Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра програмного забезпечення

Бакалаврська кваліфікаційна робота

на тему: Розробка вебсайту для персоналізованого аналізу YouTube-відео: обробка та аналітика на основі нейромереж

Виконав: студент IV курсу
групи 2ПІ-21б
спеціальності
121 – Інженерія програмного забезпечення
(шифр і назва напряму підготовки, спеціальності)
Столяр Владислав Васильович
(прізвище та ініціали)

Керівник: к.т.н., доц. каф. ПЗ Ракитянська Г.Б.
(прізвище та ініціали)
Рецензент:. к.т.н., доц. каф ЗІ Баришев Ю.В.
(прізвище та ініціали)

Допущено до захисту

Зав. кафедри ПЗ О.Н. Романюк
«09» 06 2025 р.

ВНТУ – 2025

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення
Рівень вищої освіти перший (бакалаврський)
Галузь знань 12 – Інформаційні технології
Спеціальність 121 – Інженерія програмного забезпечення
Освітньо-професійна програма – Інженерія програмного забезпечення

«ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
О. Н. Романюк
« 21 » березня 2025 р.

З А В Д А Н Н Я
НА БАКАЛАВРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Столяру Владиславу Васильовичу

1. Тема роботи – розробка вебсайту для персоналізованого аналізу YouTube-відео: обробка та аналітика на основі нейромереж.

Керівник роботи: Ракитянська Ганна Борисівна, к.т.н., доцент кафедри ПЗ, затверджені наказом вищого навчального закладу від «20» березня 2025 р. № 97.

2. Строк подання студентом роботи 2 червня 2025

3. Вихідні дані до роботи: технічне завдання на розробку вебзастосунку, опис функціональних і нефункціональних вимог до системи, результати аналізу аналогічних програмних рішень, специфікація архітектури застосунку, вимоги до інтерфейсу користувача, стандарти програмної розробки, а також перелік інструментів і технологій, що використовуються

4. Зміст текстової частини: вступ, аналіз персоналізованого аналізу YouTube-відео та постановка задач розробки, розробка моделей та алгоритмів програмного продукту, розробка програмного забезпечення системи персоналізованого аналізу відео YouTube, тестування програмного застосунку, висновки, список використаних джерел та додатки.

5. Перелік ілюстративного матеріалу: діаграми прецедентів, діаграми класів, схеми архітектури застосунку, макети користувацького інтерфейсу, фрагменти коду, результати тестування, графіки точності роботи модулів, а також скріншоти реалізованого програмного продукту.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-4	Ракитянська Г.Б., к.т.н., доцент кафедри ПЗ	24.03.25	01.06.25

7. Дата видачі завдання 24 березня 2025р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз предметної області, постановка задач	25.03.25 – 03.04.25	Вик
2	Розробка архітектури системи та вибір технологій	04.04.25 – 11.04.25	Вик.
3	Проєктування інтерфейсу користувача	12.04.25 – 19.04.25	Вик.
4	Реалізація основних програмних модулів	20.04.25 – 10.05.25	Вик
5	Тестування та розробка інструкції користувача	11.05.25 – 20.05.25	Вик
6	Оформлення матеріалів до захисту БКР	21.05.25 – 30.05.25	Вик

Студент _____ **В.В. Столяр**
(підпис) (ініціали та прізвище)

Керівник бакалаврської кваліфікаційної роботи _____ **Г.Б. Ракитянська**
(підпис) (ініціали та прізвище)

АНОТАЦІЯ

УДК 004.8

Столяр В.В. Розробка вебсайту для персоналізованого аналізу YouTube-відео: обробка та аналітика на основі нейромереж: бакалаврська кваліфікаційна робота зі спеціальності 121 – інженерія програмного забезпечення, освітня програма – інженерія програмного забезпечення. Вінниця: ВНТУ, 2025. 59 с.

На укр. мові. Бібліогр. : 28 назв ; рис. : 31; табл. 6.

У бакалаврській кваліфікаційній роботі розроблено вебсайт для персоналізованого аналізу відеоконтенту з платформи YouTube. Основна увага приділена інтеграції нейромережевої обробки для трансформації транскрипцій відео згідно з індивідуальними запитами користувача (промптами).

Реалізовано систему авторизації через Telegram, функціонал додавання підписок на канали, отримання транскрипцій та генерації модифікованого тексту. Розроблено графічний інтерфейс із підтримкою криптовалютного гаманця для оплати запитів. Система реалізована із використанням Flutter (Dart) для клієнтської частини та .NET (C#) для серверної. Проведено тестування основних функціональних модулів системи.

Отримані результати можуть бути використані в освітніх, інформаційних та дослідницьких цілях.

Ключові слова: персоналізований аналіз, YouTube-відео, нейронні мережі, обробка тексту, веброзробка, аналітика відео, telegram-авторизація, криптогаманець

ABSTRACT

UDC 004.8

Stoliar V.V. Development of a Website for Personalized Analysis of YouTube Videos: Processing and Analytics Based on Neural Networks: Bachelor's thesis in specialty 121 – Software Engineering, educational program – Software Engineering. Vinnytsia: VNTU, 2025. 59 p.

In Ukrainian. Bibliography: 28 titles; figures: 31; tables: 6.

The bachelor's qualification paper presents the development of a website for personalized analysis of video content from the YouTube platform. The main focus is on integrating neural network processing to transform video transcripts according to individual user requests (prompts).

The system implements authorization via Telegram, functionality for subscribing to channels, retrieving transcripts, and generating modified text. A graphical interface has been developed with support for a cryptocurrency wallet to process user requests. The system is implemented using Flutter (Dart) for the client side and .NET (C#) for the server side. Testing of the main functional modules of the system has been carried out.

The obtained results can be used for educational, informational, and research purposes.

Keywords: personalized analysis, YouTube videos, neural networks, text processing, web development, video analytics, Telegram authorization, crypto wallet.

ЗМІСТ

ВСТУП

Обґрунтування вибору теми дослідження. У сучасну цифрову епоху відеоконтент став однією з основних форм споживання інформації в інтернеті. Особливо це стосується таких платформ, як YouTube, які щоденно генерують мільйони годин відео, доступного для перегляду у будь-який час [1]. Стрімкий розвиток таких ресурсів спричинив зростання інтересу до засобів аналітики відео – як з боку бізнесу, так і з боку звичайних користувачів. Проте більшість існуючих інструментів аналітики орієнтовані переважно на агреговану статистику – перегляди, лайки, коментарі, демографію аудиторії [2]. Водночас глибшого аналізу змісту відео, його тематичної спрямованості, емоційного забарвлення чи релевантності для конкретного користувача – такі сервіси або не надають взагалі, або пропонують лише у форматі складних комерційних рішень.

Окрім цього, із зростанням ролі персоналізації в інтернеті все більше користувачів очікують, що системи будуть адаптуватися під їхні індивідуальні потреби, інтереси та контекст [3]. У такому середовищі виникає потреба у створенні інтелектуальних систем, здатних здійснювати глибокий аналіз відеоконтенту з урахуванням індивідуальних уподобань користувача. Це дозволяє не лише підвищити ефективність сприймання інформації, а й зменшити інформаційне перевантаження.

Надзвичайно перспективними для вирішення такого завдання є нейронні мережі [4], які здатні аналізувати не лише текстову, а й аудіовізуальну інформацію. Вони можуть виявляти ключові теми відео, здійснювати емоційний або семантичний аналіз, генерувати короткі текстові анотації або резюме. Це відкриває широкі можливості для персоналізованої обробки контенту.

Ще одним важливим аспектом сучасних застосунків є безпечна та зручна автентифікація користувача. Telegram став одним із найпопулярніших месенджерів, який дозволяє інтегрувати авторизацію через ботів, забезпечуючи водночас простоту використання та надійність [5]. Крім того, у контексті зростання популярності Web3-рішень, дедалі більше уваги приділяється

використанню криптогаманців як засобу цифрової ідентифікації [6]. Поєднання цих технологій дозволяє створити унікальне програмне рішення, що відповідає викликам часу.

Тому актуальними є питання покращення якості персоналізованого аналізу шляхом розробки інструментів аналітики відеоконтенту, здатних не лише здійснювати поверхневу статистичну оцінку, а й аналізувати зміст, емоційне забарвлення та тематичну спрямованість відео. Особливої актуальності набувають такі інструменти в контексті інтеграції з сучасними засобами авторизації, такими як Telegram або криптогаманці, що забезпечують безпечний і зручний доступ до сервісів. Тема бакалаврської кваліфікаційної роботи повністю відповідає цим тенденціям і знаходиться на перетині актуальних напрямів сучасної інформатики. Реалізація такого рішення має високу прикладну цінність і може бути корисною як для пересічних користувачів, так і для професіоналів, зацікавлених у глибшому аналізі відеоконтенту.

Зв’язок роботи з науковими програмами, планами, темами. Робота виконувалася згідно плану виконання наукових досліджень на кафедрі програмного забезпечення.

Мета та завдання дослідження. Метою бакалаврської роботи є покращення якості персонального аналізу YouTube-відео шляхом створення функціонального вебзастосунку, що забезпечує зручний доступ до аналітики відеоконтенту, а також інтеграцію новітніх технологій, таких як криптогаманець для управління фінансовими аспектами використання ресурсу. Застосунок надаватиме можливість користувачам автоматично отримувати структурований текстовий виклад відео, аналізувати його тональність, ключові ідеї, а також здійснювати фінансові операції в межах платформи.

Для досягнення поставленої мети необхідно вирішити такі завдання:

– провести аналіз існуючих рішень у сфері автоматизованого аналізу відеоконтенту та засобів візуалізації фінансової активності користувача;

– визначити способи інтеграції результатів нейромережевої обробки відео в інтерфейс користувача;

– розробити архітектуру вебсервісу та його основні компоненти, враховуючи нові функції;

– реалізувати механізми відображення результатів аналізу текстового контенту та інтеграцію криптогаманця в інтерфейс користувача;

– протестувати роботу клієнтської частини системи, оцінити зручність інтерфейсу та коректність виводу фінансової інформації.

Об’єкт дослідження – процес розробки вебсистеми для персоналізованого аналізу відео на YouTube з використанням нейромереж.

Предмет дослідження – методи та засоби реалізації вебсистеми для персоналізованого аналізу відео на YouTube з використанням нейромереж.

