal, Articles. – Режим доступу: https://discord.com/developers/docs/topics/community-resources (або відповідний розділ порталу). Discord+1

25. Mee6, Carl-bot, Dyno: Comparative Feature Overviews [Електронний ресурс] // Офіційні сайти та документація ботів. – Режим доступу: – https://mee6.xyz/ (MEE6); – https://carl.gg/ (Carl-bot); – https://dyno.gg/ (Dyno).

26. DSTU 8302:2015. Бібліографічне посилання. Загальні положення та правила складання. – К.: Мінекономрозвитку України, 2016.

**ДОДАТОК А**

## Структура вихідних кодів Discord-бота InfinityBot

У цьому додатку наведено узагальнену структура проєкту Discord-бота **InfinityBot** та основні модулі, які реалізують архітектуру, описану в пояснювальній записці.

### А.1. Структура проєкту

```
InfinityBot/
  main.py                – основний вхідний файл бота
  bot_launcher.py        – допоміжний запуск бота (режими / оточення)
  db_wrapper.py          – обгортка над Replit DB
  keep_alive.py          – підтримка активності процесу

  Modules/               – предметно-орієнтовані модулі
    AIModeration/        – трирівнева AI-модерація
      service.py         – пайплайн модерації, інтеграція з Discord
      ml.py              – виклики HuggingFace / ML API
      ml_backend.py      – бекенди моделей токсичності
      fuzzy.py           – fuzzy-патерни та антиобфускація
    Autoroles/           – автоматизація керування ролями
      service.py         – логіка видачі/зняття ролей, TTL, імунітети
      actions.py         – дії, прив'язані до панелей авторолей
      views.py           – інтерфейсні панелі в Discord
      utils.py           – утилітарні функції
    Channels/            – політики та службові дії в каналах
      service.py         – видалення, сповіщення, логування
      utils.py           – допоміжні функції
      views.py           – панелі налаштування каналів
    Config/              – конфігурація бота
      Config.py          – єдина точка доступу до конфігурацій
      store.py           – робота з Replit DB
      locales.py         – локалі та текстові ресурси
      views.py           – панелі для налаштувань
      utils.py           – утиліти конфігурації
    Security/            – безпека та ACL
      acl.py             – рівні доступу та resolve_access_level
      perm.py            – перевірки прав
      safe_io.py         – безпечна робота з файлами/експортом
      secure_view.py     – захищені інтерфейсні панелі
    Common/              – спільні утиліти, локалі, допоміжні функції
    Levels/              – система рівнів (exp, прогрес)
    Votings/             – голосування та опитування
    OneBlockedMessage/   – спеціальна логіка для блокованих повідомлень
    locales/             – файли локалізації (en/ru/uk)

  Cogs/                  – розширення з командами та логікою
  Menu/                  – меню та панелі керування
  Tools/                 – службові скрипти (перевірка локалей тощо)
  exports/               – експортовані конфігурації / дані
```

### А.2. Основні модулі та їх роль

- **Modules/AIModeration** – реалізація трирівневої AI-модерації (rule-based → ML/LLM → fallback), робота з HuggingFace, fuzzy-патернами.
- **Modules/Autoroles** – автоматизована видача/зняття ролей, тимчасові ролі (TTL), “імунітети”, панелі авторолей.
- **Modules/Channels** – застосування політик каналів, виконання рішень модерації (видалення, попередження, логування).
- **Modules/Config** – централізоване зберігання конфігурацій у Replit DB, локалі, панелі налаштувань.
- **Modules/Security (ACL)** – визначення рівнів доступу та перевірка прав для критичних команд.

**ДОДАТОК Б**

**Ключові фрагменти реалізації трирівневої AI-модерації**

У цьому додатку наведено скорочені фрагменти коду модуля AIModeration, які демонструють структуру трирівневого пайплайна AI-модерації.

**Б.1. Базова (неконтекстна) класифікація повідомлення**

```
# Modules/AIModeration/service.py (фрагмент)

async def classify_text(gid: int, content: str) -> AIResult:
    """
    Базовая (неконтекстная) классификация одного сообщения:
      Слой 1: быстрые очевидные паттерны (SPAM / NSFW) → решение без ML.
      Слой 2: ML по одному тексту (если включён).
      Слой 3: обобщённые правила + fuzzy, с учётом ml_res.
    В режиме nsfw_mode='soft' конечный NSFW допускается только при наличии
    порно-URL (см. _soft_nsfw_postprocess).
    """
    cfg = _load_config(gid)
    content_l = (content or "").lower()

    # СЛОЙ 1: очевидный SPAM/NSFW → без ML
    fast_res = _rules_fast_obvious(gid, content_l, cfg)
    if fast_res is not None:
        return fast_res

    # СЛОЙ 2: ML / HF-токсичность (один текст)
    ml_res = await ml_api.classify(content_l, guild_id=gid)

    # СЛОЙ 3: объединённые правила + fuzzy-паттерны
    final = _rules_postprocess(gid, content_l, cfg, ml_res)
    return final
```

Цей фрагмент показує послідовність: rule-based шар → ML-шар → постобробка (узагальнені правила + fuzzy).

