ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ШТУЧНОГО ІНТЕЛЕКТУ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Інтелектуальна технологія розпізнавання емоційного стану співрозмовника під час електронного листування на основі методів машинного навчання»

на здобуття освітнього ступеня бакалавра
зі спеціальності 122 Комп'ютерні науки
освітньо-професійної програми Штучний інтелект

Кваліфікаційна робота містить результати власних досліджень.
Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Юліан ШВЕЦЬ
(підпис)

Виконав:

здобувач вищої освіти Юліан ШВЕЦЬ
група ШІД-42

Керівник: Олександр ЗВЕНІГОРОДСЬКИЙ
к.т.н., доцент

Рецензент:
науковий ступінь, вчене звання _____
Ім'я, ПРІЗВИЩЕ

Київ 2025

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
Навчально-науковий інститут інформаційних технологій

Кафедра Штучного інтелекту
Ступінь вищої освіти Бакалавр
Спеціальність 122 Комп'ютерні науки
Освітньо-професійна програма Штучний інтелект

ЗАТВЕРДЖУЮ
Завідувач кафедрою Штучного інтелекту
_____ Ольга ЗІНЧЕНКО
«_____»_____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Швецю Юліану Миколайовичу

1. Тема кваліфікаційної роботи: «Інтелектуальна технологія розпізнавання емоційного стану співрозмовника під час електронного листування на основі методів машинного навчання»
   керівник кваліфікаційної роботи Олександр ЗВЕНІГОРОДСЬКИЙ, к.т.н., доцент
   затверджені наказом Державного університету інформаційно-комунікаційних технологій від «24» лютого року № 56
2. Строк подання кваліфікаційної роботи 23.05.2025 р.
3. Вихідні дані до кваліфікаційної роботи: науково-технічна література на тему аналізу емоцій в тексті з використанням штучного інтелекту, дослідження довідників з застосування великих мовних моделей.
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
   1. Огляд актуальності теми
   2. Ознайомлення з алгоритмами систем емоційного аналізу тексту
   3. Реалізація системи розпізнавання емоцій
5. Перелік графічного матеріалу: *презентація*
   1. Актуальність теми
   2. Структура нейронної мережі
   3. Інструменти для реалізації проєкту
   4. Висновки
6. Дата видачі завдання 25.02.2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз наявної науково-технічної літератури	25.02-05.03.25	виконано
2	Огляд методів розпізнавання емоцій людини з використанням ШІ	06.03-10.03.25	виконано
3	Дослідження технологій LLM (Large language model)	11.03-23.03.25	виконано
4	Ознайомлення з фреймворками, необхідними для роботи	24.03-09.04.25	виконано
5	Розробка архітектури та структури проєкту, програмна реалізація системи	10.04-14.05.25	виконано
6	Загальне тестування	15.05-18.05.25	виконано
7	Оформлення документації	18.05-23.05.25	виконано

Здобувач вищої освіти _____ Юліан ШВЕЦЬ

(підпис)

Керівник
кваліфікаційної роботи _____ Олександр ЗВЕНІГОРОДСЬКИЙ

(підпис)

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 64 стор., 22 рис., 20 джерел.

Мета роботи – розробка інформаційної технології для ефективного розпізнавання емоційного стану користувачів під час електронного листування, що дозволить підвищити якість і чутливість автоматизованих комунікаційних систем.

Об'єкт дослідження – процес розпізнавання та аналізу емоційних станів у текстових повідомленнях користувачів.

Предмет дослідження – сучасні алгоритми та технології штучного інтелекту, зокрема трансформерні моделі для глибинного навчання, які використовуються для автоматичного розпізнавання емоцій у тексті.

Методи дослідження – аналіз і узагальнення науково-технічної літератури, порівняльний аналіз існуючих алгоритмів емоційного аналізу, реалізація та тестування нейронних моделей.

Короткий зміст роботи: У роботі розглянуто актуальність і значення розпізнавання емоцій у сучасних цифрових комунікаціях. Здійснено аналіз існуючих методів, серед яких лексиконні підходи, класичне машинне навчання та глибинне навчання на базі трансформерів. Особливу увагу приділено вивченню нейронних мереж із глибинним навчанням, їх структурі та алгоритмам навчання.

У результаті дослідження розроблено інформаційну технологію, яка інтегрує трансформерну модель BERT (cardiffnlp/twitter-roberta-base-emotion). Вебзастосунок, побудований на базі Flask та Django, дозволяє ефективно розпізнавати основні емоції (радість, смуток, злість, подив), визначати емоційний контекст у реальному часі та враховувати історію взаємодії користувача для точнішого аналізу. Запропонована система може бути використана в різних сферах: онлайн-консультування, моніторинг соціальних мереж, освіта, маркетинг і психологічна підтримка.

КЛЮЧОВІ СЛОВА: ШТУЧНИЙ ІНТЕЛЕКТ, ЕМОЦІЙНИЙ АНАЛІЗ ТЕКСТУ, ТРАНСФОРМЕРИ, ГЛИБИННЕ НАВЧАННЯ, НЕЙРОННІ МЕРЕЖІ.

ABSTRACT

Text part of the bachelor's qualification work: 64 pages, 22 pictures, 20 sources.

The purpose of the work is to develop an information technology aimed at effectively recognizing users' emotional states during electronic communication, thereby enhancing the quality and responsiveness of automated communication systems.

The object of research is the process of recognizing and analyzing emotional states in users' text messages.

The subject of research encompasses modern artificial intelligence algorithms and technologies, particularly transformer models for deep learning, used for automatic emotion recognition in text.

Research methods include analysis and summarization of scientific and technical literature, comparative analysis of existing emotion recognition algorithms, and implementation and testing of neural network models.

Summary of the work: The research addresses the relevance and importance of emotion recognition in contemporary digital communications. It includes an analysis of existing methods, such as lexical approaches, classical machine learning, and transformer-based deep learning. Special attention is paid to studying neural networks with deep learning, their structures, and learning algorithms.

As a result of this study, an information technology integrating the transformer-based BERT model (cardiffnlp/twitter-roberta-base-emotion) has been developed. A web application built using Flask and Django frameworks effectively recognizes primary emotions (joy, sadness, anger, surprise), determines emotional context in real-time, and considers the user's interaction history for more accurate analysis. The proposed system can be applied in various domains, including online consulting, social media monitoring, education, marketing, and psychological support.

KEYWORDS: ARTIFICIAL INTELLIGENCE, TEXT EMOTION ANALYSIS, TRANSFORMERS, DEEP LEARNING, NEURAL NETWORKS.

ЗМІСТ

ВСТУП 11

1 ОГЛЯД АКТУАЛЬНОСТІ ТЕМИ 12

1.1 Емоції як ключ до ефективної взаємодії між людиною та штучним інтелектом 12

1.2 Актуальність задачі в сучасному цифровому світі 13

1.3 Завдання та виклики емоційного аналізу тексту 16

1.4 Історичний контекст та еволюція аналізу тексту 17

1.5 Висновки до розділу 20

2 ОЗНАЙОМЛЕННЯ З АЛГОРИТМАМИ СИСТЕМ ЕМОЦІЙНОГО АНАЛІЗУ ТЕКСТУ 21

2.1 Загальні поняття штучних нейронних мереж 21

2.2 Глибинне навчання штучних нейронних мереж 23

2.3 Алгоритми навчання штучних нейронних мереж 24

2.4 Підходи до реалізації систем емоційного аналізу тексту 25

2.5 Висновки до розділу 27

3 РЕАЛІЗАЦІЯ СИСТЕМИ РОЗПІЗНАВАННЯ ЕМОЦІЙ 28

3.1 Технологічний стек 28

3.2 Архітектура та структура проєкту 30

3.3 Як працює фронтенд 33

3.4 Як працює бекенд 36

3.5 Як працює нейромережевий аналіз. Інтеграція баз даних 38

3.6 Docker — для чого і як працює 42

3.7 Тестування системи 46

3.8 Узагальнення результатів тестування: 56

ВИСНОВКИ..... 59
ПЕРЕЛІК ПОСИЛАНЬ..... 62
ДОДАТОК А. ПРОГРАМНИЙ КОД ПРОЄКТОВАНОЇ СИСТЕМИ 64
ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ 66

ВСТУП

У сучасному цифровому суспільстві дедалі більшого значення набувають системи, здатні розуміти емоційний контекст людської комунікації. З розвитком дистанційного спілкування, зокрема в таких каналах, як електронна пошта, месенджери та онлайн-чати, емоційний аналіз тексту стає важливим інструментом для підтримки ефективної, емпатичної та безпечної взаємодії. Автоматичне визначення емоційного стану співрозмовника дозволяє значно підвищити якість обслуговування в онлайн-сервісах, розширити можливості психологічної підтримки, а також створити більш гнучкі та адаптивні інтерфейси.

Класичні підходи до аналізу емоцій базуються на словникових (лексиконних) методах або алгоритмах машинного навчання, які мають обмеження щодо точності, універсальності та здатності враховувати контекст. Завдяки розвитку технологій штучного інтелекту, зокрема трансформерних моделей та глибинного навчання, з’явилися ефективніші засоби інтерпретації текстів, здатні точно визначати емоційні відтінки повідомлень навіть у складних або неоднозначних комунікаційних ситуаціях.

У межах цієї дипломної роботи досліджується можливість створення інформаційної технології розпізнавання емоційного стану користувача під час електронного листування. Запропонований підхід ґрунтується на застосуванні трансформерної моделі для аналізу тексту.

Робота має прикладний характер і орієнтована на створення функціонального інструменту, придатного для застосування в онлайн-комунікаціях, автоматизованому консультуванні, службах підтримки, освіті та суміжних галузях.

1 ОГЛЯД АКТУАЛЬНОСТІ ТЕМИ

1.1 Емоції як ключ до ефективної взаємодії між людиною та штучним інтелектом

Упродовж тисячоліть емоції були невід’ємною частиною людського спілкування, поведінки та прийняття рішень. Вони формують наше ставлення до подій, впливають на оцінку інформації, визначають глибину сприйняття та взаємодії з навколишнім середовищем. Емоції є основою не лише індивідуального досвіду, а й соціального функціонування, оскільки саме через емоційне забарвлення ми зчитуємо наміри, настрій, відтінки сенсів у словах інших людей. Розуміння емоцій — це передумова емпатії, делікатного діалогу та успішної взаємодії.

З розвитком цифрових технологій та поширенням автоматизованих сервісів з'явилася нова потреба — адаптувати ці сервіси до особливостей людської комунікації. Якщо раніше користувач сприймав комп’ютер як "беземоційну машину", то сьогодні очікування значно змінилися. Чат-боти, голосові асистенти, онлайн-консультанти та інші інтелектуальні системи дедалі частіше стають "обличчям" компанії чи інструментом взаємодії у критичних ситуаціях (служби підтримки, медичні консультації, психологічні платформи). У таких контекстах здатність штучного інтелекту розпізнати емоційний стан користувача набуває принципового значення.

Нерозуміння емоцій або їхня хибна інтерпретація можуть призвести не лише до неефективного діалогу, а й до втрати довіри, негативного користувацького досвіду, ескалації конфліктів. Наприклад, відповідь, нейтральна з точки зору логіки, може виглядати зневажливою або холодною для людини у стресі. Навпаки, вчасне виявлення смутку, роздратування чи збудження дозволяє системі адаптувати свою відповідь, змінити тон, запропонувати підтримку чи перенаправити запит до "живого" оператора.

У зв’язку з цим виникає завдання — навчити штучні інтелекти "розуміти" людину на емоційному рівні. І хоча комп’ютер не має емоцій у людському сенсі,

він може і повинен навчитися розпізнавати емоційні сигнали у тексті, голосі, міміці. Одним із найбільш поширених і доступних каналів для такого аналізу є текстове спілкування, яке домінує в онлайн-взаємодії. Саме текст є джерелом даних, з якого система може дізнатися про настрій користувача — через лексику, пунктуацію, граматичні конструкції, навіть довжину речень.

Розпізнавання емоцій у тексті є складним міждисциплінарним завданням, що поєднує лінгвістику, психологію, інформатику та нейронні мережі. Протягом останнього десятиліття ця галузь зазнала стрімкого розвитку завдяки впровадженню глибинного навчання, зокрема архітектури трансформерів (Transformers), яка дозволяє моделі бачити контекст не лише навколо слова, а й у масштабі всього повідомлення.

Таким чином, створення сервісу, здатного розпізнавати емоційний стан користувача на основі тексту — це не лише технічне завдання, а й соціальна необхідність. Воно спрямоване на підвищення якості взаємодії між людиною і машиною, на гуманізацію цифрового середовища та на створення інтелектуальних систем нового покоління — більш чутливих, делікатних, уважних до людини.

1.2 Актуальність задачі в сучасному цифровому світі

У XXI столітті цифрова комунікація стала основною формою взаємодії між людьми, компаніями, установами та державними органами. Кожного дня мільярди повідомлень передаються через електронну пошту, месенджери, чат-боти, соціальні мережі та платформи онлайн-обслуговування. У цьому величезному обсязі інформації ключову роль починають відігравати **емоції**, які супроводжують кожне повідомлення. Вони стають не лише засобом вираження ставлення, а й цінним джерелом даних, здатним розкрити приховані наміри, потреби або проблеми користувача.

Проблема полягає в тому, що більшість цифрових систем залишається “емоційно глухою”. Вони оперують фактами, ключовими словами, структурами запитів, але не враховують емоційного фону, який може кардинально змінити

значення повідомлення. Саме тому здатність системи розпізнати емоційний контекст повідомлення — критично важливий чинник для розвитку сучасних інтерфейсів.

Актуальність задачі емоційного аналізу тексту визначається щонайменше чотирма глобальними тенденціями:

1. Глобальна цифровізація комунікації. З кожним роком збільшується кількість користувачів, які віддають перевагу письмовому спілкуванню в онлайн-просторі. Компанії, державні установи, банки, лікарі, консультанти — усі переходять на дистанційне обслуговування. У такому контексті текстове повідомлення часто є єдиним каналом зворотного зв’язку між клієнтом і системою.
2. Поява “чутливих” до контексту систем. Прості чат-боти, які працюють за сценарієм “якщо → тоді”, поступово втрачають свою актуальність. Їх замінюють **інтелектуальні агенти**, які повинні адаптувати свою поведінку до тону, настрою та навіть емоційної стабільності користувача. Без здатності розпізнавати емоції така адаптація неможлива.
3. Соціальна потреба в емпатії з боку ШІ. З розвитком AI-систем зростає вимога до їх “людяності”. У психології, освіті, медичному консультуванні важливо, щоб система вміла розпізнати ознаки стресу, тривоги, збудження або, навпаки, апатії. Це дає змогу уникати загострень, підвищити якість обслуговування й зміцнити довіру користувача
4. Зростання ролі емоцій в аналітиці та прогнозуванні. Маркетингові дослідження, соціологічні опитування, моніторинг відгуків — усі ці галузі дедалі частіше використовують емоційні метрики як доповнення до кількісних. Емоції дозволяють не лише фіксувати реакцію, а й прогнозувати поведінку.

Приклади застосування в реальних умовах

- Онлайн-служби підтримки системи, які розпізнають емоції в повідомленнях користувача, можуть автоматично змінювати свою поведінку: переходити на м’який стиль спілкування у разі виявлення

роздратування, надавати додаткову інформацію при виявленні розгубленості, або передавати запит “живому” оператору при високому рівні емоційного збудження.

- Освіта — у платформах дистанційного навчання аналіз емоцій студентів у чатах або відповідях дозволяє виявляти рівень залученості, задоволення або перевантаження. Це відкриває перспективи для адаптивного навчання, коли складність або тон викладу автоматично підлаштовується під настрій учня.
- Психологічні сервіси у цифровій психотерапії аналіз текстових повідомлень дозволяє відстежувати емоційні зміни клієнта з часом. Такі сервіси, як Woebot чи Wysa, вже зараз використовують алгоритми емоційної класифікації для надання рекомендацій або попередження кризових станів.
- Маркетинг і бренд-моніторинг компанії аналізують тональність і емоційне забарвлення відгуків про продукти. Це дозволяє не тільки класифікувати повідомлення за полярністю, а й виявляти глибші емоційні тенденції — наприклад, страх перед змінами, відчуття зради брендом, очікування інновацій тощо.
- HR-аналітика під час попереднього відбору кандидатів системи можуть автоматично аналізувати мотиваційні листи або відкриті відповіді, визначаючи ступінь ентузіазму, сумніву, самоподачі тощо.

Усе вищезазначене свідчить про те, що емоційний аналіз тексту вже давно вийшов за межі теоретичних досліджень і став реальним інструментом в арсеналі сучасного бізнесу, науки та державного управління. Його важливість лише зростатиме, а отже — розвиток і вдосконалення таких систем є не просто технічною інновацією, а **суспільною потребою**, на яку дипломна робота прагне дати практичну відповідь.

1.3 Завдання та виклики емоційного аналізу тексту

Емоційний аналіз тексту, або ж **emotion detection**, — це специфічне підзавдання в межах ширшої дисципліни аналізу тональності (**sentiment analysis**), яке має на меті не просто визначити, позитивне чи негативне повідомлення, а виявити конкретну емоцію, що в ньому виражена. До таких емоцій зазвичай належать: радість, сум, страх, злість, здивування, любов, огиду тощо. Залежно від прикладної задачі, емоцій може бути більше або менше — від базових семи (за класифікацією Екмана) чи мінімальних двох (злість та радість) до десятків відтінків у психологічно орієнтованих дослідженнях.