Методи дослідження. У процесі дослідження використовувалися: теорія обробки природної мови для інтеграції результатів текстового аналізу в інтерфейс користувача; результати глибокого навчання для використання функцій автоматичної транскрипції та аналізу тональності в інтерфейсі застосунку; комп’ютерне моделювання для проектування логіки взаємодії інтерфейсу з аналітичними модулями; вебтехнології (Flutter, Dart) для створення архітектури та реалізації клієнтського модуля вебзастосунку.

Новизна отриманих результатів.

1. Модель системи, яка поєднує можливості аналізу відеоконтенту, інтеграції платіжного функціоналу та взаємодії з користувачем, спроєктована з урахуванням сучасних вимог до персоналізованих вебсервісів. Її архітектура дозволяє масштабувати застосунок та доповнювати новими модулями без значних змін у базовому функціоналі.

2. Алгоритми обробки відео на основі нейромереж, що дозволяють проводити багаторівневу обробку контенту – від транскрибування до аналізу

ключових ідей та емоційного забарвлення тексту, – оптимізовано для швидкої роботи в реальному часі та мінімізації участі людини у процесі.

Практична цінність отриманих результатів. Практична цінність отриманих результатів полягає у можливості використання вебсервісу для аналізу YouTube-відео в різних сферах діяльності. Зокрема, програму можна застосовувати у дослідницьких цілях для вивчення інформаційного впливу контенту, в освітньому процесі для формування тематичних добірок відео на основі змісту, а також у маркетинговій сфері – для швидкої оцінки та змісту публікацій конкурентів або цільової аудиторії.

Особистий внесок. Усі наукові результати отримано здобувачем самостійно.

Апробація роботи. Результати роботи доповідалися на LIV Всеукраїнській науково-технічній конференції факультету інформаційних технологій та комп’ютерної інженерії (2025), що відбулася у Вінницькому національному технічному університеті, і опубліковано тези, де представлено основні аспекти реалізації вебзастосунку для персонального аналізу відео з платформи YouTube.

Публікації. За тематикою дослідження опубліковано наукову працю у збірнику матеріалів конференції «LIV Всеукраїнській науково-технічній конференції факультету інформаційних технологій та комп’ютерної інженерії (2025)» [7].

1 АНАЛІЗ ПЕРСОНАЛІЗОВАНОГО АНАЛІЗУ YOUTUBE-ВІДЕО ТА ПОСТАНОВКА ЗАДАЧ РОЗРОБКИ

1.1 Аналіз предметної області інтелектуального аналізу відеоконтенту YouTube

YouTube є однією з найбільших платформ для розповсюдження відеоконтенту, яка охоплює понад 2,5 мільярда активних користувачів щомісяця. Щоденно на платформу завантажується понад 500 годин відео щохвилини, що створює величезний масив інформації [8]. Відеоконтент охоплює різні сфери, включаючи освіту, науку, розваги, бізнес та технології. Завдяки своїй доступності та масштабності YouTube є основним джерелом контенту для користувачів по всьому світу.

Робота з відеоконтентом на YouTube базується на складних алгоритмах персоналізованих рекомендацій, які враховують інтереси користувача, історію переглядів, взаємодію з контентом та загальні тренди. Автоматизована аналітика дозволяє авторам отримувати дані щодо переглядів, середньої тривалості перегляду, показника взаємодії (лайки, коментарі, підписки) та інших метрик, що дає змогу оцінити ефективність відео [9].

Проте, незважаючи на високий рівень автоматизації, аналіз відеоконтенту стикається з низкою проблем [10]. По-перше, великий обсяг відео ускладнює швидке знаходження релевантного контенту. По-друге, суб'єктивність змісту відео та його інтерпретації створює труднощі у класифікації та оцінюванні. По-третє, мовні бар’єри обмежують доступ до контенту для аудиторії, яка не володіє мовою оригіналу. Крім того, автори відео можуть маніпулювати заголовками та описами, що ускладнює точне розуміння суті матеріалу.

Ці виклики створюють потребу у розробці спеціалізованих рішень для глибшого аналізу відео. Інтеграція технологій штучного інтелекту та нейромереж може покращити автоматичну обробку змісту відео, дозволяючи користувачам

отримувати персоналізований аналіз текстового контенту, оцінку основних ідей та тенденцій, а також інструменти для більш ефективного споживання інформації.

1.2 Огляд і порівняння існуючих рішень

Сучасний ринок містить низку програмних рішень для аналізу відеоконтенту на платформі YouTube. Вони можуть виконувати такі функції, як збір статистики, автоматична транскрипція аудіо у текст, аналіз популярності контенту та рекомендації щодо SEO-оптимізації. Проте жоден з них не пропонує комплексного персоналізованого аналізу відео з урахуванням конкретних запитів користувача.

У цьому розділі розглянемо найбільш популярні сервіси, які частково або повністю реалізують функції аналізу YouTube-відео:

– YouTube Analytics (Google LLC);
– Vidooly;
– TubeBuddy;
– Sonix.ai (Sonix, Inc.);
– Watson Speech to Text (IBM).

YouTube Analytics (Google LLC) – офіційний інструмент YouTube, що дозволяє авторам аналізувати статистику переглядів, активність користувачів, демографічні показники аудиторії та залученість контенту (див. рисунок 1.1). Основний недолік сервісу – відсутність можливостей для глибокого аналізу змісту відео, транскрипції тексту чи аналізу тональності автора [11].

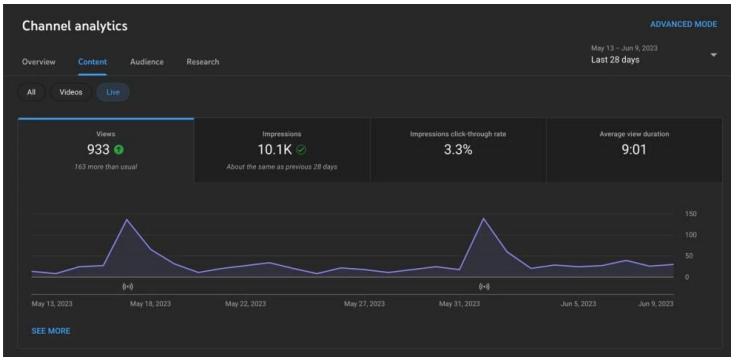


Рисунок 1.1 – Інтерфейс YouTube Analytics

Vidooly – маркетингова платформа для аналітики відеоконтенту, яка надає дані про тренди, залученість користувачів та конкурентів (див. рисунок 1.2). Вона використовується здебільшого компаніями та блогерами для покращення ефективності контенту [12]. Головний недолік – акцент лише на маркетинговій аналітиці, без можливості глибокого змістового аналізу відео.

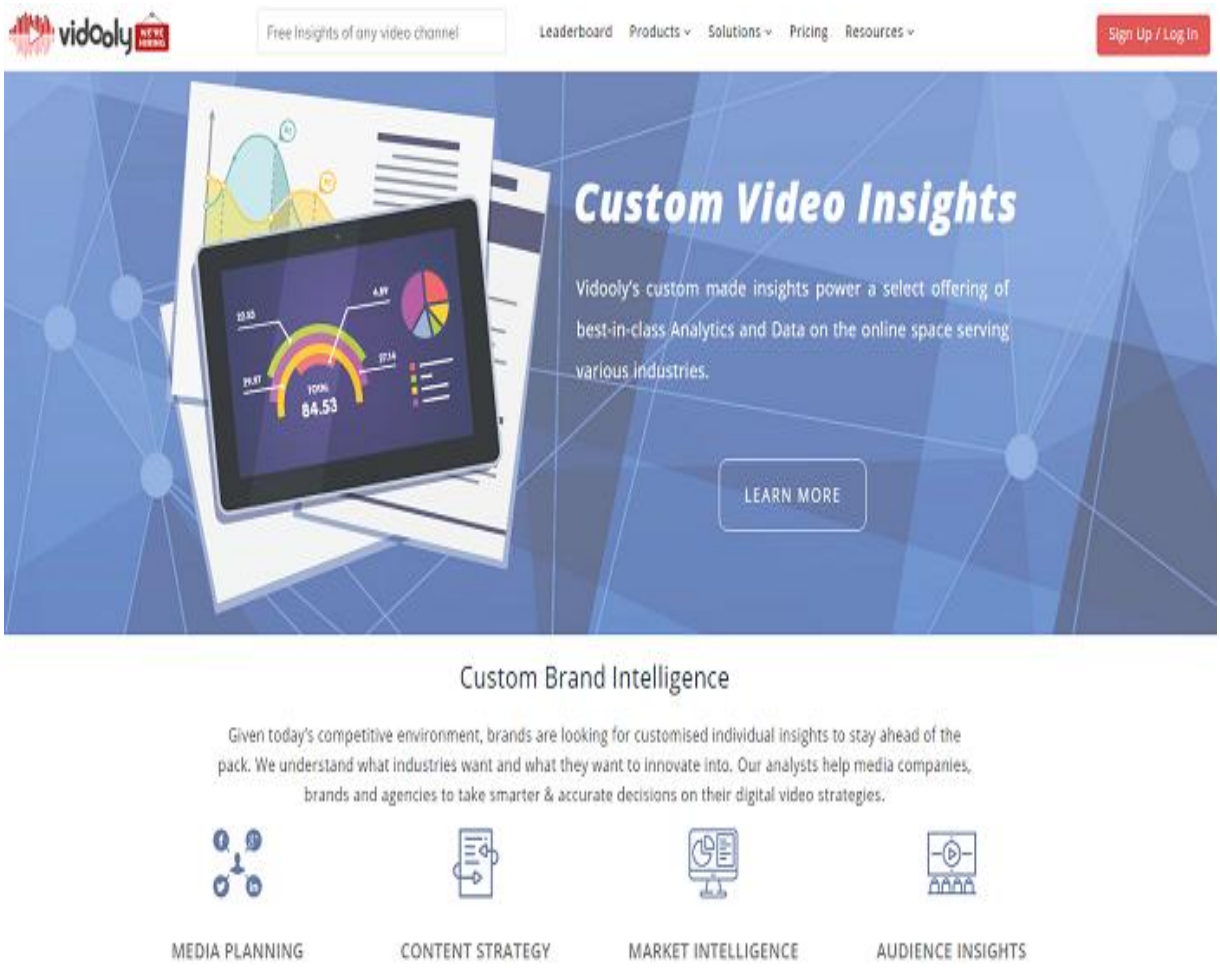


Рисунок 1.2 – Головна сторінка Vidooly

TubeBuddy – браузерне розширення для YouTube, що допомагає авторам оптимізувати свої відео за допомогою інструментів для аналізу ключових слів, підбору тегів і SEO-оптимізації (див. рисунок 1.3). Його головний мінус – відсутність аналізу змісту відео та персоналізованого підходу до обробки інформації [13].