**Б.2. Контекстна LLM-модерація діалогу**

```
# Modules/AIModeration/service.py (фрагмент)

async def classify_text_contextual(
    message: discord.Message,
    history_limit: int = 5
) -> AIResult:
    """
    Контекстная классификация:
      Слой 1: быстрые паттерны (SPAM/NSFW) по самому сообщению.
      Слой 2: диалоговый LLM (HF) с контекстом.
      Слой 3: неконтекстный fallback (одиночное сообщение: ML+rules).
```

```
    В режиме nsfw_mode='soft' любой NSFW без порно-URL будет понижен
    до PROFANITY (см. _soft_nsfw_postprocess).
    """
    guild = message.guild
    if not isinstance(guild, discord.Guild):
        # DMs / системные — без контекста, только по тексту
        return await classify_text(0, message.content or "")

    cfg = await asyncio.to_thread(_load_config, guild.id)
    # ... отримання історії діалогу та формування промпту для LLM ...
    # raw = await ml_api.classify_dialog(messages, guild_id=guild.id)
    # ... постобробка та fallback ...
```

**Б.3. Виклик HuggingFace / ML-бекенду**

```
# Modules/AIModeration/ml.py (фрагмент)

HF_ENDPOINT_ENV = "HF_MODERATION_ENDPOINT"
HF_TOKEN_ENV = "HF_MODERATION_TOKEN"

async def classify(text: str, *, guild_id: int | None = None) -> Optional[Dict[str, Any]]:
    """
    Старый API, который уже использует твой код:
      → dict {'label': 'TOXIC', 'score': 0.93, 'reason': '...'} или None.
    Работает по ОДНОМУ тексту (без явного контекста).
    """
    text = (text or "").strip()
    if not text:
        return None

    messages = [{"role": "user", "content": text}]
    raw = await _hf_call_messages(messages)
    if not raw:
        return None

    return _normalize_response(raw)
```

**Б.4. Fuzzy-патерни та антиобфускація**

```
# Modules/AIModeration/fuzzy.py (фрагмент)

# Разделители между буквами (символы, пробелы, подчёркивание и т.п.)
SEP = r"(?:[\W_]+)?"

# Базовая карта латиница↔кириллица + leetspeak
CHAR = {
    "a": "[aа@4]",
    "b": "[bб6]",
    "c": "[cс]",
    "e": "[eе3]",
    "g": "[g9]",
    "h": "[hн]",
```

```
    # ...
}

def word_to_regex(word: str) -> str:
    tokens = []
    for ch in word.lower():
        grp = CHAR.get(ch, re.escape(ch))
        tokens.append(grp)
    body = SEP.join(tokens)
    return rf"\b{body}\b"

def build_fuzzy_patterns(base_terms: List[str]) -> List[re.Pattern]:
    pats: List[re.Pattern] = []
    for w in base_terms:
        w = (w or "").strip()
        if not w:
            continue
        try:
            pats.append(re.compile(word_to_regex(w), re.IGNORECASE))
        except re.error:
            continue
    return pats
```

Цей код показує, як слова (у тому числі з можливими leetspeak/замінами) перетворюються на регулярні вирази, стійкі до обфускації.

**ДОДАТОК В**

## Приклади конфігурацій та тестових даних для AI-модерації й автоматизації

### В.1. Приклад конфігурації AI-модерації в Replit DB

Наведено узагальнений приклад структури записів у Replit DB для одного Discord-сервера (ідентифікатор умовний 123456789012345678):

```
config:global = {
  "owner_id": 987654321098765432,
  "locale": "uk-UA",
  "log_level": "INFO"
}

config:guild:123456789012345678 = {
  "ai": {
    "enabled": true,
    "mode": "soft",
    "nsfw_mode": "soft",
    "thresholds": {
      "TOXIC": 0.75,
      "SPAM": 0.80,
      "NSFW": 0.80
    }
  },
  "log_channel_id": 111222333444555666,
  "staff_role_id": 777888999000111222
}

state:guild:123456789012345678:autoroles = {
  "panels": [
    {
      "message_id": 555666777888999000,
      "emoji": "✅",
      "discord_role_id": 864200000000000001,
      "ttl_seconds": 86400
    }
  ]
}
```