- Основні завдання систем емоційного аналізу включають:
- Класифікацію тексту за емоційною ознакою — визначення, яку саме емоцію передає повідомлення.
- Визначення інтенсивності емоції — не лише категорія, а й сила її вираження.
- Обробку контексту — аналіз повідомлень у діалозі, серії або часовому ряді.
- Врахування індивідуального стилю користувача — адаптація моделі до того, як конкретна людина виражає емоції.
- Інтерпретацію неоднозначних повідомлень — робота з іронією, сарказмом, гумором, подвійними значеннями.
- Мультимодальну інтеграцію — комбінування тексту з голосом, мімікою або поведінкою (у розширених системах).
- Головні труднощі, з якими стикається ця галузь:
- Мовна неоднозначність: багато слів мають емоційне значення лише в певному контексті. Наприклад, слово "сильно" може вказувати як на позитивну емоцію ("сильно люблю"), так і на негативну ("сильно дратує").

- Суб’єктивність інтерпретації: одне й те саме повідомлення різні люди можуть сприймати по-різному — залежно від культурного контексту, настрою, попереднього досвіду.
- Недостатність позначених даних: для навчання моделей необхідні великі масиви текстів із вказаними емоціями. Такі корпуси є, але часто — обмежені або нерепрезентативні.
- Сарказм та іронія: моделі, навіть потужні, часто не можуть вловити, коли емоція виражена непрямо або навмисно спотворена. Це залишається одним із найскладніших аспектів для NLP.
- Мовна варіативність: повідомлення можуть бути написані сленгом, діалектизмами, із помилками, емодзі або змішаними мовами — що ускладнює обробку.
- Контекст взаємодії: без розуміння попередніх повідомлень або ситуації, в якій відбувається спілкування, система може дати некоректну відповідь.

Усі ці виклики вимагають комплексного підходу до побудови системи розпізнавання емоцій. Простих алгоритмів недостатньо — необхідна інтеграція сучасних моделей машинного навчання, архітектур трансформерів, адаптивних механізмів врахування контексту та гнучкої взаємодії з користувачем.

1.4 Історичний контекст та еволюція аналізу тексту

Ще задовго до появи сучасних глибоких нейронних мереж дослідники й інженери прагнули навчити комп’ютерні системи розпізнавати не лише зміст, а й емоційне забарвлення текстових повідомлень. Аналіз емоцій в тексті, як окрема підгалузь обробки природної мови (NLP — Natural Language Processing), почав формуватися ще в кінці XX століття — у період, коли з’явились перші системи автоматичного розпізнавання тональності (sentiment analysis). Початково такі системи ґрунтувалися виключно на лексичних правилах, де ключовими були словники, що містили позитивні або негативні слова з відповідними ваговими коефіцієнтами.

Ці словникові підходи, хоча й обмежені, заклали фундаментальні уявлення про можливість емоційної інтерпретації мови на машинному рівні. Зокрема, у 2002–2004 роках активно використовувалися бази типу SentiWordNet та General Inquirer, які дозволяли прив’язати кожне слово до емоційного класу. Методики тоді будувалися за принципом простого підрахунку співвідношення позитивних і негативних слів у реченні, а самі емоції зводились до найпростіших категорій — “позитивне”, “негативне”, іноді — “нейтральне”.

З розвитком комп’ютерних потужностей та зростанням обсягів цифрових текстів у публічному просторі (блоги, форуми, соцмережі) виникла потреба у більш точних методах, здатних враховувати не лише окремі слова, а й контекст, у якому вони вживаються. Саме тоді на передній план починає виходити машинне навчання, зокрема методи на основі класичних алгоритмів: Naive Bayes, SVM (підтримувальні векторні машини), логістична регресія. Ці підходи вимагали побудови *ознакового простору* (features) — наприклад, за допомогою TF-IDF, Bag-of-Words, n-грам тощо. Модель “навчалась” на заздалегідь розмічених даних і виводила ймовірність приналежності нового тексту до певної емоційної категорії. У порівнянні з лексичними методами, ці моделі були точнішими, здатними враховувати статистичні зв’язки між словами, проте все ще залишались “сліпими” до контексту — тобто не розрізняли, наприклад, сарказм або подвійне заперечення.

Наступним великим етапом стало впровадження глибинного навчання (deep learning). Близько 2013–2015 року дослідники почали активно використовувати нейронні мережі, зокрема RNN (рекурентні нейромережі) та їх модифікації — LSTM і GRU. Ці моделі вже вміли враховувати послідовність слів у тексті, тобто формально “пам’ятали” попередній контекст, що дало змогу підвищити якість емоційної класифікації. Вони використовувались у парі з **word embedding’ами** — наприклад, Word2Vec або GloVe — які кодували кожне слово у вигляді багатовимірного вектора. Це дозволило відходити від буквального сприйняття слова до його "семантичного відображення", наближеного до людського розуміння значення.

Однак справжній прорив відбувся у 2017 році, коли було опубліковано знакову статтю “Attention Is All You Need”, що представила архітектуру Transformer. На її основі була розроблена низка революційних моделей, серед яких найбільш відомою стала BERT (Bidirectional Encoder Representations from Transformers)**,** представлена Google у 2018 році. На відміну від попередніх моделей, які читали текст здебільшого зліва направо, BERT здійснює двонаправлену обробку, тобто враховує слова як зліва, так і справа від поточного — що дало змогу моделі краще “розуміти” повний контекст.

BERT та її похідні — DistilBERT, RoBERTa, ALBERT, Electra — почали активно застосовуватись у задачах емоційної класифікації. Особливо помітною стала поява моделей, попередньо *навчених спеціально для емоційного аналізу* (fine-tuned on emotion datasets), зокрема cardiffnlp/twitter-roberta-base-emotion, яка була використана у рамках даного проєкту. Вона навчена на відкритому корпусі GoEmotions (від Google), що містить понад 58 тис. анотацій з емоційною розміткою на англомовних текстах.

Таким чином, за останні 20 років відбулося колосальне перетворення: від примітивних словників до складних, багатомільйонних моделей із сотнями параметрів і можливістю обробляти емоційний підтекст на рівні, близькому до людського. Емоційний аналіз із прикладного інструмента для маркетологів перетворився на *потужний міждисциплінарний механізм*, який використовують у сфері охорони здоров’я, освіти, кібербезпеки, автоматизованого консультування та, що найважливіше, в архітектурі "чутливих" інтерфейсів взаємодії людини та штучного інтелекту.

Сучасні моделі, доступні через платформи типу HuggingFace, дозволяють у кілька рядків коду реалізувати обчислення, які ще 10 років тому вимагали б кількох тижнів академічної роботи. Завдяки цьому аналіз емоцій став не лише доступним, а й гнучким — кожен розробник може легко інтегрувати такі інструменти у власний вебзастосунок, налаштувати під свої потреби та розширити, наприклад, до обробки історії діалогу, як це реалізовано у межах цієї дипломної роботи.

1.5 Висновки до розділу

У першому розділі здійснено комплексний огляд теоретичних засад аналізу емоцій у тексті, що є критично важливим етапом у побудові інформаційної технології розпізнавання емоційного стану користувача. Детально розглянуто понятійний апарат, зокрема визначення емоції, емоційного стану, особливостей їх прояву у текстових повідомленнях, а також ролі емоцій у цифровій комунікації. Окреслено основні підходи до емоційного аналізу: лексиконні, статистичні та нейромережеві, їхні переваги, недоліки та сфери застосування.

Розділ також містить історичний огляд еволюції відповідних методів — від простих словникових стратегій до сучасних глибоких моделей з використанням штучного інтелекту. Це дозволяє простежити розвиток галузі й підкреслити актуальність автоматизованого аналізу емоцій у сучасному інформаційному середовищі. Таким чином, перший розділ формує надійну теоретичну основу, що обґрунтовує вибір відповідних технологій для подальшої реалізації системи.

2 ОЗНАЙОМЛЕННЯ З АЛГОРИТМАМИ СИСТЕМ ЕМОЦІЙНОГО АНАЛІЗУ ТЕКСТУ

2.1 Загальні поняття штучних нейронних мереж

Штучні нейронні мережі (ШНМ) — це математичні моделі, які намагаються відтворити принципи функціонування біологічної нервової системи людини або інших живих істот. Ці моделі складаються з численних штучних нейронів, що об’єднані у складні структури і взаємодіють один з одним. Основною метою створення ШНМ є моделювання здатності мозку людини навчатися та вирішувати складні завдання, такі як класифікація, прогнозування та аналіз природної мови. Штучні нейронні мережі набули великої популярності завдяки здатності ефективно навчатися на великих обсягах даних і знаходити закономірності, які важко виявити традиційними алгоритмічними методами.

Структурно штучні нейронні мережі зазвичай мають три основних рівні:

- Вхідний шар, який відповідає за прийом та попередню обробку початкових даних. Цей шар забезпечує початковий етап роботи мережі, направляючи дані далі.
- Приховані шари, які є проміжними і відповідають за складну математичну обробку інформації. На цьому етапі відбувається виділення ключових особливостей і формування складних внутрішніх уявлень, що забезпечують ефективне навчання нейронної мережі.
- Вихідний шар, що відповідає за видачу фінального результату. Залежно від поставленої задачі це може бути прогноз, класифікація, оцінка ймовірності тощо.

На рисунках 2.1 та 2.2 можна побачити структуру штучних нейронних мереж: одношарової та багатошарової.

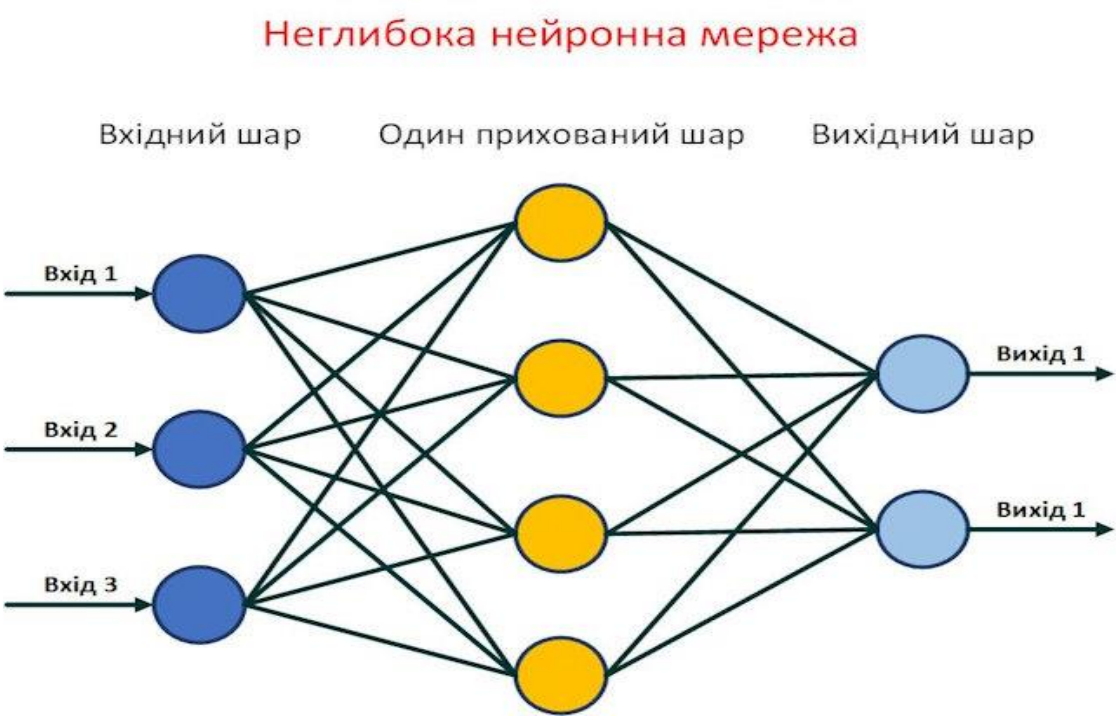


Рис. 2.1 – Одношарова нейронна мережа.

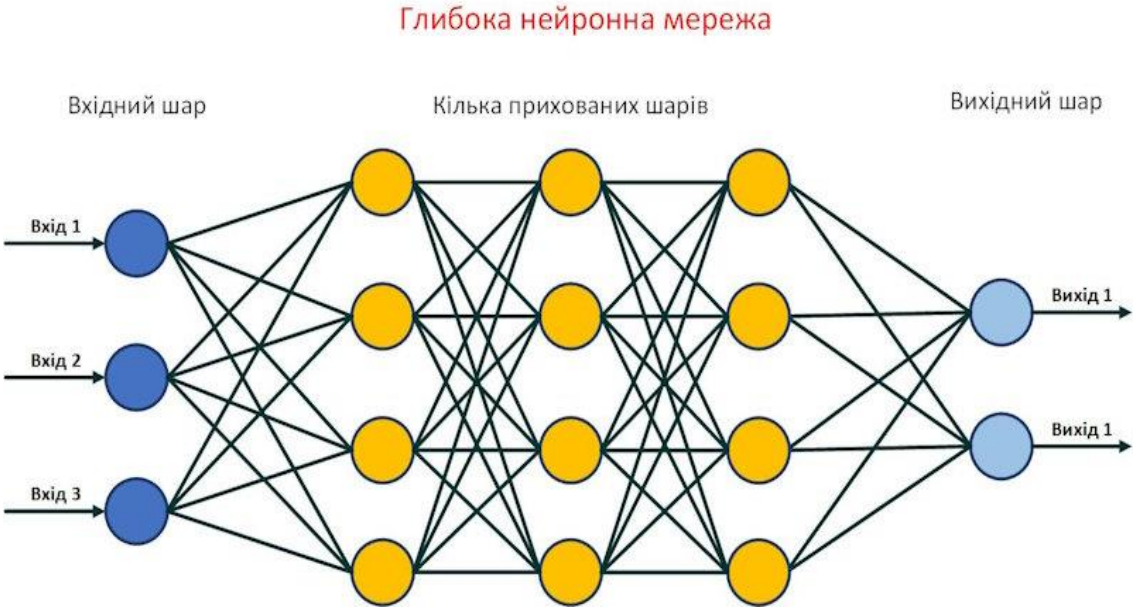


Рис. 2.2 – Багатошарова нейронна мережа.

До важливих переваг ШНМ можна віднести:

- адаптивність і здатність навчатися на основі наданих наборів даних;
- можливість самоорганізації, завдяки якій мережа може автоматично структурувати та систематизувати інформацію;
- можливість паралельної обробки даних, що дозволяє забезпечувати роботу в режимі реального часу;
- відмовостійкість завдяки надлишковості інформації та великій кількості взаємопов’язаних нейронів, що дає змогу уникати критичних збоїв у роботі мережі навіть у разі часткового пошкодження.

2.2 Глибинне навчання штучних нейронних мереж

Глибинне навчання (Deep Learning) є окремою, спеціалізованою галуззю машинного навчання, яка зосереджується на використанні нейронних мереж із великою кількістю прихованих шарів. Цей напрямок став особливо актуальним завдяки зростанню обчислювальних потужностей комп’ютерних систем та великих обсягів доступних даних для навчання. Саме через значну кількість прихованих шарів такі нейронні мережі можуть виявляти дуже складні закономірності та патерни в даних, які залишаються недоступними для простіших або класичних методів машинного навчання.

Однією з важливих особливостей глибоких нейронних мереж є їхня багатошарова структура, що дозволяє мережі поступово переходити від конкретних особливостей даних до більш узагальнених та абстрактних уявлень. Кожен наступний шар нейронної мережі отримує оброблену інформацію з попереднього, завдяки чому поступово збільшується рівень абстракції та узагальнення даних. Це особливо корисно у задачах із великим обсягом інформації та високою складністю структури.

- Глибокі нейронні мережі широко застосовуються у різноманітних завданнях, таких як:
- розпізнавання тексту та аналіз емоцій, де мережа вчиться інтерпретувати зміст і контекст написаного;

- розпізнавання зображень, відео та комп’ютерне бачення загалом;
- створення рекомендаційних систем, які можуть ефективно пропонувати користувачам найбільш релевантний контент;
- автоматичне управління автомобілями, включаючи аналіз ситуації на дорозі та прийняття швидких рішень.

Найпоширенішими типами глибоких нейронних мереж є згорткові нейронні мережі (CNN), рекурентні нейронні мережі (RNN), серед яких особливо виділяються LSTM та GRU, і трансформерні моделі, такі як BERT, що останнім часом набули надзвичайної популярності.

Трансформерні моделі є особливо ефективними в задачах аналізу природної мови, завдяки своїй здатності ефективно враховувати контекст слів у реченні за допомогою механізму уваги (attention). Це дозволяє досягти високої точності при аналізі емоційних станів, ідентифікації настроїв користувачів та виконання інших складних задач. Зокрема, моделі BERT, DistilBERT і RoBERTa стали стандартом сучасних систем емоційного аналізу тексту, значно покращуючи можливості програмних засобів у цій сфері.

2.3 Алгоритми навчання штучних нейронних мереж

Алгоритми навчання штучних нейронних мереж є важливою складовою, що забезпечує їх здатність ефективно вирішувати поставлені завдання. Основні типи навчання включають контрольоване, неконтрольоване та напівконтрольоване навчання.