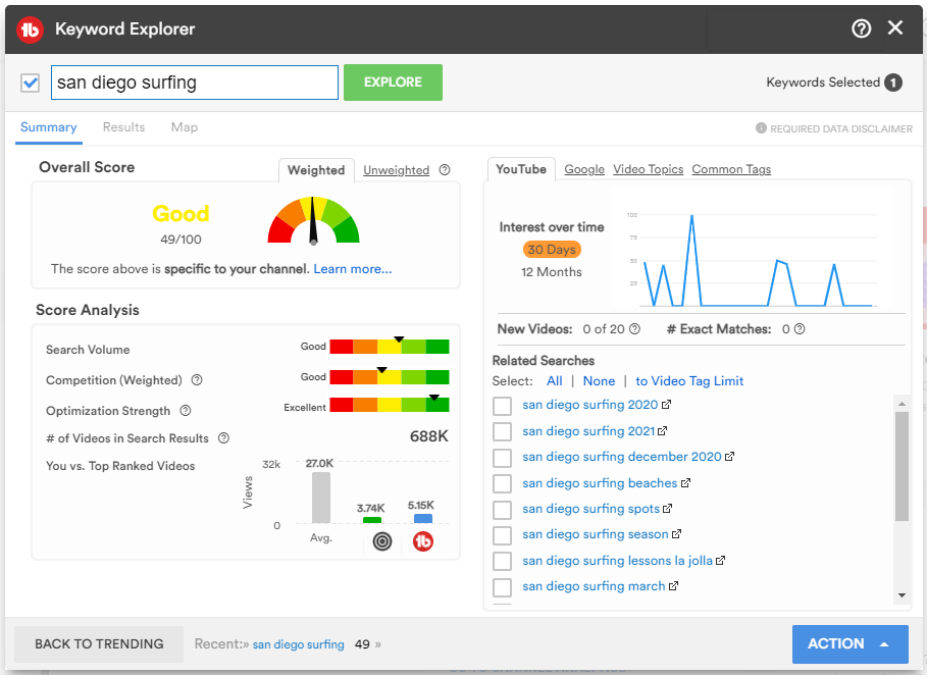


Рисунок 1.3 – Приклад використання TubeBuddy

Sonix.ai (Sonix, Inc.) – хмарний сервіс автоматичної транскрипції аудіо та відео, що підтримує кілька мов (див. рисунок 1.4). Він дозволяє конвертувати мовлення у текст, проте не пропонує подальшого аналізу отриманого тексту, що є його основним недоліком [14].

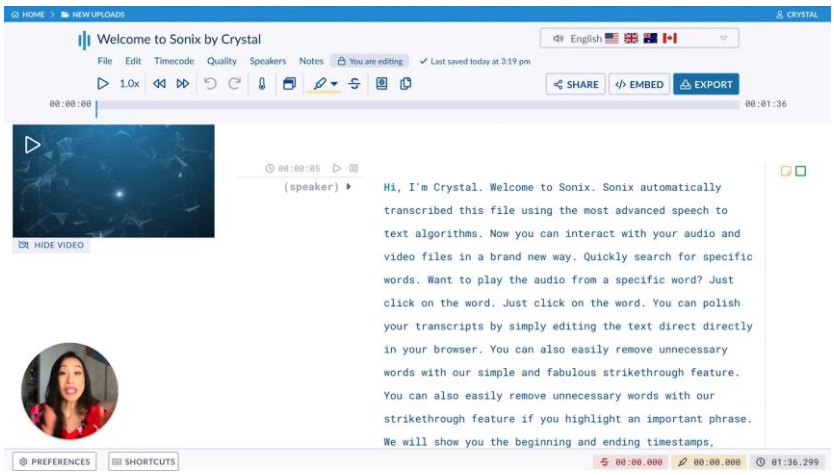


Рисунок 1.4 – Приклад використання Sonix.ai

Watson Speech to Text (IBM) – потужний інструмент на основі штучного інтелекту для перетворення мовлення у текст (див. рисунок 1.5). Він підтримує

аналіз тональності та емоцій, але вимагає складної інтеграції у власні рішення та не орієнтований на персоналізований аналіз YouTube-відео [15].

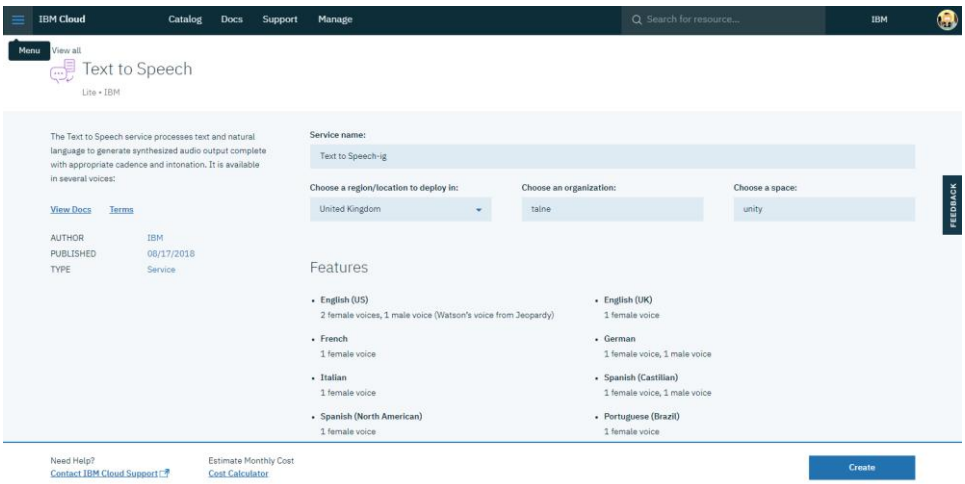


Рисунок 1.5 – Інтрефейс Watson Speech to Text

Для узагальнення проведеного аналізу було складено таблицю 1.1, яка містить порівняння функціональних можливостей розглянутих рішень.

Таблиця 1.1 – Порівняльний аналіз існуючих рішень

Функціонал	YouTube Analytics	Vidooly	TubeBuddy	Sonix.ai	IBM Watson STT	Власна розробка
1	2	3	4	5	6	7
Транскрипція відео	-	-	-	+	+	+
Аналіз змісту	-	-	-	-	+	+
Персоналізований аналіз	-	-	-	-	-	+
Аналіз емоцій та тональності	-	-	+	-	-	+
Аналітика аудиторії	+	+	-	+	+	+
Загальна оцінка	20%	20%	20%	40%	60%	100%

Аналіз існуючих рішень показав, що жоден з розглянутих сервісів не надає комплексного персоналізованого аналізу YouTube-відео. Деякі з них мають окремі корисні функції, такі як автоматична транскрипція (Sonix.ai, IBM Watson STT), аналітика аудиторії (YouTube Analytics, Vidooly) або аналіз тональності мовлення (IBM Watson STT). Однак жоден із цих сервісів не поєднує всі необхідні можливості для повноцінного аналізу контенту відео.

Як видно з таблиці 1.1, найбільш функціональним серед наявних рішень є IBM Watson Speech to Text, який отримав загальну оцінку 60%. Водночас наш продукт охоплює всі ключові функції аналізу, досягаючи 100% відповідності критеріям. Це підтверджує доцільність розробки власного програмного рішення, яке інтегрує автоматичну транскрипцію, аналіз змісту, персоналізовану обробку даних та аналітику думок автора відео, що зробить його унікальним на ринку.

1.3 Аналіз методів розв’язання задач

Розробка програмного забезпечення для персоналізованого аналізу YouTube-відео потребує вибору ефективних методів обробки тексту, аналізу тональності, транскрипції аудіо та візуалізації даних. У цьому розділі розглянуто основні методи, які можна застосувати для реалізації ключових функцій системи.

Однією з основних задач є автоматична транскрипція аудіо у текст. Для цього використовуються як традиційні, так і сучасні методи. Традиційні методи, такі як Hidden Markov Models (HMM) та Gaussian Mixture Models (GMM), ґрунтуються на статистичних моделях і акустичних особливостях мовлення, проте вони поступаються в точності сучасним нейромережевим моделям [16]. Серед передових рішень виділяються нейронні мережі, що використовують глибокі згорткові та рекурентні архітектури. До найпопулярніших моделей належать DeepSpeech від Mozilla, Whisper від OpenAI та Wav2Vec 2.0 від Meta. Whisper є однією з найбільш точних моделей, оскільки підтримує багатомовну транскрипцію та може працювати навіть із зашумленими аудіофайлами.

Після транскрипції відео необхідно виконати аналіз тексту та визначити ключові фрази. Одним із базових методів є Term Frequency-Inverse Document Frequency (TF-IDF), який визначає важливість слів у тексті на основі їх частотності, проте він не враховує контекст [17]. Інший підхід, Latent Semantic Analysis (LSA), застосовує сингулярне розкладання матриць для визначення семантичних зв'язків між словами, що дозволяє покращити якість аналізу. Найбільш точними вважаються методи на основі глибокого навчання, такі як Word2Vec, FastText та BERT. Особливо ефективним є BERT, який враховує контекст слів і здатен працювати з багатозначними поняттями.

Ще однією важливою задачею є аналіз тональності тексту, який дозволяє визначити емоційний фон відео та ставлення автора до певних тем. Найпростіший підхід базується на лексичних ресурсах, де позитивні та негативні слова визначають тональність тексту. Проте цей метод не завжди ефективний для складних речень. Більш точні результати дають методи машинного навчання, такі як Naïve Bayes, SVM та Random Forest, які використовуються для класифікації текстів на позитивні, негативні та нейтральні. Найкращі результати демонструють моделі глибокого навчання, зокрема LSTM та Transformers (BERT, RoBERTa, DistilBERT), які здатні враховувати контекст та граматичні особливості тексту.

Останнім важливим аспектом є візуалізація даних та аналітики, що дозволяє подати результати аналізу у зручному вигляді. Використовуються різні графічні методи, такі як стовпчикові діаграми, кругові діаграми та лінійні графіки, які допомагають відобразити частотність ключових тем або зміну тональності в часі. Для представлення популярних тем у відео застосовується метод хмар слів (Word Cloud), що дозволяє швидко оцінити основні теми розмови. Також корисним є сентиментний графік (Sentiment Timeline), який відображає зміну емоційного тону відео протягом його тривалості.

Аналіз існуючих методів показав, що для якісного аналізу відео доцільно використовувати сучасні нейромережеві підходи, такі як Whisper для транскрипції, BERT для визначення ключових фраз та аналізу змісту, а також

глибокі рекурентні моделі для оцінки тональності тексту. Візуалізація результатів у вигляді графіків та хмар слів дозволяє зробити висновки зрозумілими для користувача. Вибір цих методів забезпечить високу точність аналізу та зручність використання системи.

1.4 Постановка задач

На основі проведеного аналізу предметної області, існуючих аналогів та методів розв’язання поставлених задач сформульовано ключові вимоги до вебсайту для персоналізованого аналізу YouTube-відео. Основною метою процес покращення якості шляхом розробки програмного продукту, який дозволить користувачам отримувати глибоку аналітику відеоконтенту за допомогою сучасних методів обробки тексту та нейромережевого аналізу.

Завдання включають декілька основних напрямків. Першочерговим є розробка алгоритму автоматичної транскрипції аудіо у текст, який забезпечить точне та швидке перетворення мовлення у текстову форму. Для цього необхідно інтегрувати ефективні мовні моделі, які підтримують багатомовну обробку та працюють у реальному часі. Наступним завданням є аналіз змісту відео, що включає виділення ключових тем, пошук важливих фраз та визначення їхньої релевантності. Це потребує використання сучасних методів Natural Language Processing (NLP) та моделей на основі трансформерів.

Окремим аспектом є аналіз тональності тексту, що дозволить визначати емоційний настрій автора відео. Реалізація цього завдання передбачає розробку алгоритмів, які будуть класифікувати текст на позитивний, негативний або нейтральний, а також відстежувати зміни емоційної забарвленості упродовж відео. Для підвищення точності аналізу необхідно використовувати моделі глибокого навчання.

Постановка задачі:

– розробити авторизацію користувачів для персоналізованого доступу до сервісу;

– розробити можливість підписки на YouTube-блогерів та перегляду їхнього контенту;

– реалізувати отримання оригінального тексту відео та його модифікацію за допомогою нейромережевої обробки відповідно до заданого користувачем промпту;

– розробити зручний та адаптивний графічний інтерфейс програмного забезпечення;

– здійснити тестування програмного забезпечення згідно поставлених задач.

Ще одним важливим завданням є створення інтуїтивно зрозумілого інтерфейсу вебсервісу, що забезпечить зручний доступ до результатів аналізу. Крім того, система повинна мати можливість персоналізованого налаштування, що дозволить користувачам обирати критерії аналізу відповідно до їхніх потреб.

Підсумовуючи, основні задачі, які необхідно вирішити у межах бакалаврської кваліфікаційної роботи, включають розробку алгоритму транскрипції аудіо у текст, аналіз змісту та тональності відео, а також створення зручного та ефективного інтерфейсу для представлення результатів. Реалізація цих задач дозволить створити потужний інструмент для персоналізованого аналізу відеоконтенту, що значно спростить роботу з великим обсягом інформації та покращить розуміння ключових ідей, викладених у відео.

1.5 Висновки

У першому розділі був проведений детальний аналіз предметної області, включаючи вивчення ключових аспектів автоматизованого аналізу відеоконтенту на платформі YouTube. Розглянуто сучасні тенденції, пов'язані з обробкою відео, та виконано аналіз існуючих програмних рішень, таких як YouTube Analytics, Vidooly, TubeBuddy, Sonix.ai та IBM Watson Speech to Text. У ході порівняння функціональних можливостей цих рішень було виявлено їхні переваги, недоліки

та обмеження.

Особливу увагу приділено недолікам, які перешкоджають виконанню комплексного персоналізованого аналізу відео. Ці фактори стали основою для формулювання вимог до власного програмного продукту, здатного інтегрувати транскрипцію, семантичний аналіз та аналіз тональності тексту, забезпечуючи високий рівень точності й інтуїтивно зрозумілий інтерфейс.

Запропоновані методи, зокрема використання сучасних нейромережевих моделей, таких як Whisper, BERT та методи візуалізації даних, підтвердили свою ефективність. Висновки, отримані в результаті аналізу, дозволили обґрунтувати необхідність створення нового вебсервісу, який вирішуватиме проблеми, не охоплені існуючими аналогами.

У фінальній частині розділу були сформульовані ключові задачі, включаючи транскрипцію аудіо, глибокий аналіз тексту та створення користувацького інтерфейсу. Виконання цих завдань має забезпечити реалізацію комплексного рішення, яке відповідатиме сучасним потребам аналізу відеоконтенту.

2 РОЗРОБКА МОДЕЛЕЙ ТА АЛГОРИТМІВ ПРОГРАМНОГО ПРОДУКТУ

2.1 Розробка інтерфейсу для персоналізованого аналізу відео YouTube

У межах розробки програмного забезпечення важливим етапом є створення користувацького інтерфейсу (UI), який забезпечує інтуїтивну, зручну та логічну взаємодію між користувачем і системою. Основна мета інтерфейсу – забезпечити доступ до ключових функцій застосунку, орієнтуючись на потреби користувача та особливості взаємодії з платформою YouTube.

Для реалізації інтерфейсу було обрано фреймворк Flutter, який дозволяє створювати адаптивні та кросплатформені вебінтерфейси. Його використання забезпечує швидку розробку, просте масштабування та зручне компонування елементів. На головній сторінці застосунку користувача зустрічає панель із двома основними вкладками: «Subscriptions» та «Content». У першій вкладці відображаються підписані канали, у другій – доступні відео для аналізу.

Користувач має змогу додати YouTube-канал за допомогою введення псевдоніма автора на вкладці Subscriptions. Після цього система надсилає запит на сервер, де відбувається перевірка валідності каналу – тобто чи існує такий канал на платформі YouTube. У випадку позитивного результату, канал додається до списку підписок користувача.

Для перегляду відео необхідно вибрати один із раніше доданих каналів, після чого у вкладці Content відображається список доступних відео цього автора. Натиснувши на будь-яке з відео, відкривається діалогове вікно, у якому користувач може переглянути оригінальний текст та модифікований текст відео. Оригінальний текст формується автоматично на основі розпізнавання мовлення у відео, тоді як модифікований текст генерується лише за умови, якщо користувач задав персоналізований промпт для цього каналу. Також реалізовано можливість видалення каналу зі списку підписок та редагування промпту.

Для реалізації логіки взаємодії з інтерфейсом було створено схему інтерфейсу, що зображена на рисунку 2.1.

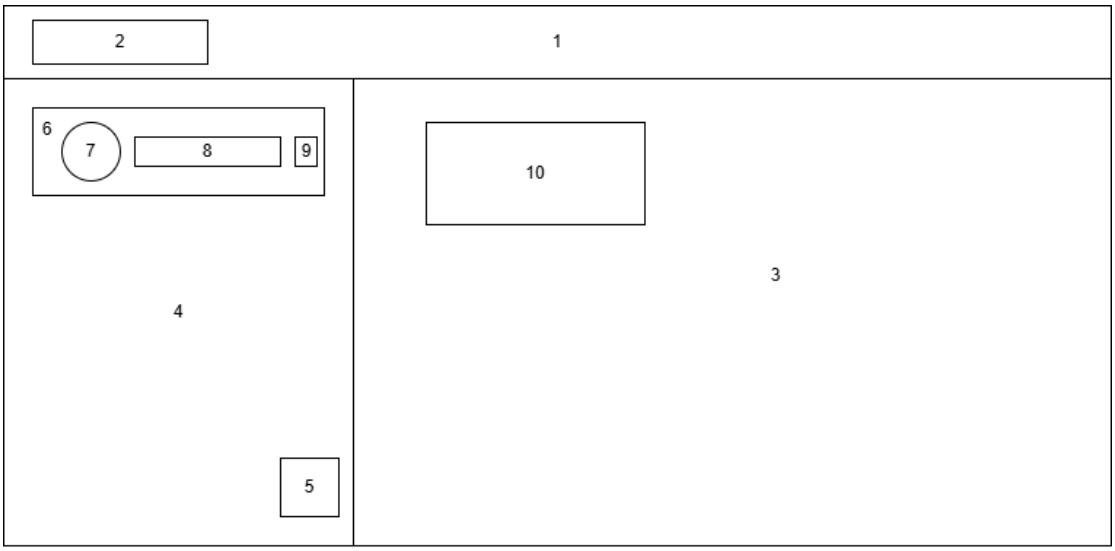


Рисунок 2.1 – Схема інтерфейсу головної сторінки

У схемі були використані такі ключові компоненти Flutter:

1) AppBar – верхня навігаційна панель, що відображається на всіх сторінках і містить заголовок та інші дії.

2) AppBarTitle – текстовий елемент, що задає назву в межах AppBar.

3) Content – основна вкладка, яка відображає список відео для аналізу.

4) Subscription – вкладка зі списком підписок користувача на YouTube-канали.

5) FloatingActionButton – кнопка швидкої дії, яка відкриває діалогове вікно для додавання нового каналу.

6) Card – контейнер, що представляє окремий канал у списку підписок, візуально виділяючи його.

7) CircleAvatar – елемент, що виводить аватар обраного YouTube-каналу всередині картки.

8) Card.title – назва каналу, відображена в межах відповідної картки.

9) PopupMenuButton – кнопка, що відкриває меню з такими додатковими діями як встановлення промпту та видалення каналу.

10) ContentGridItem – плитка, яка представляє окреме відео у вкладці Content.

Незважаючи на наявність схожих інструментів, більшість із них не пропонують користувачам гнучкого механізму взаємодії з відеоконтентом YouTube у контексті персоналізованого аналізу. Саме тому було прийнято рішення про створення власного інтерфейсу вебзастосунку, в якому користувачі мають можливість додавати канали за псевдонімом, перевіряти їхню валідність, переглядати список відео обраного автора та аналізувати вміст відео в окремому діалоговому вікні. Такий підхід дозволяє сформувати логічну, інтуїтивну структуру взаємодії з системою, що є фундаментом для подальшої розробки повноцінного функціоналу персоналізованого аналізу відеоконтенту.

2.2 Розробка дизайну користувацького інтерфейсу на основі принципів UX/UI та колористики

Дизайн інтерфейсу відіграє ключову роль у сприйнятті системи користувачем, оскільки саме візуальна складова першою формує враження про програмний продукт [18]. Основними завданнями на цьому етапі розробки є забезпечення зручності, інтуїтивності, візуальної привабливості та відповідності ергономічним вимогам.