Контрольоване навчання передбачає використання попередньо розмічених даних, що містять правильні відповіді або результати. Мережа намагається мінімізувати помилки шляхом коригування вагових коефіцієнтів на основі різниці між передбаченим та фактичним результатом. Найпоширенішими алгоритмами в контрольованому навчанні є зворотне поширення помилки (backpropagation), стохастичний градієнтний спуск (stochastic gradient descent) і алгоритм Левенберга-Марквардта.

Неконтрольоване навчання використовується, коли мережі надаються не розмічені дані, і вона самостійно намагається знайти приховані закономірності або

структури в цих даних. Типовими алгоритмами в цьому напрямі є кластеризація, наприклад, алгоритм k-середніх (k-means) або самоорганізовані карти Кохонена (Self-Organizing Maps, SOM).

2.4 Підходи до реалізації систем емоційного аналізу тексту

Існує кілька підходів до реалізації автоматичного емоційного аналізу тексту. Вони розвивалися паралельно з еволюцією інструментів обробки природної мови та машинного навчання, і кожен з них має свої переваги та обмеження. Сучасні системи зазвичай комбінують кілька підходів, але історично їх можна умовно поділити на три основні класи: лексиконні методи, класичне машинне навчання та глибинне навчання з трансформерами.

2.1.1 Лексиконні (словникові) методи

Ці методи базуються на попередньо складених словниках, у яких кожному слову приписується певна емоційна характеристика (наприклад, "joy", "sadness", "anger") або оцінка (наприклад, -1 до +1 за шкалою тональності).

- Переваги:
- Прості у реалізації.
- Не вимагають навчання моделі.
- Добре працюють для базового аналізу та швидкого прототипування.
- Недоліки:
- Не враховують контекст слова в реченні.
- Не здатні обробляти складні конструкції, іронію або багатозначність.
- Погано працюють із неформальними текстами (наприклад, месенджери, форуми).

2.1.2 Класичне машинне навчання

Цей підхід передбачає **попередню обробку тексту** (токенізація, нормалізація, видалення стоп-слів) і перетворення його у векторну форму (наприклад, TF-IDF або Bag of Words), після чого застосовуються моделі типу:

- Naive Bayes
- Support Vector Machine (SVM)

- Logistic Regression
- Decision Trees
- Переваги:
- Працює краще за словникові методи.
- Досить швидке навчання.
- Добре адаптується до специфіки мовлення в обраній предметній галузі.
- Недоліки:
- Модель не розуміє значення слів — лише частоти.
- Не враховується порядок слів.
- Не має "пам’яті" про контекст.

2.1.3 Глибинне навчання та трансформери

Найсучасніші системи емоційного аналізу використовують глибокі нейромережі, зокрема архітектуру трансформерів (Transformers), яка забезпечує контекстуальне кодування слів. На відміну від рекурентних моделей, трансформери здатні паралельно обробляти послідовності тексту й ефективно враховують весь контекст речення.

- Серед найвідоміших моделей:
- *BERT* (Bidirectional Encoder Representations from Transformers)
- *DistilBERT* (спрощена та пришвидшена версія BERT)
- *RoBERTa, XLNet, ALBERT*
- Моделі від *HuggingFace* (наприклад, cardiffnlp/twitter-roberta-base-emotion)
- Переваги:
- Висока точність.
- Урахування повного контексту (до і після слова).
- Можливість тонкої адаптації (fine-tuning) до власного датасету.
- Недоліки:
- Вимагає великих обчислювальних ресурсів (GPU/TPU).
- Складність реалізації без відповідного досвіду.

- Чутливість до балансу класів та якості навчального набору.

З огляду на детальний аналіз переваг і недоліків кожного з розглянутих підходів — від простих лексиконних методів до глибоких нейромережевих архітектур, а також враховуючи практичні аспекти розгортання, точність, вимоги до ресурсів, масштабованість і гнучкість — я вирішив зупинитися саме на використанні моделі cardiffnlp/twitter-roberta-base-emotion з репозиторію HuggingFace Transformers. Ця модель, на мою думку, найкраще поєднує в собі ключові характеристики, необхідні для ефективної реалізації задачі емоційного аналізу тексту: високу точність класифікації, помірні обчислювальні потреби, простоту інтеграції у вебсистему та наявність відкритої документації й підтримки спільноти.

2.5 Висновки до розділу

У другому розділі проаналізовано технічні принципи функціонування сучасних систем емоційного аналізу, зокрема ті, що базуються на штучних нейронних мережах. Висвітлено основи побудови ШНМ: їх архітектуру, рівневу структуру (вхідний, приховані та вихідний шари), способи представлення даних, а також принципи функціонування нейронів. У межах цього розділу розглянуто ідеї, що лежать в основі глибинного навчання, та пояснено, як саме багатоетапна обробка дозволяє мережам поступово формувати узагальнене уявлення про структуру даних.

Особливу увагу приділено вивченню різних типів нейронних мереж — згорткових, рекурентних і трансформерних — із роз’ясненням їхніх переваг у задачах обробки природної мови. Значну частину тексту присвячено опису алгоритмів навчання нейронних мереж: контрольованого, неконтрольованого й напівконтрольованого, а також методів оптимізації (наприклад, стохастичний градієнтний спуск, backpropagation тощо).

Таким чином, розділ дає всебічне уявлення про технічну базу, яка лежить в основі сучасних систем розпізнавання емоцій у тексті, й готує ґрунт для практичної реалізації подібних технологій.

3 РЕАЛІЗАЦІЯ СИСТЕМИ РОЗПІЗНАВАННЯ ЕМОЦІЙ

3.1 Технологічний стек

Для реалізації фронтенд-сервісу в межах цього проєкту було обрано фреймворк Flask, який належить до категорії так званих мікрофреймворків і відомий своєю легкістю, гнучкістю та швидкістю розгортання. Його архітектура дозволяє будувати компактні вебсервіси без зайвої складності, що особливо важливо на етапі створення MVP (minimum viable product). Flask не нав’язує суворих структур, що дає змогу адаптувати проєктну логіку під конкретні потреби, не обмежуючи розробника. Саме з цієї причини він був обраний як ідеальне рішення для створення посередницького API-шлюзу між користувацьким інтерфейсом (реалізованим у вигляді HTML/JS-сторінки) та аналітичним ядром системи.

Основним функціональним призначенням Flask у рамках даного стеку є:

- отримання запитів з фронтенду;
- обробка сесій користувача (включно з генерацією session_hash);
- маршрутизація цих запитів до головного бекенду (Django);
- а також зворотна передача результатів аналізу користувачеві у зручному форматі JSON**.**

Такий розподіл відповідальностей відповідає принципу Separation of Concerns, згідно з яким кожен компонент системи виконує чітко визначену функцію.

Аналітична частина реалізована на базі фреймворка Django, який, на відміну від **Flask**, належить до повноцінних вебфреймворків із чіткою структурою, вбудованими інструментами безпеки, валідації, авторизації та роботи з базами даних. Для побудови API в Django інтегровано Django Rest Framework (DRF) — розширення, що дозволяє легко й ефективно створювати RESTful-сервіси. Використання DRF значно спрощує процес серіалізації, обробки запитів, налаштування маршрутів та взаємодії з ORM, а також дає змогу швидко масштабувати API у разі розширення функціоналу. Django було обрано не лише

завдяки його зрілості, а й завдяки великій спільноті, якій належить активна роль у підтримці, виправленні помилок та розробці додаткових модулів.

Ключовими факторами вибору Django у цьому проєкті стали:

- чітка та логічна структура застосунків;
- інтеграція з потужною ORM (що може використовуватись паралельно з raw SQL);
- зручна розділеність конфігурацій і логіки;
- висока швидкість розробки функціоналу.

Для зберігання історії запитів, користувачів та результатів аналізу використовується реляційна база даних MySQL — одна з найпопулярніших і найстабільніших СУБД із відкритим кодом. MySQL була обрана завдяки:

- високій швидкодії при роботі з запитами;
- надійній інтеграції як із Django (через mysqlclient), так і з Flask (через SQLAlchemy);
- широкій підтримці хостингів, моніторингу та резервного копіювання;
- наявності інструментів для аналізу запитів і діагностики продуктивності.

У Flask-компоненті для взаємодії з базою використано ORM SQLAlchemy, що дозволяє абстрагуватись від синтаксису SQL і працювати з даними у вигляді об’єктів Python. У Django, натомість, реалізовано прямі SQL-запити (raw()), що дає гнучкість і контроль на рівні складних аналітичних операцій.

Для виконання нейромережевого емоційного аналізу тексту інтегровано модель cardiffnlp/twitter-roberta-base-emotion з бібліотеки HuggingFace Transformers. Ця модель є оптимізованою версією BERT, натренованою на великому корпусі емоційно маркованих даних. Вона дозволяє без додаткового донавчання розпізнавати чотири базові емоцій — anger, joy, optimism, sadness — із високою точністю та мінімальними часовими затратами на інференс. Завдяки попередньому тренуванню та відкритій документації, дана модель є ідеальним варіантом для проєктів, які вимагають швидкої інтеграції ШІ без розгортання власної інфраструктури для навчання.

Усі компоненти проєкту — фронтенд, Flask, Django, база даних, нейромережева логіка — були об’єднані в єдину систему за допомогою Docker. Використання контейнеризації забезпечує ізоляцію середовища, що дозволяє уникнути конфліктів між залежностями та бібліотеками. Завдяки Docker Compose система була розбита на кілька незалежних сервісів:

- Flask — посередник між інтерфейсом і ядром,
- Django — бекенд для обчислень та API,
- MySQL — СУБД для збереження станів,
- init-db — окремий контейнер для ініціалізації структури таблиць.

Усі ці сервіси взаємодіють між собою через внутрішню **Docker**-мережу, що забезпечує стабільність і контрольовану передачу даних. Подібний підхід значно спрощує розгортання: систему можна запустити за однією командою docker compose up, що особливо зручно при розробці, тестуванні або перенесенні на інший сервер.

Короткий огляд стеку

1) Frontend: HTML/CSS/JS (Vanilla JS + шаблони)
2) Backend #1 (Flask): приймає запити з фронтенду, керує сесією, перенаправляє до Django
3) Backend #2 (Django DRF): логіка обробки тексту, взаємодія з нейромережею та базою
4) ML-модель: cardiffnlp/twitter-roberta-base-emotion
5) База даних: MySQL
6) ORM/SQL: SQLAlchemy (Flask), Raw SQL (Django)
7) Контейнеризація: Docker + Docker Compose

3.2 Архітектура та структура проєкту

Архітектура програмного забезпечення, як і його фізична структура на рівні файлів та директорій, є важливим аспектом загальної організації проєкту. В межах реалізації системи емоційного аналізу тексту було обрано структуру, що відповідає

принципам розділення відповідальностей (Separation of Concerns) та забезпечує високу модульність, масштабованість і легкість у підтримці коду.

На рисунку 3.1 зображено загальну логічну структуру проєкту, що включає декомпозицію за функціональними блоками:

- Директорія frontend/ містить код, відповідальний за взаємодію з користувачем. Тут зосереджено HTML-шаблони, CSS-стилі та JavaScript-логіку, а також код Flask-сервера, що виступає проміжною ланкою між клієнтом і бекендом. Об’єднання Flask і фронтенду в одну директорію логічне, оскільки Flask тут виконує роль “обгортки” для клієнтського інтерфейсу.
- Директорія backend/ містить основний обчислювальний блок системи — застосунок на Django, що відповідає за реалізацію REST API, обробку запитів, виклики нейромережевої моделі та роботу з базою даних. Такий поділ дозволяє повністю ізолювати ядро бізнес-логіки від презентаційного шару.
- В окремій директорії data/ розміщуються скрипти ініціалізації, які потрібні для налаштування структури БД на початковому етапі. Наприклад, тут розміщується create_db.py, який створює необхідні таблиці, якщо вони відсутні. Це забезпечує стабільний запуск проєкту “з нуля” навіть на новому сервері без ручного втручання.

Такий поділ на логічно незалежні компоненти дозволяє:

- працювати над кожним блоком окремо без ризику порушити решту системи;
- легко тестувати функціональність модулів у ізоляції;
- змінювати або оновлювати окремі частини — наприклад, замінити модель ШІ або реалізувати новий фронтенд — без потреби торкатися всієї архітектури.

Цей підхід сприяє паралельній розробці, де одна команда може займатися клієнтською частиною, інша — API або обчисленнями. Завдяки цьому спрощується масштабування проєкту й оптимізується командна робота. Подібна структура

також дає змогу швидко інтегрувати нові технології — наприклад, у майбутньому можна буде замінити Flask на FastAPI або розбити бекенд на мікросервіси.

Окрім цього, обрана архітектура спрощує налаштування CI/CD-процесів. Кожен компонент має свою точку входу, залежності та може бути незалежно протестований і зібраний. Завдяки Docker, усі сервіси упаковані в окремі контейнери, що запускаються одночасно в єдиній мережі. Застосування Docker Compose дозволяє описати цю структуру у вигляді конфігураційного файлу docker-compose.yml, що значно полегшує розгортання проєкту.

Управління параметрами конфігурації здійснюється централізовано через файл *.env*, який містить значення для змінних середовища: паролі, порти, імена бази даних, параметри мережі тощо. Такий підхід дозволяє:

- гнучко адаптувати систему до різних середовищ (локальна розробка, staging, production);
- уникати хардкоду;
- забезпечити безпечніше зберігання конфіденційної інформації.

У результаті проєкт має чітку, логічну та відкріплену структуру, яка дозволяє не лише ефективно реалізовувати поточний функціонал, а й залишає великий простір для подальшого розширення, оптимізації та підтримки. Ця структура є добрим прикладом реалізації принципів сучасної інженерії програмного забезпечення, де з самого початку закладаються основи стабільності, гнучкості та масштабованості.

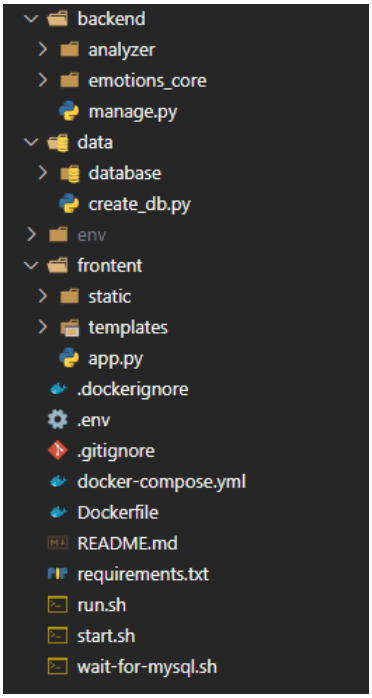


Рисунок 3.1 - Структура проєкту у середовищі розробки Visual Studio Code

3.3 Як працює фронтенд

Фронтенд частина розробленої системи є важливою складовою загальної архітектури та відповідає за **інтерактивну взаємодію з користувачем**, забезпечуючи зручний інтерфейс для введення запитів та відображення результатів емоційного аналізу. В межах обраної архітектури ця частина реалізована з використанням мікрофреймворку Flask, який у даному випадку виконує функцію посередницького логічного шару між користувацьким інтерфейсом і основною обчислювальною логікою, реалізованою у фреймворку Django.

Таке архітектурне рішення дозволяє досягти чіткого розділення відповідальностей між компонентами, що є принципом сучасної інженерії програмного забезпечення. Зокрема, фронтенд зосереджується на презентації та зборі вхідних даних, Flask займається маршрутизацією, обробкою сесій та перенаправленням запитів, а Django реалізує складну бізнес-логіку і взаємодію з базою даних та нейромережею.

Після завантаження сторінки користувач бачить веб-інтерфейс, створений за допомогою класичних вебтехнологій — HTML (структура сторінки), CSS (стилізація елементів) та JavaScript (реактивність і динаміка). Інтерфейс

складається з інтуїтивно зрозумілих елементів: текстового поля для введення повідомлення, кнопки надсилання запиту, а також блоків, призначених для відображення результатів аналізу, включно з візуалізацією емоцій, коефіцієнтів та історії запитів. Такий підхід дозволяє досягти максимальної зручності для кінцевого користувача — без перевантаження елементами, але з наявністю усієї необхідної функціональності.

При взаємодії з інтерфейсом — зокрема, коли користувач вводить повідомлення та натискає кнопку — активується клієнтський JavaScript-код (розміщений у файлі script.js). Він відповідає за формування HTTP-запиту, зокрема, POST-запиту, що містить текстове значення, введене користувачем, і передає його на серверну частину. Запит спрямовується на маршрут /process_query, який реалізований на Flask-сервері.

У свою чергу, Flask виконує роль початкового обробника запиту. Насамперед, він перевіряє наявність у користувача унікального сесійного ідентифікатора (session_hash). Цей ідентифікатор генерується автоматично у випадку, якщо його не виявлено, і зберігається на боці клієнта (наприклад, через cookie або у локальному сховищі браузера). Такий підхід дозволяє системі підтримувати контекстну взаємодію, навіть у відсутності повноцінної авторизації, тобто користувач лишається “анонімним”, але його запити логічно пов’язані між собою.

Після обробки первинної інформації, Flask формує вторинний запит — цього разу до внутрішнього API, реалізованого на базі Django REST Framework. У цьому запиті передається текст повідомлення, session_hash, а також — за необхідності — службові метадані (час, мова, джерело). Таким чином, Flask виконує роль проксі-сервісу, що не змінює дані, а лише переадресовує їх на потрібний обробник у системі.