У процесі створення дизайну вебінтерфейсу для системи персоналізованого аналізу відео YouTube було прийнято рішення використовувати мінімалістичний та сучасний стиль, у якому акцент зроблено на чистоту інтерфейсу, зрозумілу ієрархію елементів та логіку взаємодії. Основна палітра кольорів побудована на темних тонах із фіолетовими відтінками, що відповідає вимогам доступності та забезпечує комфортну взаємодію протягом тривалого часу.

Інтерфейс застосунку поділяється на кілька основних екранів. Першим і обов’язковим кроком є сторінка авторизації, яка реалізована у вигляді окремого інтерфейсного компонента. Для входу в систему користувач повинен ввести верифікаційний код, який він отримує через спеціального Telegram-бота. Така реалізація забезпечує як зручність, так і безпеку доступу. Інтерфейс сторінки

авторизації містить коротке інструктивне повідомлення, поле введення коду, кнопку підтвердження, а також індикацію помилок або успішної авторизації.

Після успішної авторизації користувач потрапляє на основну частину інтерфейсу, яка поділена на дві основні вкладки:

Subscriptions – реалізовано бокову панель, у якому відображається список каналів користувача. Передбачено інтуїтивне поле для введення псевдоніма та кнопки «Додати», «Видалити» і «Встановити промпт».

Content – розміщується у центральній області інтерфейсу. Після вибору каналу зі списку підписок, система відображає доступні відео у вигляді сітки карток або списку.

Діалогове вікно перегляду текстів розроблено у вигляді модального компоненту з двома вкладками: «Оригінальний текст» і «Модифікований текст». Якщо для каналу не задано промпт, блок із модифікованим текстом міститиме відповідне повідомлення.

У розробці дизайну було враховано принципи адаптивності, що дозволяє коректно відображати інтерфейс на різних пристроях, включаючи планшети та ноутбуки з різною роздільною здатністю. Також дотримано принципів послідовності стилів, інформативності елементів, контрастності шрифтів та візуальної ієрархії, що підвищує загальну юзабіліті системи.

У результаті розробки вдалося створити інтерфейс, який є водночас простим у використанні, естетично приємним і функціонально готовим до подальшого масштабування. Це створює підґрунтя для розширення можливостей системи, зокрема впровадження інструментів пошуку, фільтрації та візуалізації аналітики.

2.3 Розробка моделі фронтенд системи аналізу відео YouTube

Ефективне функціонування вебзастосунку для персоналізованого аналізу відео на YouTube вимагає чіткого розмежування відповідальностей між його складовими, що досягається шляхом моделювання структури системи. Основна мета цього процесу – побудова логічної архітектури, яка забезпечить

масштабованість, зручну підтримку та гнучке розширення функціоналу в майбутньому.

У межах архітектурної моделі основну роль відіграє клієнтська частина, яка забезпечує ключові функції взаємодії користувача із системою. Зокрема, реалізовано авторизацію через Telegram, роботу з каналами, вибір відео для аналізу, перегляд результатів, а також інтегровано криптогаманець. Останній дозволяє фіксувати історію поповнень та витрат під час запитів до оригінального та модифікованого тексту для обраного відео. Така функціональність є важливою частиною персоналізації сервісу, оскільки забезпечує користувача повною інформацією про використані ресурси в межах системи.

Загальну структуру компонентів, зосереджену на клієнтській частині, наведено у таблиці 2.1.

Таблиця 2.1 – Опис модулю фронтенд

Компонент	Технологія	Функціональні обов’язки
1	2	3
Фронтенд	Flutter (Dart)	– Авторизація через Telegram (введення коду) – Інтерфейс вкладок Subscriptions і Content – Вибір відео та перегляд текстів у модальному вікні – Керування каналами: додавання, видалення, встановлення промптів – Інтеграція з API бекенду та мовної моделі – Відображення балансу, історії поповнень і витрат у криптогаманці

Крім логічної архітектури, було також розроблено загальну модель роботи програми у вигляді діаграми, що забезпечує зв’язок між користувачем, підписками, відео та результатами аналізу, яка зображена на рисунку 2.2.

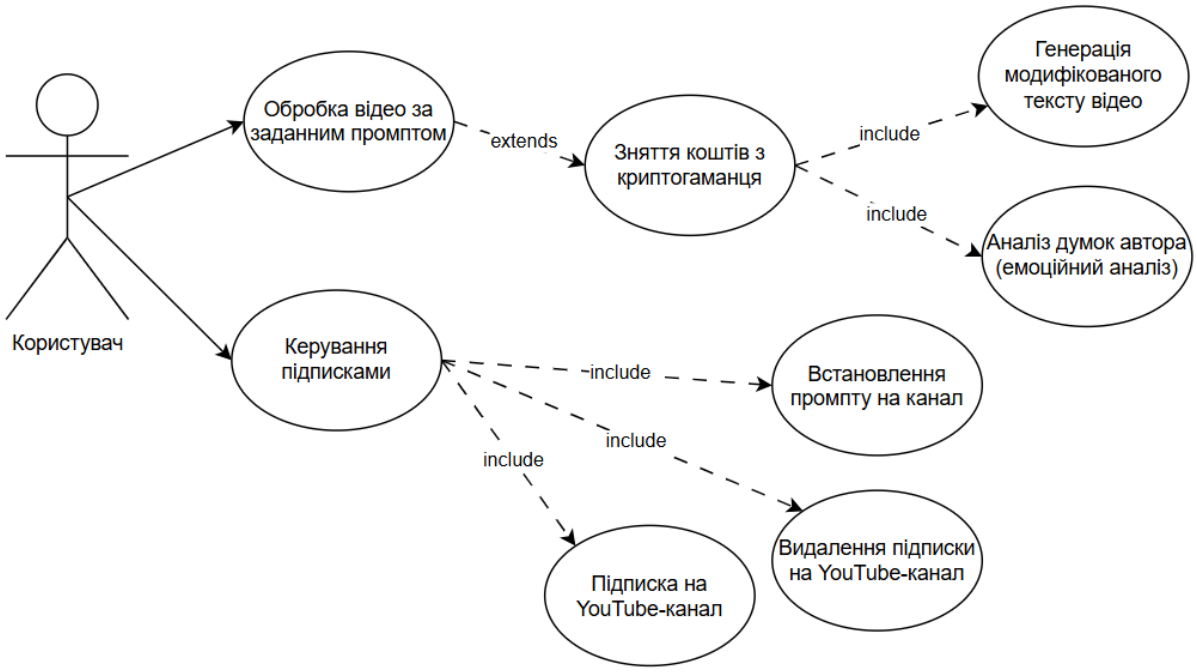


Рисунок 2.2 – Загальна модель роботи програми

Дана діаграма демонструє загальну логіку роботи системи з точки зору користувача. Передбачено можливість взаємодії з підписками, перегляду оригінальних і модифікованих текстів відео, а також персоналізованих запитів через налаштування промптів. Всі мережеві запити до серверної логіки та мовної моделі здійснюються через визначені API. Структура передбачає масштабованість і дає змогу легко додавати нові функції, такі як аналітика, фільтрація або візуалізація текстових даних, без порушення існуючої архітектури. Графічну схему було побудовано за допомогою інструмента draw.io [19].

2.4 Розробка алгоритмів роботи програмних модулів аналізу відеоконтенту з використанням нейромереж

Алгоритми, що лежать в основі аналізу відеоконтенту в розробленій системі, спрямовані на послідовне отримання, обробку й модифікацію текстової інформації з відео за допомогою нейромереж. Ці алгоритми дозволяють

персоналізувати зміст відповідно до очікувань користувача, шляхом генерації адаптованих інтерпретацій оригінального тексту.

Основні етапи обробки реалізовано у вигляді взаємодії між фронтендом, бекендом та нейромережним модулем. Алгоритм функціонування програмних модулів можна умовно поділити на три ключові блоки.

Загальна послідовність виконання алгоритмів роботи системи відображена на блок-схемі, що наведена на рисунку 2.3. Вона ілюструє основні етапи: авторизацію користувача, взаємодію з підписками та відеоконтентом, а також обробку текстової інформації нейромережею для подальшого персоналізованого аналізу.

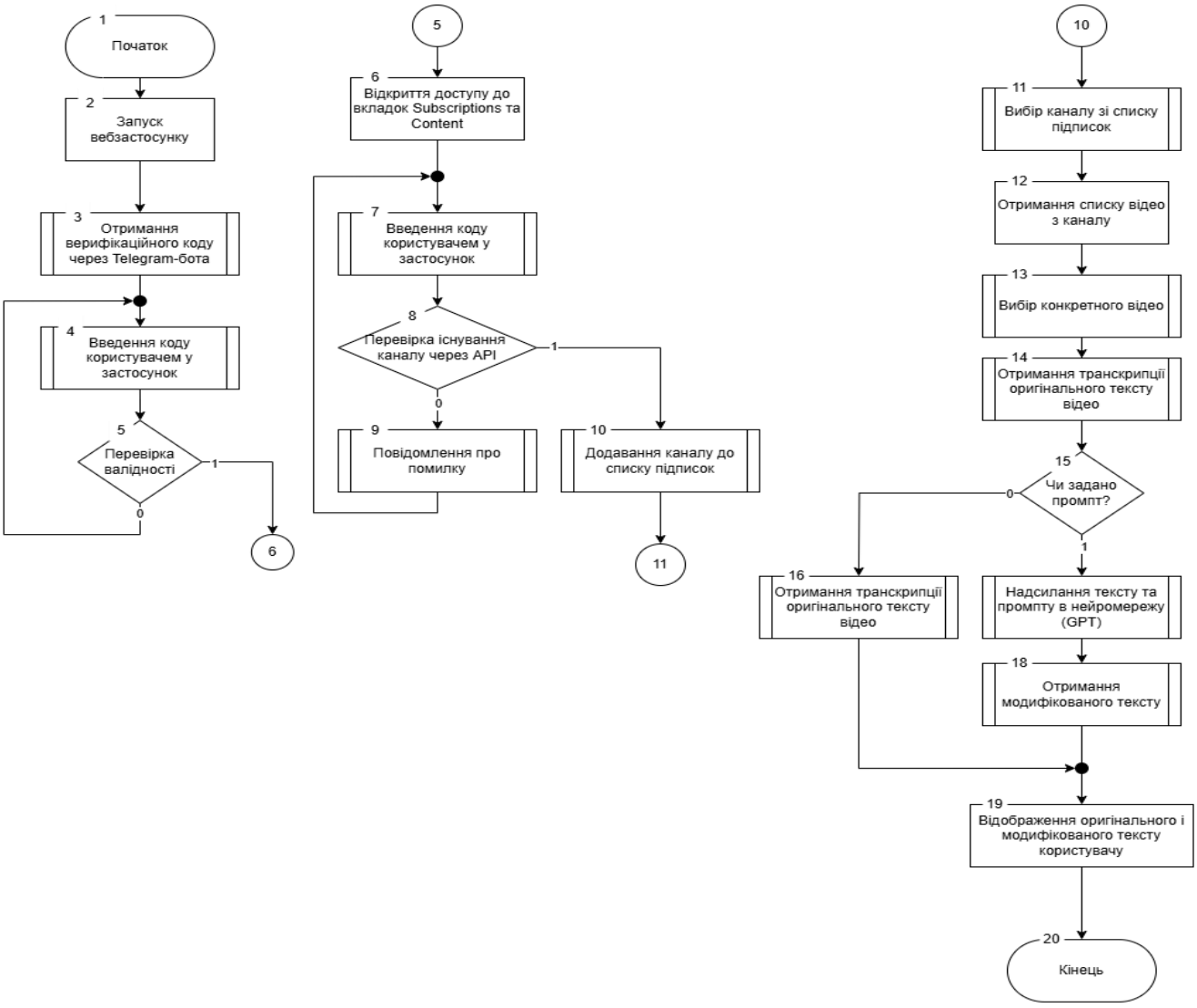


Рисунок 2.3 – Блок-схема виконання алгоритмів роботи системи

Авторизація користувача:

1) Користувач відкриває вебзастосунок і потрапляє на сторінку авторизації.

2) Користувач отримує верифікаційний код через Telegram-бота.

3) Введенням коду в застосунку він підтверджує особу.

4) Після підтвердження користувач отримує доступ до вкладок із підписками та відео.

Робота з підписками та відео:

1) Користувач вводить псевдонім автора YouTube-каналу.

2) Застосунок надсилає запит на сервер для перевірки валідності каналу через API.

3) Якщо канал існує, він додається у вкладку Subscriptions.

4) Користувач обирає канал і переходить до вкладки Content, де отримує список відео цього каналу.

5) При виборі відео відбувається запит до сервера на отримання розшифрованого тексту (транскрипції).

6) Оригінальний текст відображається у діалоговому вікні.

7) Якщо до каналу встановлено промпт – текст надсилається в нейромережний модуль для генерації модифікованого тексту.

8) Результат (модифікований текст) повертається і відображається поруч з оригіналом.

Взаємодія з нейромережею:

1) Алгоритм генерації модифікованого тексту:

2) Сервер формує запит у вигляді пари «оригінальний текст + промпт».

3) Запит надсилається до нейромережного модуля типу GPT.

4) Модель аналізує текст і формує відповідь згідно з вказаною інструкцією (промптом).

5) Отриманий модифікований текст повертається на сервер і передається у фронтенд для відображення.

Такий підхід дозволяє користувачам отримувати адаптовані версії тексту з відео відповідно до власних цілей – наприклад, скорочену версію, текст, переформульований під певну тематику, або спрощене пояснення.

Алгоритми створені таким чином, щоб забезпечити масштабованість, розширюваність та високу чутливість до якості вхідних промптів. Надалі можливе впровадження більш гнучких механізмів редагування промптів або навчання власної кастомної моделі для ще більш точної персоналізації аналізу [20].

2.5 Висновки

У цьому розділі було проведено детальний аналіз поставленої задачі та виконано формалізацію вхідних даних, необхідних для реалізації вебзастосунку для персоналізованого аналізу відео з платформи YouTube. Було розглянуто основні вимоги до функціоналу системи, а також обґрунтовано доцільність її розробки в умовах зростаючого обсягу відеоконтенту та потреби в його адаптивному сприйнятті.

Під час проєктування інтерфейсу враховувались принципи зручності, логічності структури та послідовності користувацького досвіду. Окрему увагу приділено дизайну: вибрано сучасний стиль для візуальної простоти. Сформовано структурну модель системи та описано алгоритми роботи ключових програмних модулів, зокрема модулів отримання відео, аналізу тексту з використанням нейромереж і генерації результату відповідно до промпту користувача.

Проведене моделювання та опис алгоритмів створює надійну основу для подальшої реалізації програмного продукту, який буде здатен ефективно обробляти інформацію з відео YouTube відповідно до індивідуальних потреб користувачів.

3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ СИСТЕМИ ПЕРСОНАЛІЗОВАНОГО АНАЛІЗУ ВІДЕО YOUTUBE

3.1 Варіантний аналіз і обґрунтування вибору засобів реалізації

У межах створення системи для персоналізованого аналізу відео з YouTube виникла потреба обрати оптимальні засоби реалізації, що відповідали б вимогам масштабованості, продуктивності та можливості подальшого розвитку системи. Було проведено варіантний аналіз різних підходів і технологій для реалізації як серверної, так і клієнтської частини застосунку.

У якості основи для серверної частини було обрано стек технологій .NET у поєднанні з архітектурними підходами Clean Architecture та Domain-Driven Design (DDD). Такий вибір обумовлений бажанням досягти високої гнучкості коду, ізоляції бізнес-логіки від інфраструктурних залежностей та зручності масштабування проєкту. Крім того, .NET забезпечує продуктивне середовище для побудови високонавантажених систем, що є критично важливим у контексті обробки відеоконтенту.

Для організації взаємодії між сервісами використано мікросервісну архітектуру з вертикальним розподілом відповідальностей. Кожен мікросервіс відповідає за окрему бізнес-функцію: обробку відео, зберігання даних, роботу з каналами, генерацію текстів тощо. Для комунікації між сервісами застосовано два брокери повідомлень: RabbitMQ, що відповідає за внутрішню подієву взаємодію між мікросервісами, та Azure Service Bus, який використовується для обміну повідомленнями між системою та зовнішніми службами, наприклад, телеграм-ботом [21].

Зберігання даних реалізовано з використанням двох баз даних. PostgreSQL застосовується для структурованих реляційних даних, таких як інформація про користувачів, канали та відео. Hive використовується для зберігання та аналізу великих обсягів текстових даних, зокрема – результатів транскрибування та модифікації відео за допомогою нейромереж.

Для хостингу та розгортання інфраструктури використовується хмарна платформа Microsoft Azure, яка забезпечує високу надійність, масштабованість і можливість гнучкого керування обчислювальними ресурсами [22].

Клієнтська частина системи реалізована за допомогою фреймворку Flutter, який дозволяє створювати сучасний, швидкий та кросплатформений інтерфейс. Такий підхід забезпечує зручність роботи з системою як з мобільних пристроїв, так і з браузера.

У результаті варіантного аналізу обрано сучасний та надійний набір технологій, який дозволяє створити масштабовану, гнучку й ефективну систему аналізу відеоконтенту з використанням нейромереж.

3.2 Аналіз програмних модулів системи персоналізованого аналізу відео YouTube

Система персоналізованого аналізу відео на YouTube побудована за принципами Clean Architecture у поєднанні з підходом Domain-Driven Design, що дозволило розділити її логіку на окремі рівні: доменний, аплікаційний, інфраструктурний і зовнішній [23]. Це забезпечило чітке структурування програмного коду та спростило підтримку, тестування й масштабування системи.

Така структурна організація дозволяє зосередити функціональність кожного модуля на окремому завданні, що значно полегшує розробку, тестування та внесення змін. Завдяки цьому кожен компонент системи можна оновлювати або масштабувати незалежно, не впливаючи на роботу інших частин застосунку, що особливо важливо для мікросервісної архітектури.

Кожна бізнес-функція системи винесена в окремий мікросервіс, що дозволяє незалежно керувати її логікою та забезпечує гнучкість при розгортанні в хмарному середовищі Azure. Мікросервіси взаємодіють між собою за допомогою RabbitMQ, а зовнішні запити обробляються через Azure Service Bus. Така схема дозволяє обробляти великі обсяги даних у режимі реального часу без втрати продуктивності.

До основних модулів системи належать:

– Модуль автентифікації, який реалізує авторизацію користувача за допомогою верифікаційного коду, отриманого через Telegram-бота. Успішна авторизація відкриває доступ до основного функціоналу системи.

– Модуль керування підписками, що надає можливість додавати YouTube-канал за псевдонімом, перевіряти його валідність на сервері та зберігати у вкладці Subscriptions. Передбачено функції видалення каналу або призначення для нього промпту.

– Модуль отримання відео, що відповідає за завантаження відео з обраного каналу та відображення їх у вкладці Content. Відео можна переглядати в діалоговому вікні разом із текстом після обробки.

– Модуль нейромережевого аналізу, що реалізує обробку відео за заданим промптом. Користувач має можливість отримати змінений текст (на основі оригінального) після обробки через нейромережеву модель. Цей модуль ізольований і взаємодіє з іншими через чергу повідомлень.

– Модуль збереження та обробки даних, що базується на PostgreSQL для зберігання структурованих даних (користувачі, канали, відео), а також Hive – для масштабованого зберігання результатів транскрипції та текстового аналізу.

– Модуль телеграм-бота, що обслуговує взаємодію з користувачем для автентифікації та верифікації.

Кожен модуль містить відповідний набір класів, сервісів, репозиторіїв та DTO, що відповідає принципам SOLID та дозволяє легко адаптувати бізнес-логіку. Для взаємодії між модулями розроблені контракти у вигляді DTO та подій.

На рисунку 3.1 наведено UML-діаграму послідовностей взаємодії основних модулів системи персоналізованого аналізу відео з YouTube, яка демонструє ключові етапи обробки запиту користувача – від авторизації до отримання модифікованого тексту на основі аналізу відео.