На Django-бекенді запускається основна логіка обробки: проводиться емоційний аналіз тексту, за необхідності — з урахуванням історії попередніх запитів. На виході формується структурований JSON-об’єкт, що містить результати аналізу: основну емоцію, розподіл ймовірностей по всіх емоціях,

інтегральний емоційний “score”, а також контекстно зважений результат з урахуванням історії.

Цей JSON-об’єкт повертається до Flask, який у свою чергу відповідає клієнту. JavaScript-код на стороні користувача обробляє відповідь і динамічно оновлює вміст сторінки. Зокрема, на основі отриманих даних:

- виводиться визначений емоційний стан користувача;
- показується розподіл емоційних коефіцієнтів у вигляді таблиці, графіка або індикаторів;
- відображається історія попередніх запитів, прив’язана до конкретної сесії;
- за можливості — відображається емоційна динаміка (порівняння з попередніми результатами).

Усе це відбувається без повного перезавантаження сторінки, завдяки використанню AJAX-підходу в JavaScript. Така реалізація забезпечує кращу реактивність, знижує затримки й підвищує зручність користування системою.

Варто особливо наголосити, що Flask у цій архітектурі не виконує обчислювальних завдань. Він не проводить жодного аналізу тексту, не взаємодіє з моделлю ШІ й не зберігає дані у базі. Його роль зводиться до:

- обробки HTTP-запитів;
- створення або зчитування сесій;
- формування проксій-запитів до Django;
- повернення відповіді назад у фронтенд.

Це рішення дозволяє уніфікувати точку входу у систему, зробити її більш керованою, а також забезпечити кращу роздільність логіки, що суттєво полегшує супровід, масштабування, дебагінг та тестування окремих компонентів. Такий розподіл є типовим для систем, які будуються з прицілом на подальше мікросервісне або багатошарове розгортання, з можливістю незалежного масштабування фронтенду, шлюзу й обчислювального ядра.

3.4 Як працює бекенд

Попри те, що саме Flask виступає технічною точкою входу для запитів, які надходять із фронтенду, основна обчислювальна логіка проєкту реалізована у Django — потужному високорівневому фреймворку для розробки вебзастосунків, який у цій архітектурі виконує функцію центрального бекенд-компонента, відповідального за реалізацію бізнес-логіки, управління даними, запуск штучного інтелекту та формування структурованої відповіді для користувача.

Django обрано не випадково — цей фреймворк не лише забезпечує надійний базис для побудови масштабованих систем, але й має високу гнучкість у роботі з БД, підтримку REST API через DRF, а також зручну інтеграцію з модулями машинного навчання. У межах нашої системи саме Django реалізує логіку, пов’язану з емоційним аналізом тексту, збереженням запитів, ідентифікацією користувачів та контекстною обробкою історії взаємодій.

Коли користувач формує повідомлення через клієнтський інтерфейс і надсилає його, цей запит передається спершу на Flask-сервер, який, після первинної обробки, формує внутрішній виклик до API Django. Цей запит надсилається на маршрут /api/process_query — спеціально створену ендпоінт-функцію, що відповідає за глибоку обробку текстових даних. У межах запиту передаються усі ключові параметри:

- текст самого повідомлення,
- сесійний ідентифікатор session_hash,
- а також службові метадані, якщо вони необхідні для додаткової логіки (наприклад, тип запиту, час доби, тощо).

На стороні Django початковим кроком є ідентифікація користувача. Система здійснює пошук у таблиці користувачів за значенням session_hash. Якщо такого користувача не знайдено, то автоматично створюється новий запис у БД. Цей підхід дозволяє підтримувати індивідуальність сесії, навіть у разі відсутності повноцінної авторизації — тобто зберігається анонімність, але не втрачається зв’язок між діями

користувача. Це важливо для аналізу послідовності емоційних станів у часі та формування персоналізованого контексту.

Далі запит передається до основної логіки — функції analyze_emotions(), яка виконує базовий аналіз вхідного тексту. На цьому етапі система працює з повідомленням у "чистому" вигляді, без урахування історії попередніх звернень. Текст токенізується, обробляється через модель cardiffnlp/twitter-roberta-base-emotion, і на виході формується набір логітів, які конвертуються у ймовірності за допомогою Softmax. Кожне значення відповідає певній емоційній категорії (anger, joy, optimism, sadness), і модель повертає набір коефіцієнтів, що визначають інтенсивність кожної з емоцій. Найбільше значення вважається домінуючою емоцією повідомлення.

Після цього система виконує додатковий поглиблений аналіз, який реалізовано у функції analyze_with_history(). На цьому етапі система звертається до таблиці query_history, де зберігається повна хронологія попередніх запитів поточного користувача. Витягнувши останні N записів, система виконує агрегацію історичних даних — обчислює середні значення емоцій, виводить динаміку та зважує поточний результат з урахуванням попереднього емоційного фону. Таким чином формується більш точна, контекстно обґрунтована оцінка, яка враховує не лише окреме повідомлення, а й емоційний стан у динаміці.

У результаті система формує два паралельні результати:

- Чистий результат — базовий аналіз поточного повідомлення без урахування попередніх дій.
- Історичний результат — поглиблена оцінка з урахуванням попередніх запитів користувача.

Крім того, відповідь містить також масив з історією запитів, що дозволяє відображати її у клієнтському інтерфейсі та виводити наочні графіки або списки. Вся ця інформація інкапсулюється в структурований JSON-об’єкт, який включає:

- поточну емоцію,
- розподіл імовірностей,
- історію попередніх запитів,

- часові мітки та службові параметри.

Одночасно з формуванням відповіді, новий запит також додається до бази у таблицю query_history, де зберігається:

- час запиту,
- ідентифікатор користувача,
- отриманий розподіл емоцій,
- категорія повідомлення (якщо передбачено класифікацію),
- значення інтегрального емоційного “score”.

Формування відповіді завершується передачею об’єкта назад на сторону Flask, який, не змінюючи структури або змісту JSON, просто переправляє її на фронтенд. Це завершує цикл взаємодії між користувачем і обчислювальним ядром системи.

Такий поділ відповідальностей, де Flask виконує функцію маршрутизатора, а Django — обчислювача та координатора логіки, дозволяє побудувати гнучку, розширювану й стабільну систему, у якій кожен компонент виконує чітко визначену роль.

3.5 Як працює нейромережевий аналіз. Інтеграція баз даних

3.5.1 Нейромережевий аналіз

У центрі інтелектуального ядра системи емоційного аналізу тексту розташована попередньо навчена трансформерна модель cardiffnlp/twitter-roberta-base-emotion. Ця модель належить до класу нейронних мереж, створених на базі архітектури BERT (Bidirectional Encoder Representations from Transformers) — однієї з найпотужніших та найвпливовіших моделей для обробки природної мови (NLP), розробленої лабораторією Google AI.

Коротка довідка про BERT

BERT — це двонаправлена трансформерна модель, що означає її здатність аналізувати контекст слів одночасно зліва направо і справа наліво. Така особливість дозволяє їй не просто зчитувати значення слова, а розуміти його

семантичний сенс у конкретному контексті. Наприклад, слово “світлий” у реченні “світлий настрій” і “світлий день” може мати різне емоційне забарвлення — і модель здатна це враховувати.
BERT і його спрощені варіації (така як RoBERTa) стали стандартом для завдань класифікації, семантичного аналізу, пошуку схожих текстів та, зокрема, виявлення емоційного стану в коротких і довгих фрагментах тексту.

Модель імпортується та використовується за допомогою бібліотеки HuggingFace Transformers, яка вважається галузевим стандартом у сфері відкритих рішень для трансформерних моделей. Ця бібліотека надає зручний інтерфейс для роботи з численними моделями, зокрема з RoBERTa, що використовується у проєкті. За основу взято модель cardiffnlp/twitter-roberta-base-emotion, яка була спеціально натренована на великій кількості коротких текстів із Twitter для класифікації емоцій.

Після отримання текстового запиту від користувача система виконує етап токенізації — текст перетворюється у формат, придатний для обробки трансформером. Для цього використовується спеціальний токенізатор:

```
inputs = tokenizer(text, return_tensors="pt", truncation=True)
```

Рисунок 3.2 – Токен для обробки тексту

У результаті формується **тензор** — багатовимірна числова структура, яка передає змістовну структуру тексту. Кожне слово або його частина кодується у вигляді унікального числового представлення відповідно до словника моделі.

Ці дані передаються до входу моделі, яка генерує **логіти** — “сирі” числові активації емоцій. Для подальшого аналізу логіти нормалізуються за допомогою функції Softmax, яка перетворює їх на набір ймовірностей. Це дозволяє визначити, з якою впевненістю модель “бачить” ту чи іншу емоцію в тексті.

На відміну від більш класичних моделей, ця трансформерна архітектура навчається розпізнавати такі емоції:

- Anger — Злість
- Joy — Радість

- Optimism — Подив
- Sadness — Смуток

Ці емоції є результатом дообробки вихідних класів моделі **CardiffNLP**, яка за замовчуванням працює з англомовними твітами, але може бути адаптована для ширшого спектру текстів. З чотирьох значень найбільша ймовірність визначає домінуючу емоцію, яка стає основною міткою емоційного стану повідомлення.

Крім того, обчислюється **емоційний score** — показник інтенсивності емоції. Він дозволяє оцінити не лише наявність емоції, а й її силу, що особливо корисно для повідомлень із низьким або навпаки — надмірним емоційним навантаженням.

Якщо увімкнено режим аналізу з урахуванням історії, система додатково звертається до бази даних, де зберігаються попередні повідомлення користувача. Зчитавши останні N запитів, вона:

1. Обчислює середні емоційні значення;
2. Порівнює їх із поточним результатом;
3. Формує зважений емоційний профіль.

Цей підхід дозволяє виявляти тренди у зміні емоційного стану, що є ключовим для систем моніторингу психологічного благополуччя. Модель не лише фіксує емоцію «тут і зараз», а й допомагає зрозуміти, чи є ця емоція тимчасовою реакцією чи стабільною тенденцією, що наближає аналіз до рівня людської інтуїції.

3.5.2. База даних

Фундаментом збереження даних у проєкті виступає реляційна система управління базами даних (СУБД) — зокрема, MySQL, яка виконує роль основного довготривалого сховища інформації. Уся система персоналізації, контекстної аналітики та відтворюваності емоційного аналізу базується на здатності зберігати й обробляти історичні дані, що належать до конкретного користувача. Саме тому обрана СУБД повинна бути продуктивною, масштабованою та надійною, із підтримкою складних SQL-запитів, транзакцій і механізмів узгодження даних.

MySQL — одна з найпоширеніших і найстабільніших систем з відкритим кодом, яка зарекомендувала себе в тисячах високонавантажених проєктів. Вона

відмінно підтримується сучасними вебфреймворками, включаючи як Django (через mysqlclient або PyMySQL), так і Flask (через ORM SQLAlchemy), що робить її логічним вибором для побудови сумісної архітектури.

У проєкті реалізовано дві основні сутності, які зберігаються у базі даних і складають логічну основу всієї взаємодії користувача із системою:

Таблиця users

Ця таблиця відповідає за ідентифікацію кожного користувача. Для цього використовується поле session_hash, яке виконує роль унікального ідентифікатора сесії. Цей підхід дозволяє:

- підтримувати псевдоанонімну взаємодію, без обов'язкової реєстрації;
- зберігати контекст взаємодії, навіть коли користувач повертається до сервісу з того самого пристрою;
- розмежовувати дані між різними сесіями, не вдаючись до складних механізмів авторизації.

Така реалізація є зручною у випадках, коли важливе збереження історії звернень, але немає потреби в реальному акаунті чи персональних даних користувача.

Таблиця query_history

Це ключова таблиця для збереження результатів емоційного аналізу. Вона накопичує всю історію запитів, здійснених кожним користувачем, включаючи як поточні повідомлення, так і їх контекстні метадані. Для кожного запису фіксуються:

- час запиту (timestamp);
- емоційні результати — набір ймовірностей за чотирма емоціями;
- основна (домінуюча) емоція;
- інтегральний емоційний "score" — загальний індекс інтенсивності;
- емоційне охоплення (scope) — кількість емоцій, що були виявлені з помітною ймовірністю;
- зв'язок із користувачем — зовнішній ключ до таблиці users.

У майбутньому, за потреби, до цієї таблиці можуть бути додані додаткові поля — наприклад, категорія повідомлення, мова, геолокація або тег контексту (позитивний/негативний запит, службова команда, тощо).

Роль бази в емоційному аналізі. На відміну від класичних одноразових інтерфейсів, які обробляють лише поточний запит, розроблена система передбачає глибинну аналітику з урахуванням історії. Саме база даних є тією інфраструктурною основою, яка дозволяє:

- формувати емоційний профіль користувача у динаміці;
- виводити середні значення емоцій за певний період;
- відстежувати тенденції та циклічність у настрої;
- аналізувати контекст змін — наприклад, послідовності "злість → смуток → апатія";
- фільтрувати запити за категоріями, фреймами або скоупами.

Також БД дає змогу реалізувати функціонал візуального графіку змін емоцій у часі, що відкриває шлях до побудови інтерфейсів на кшталт “емоційного щоденника” або дашборду самоспостереження.

3.6 Docker — для чого і як працює

Уся система Emotions Tracking Software реалізована із використанням технології контейнеризації, зокрема за допомогою платформи Docker, яка на сьогодні є одним із найпоширеніших інструментів у галузі DevOps та сучасної програмної інженерії. Застосування Docker дозволяє створити ізольоване, незалежне середовище для кожного компонента системи, що значно підвищує стабільність роботи, відтворюваність результатів та знижує залежність від операційної системи, драйверів і версій бібліотек.

Docker забезпечує стандартизовану упаковку застосунку разом з усіма його залежностями, конфігураційними файлами, службовими скриптами та службовим середовищем. У результаті формується так званий контейнер, який є легковаговою віртуалізованою одиницею виконання, що працює ізольовано, але без створення повноцінної віртуальної машини. Завдяки цьому запуск програмного забезпечення

в різних середовищах (на локальному комп’ютері, на сервері, у хмарі тощо) відбувається однаково — що повністю усуває класичну проблему “у мене працює, а в тебе — ні”.

- Архітектура розгортання
- Усі основні компоненти системи, а саме:
- база даних MySQL,
- бекенд на Django,
- проміжний сервер на Flask,
- сервіс ініціалізації бази даних (init-db)

Розгортаються як окремі незалежні контейнери, які керуються через конфігураційний файл docker-compose.yml (Див. рисунок 1.2). Цей файл визначає взаємозв’язки між сервісами, порядок їх запуску, параметри мережі, проброс портів, змінні середовища тощо. Таким чином, Docker не лише створює контейнер для кожного модуля, а й забезпечує їх координацію у межах єдиного цілісного середовища.

Усі сервіси взаємодіють між собою через внутрішню мережу Docker, яка створюється автоматично при запуску. Це дозволяє сервісам обмінюватися даними за допомогою логічних імен контейнерів, що спрощує конфігурацію (наприклад, замість IP-адреси база доступна як mysql).

Порядок запуску та логіка ініціалізації. Під час запуску системи спершу активується контейнер MySQL, у межах якого задаються параметри конфігурації, зокрема змінна MYSQL_ROOT_PASSWORD, що відповідає за встановлення початкового адміністративного паролю. Після готовності бази запускається контейнер init-db, завданням якого є одноразова ініціалізація структури БД. Скрипт create_db.py, що виконується в цьому контейнері, створює необхідні таблиці (наприклад, users, query_history) та виконує первинну підготовку до взаємодії з іншими модулями.

Цей контейнер має ефемерний характер: після виконання скрипта його робота припиняється, адже подальше існування ініціалізаційної логіки є

недоцільним. Така ізоляція дозволяє уникати конфліктів між службовими та основними контейнерами та спрощує процес супроводу системи.

Після цього послідовно активуються:

- бекенд-компонент Django, який відкриває порт 8000 і очікує на API-запити від Flask;
- фронтенд-проксі на Flask, який працює на порту 8080 і приймає запити користувача через браузер.

Усі ці сервіси керуються через єдиний файл docker-compose.yml, що дозволяє запустити або перезапустити всю архітектуру єдиною командою, без потреби ручного налаштування окремих елементів.

Збереження даних, мережа, логування. Для збереження даних між сесіями використовується спеціальний Docker-том mysql_data, який дозволяє уникнути втрати інформації навіть у разі зупинки або перезапуску контейнера MySQL. Це забезпечує персистентність інформації — фундаментальну вимогу до будь-якої аналітичної системи.

Файл docker-compose.yml також визначає мережеву взаємодію між контейнерами, забезпечуючи ізоляцію від зовнішнього середовища та безпеку. При потребі можна задавати проброс портів, налаштовувати зовнішній доступ або змінювати поведінку контейнера під час перезапуску.

Крім того, Docker надає зручні інструменти для моніторингу та аналізу журналів логів. Наприклад, команда docker compose logs -f дозволяє відстежувати в реальному часі, як система реагує на запити, де виникають помилки або як веде себе кожен сервіс під навантаженням.

Автоматизація та переваги. Завдяки Docker, розгортання системи стало максимально простим: будь-який розробник або DevOps-інженер може відтворити повну інфраструктуру за кілька хвилин, не витрачаючи час на встановлення Python, бібліотек, драйверів чи налаштування середовища. Це значно знижує поріг входу, мінімізує кількість помилок під час розгортання і дозволяє зосередитися на логіці системи.

Ще одна перевага — кросплатформеність: незалежно від того, працює користувач на Windows, Linux чи macOS, запуск відбуватиметься однаково, оскільки Docker стандартизує середовище виконання. Такий підхід особливо вигідний для командної розробки, коли у кожного члена команди різна конфігурація системи.

Отже, у рамках даного проєкту Docker виступає не просто засобом запуску застосунку, а стратегічним елементом архітектури, який:

- забезпечує ізоляцію, повторюваність і надійність;
- дозволяє централізовано керувати сервісами;
- підтримує гнучку масштабованість та легке оновлення окремих компонентів без зупинки всієї системи.

Його використання суттєво покращує якість життєвого циклу програмного забезпечення, полегшує CI/CD (безперервну інтеграцію та деплоймент), сприяє автоматизації процесів і створює умови для подальшого масштабування платформи без суттєвих змін у кодовій базі чи інфраструктурі.

```yaml
version: '3.9'

services:
  mysql:
    image: mysql:8
    ports:
      - "3307:3306"
    environment:
      MYSQL_ROOT_PASSWORD: "0000"
    volumes:
      - mysql_data:/var/lib/mysql
    healthcheck:
      test: ["CMD", "mysqladmin", "ping", "-h", "localhost"]
      interval: 5s
      timeout: 3s
      retries: 10


  init-db:
    build:
      context: .
      dockerfile: Dockerfile
    depends_on:
      mysql:
        condition: service_healthy
    env_file: .env
    command: ["sh", "./wait-for-mysql.sh"]
    working_dir: /

  backend:
    build:
      context: .
      dockerfile: Dockerfile
    depends_on:
      - mysql
      - init-db
    ports:
      - "8000:8000"
    env_file: .env
    command: ["python", "manage.py", "runserver", "0.0.0.0:8000"]
    working_dir: /backend

  frontend:
    build:
      context: .
      dockerfile: Dockerfile
    depends_on:
      - backend
    ports:
      - "8080:8080"
    env_file: .env
    command: ["python", "app.py"]
    working_dir: /frontent

volumes:
  mysql_data:

```

Рисунок 3.3 - Головний конфігураційний docker-файл - docker-compose.yml

3.7 Тестування системи

Перед введенням системи в експлуатацію, а також із метою підтвердження її функціональної коректності, було проведено тестування ключових компонентів

розробленого застосунку. Особливу увагу зосереджено на перевірці працездатності та коректності роботи механізму емоційного аналізу тексту, реакції системи на типові та граничні сценарії, а також загальної стабільності вебінтерфейсу й взаємодії між модулями.

Зміна нейромережевої моделі з distilbert-base-uncased-emotion на cardiffnlp/twitter-roberta-base-emotion стала відповіддю на виявлені обмеження початкового рішення. Попри те, що модель DistilBERT мала певні переваги в швидкості обробки, її якість результатів у конкретних кейсах виявилася недостатньо стабільною. Модель cardiffnlp/twitter-roberta-base-emotion, у свою чергу, має кращу репрезентативність для коротких, неформальних текстів, наближених до тих, які використовуються в чатах, листуванні або соціальних мережах. Її попереднє навчання на великому корпусі твітів дозволило досягти вищої точності та релевантності емоційної класифікації, що зробило її оптимальним вибором для поточної задачі.

Процедура тестування охоплювала як позитивні сценарії (тобто очікувані, звичайні умови роботи системи), так і негативні (нестандартні, граничні, потенційно проблемні запити). Для кожного тесту фіксувалися вхідні дані, очікувана реакція системи, фактична відповідь, а також супровідні коментарі щодо поведінки на фронтенді й у системних логах.

Особливу увагу приділено таким аспектам:

- адекватності класифікації емоцій у межах нової моделі (якість, стабільність, відповідність змісту);
- роботі механізму session_hash і збереженню історії запитів;
- правильності формування відповіді з урахуванням або без урахування історії;
- відображенню результатів на фронтенді (без помилок рендерингу або втрати даних);
- реакції на порожні або некоректні запити;
- збереженню даних у базі при повторних зверненнях;

- стабільності системи при кількох запитах поспіль та після перезапуску контейнерів.

Подані нижче скріншоти і таблиці демонструють результати тестових сценаріїв, супроводжуються коментарями щодо очікуваної та фактичної поведінки системи, і дають змогу об'єктивно оцінити рівень її готовності до практичного використання.

1. Перш за все, при запуску системи активується смоук-тест — тридцятиразова перевірка роботи бекенду (Рис. 3.4). В разі успіху термінал можна закривати, якщо ж буде провал — треба перезапустити процес.

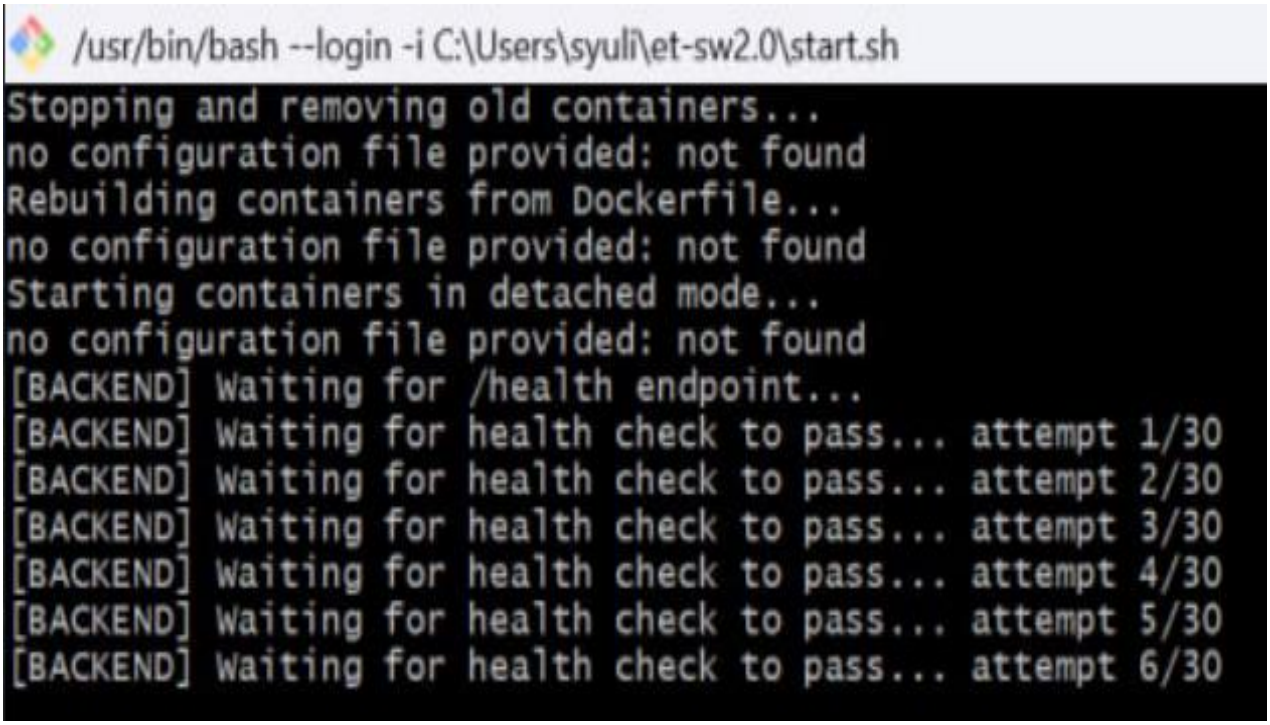


Рисунок 3.4 - “Health check’и бекенду”

2. Перевіряємо позитивні інпути, кожен по 3 рази.
3. Почнемо зі смутку (sadness):

Текст: “Соня порвала зі мною вчора”. Очікуваний та фактичний результат збігаються, ШІ трактував переважаючу чисту емоцію в тексті як смуток. Бінарний статус емоції: негативна (рис. 3.5).

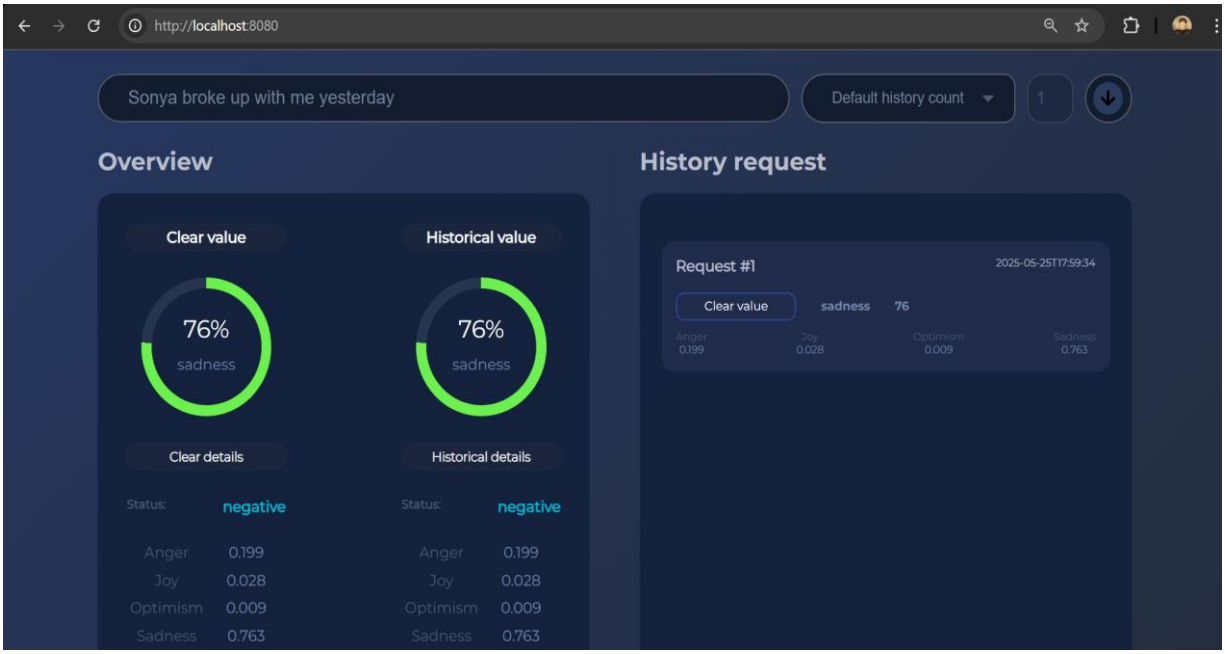


Рисунок 3.5 - Перша перевірка правильності інтерпретації емоції смутку.

Текст: “Я почуваюся таким самотнім. Я навіть не маю з ким поговорити”. Результат: 98% sadness. Статус емоції: негативний (рис. 3.6).

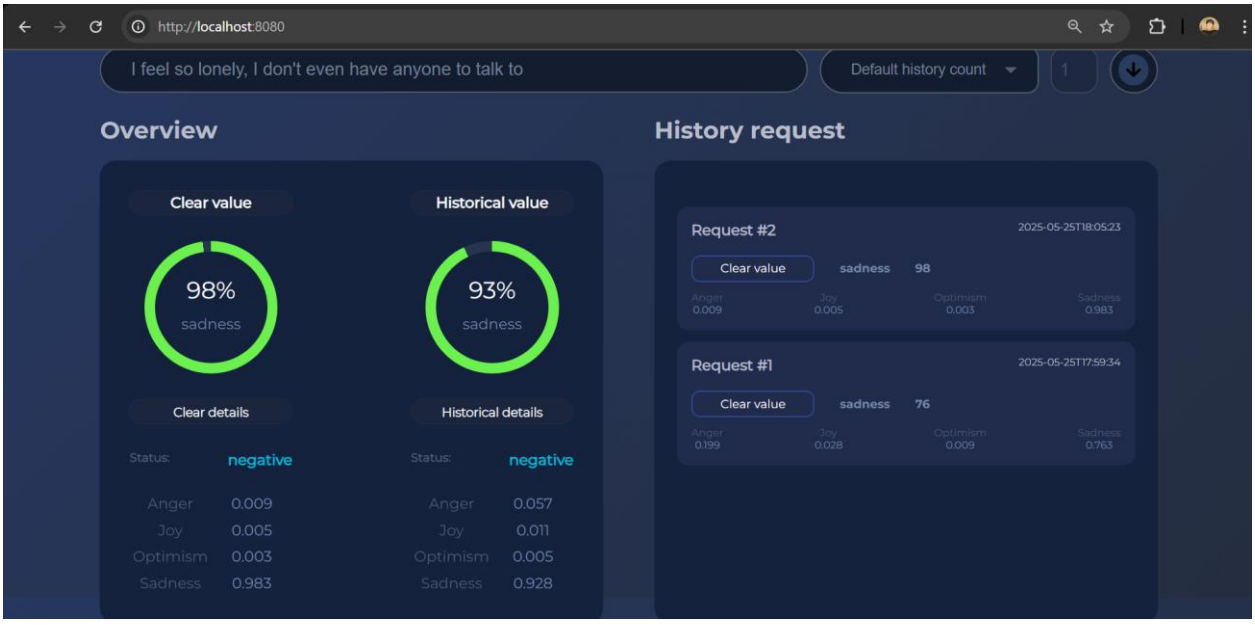


Рисунок 3.6 - Друга перевірка.

Текст: “Я некрасивий”. Чисте значення домінуючої емоції: 77% сум. Статус: негативний (рис. 3.7).

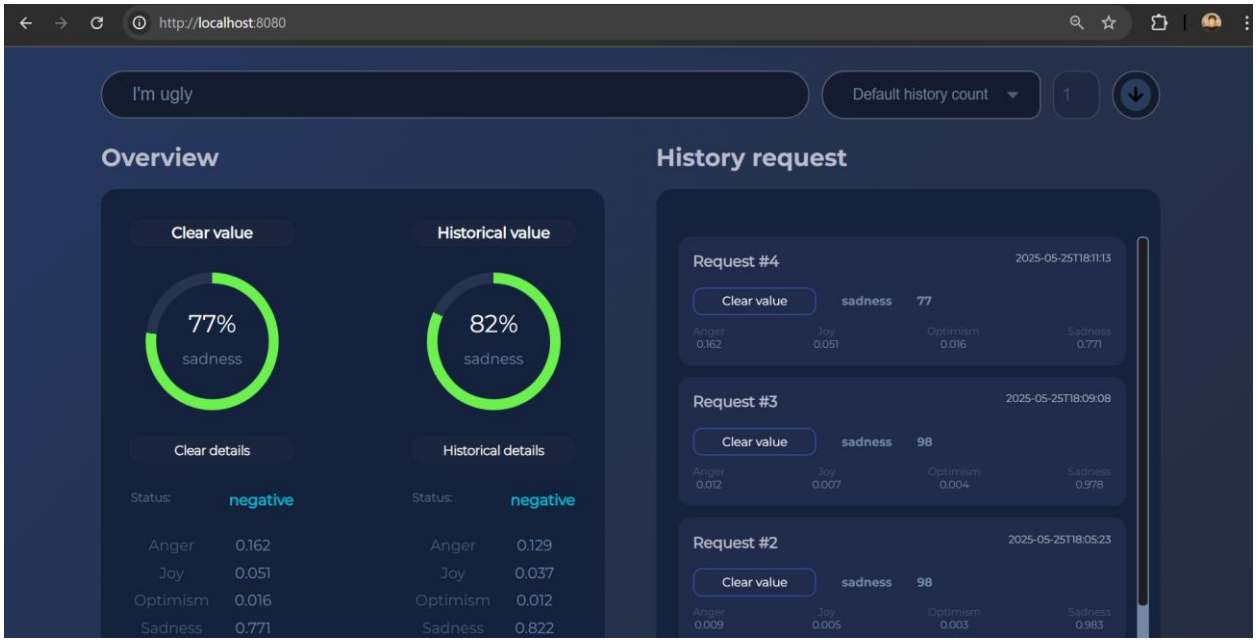


Рисунок 3.7 - Третя перевірка.

Тепер перевіримо оптимізм.

Текст: “Вище ніс, хлопче, все буде добре!” Результат: 76% оптимізм. Бінарний статус: позитивний. Тест пройдено (рис. 3.8).

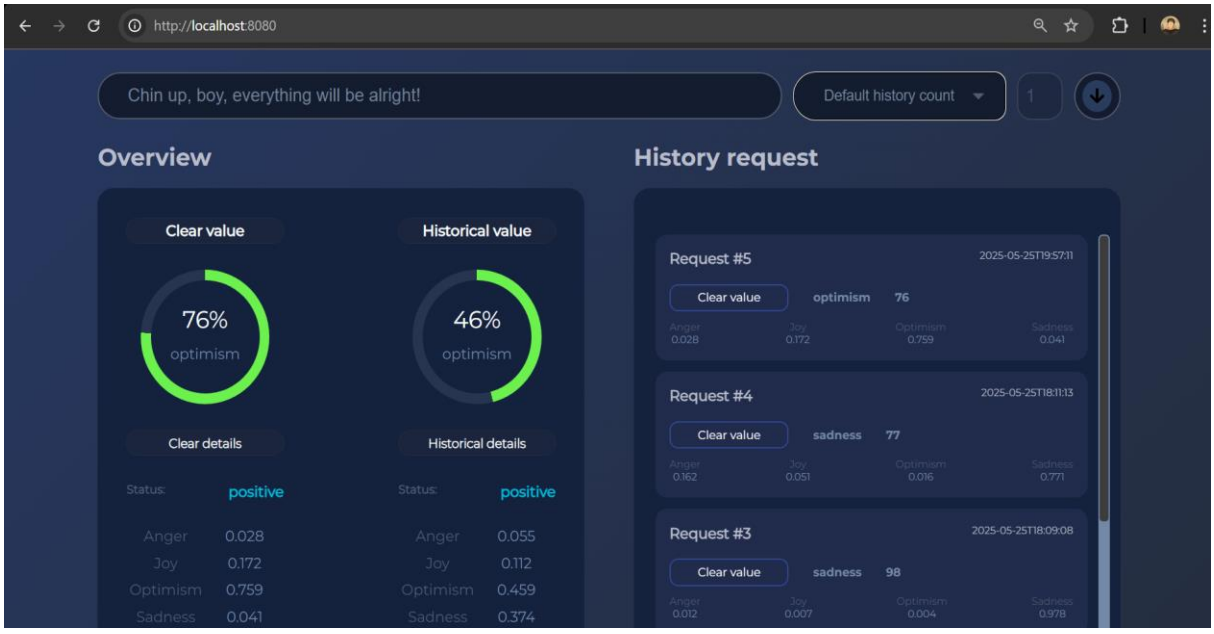


Рисунок 3.8 - Перша перевірка на оптимізм.