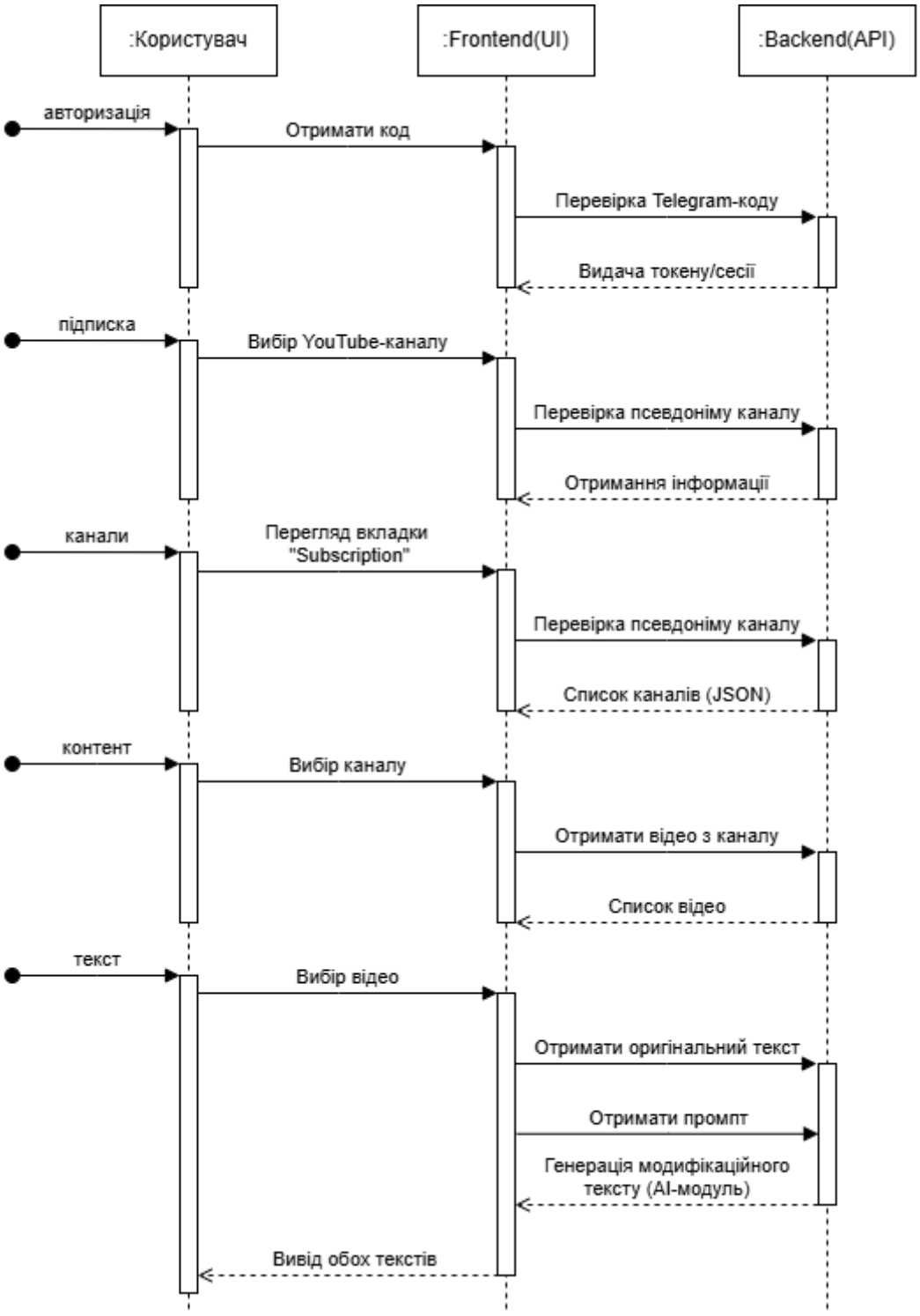


Рисунок 3.1 – Діаграма послідовностей

3.3 Реалізація модуля авторизації користувача

Першим екраном, з яким взаємодіє користувач, є сторінка авторизації. Вхід у систему реалізовано за допомогою одноразового верифікаційного коду, який

користувач отримує через Telegram-бота, прив’язаного до свого облікового запису. Це рішення було обрано з міркувань безпеки, оскільки такий підхід дозволяє уникнути зберігання паролів, використовуючи натомість тимчасовий код. Після того як код введено, на сервер надсилається запит на його перевірку. У разі успішної верифікації користувач отримує доступ до головного інтерфейсу застосунку, а інформація про авторизацію тимчасово зберігається у локальному сховищі за допомогою Hive, що дозволяє не вводити код при кожному відкритті застосунку.

Архітектурні особливості:

– Мікросервіс написаний на .NET.

– Telegram Bot працює як окремий сервіс.

– Авторизація базується на JWT-токенах.

– Використовується шина повідомлень для обміну подіями авторизації (наприклад, через RabbitMQ).

На рисунку 3.2 зображено програмну реалізацію авторизації користувача через Telegram-бота

```
[ApiController]
[Route("api/user/log-in")]
public class AuthController : ControllerBase
{
    private readonly IVerificationService _verificationService;

    public AuthController(IVerificationService verificationService)
    {
        _verificationService = verificationService;
    }

    [HttpPost("request-code")]
    public async Task<IActionResult> RequestVerificationCode([FromBody] AuthRequestDto request)
    {
        await _verificationService.SendVerificationCodeAsync(request.PhoneNumber);
        return Ok("Код надіслано через Telegram.");
    }

    [HttpPost("verify-code")]
    public async Task<IActionResult> VerifyCode([FromBody] VerificationDto dto)
    {
        var token = await _verificationService.VerifyCodeAsync(dto.PhoneNumber, dto.Code);
        if (token == null)
            return Unauthorized("Невірний код.");

        return Ok(new { token });
    }
}
```

Рисунок 3.2 – Авторизація користувача через Telegram-бота

Основні компоненти:

– AuthController.cs – приймає запити авторизації.

– TelegramBotService.cs – відправляє верифікаційний код.

– VerificationService.cs – перевіряє код.

– UserService.cs – управляє користувачами в БД.

Для реалізації механізму авторизації користувача через Telegram-бота використовується послідовність перевірок і взаємодій між клієнтською частиною, сервером і самим ботом, що схематично подано на рисунку 3.3.

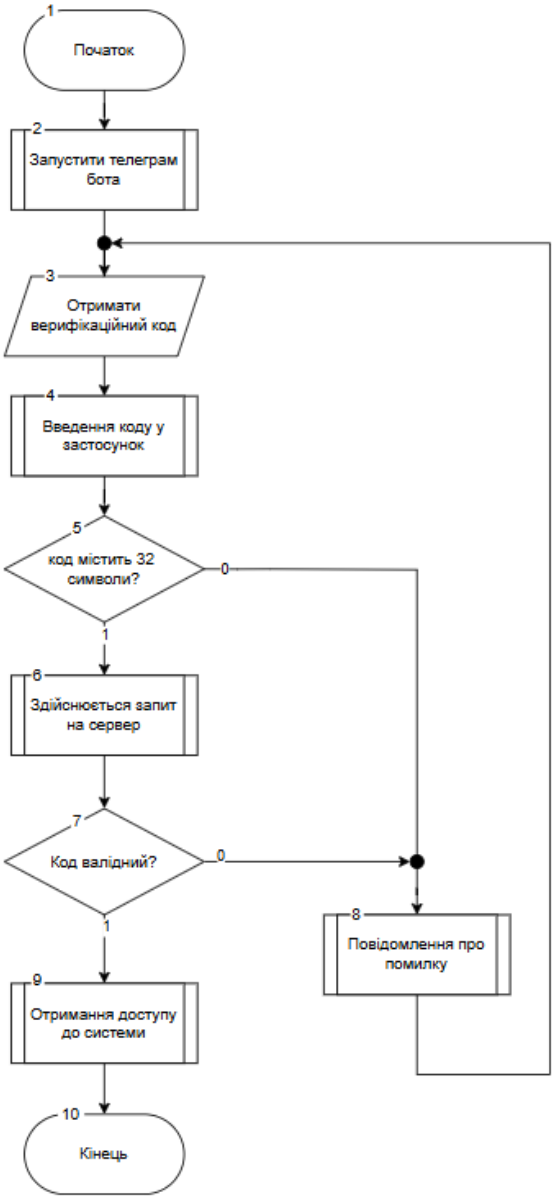


Рисунок 3.3 – Алгоритм авторизації користувача через Telegram-бота.

3.4 Реалізація модуля керування підписками

Модуль керування підписками реалізує функціонал, пов’язаний із додаванням, перевіркою, видаленням та встановленням промпту для YouTube-каналів, на які користувач хоче підписатися. Основний функціонал користувача починається з головного екрана, який містить дві основні вкладки: Subscriptions та Content. У вкладці Subscriptions реалізовано можливість додавання YouTube-каналів за допомогою введення псевдоніма автора. При цьому здійснюється перевірка валідності введеного псевдоніма через запит до сервера: якщо канал існує, він зберігається у списку підписок. Для кожного підписаного каналу користувач має можливість встановити індивідуальний промпт – текстову інструкцію для модифікації змісту відео під час обробки. Також, за необхідності, канал можна видалити. Завдяки такій логіці взаємодії користувач може гнучко формувати перелік авторів, чий контент його цікавить, а також керувати контекстом аналізу.

На серверному боці ключовим елементом є контролер ChannelController, який обробляє запити на додавання каналу, перевірку псевдоніма, отримання списку каналів користувача та редагування промптів. Реалізації методу для додавання нового підписаного каналу зображена на рисунку 3.4

```
// Метод у ChannelController для додавання нового каналу
[HttpPost("add-channel")]
public async Task<IActionResult> AddChannel([FromBody] AddChannelRequest request)
{
    var result = await _channelService.AddChannelAsync(request.UserId, request.Alias);
    if (!result.IsValid)
        return BadRequest(result.ErrorMessage);

    return Ok(result.Channel);
}
```

Рисунок 3.4 – Оформлення підписки на новий канал

У свою чергу, клієнтська частина на Flutter реалізує відображення підписок через віджет Card, який містить аватар каналу, назву, кнопку контекстного меню (PopupMenuButton) для видалення або встановлення промпту. Подання відповіді

сервера адаптується через відповідні DTO, які передаються через Service Layer до UI.