Текст: “Я не здамся, доки не досягну своєї мети”. 92% оптимізму, статус емоції: позитивний (рис. 3.9).

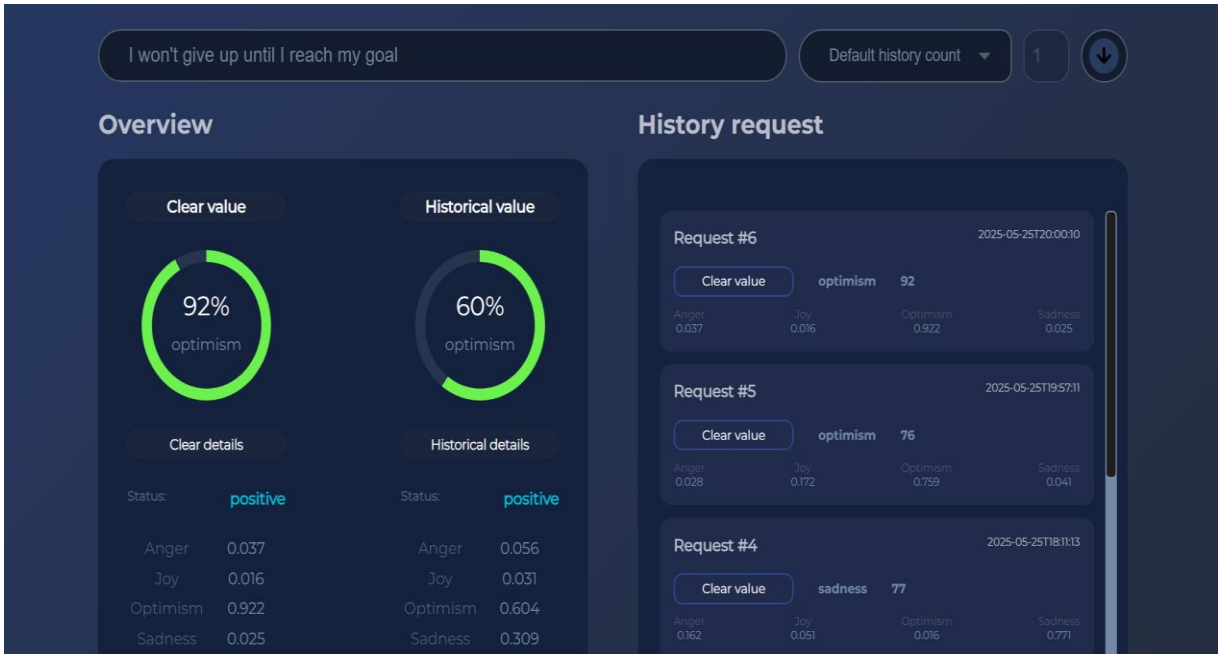


Рисунок 3.9 - Друга перевірка.

Текст: “Лікар сказав, що мої шанси на відовлення дуже високі. Звучить багатообіцяюче”. Показник оптимізму, що складає 88% та позитивний бінарний статус свідчать про те, що цю перевірку ШІ також пройшов (рис. 3.10).

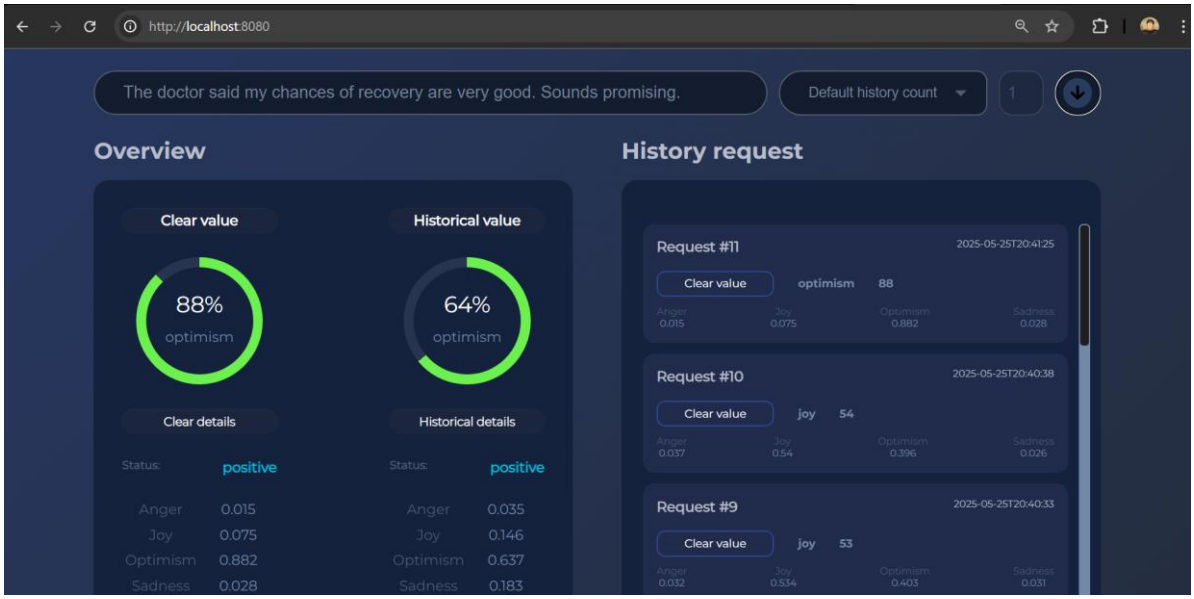


Рисунок 3.10 - Третя перевірка.

Наступний на черзі гнів.

Текст: “Я поб’ю тебе, якщо ти не замовкнеш прямо зараз”. Відсоток гніву – 97%. Бінарний статус: негативний (рис. 3.11).

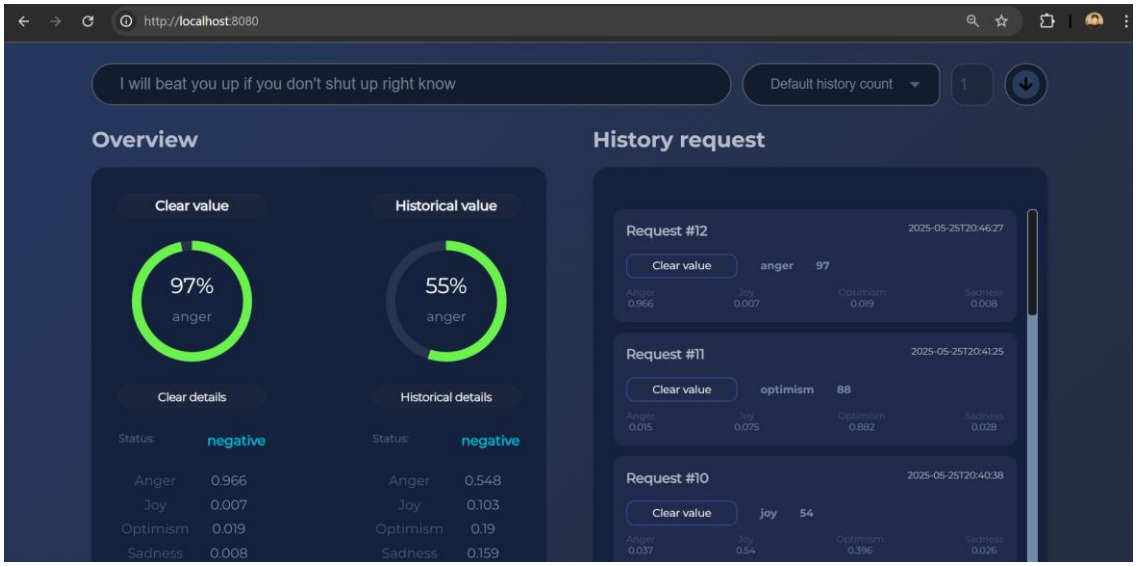


Рисунок 3.11 - Перший тест визначення злості.

Текст: Повір мені, якщо сьогодні ти знову запізнишся на роботу, то ти зіштовхнешся з такими наслідками, які ти навіть не міг уявити. 91% гніву, статус емоції: негативний (рис. 3.12).

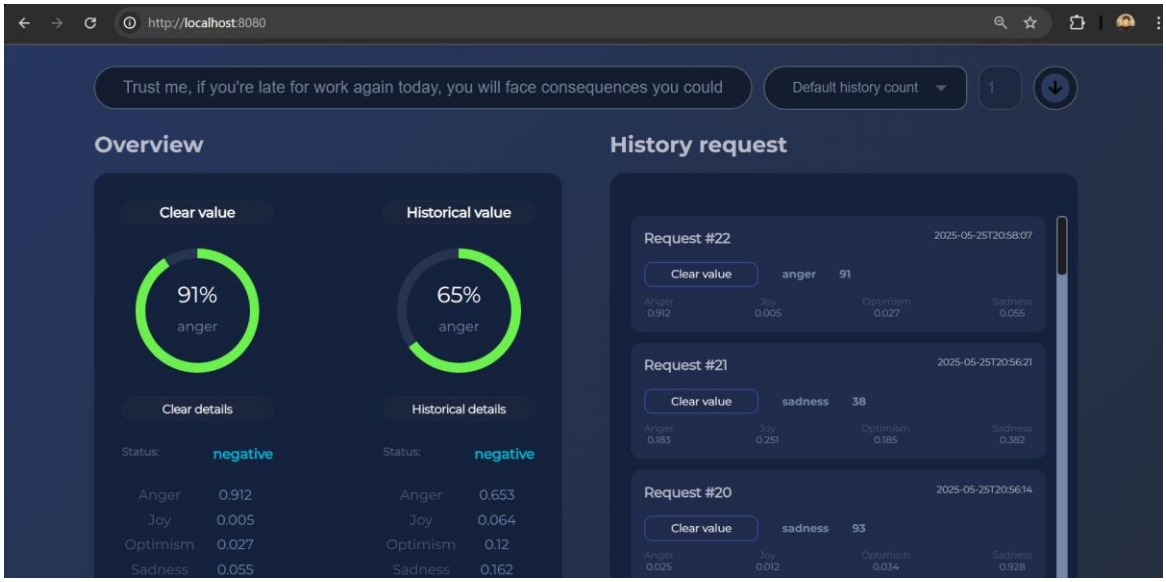


Рисунок 3.12 - Друга перевірка.

Текст: “Перестань гратися і зроби щось нарешті” Показник гніву в 83% та негативний бінарний статус свідчать про те, що цей тест також успішно пройдено (рис. 3.13).

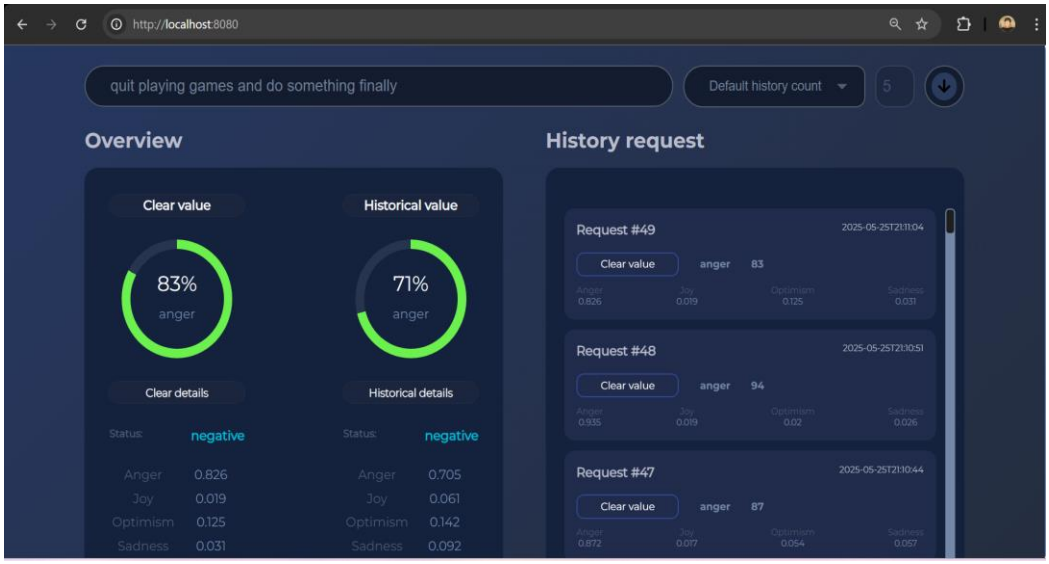


Рисунок 3.13 - Третій тест.

Остання емоція: радість.

Текст: “Я щойно виграв у лотерею” Результат: 82% радості, статус емоції: позитивний. Тест успішно пройдено (рис. 3.14).

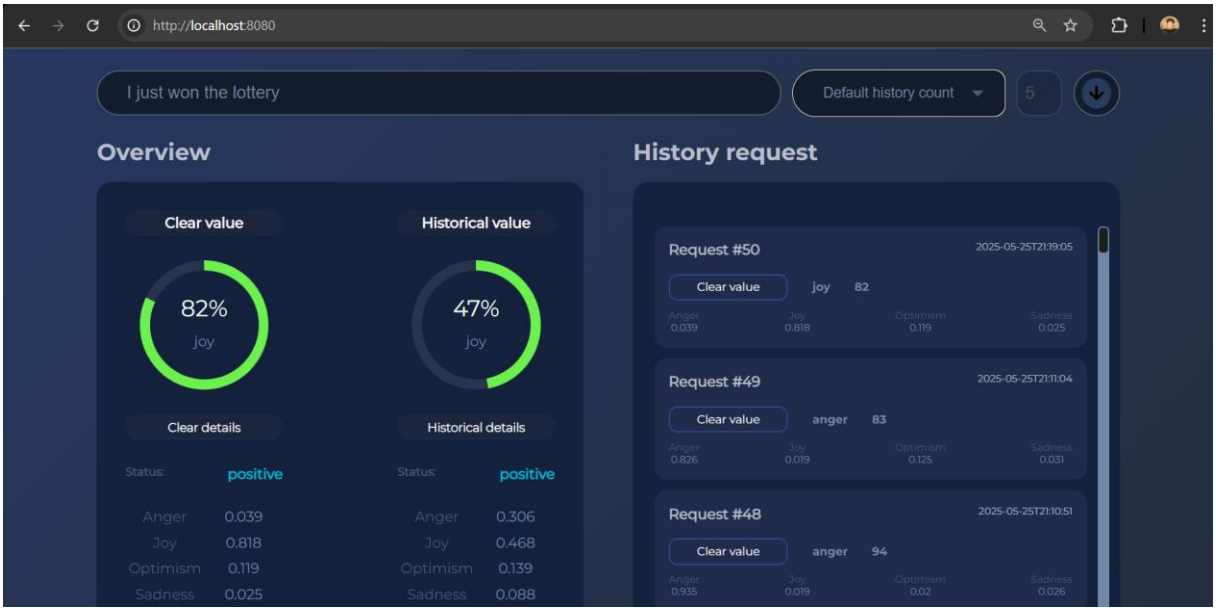


Рисунок 3.14 - Перша перевірка інтерпретації радості

Текст: “Мені б хотілося погладити це маленьке цуценя” 89% радості та позитивний статус емоції (рис. 3.15).

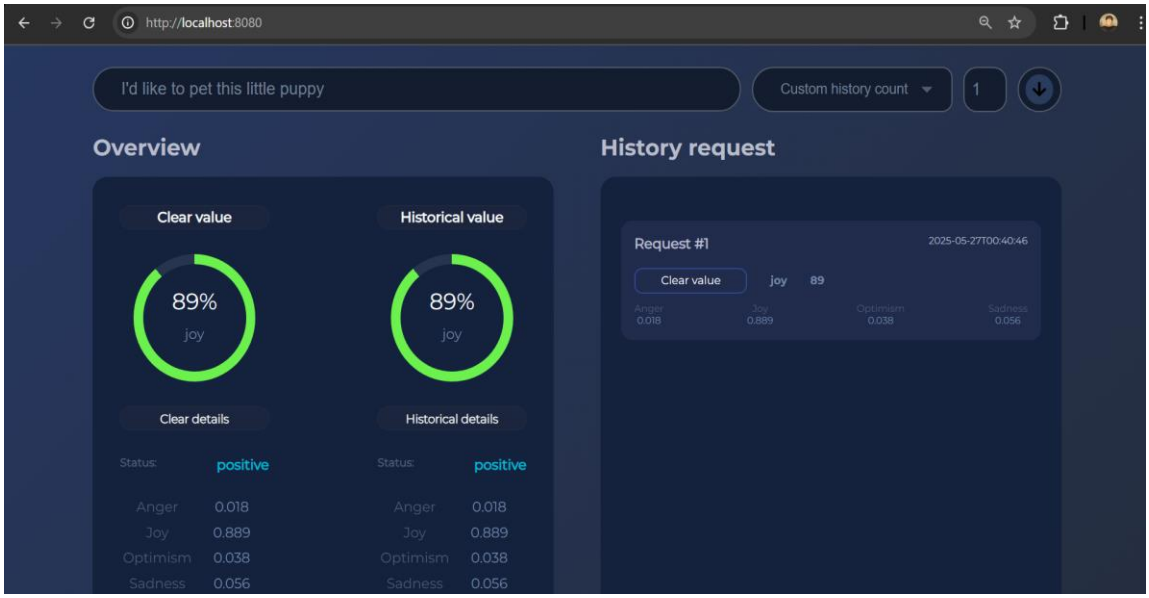


Рисунок 3.15 - Друга перевірка

Текст: “Дякую за такий чудовий подарунок, Артуре” Показник радості: 94%. Статус емоції: позитивний (рис. 3.16).

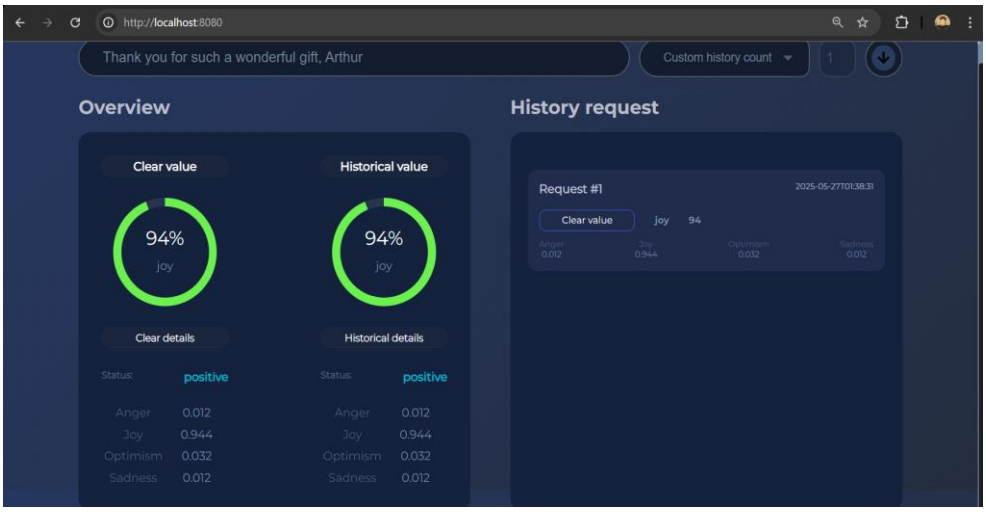


Рисунок 3.16 - Третя перевірка

З позитивних запитів варто також перевірити реакцію моделі на текст без жодного емоційного забарвлення.

Текст: “Джим”. Як видно зі скріншоту, якоїсь однієї абсолютно домінуючої емоції (більше 50%) тут немає. Воно й не дивно, адже це просто ім’я, яке саме по собі не несе в собі ані злості, ані радості, оптимізму чи суму. З наведеного нижче скріншоту видно, що ШІ це розуміє (рис. 3.17).

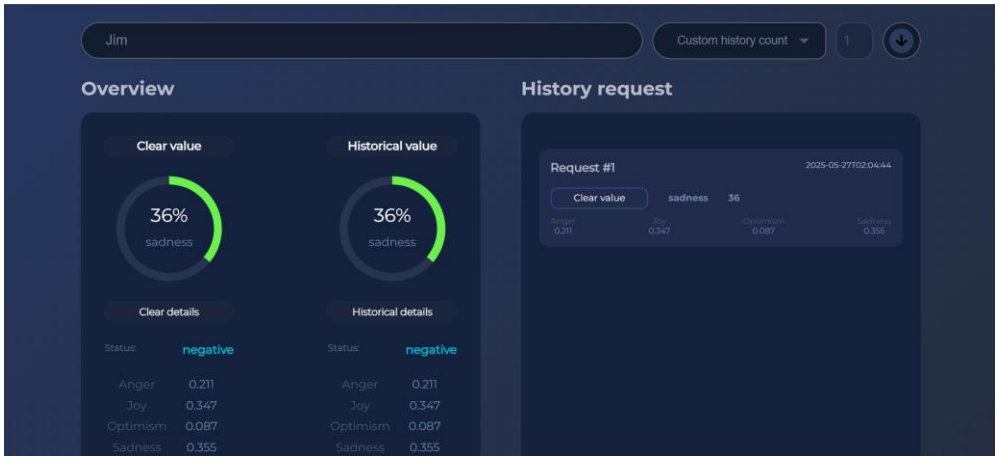


Рисунок 3.17 - Перша перевірка реакції моделі на фразу з нейтральним емоційним забарвленням

Текст: “Спускайся, хлопче”. Спостерігаємо ідентичну ситуацію, що й в минулому випадку. Відштовхуючись лише від цих слів, зробити висновок про емоцію, з якою вони використовуються, не є можливим, тому домінуюча емоція відсутня (рис. 3.18).

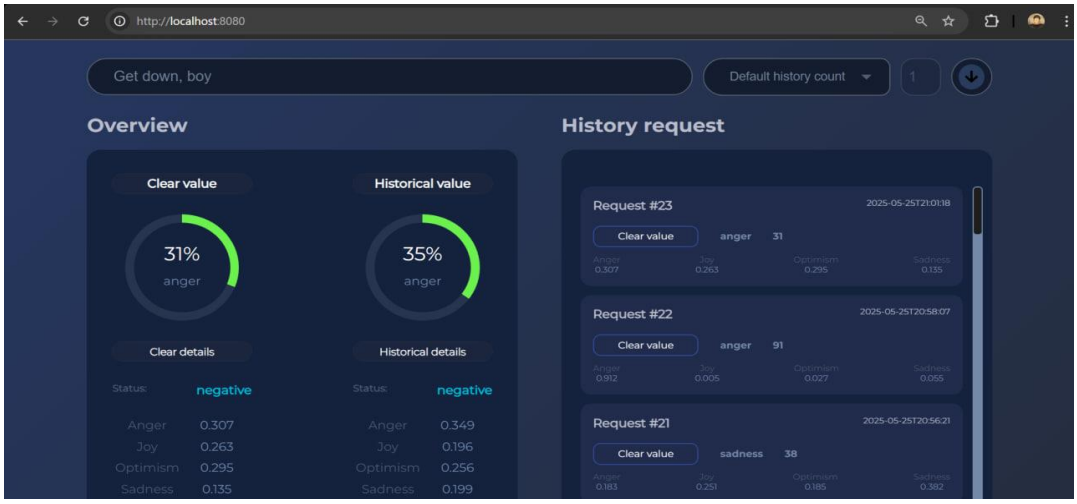


Рисунок 3.18 - Друга перевірка

Текст: “Нью-йорк не є столицею Сполучених Штатів Америки”. Маємо факт, що не виражає жодну з емоцій у вакуумі. Результат цієї перевірки ідентичний двом попереднім. ШІ, як і очікувалося, не виявив якоїсь основної переважаючої емоції (рис. 3.19).

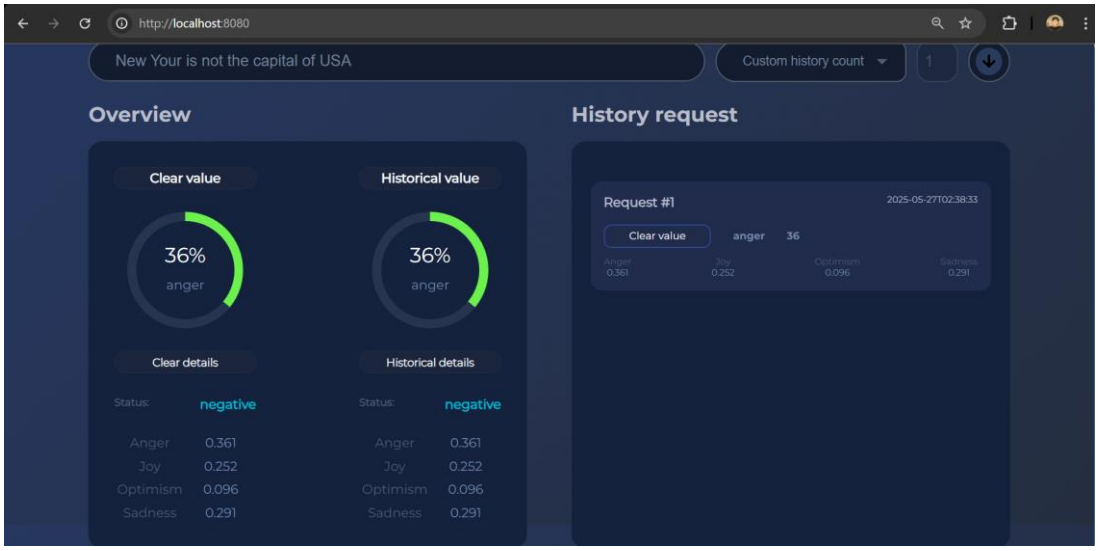


Рисунок 3.19 - Третя перевірка

Після успішного проходження “позитивних тестів” не завадить також провести й “негативний”. Якщо ми, скажімо, залишимо поле вводу пустим і спробуємо натиснути на кнопку, що запускає інтерпретацію емоцій, то сервіс повідомить нам про те, що ми нічого не ввели та попросить ввести запит.

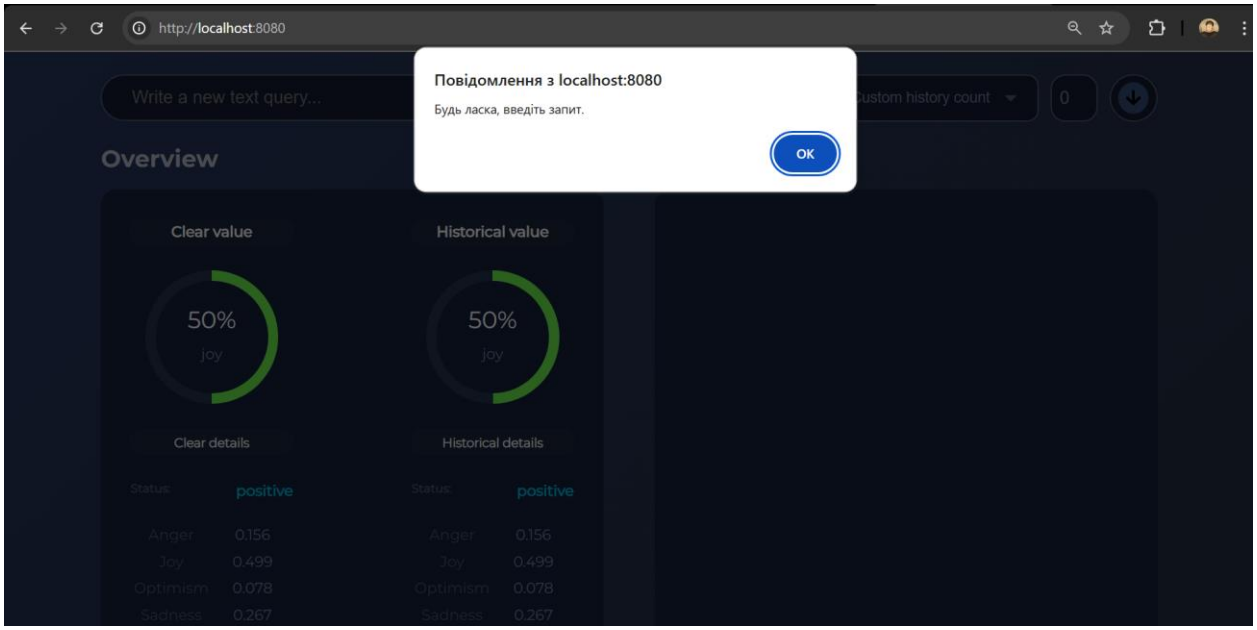


Рисунок 3.20 - Перевірка реакції системи на відправку порожнього запиту.

3.8 Узагальнення результатів тестування:

Тестування інформаційної системи емоційного аналізу мало на меті підтвердження її працездатності, логічної цілісності та готовності до практичного застосування. Основну увагу було приділено перевірці коректної взаємодії між

усіма компонентами архітектури, стабільності інтерфейсу, відповідності результатів очікуванням користувача, а також загальному функціонуванню системи в умовах типового використання.

На етапі розробки було прийнято рішення замінити початкову модель емоційної класифікації на більш релевантну задачі — cardiffnlp/twitter-roberta-base-emotion. Така заміна зумовлена потребою у підвищенні точності аналізу для коротких і неформальних текстів, характерних для електронного листування. Нова модель, натренована на корпусі твітів, продемонструвала вищу чутливість до емоційного забарвлення та кращу відповідність реальному контексту користувацької мови.

У процесі тестування було перевірено такі аспекти:

- коректність маршрутизації даних від інтерфейсу користувача до обчислювального модуля;
- точність емоційної класифікації на рівні як ізольованих повідомлень, так і серійних запитів;
- обробку стандартних повідомлень з вираженим емоційним навантаженням (радість, смуток, злість, оптимізм);
- стійкість системи до повторних звернень;
- відображення результатів на фронтенді без помилок структури або відмов візуалізації;
- стабільність контейнеризованого розгортання при зупинці та повторному запуску сервісів.

Фронтенд частина системи успішно обробляє запити користувача, передає дані на проміжний сервер Flask, який, у свою чергу, виконує перенаправлення до аналітичного бекенду на Django. Обчислювальний модуль формує результат аналізу у структурованому форматі та повертає його назад на інтерфейс, де він виводиться у вигляді основної емоції, коефіцієнтів та аналітичного узагальнення. Система не демонструвала збоїв, затримок або втрати даних при обробці запитів.

Під час перевірки були також враховані ситуації з порожніми запитами або повідомленнями, що не несуть чітко вираженого емоційного навантаження. Усі

подібні випадки оброблялися коректно — без аварійної поведінки або критичних помилок. Механізм псевдоідентифікації користувача на основі session_hash забезпечив підтримку сесійної послідовності, що дозволило зберігати контекст аналізу при кількох запитах поспіль.

Docker-інфраструктура показала себе як надійний засіб ізоляції компонентів. Завдяки використанню docker-compose, система запускалася без додаткового налаштування, з автоматичним формуванням мережі, пробросом портів та стабільною ініціалізацією контейнерів. Жодних труднощів із повторним запуском чи збереженням працездатності після переривання роботи не було зафіксовано.

Таким чином, система пройшла повний цикл функціонального тестування. Всі ключові компоненти продемонстрували відповідність вимогам, логічну узгодженість та технічну стабільність. Отримані результати свідчать про готовність застосунку до впровадження в умовах реального використання або до подальшого розширення з урахуванням нових вимог.

ВИСНОВКИ

Мета роботи: метою даної дипломної роботи було створення інформаційної технології, здатної розпізнавати емоційний стан співрозмовника під час текстової комунікації в онлайн-середовищі. Така задача є актуальною в умовах сучасного цифрового суспільства, де дедалі більше особистих, ділових і соціальних контактів відбувається у форматі листування, без візуальних або голосових сигналів. У цих умовах виникає необхідність в автоматизованих системах, здатних інтерпретувати приховані емоції користувача, орієнтуючись виключно на зміст тексту.

Передбачалося, що реалізована система зможе не лише визначати базову емоцію в кожному окремому повідомленні, але й враховувати історичний контекст попередньої комунікації. Це дозволяє отримати точніший, динамічний профіль емоційного стану користувача, що, у свою чергу, може бути застосовано у сферах психологічного консультування, аналізу клієнтської підтримки, освітніх платформ, систем модерації або адаптивних користувацьких інтерфейсів.

Виконані дії: У рамках дипломного проєкту було реалізовано повноцінну систему, що поєднує сучасні досягнення в галузі штучного інтелекту, глибинного навчання та веброзробки. Зокрема:

- Проведено огляд сучасного стану задачі емоційного аналізу тексту, а також історії її розвитку — від перших словникових підходів до сучасних нейромережевих рішень на базі трансформерів.
- Обґрунтовано вибір моделі cardiffnlp/twitter-roberta-base-emotion як компромісного рішення, що поєднує швидкодію та точність класифікації для чотирьох емоцій: Anger, Joy, Optimism Sadness.
- Розроблено клієнт-серверну архітектуру на основі двох незалежних бекенд-компонентів: Flask відповідає за маршрутизацію, сесії та взаємодію з фронтендом, а Django + DRF реалізує основну логіку обробки, роботу з базою даних і виклик нейромережі.
- Побудовано простий, але функціональний вебінтерфейс, який дозволяє користувачеві надсилати текстові повідомлення й одразу бачити

результати емоційного аналізу, включаючи графічне представлення домінуючої емоції, коефіцієнтів та історії.

- Забезпечено збереження даних у реляційній СУБД MySQL, де зберігаються як інформація про користувачів (на основі session_hash), так і повна історія їхніх запитів (таблиця query_history).
- Впроваджено модуль контекстного аналізу, який комбінує поточне повідомлення з N попередніми, формуючи зважений емоційний профіль.
- Реалізовано контейнеризацію системи за допомогою Docker і Docker Compose, що дозволяє швидко розгортати проєкт незалежно від середовища, уникати конфліктів між бібліотеками та підтримувати цілісність конфігурації.
- Проведено тестування системи: перевірено поведінку при стандартних і некоректних запитах, перевірено стабільність контейнерів після перезапуску, реакцію на порожні або надмірні повідомлення.

Отримані результати: Результати реалізації проєкту свідчать про досягнення поставленої мети. Було створено технологічно завершене програмне рішення, що:

- Автоматично класифікує емоції користувача на основі тексту повідомлення;
- Працює як ізольовано, так і з урахуванням попередніх запитів в межах однієї сесії;
- Зберігає інформацію для подальшого аналізу та візуалізації;
- Гарантує стабільність і повторюваність роботи через контейнеризацію;
- Може бути легко масштабоване, модифіковане або розгорнуте в новому середовищі без втрати функціональності;
- Придатне для практичного застосування у широкому спектрі задач.

Окремо варто зазначити, що реалізоване рішення має відкриту архітектуру — тобто може бути адаптоване під специфіку нових задач, мов, сфер або вимог, без потреби у принциповому перегляді основної логіки.

Перспективи розвитку**:** Система, реалізована в межах дипломної роботи, є гнучкою та розширюваною. Серед можливих напрямів її подальшого вдосконалення можна виділити:

- Підключення мультимовних моделей (наприклад, mBERT або XLM-R), що дозволить розширити функціонал на інші мови;
- Інтеграцію з месенджерами (Telegram, WhatsApp), браузерними розширеннями або мобільними застосунками;
- Створення адміністративної панелі або аналітичного модуля з графіками динаміки емоцій;
- Використання розширених емоційних класифікаторів з більшою кількістю емоцій або спектральною моделлю (від "щастя" до "апатії");
- Створення інтерфейсу для зворотного зв’язку, який дозволить уточнювати результати класифікації з боку користувача;
- Побудову системи, що навчається на основі анонімізованих даних користувачів для покращення точності.

Загальний висновок**:** Таким чином, дипломна робота продемонструвала успішну реалізацію комплексної прикладної задачі — розпізнавання емоційного стану людини на основі текстового листування з урахуванням контексту. Було застосовано сучасні технології в галузі NLP, web development та DevOps, що свідчить про міждисциплінарний характер проєкту. Отримані результати підтверджують ефективність обраного підходу, а сама система має як практичну цінність, так і науковий потенціал для подальших досліджень у сфері аналізу людських емоцій в цифровому середовищі.

ПЕРЕЛІК ПОСИЛАНЬ

1. Liu, Y., Ott, M., Goyal, N. et al. RoBERTa: A Robustly Optimized BERT Pretraining Approach. Hugging Face Documentation, 2025. URL: https://huggingface.co/docs/transformers/model_doc/roberta
2. Barbieri, F., Camacho-Collados, J., Espinosa-Anke, L., Neves, L. CardiffNLP/twitter-roberta-base-emotion. Hugging Face Repository, 2025. URL: https://huggingface.co/cardiffnlp/twitter-roberta-base-emotion
3. CardiffNLP. TweetEval Benchmark Repository. GitHub, 2025. URL: https://github.com/cardiffnlp/tweeteval/tree/main
4. Ebrahimi, M., et al. Effective Black-Box Testing of Emotion Classification Models. arXiv preprint, 2024. URL: https://arxiv.org/abs/2407.20884
5. freeCodeCamp. Python Microservices Web App (with React, Django, Flask) – Full Course. Class Central, 2024. URL: https://www.classcentral.com/course/freecodecamp-python-microservices-web-app-158271
6. Python Software Foundation. Creation of Virtual Environments. Python Documentation, 2025. URL: https://docs.python.org/3/tutorial/venv.html
7. Pallets Projects. Flask Documentation. Flask, 2025. URL: https://flask.palletsprojects.com/en/latest/
8. Django Software Foundation. Django Documentation. Django, 2025. URL: https://docs.djangoproject.com/en/stable/
9. Tom Christie. Django REST Framework Documentation. 2025. URL: https://www.django-rest-framework.org/
10. JeffyJeff. The Beginner’s Guide to Docker. Medium, 2024. URL: https://medium.com/@JeffyJeff/the-beginners-guide-to-docker-fa4c4d3181e7
11. DevLawrence. Getting Started with REST API: Beginner’s Tutorial. DEV Community, 2024. URL: https://dev.to/devlawrence/getting-started-with-rest-api-beginners-tutorial-4nki

12. Doucet, A. et al. Emotion Detection Using RoBERTa for Multilingual Tweets. SIGMORPHON Workshop, 2021. URL: https://aclanthology.org/2021.sigmorphon-1.12/
13. Alhagry, S., Aly, A. A Survey on Sentiment and Emotion Analysis for Natural Language Processing. IEEE Access, 2020. URL: https://ieeexplore.ieee.org/document/9099222
14. Raj, D. Hands-On Natural Language Processing with Python. O'Reilly Media, 2020. URL: https://www.oreilly.com/library/view/hands-on-natural-language/9781789807325/
15. Adiwardana, D. et al. Neural Approaches to Conversational AI. Microsoft Research, 2020. URL: https://arxiv.org/abs/2004.13637
16. Rao, D., et al. Docker for Data Science: Building Scalable and Modular Data Infrastructure. O'Reilly Media, 2018. URL: https://www.oreilly.com/library/view/docker-for-data/9781484230114/
17. Vincent, W. S. Django for APIs: Build Web APIs with Python & Django. Self-published, 2021. URL: https://djangoforapis.com/
18. Barbieri, F., et al. Twitter Sentiment and Emotion Analysis with Pre-trained Transformers. EMNLP 2021. URL: https://arxiv.org/abs/2108.12409
19. Mathur, P., et al. Practical Deep Learning for Cloud, Mobile, and Edge. O'Reilly Media, 2021. URL: https://www.oreilly.com/library/view/practical-deep-learning/9781492034841/
20. Tunstall, L., von Platen, P., Le Scao, T., Wolf, T. Natural Language Processing with Transformers: Building Language Applications with Hugging Face. O'Reilly Media, 2022. URL: https://www.oreilly.com/library/view/natural-language-processing/9781098103231/

ДОДАТОК А. ПРОГРАМНИЙ КОД ПРОЄКТОВАНОЇ СИСТЕМИ

```
from transformers import AutoTokenizer, AutoModelForSequenceClassification
import torch
from torch.nn.functional import softmax

# Завантаження моделі CardiffNLP
MODEL_NAME = "cardiffnlp/twitter-roberta-base-emotion"
tokenizer = AutoTokenizer.from_pretrained(MODEL_NAME)
model = AutoModelForSequenceClassification.from_pretrained(MODEL_NAME)

# Порядок емоцій згідно моделі
emotion_labels = ['anger', 'joy', 'optimism', 'sadness']

def analyze_emotions(text: str):
    if not text.strip():
        return "neutral", {label: 0.0 for label in emotion_labels}

    inputs = tokenizer(text, return_tensors="pt", truncation=True)
    with torch.no_grad():
        logits = model(**inputs).logits
        probs = softmax(logits, dim=1).squeeze().tolist()

    emotions = {label: round(prob, 3) for label, prob in zip(emotion_labels, probs)}
    emotions["score"] = round(sum(emotions.values()) / len(emotion_labels), 3)

    # Визначення тону
    dominant_emotion = max(emotions, key=lambda k: emotions[k] if k in emotion_labels
else -1)
    tone = "positive" if dominant_emotion in ['joy', 'optimism'] else "negative"
```

```
    return tone, emotions

def analyze_with_history(current_text: str, history_texts: list[str], history_scores: list[dict]):
    tone_current, current_emotions = analyze_emotions(current_text)

    if not history_scores:
        return tone_current, current_emotions

    # Усереднення історичних емоцій
    history_avg = {label: 0.0 for label in emotion_labels}
    for record in history_scores:
        for label in emotion_labels:
            history_avg[label] += record.get(label, 0.0)
    for label in history_avg:
        history_avg[label] /= len(history_scores)

    # Об'єднання з поточними (усереднення 50/50)
    combined_emotions = {
        label: round((current_emotions[label] + history_avg[label]) / 2, 3)
        for label in emotion_labels
    }
        combined_emotions["score"] = round(sum(combined_emotions.values()) / len(emotion_labels), 3)
    dominant = max(combined_emotions, key=combined_emotions.get)
    tone = "positive" if dominant in ['joy', 'optimism'] else "negative"

    return tone, combined_emotions
```

ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ КАФЕДРА
ШТУЧНОГО ІНТЕЛЕКТУ

КВАЛІФІКАЦІЙНА РОБОТА

На тему:

«Інтелектуальна технологія розпізнавання емоційного стану співрозмовника під час електронного листування на основі методів машинного навчання»

Виконав: здобувач вищої освіти гр. ШІД-42
Юліан ШВЕЦЬ
Керівник: кандидат технічних наук, доцент
Олександр ЗВЕНІГОРОДСЬКИЙ

Постановка Задачі

Мета роботи: створення веб-орієнтованої інформаційної системи для автоматичного аналізу емоційного стану людини на основі тексту електронного листування.

Об'єкт дослідження: процес автоматичного розпізнавання емоцій за текстовими повідомленнями.

Предмет дослідження: методи та засоби розпізнавання емоційного стану у текстових повідомленнях.

Основні завдання:

1. Аналіз сучасних підходів і методів класифікації емоцій у тексті.
2. Вибір та освоєння технологій розробки.
3. Створення архітектури та прототипу системи.
4. Розробка дизайну веб-ресурсу.
5. Тестування реалізованої інформаційної системи.

Актуальність теми

У XXI столітті текстова комунікація стала головним каналом взаємодії в бізнесі, освіті, психологічній підтримці, медичних послугах і цифровій бюрократії.

Емоції в текстах несуть важливу інформацію про внутрішній стан, наміри, рівень стресу чи задоволення користувача — навіть коли він цього прямо не озвучує.

Більшість цифрових систем досі залишаються “емоційно нечутливими”: вони аналізують лише зміст повідомлень, не враховуючи емоційний фон, тональність чи зміну настрою.

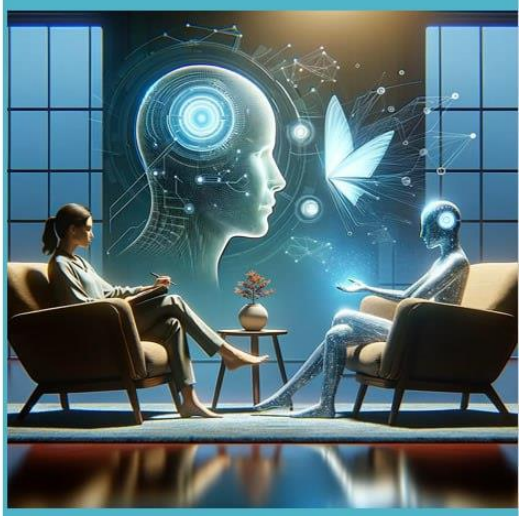
Натомість сучасні сервіси дедалі частіше потребують гнучкої адаптації до емоцій — як у чат-ботах, так і в автоматизованих платформах зворотного зв’язку чи підтримки.

Аналіз емоцій виявляється критично важливим у дистанційному навчанні, службах підтримки, цифровій психотерапії, а також у брендовому та соціальному моніторингу.

Здатність системи виявляти роздратування, тривожність чи байдужість дозволяє вчасно змінити стиль спілкування, надати підтримку або передати запит оператору.

Це відкриває шлях до створення сервісів, здатних не лише відповідати, а й розуміти емоційний контекст — що є ключовою вимогою до "людяного" ШІ.

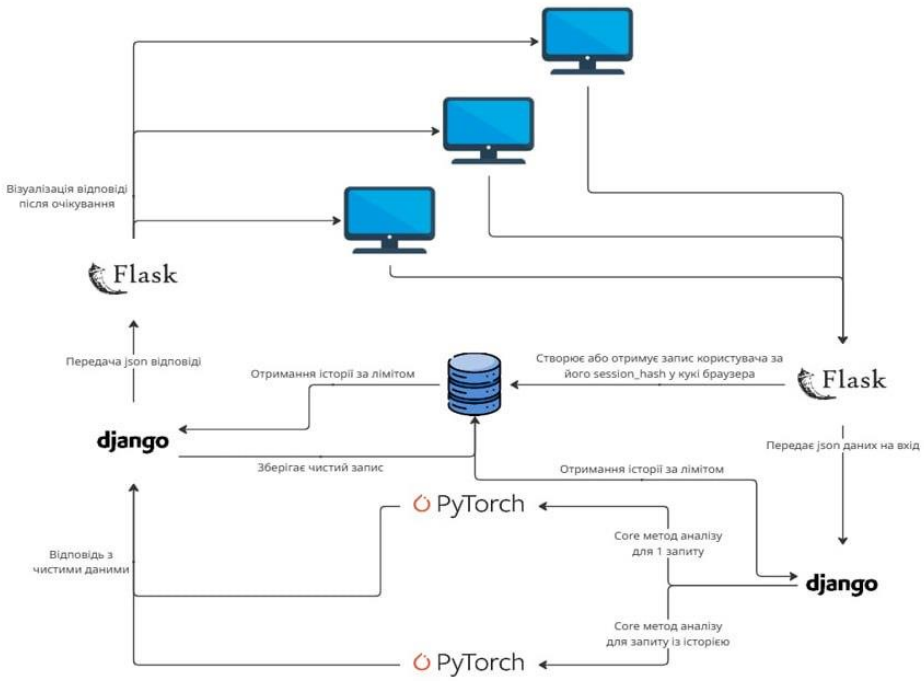
Розробка таких рішень — це не лише технічна інновація, а й відповідь на запит суспільства на глибші, чутливіші цифрові взаємодії.

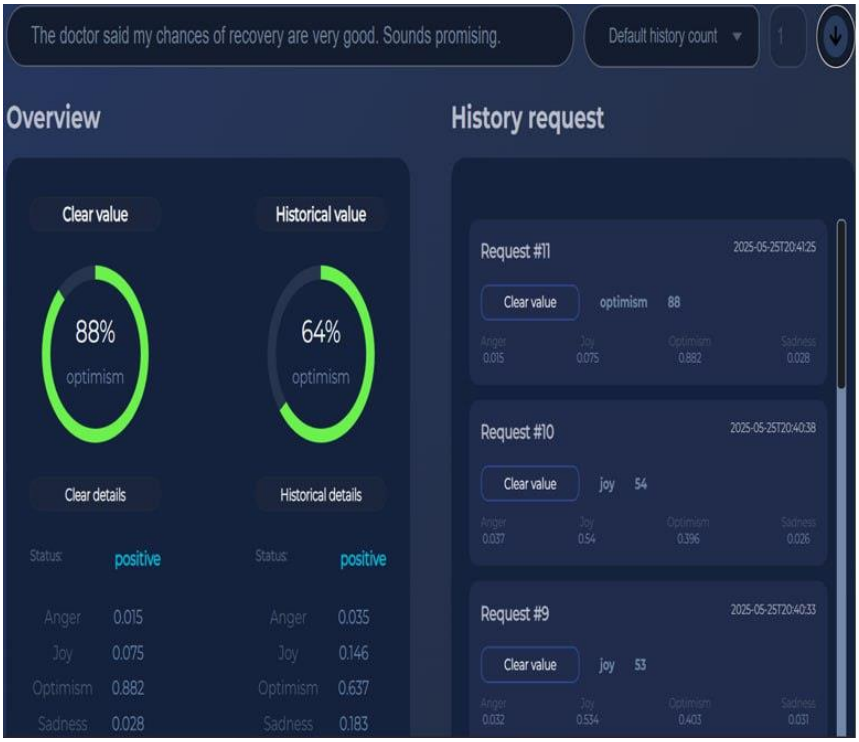


Використані технології та інструменти

- Frontend: HTML, CSS, JavaScript
- Backend: Flask (маршрутизація запитів), Django DRF (аналітичний бекенд)
- ML-модель: HuggingFace Transformers (twitter-roberta-base-emotion)
- База даних: MySQL
- ORM/SQL: SQLAlchemy (Flask), Raw SQL (Django)

Архітектура системи





Опис роботи системи

1. Користувач вводить текст повідомлення через вебінтерфейс.
2. Запит передається на Flask-сервер, що керує сесією та маршрутизацією.
3. Flask передає запит до Django DRF для аналізу.
4. Django DRF токенізує текст, проводить емоційний аналіз за допомогою RoBERTa-моделі.
5. Django повертає структурований результат емоційного аналізу через Flask на фронтенд.
6. Результат відображається користувачу у зрозумілій формі (основна емоція, коефіцієнти, історія запитів).

Висновки

Проведено аналіз сучасних методів і підходів до автоматичного розпізнавання емоцій людини з тексту з використанням нейромережевих моделей.

В результаті роботи було:

- Розглянуто сучасні рішення та методики розпізнавання емоцій.
- Створено архітектуру системи з використанням популярних вебтехнологій.
- Успішно інтегровано сучасну високоточну нейромережеву модель.

Система потребує певних доробок і оптимізації, після яких може бути впроваджена у реальне використання.

Запропоноване рішення може знайти широке практичне застосування у таких галузях:

- психологічна підтримка та консультування,
- автоматизоване оцінювання якості клієнтської підтримки,
- персоналізовані навчальні та освітні платформи,
- онлайн-маркетинг та соціальні дослідження.