Процес додавання YouTube-каналу до списку підписок користувача реалізується на основі перевірки існування каналу за введеним псевдонімом. Деталі реалізації схематично наведено на рисунку 3.5

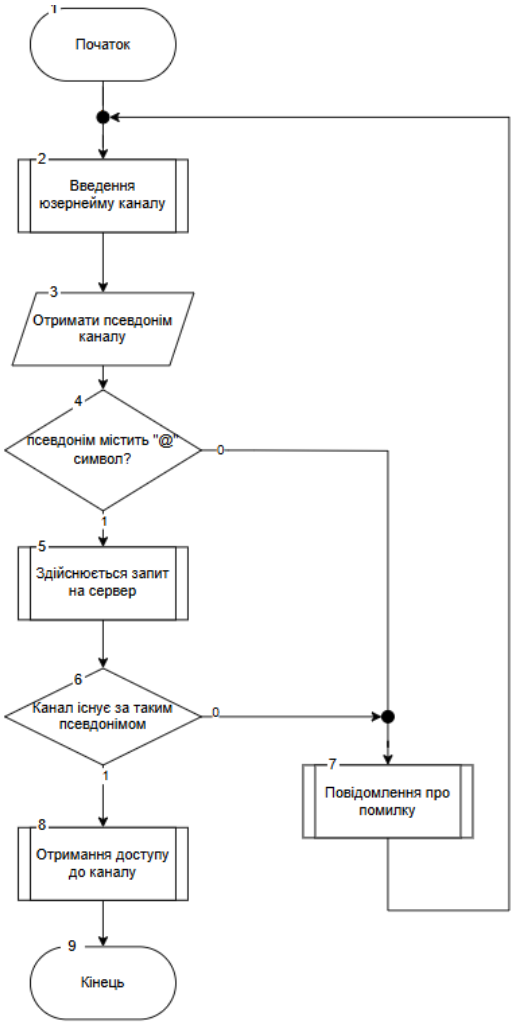


Рисунок 3.5 – Алгоритм додавання YouTube-каналу за псевдонімом.

3.5 Реалізація модуля отримання відео для вибраного каналу

Модуль отримання відео відповідає за завантаження відеоматеріалів з обраного каналу та відображення їх у вкладці Content. Його основна задача – надати користувачеві доступ до списку відео, які можна проаналізувати. Уся логіка передачі даних побудована на основі взаємодії з REST API, який

обслуговується бекендом, реалізованим на платформі .NET у межах мікросервісної архітектури. Взаємодія між окремими мікросервісами організована через два брокери повідомлень: Azure Service Bus та RabbitMQ, які відповідають за стабільну та надійну обробку запитів. Після вибору каналу на клієнті відбувається запит до бекенду, який повертає список відео цього автора. Кожне відео подається у вигляді картки, при натисканні на яку відкривається діалогове вікно з доступом до текстового вмісту (див. рисунок 3.6).

```
Future<List<Video>> fetchChannelVideos(String channelId) async {
  final response = await http.get(Uri.parse('$baseUrl/videos/$channelId'));
  if (response.statusCode == 200) {
    final jsonData = json.decode(response.body) as List;
    return jsonData.map((video) => Video.fromJson(video)).toList();
  } else {
    throw Exception('Failed to load videos');
  }
}
```

Рисунок 3.6 – Отримання списку відео з бекенду через REST API

Взаємодія користувача із вкладкою Subscriptions для перегляду списку відео реалізується за допомогою запиту на сервер, як зображено на рисунку 3.7.

Рисунок 3.7 – Алгоритм отримання відео для вибраного каналу

3.6 Розробка модуля нейромережевого аналізу

Модуль нейромережевого аналізу є ключовим компонентом системи персоналізованого аналізу, який відповідає за обробку відео на основі заданого промпту користувача. Обробка починається з клієнтської взаємодії: після вибору каналу у вкладці Subscriptions, користувач переходить до вкладки Content, де відображається список відео, отриманих із сервера. Відображення базується на запиті до бекенду, який повертає як оригінальний текст, згенерований за допомогою OpenAI Whisper, так і модифікований текст, сформований на основі встановленого промпту. Обидва варіанти тексту показуються у діалоговому вікні, що відкривається після натискання на відео.

На рисунку 3.8 зображена логіка обробки, яка реалізована у відповідному контролері, де за запитом до відео виконується транскрипція за допомогою OpenAI Whisper, після чого результат передається до моделі генерації тексту, якщо промпт користувача заданий.

```
[HttpGet("get-video-details/{channelId}/{videoId}")]
public async Task<IActionResult> GetVideoDetails(string channelId, string videoId)
{
    try
    {
        var video = await _videoService.GetVideoByIdAsync(channelId, videoId);
        if (video == null)
        {
            return NotFound(new { isOk = false, errors = new[] { "Video not found" } });
        }
        string originalText = await _whisperService.TranscribeAsync(video.Url);
        string modifiedText = string.Empty;
        if (!string.IsNullOrWhiteSpace(video.Prompt))
        {
            modifiedText = await _gptService.ModifyTextAsync(originalText, video.Prompt);
        }
        var result = new
        {
            isOk = true,
            errors = Array.Empty<string>(),
            value = new
            {
                items = new[]
                {
                    new {
                        id = video.Id,
                        title = video.Title,
                        originalText = originalText,
                        modifiedText = modifiedText
                    }
                }
            }
        };

        return Ok(result);
    }
    catch (Exception ex)
    {
        return StatusCode(500, new { isOk = false, errors = new[] { ex.Message } });
    }
}
```

Рисунок 3.8 – Логіка транскрипції відео за допомогою нейромережі

Цей підхід дозволяє гнучко масштабувати обробку та модифікацію контенту, зберігаючи високий рівень персоналізації. Важливо, що взаємодія з нейромережею відбувається на рівні окремого сервісу, whisperService для транскрипції та gptService для генерації тексту, що відповідає принципам чистої архітектури та забезпечує незалежність окремих компонентів системи.

Отримання та обробка транскрипцій відео, а також формування модифікованого тексту на основі заданого промпту, схематично зображено на рисунку 3.9.

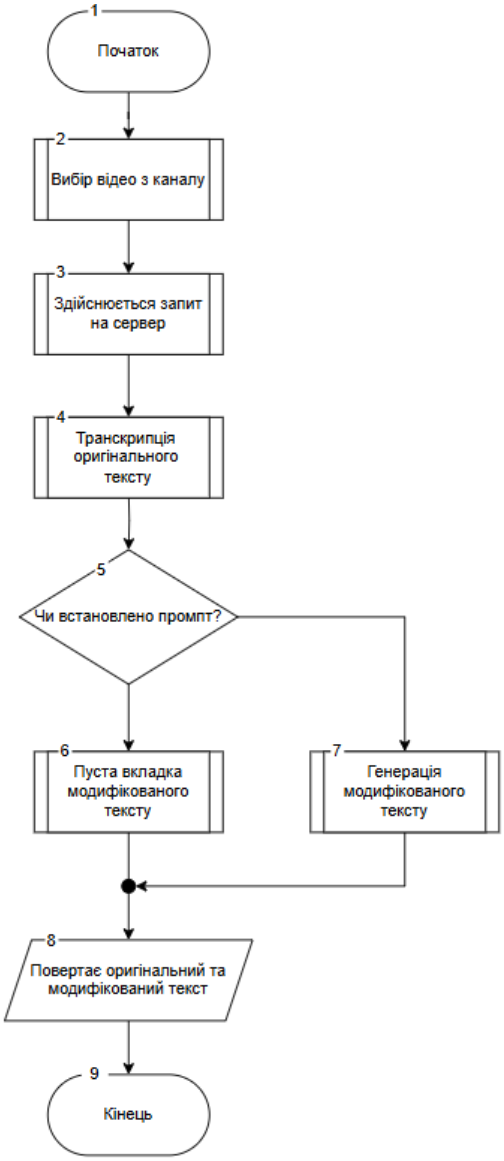


Рисунок 3.9 – Алгоритм перегляду оригінального та модифікованого тексту відео.

3.7 Висновки

У третьому розділі було здійснено детальну розробку програмного забезпечення системи персоналізованого аналізу відеоконтенту з платформи YouTube. На основі варіантного аналізу обґрунтовано вибір технологій та підходів до реалізації, серед яких ключову роль відіграють Clean Architecture, принципи DDD, .NET-технології, мікросервісна архітектура та використання хмарної інфраструктури Azure з брокерами повідомлень Azure Service Bus та RabbitMQ. Для зберігання та обробки даних було обрано PostgreSQL як основну реляційну базу та Hive для локального кешування на клієнтській частині.

Було розглянуто розробку основних функціональних компонентів системи: модуль авторизації через Telegram-бота, додавання YouTube-каналів за псевдонімом, перегляд контенту з підписаних каналів, а також механізм отримання та трансформації текстової інформації з відео за допомогою нейромереж. Схематичне моделювання алгоритмів дозволило візуалізувати логіку роботи кожного з етапів взаємодії користувача з системою.

Таким чином, реалізований програмний продукт не лише відповідає поставленим вимогам до персоналізованого аналізу відео, але й демонструє потенціал до масштабування та інтеграції додаткових аналітичних інструментів у майбутньому.

4 ТЕСТУВАННЯ ПРОГРАМНОГО ЗАСТОСУНКУ

4.1 Методологія тестування програмного забезпечення

Тестування програмного забезпечення є невід'ємною частиною процесу розробки і гарантує стабільну, безпечну та передбачувану роботу системи. Для перевірки функціональності, зручності та надійності вебзастосунку, розробленого для персоналізованого аналізу відео з платформи YouTube, було обрано поєднання модульного (unit testing), інтеграційного та ручного тестування [24] .

Модульне тестування застосовувалося для перевірки окремих логічних компонентів бекенду, зокрема модулів обробки запитів до API YouTube, роботи з базами даних (PostgreSQL, Hive), взаємодії з нейронною мережею для модифікації тексту, а також з брокерами повідомлень RabbitMQ та Azure Service Bus. Тестування здійснювалося в межах кожного мікросервісу, що відповідає архітектурному підходу Clean Architecture і DDD.

Інтеграційне тестування проводилося для перевірки коректної взаємодії між мікросервісами та клієнтською частиною, реалізованою на Flutter. Окрема увага приділялася перевірці сценаріїв: додавання YouTube-каналу за псевдонімом, отримання списку відео, встановлення промпту до каналу, відображення діалогового вікна з текстами тощо. Було протестовано також обробку виключень, зокрема ситуацій, коли вказаний канал не існує або неможливо отримати дані з YouTube API.

Ручне тестування використовувалося для перевірки інтерфейсу користувача, його реакції на введення, а також дотримання UX/UI принципів. Особлива увага приділялася роботі вкладок Subscriptions і Content, доступності функцій авторизації через Telegram, та зручності навігації по системі.

Таким чином, комплексне використання декількох методів тестування дозволило забезпечити всебічну перевірку функціоналу застосунку як з технічної, так і з користувацької точок зору

4.2 Тестування програмного забезпечення

У межах процесу тестування було здійснено перевірку функціонування ключових компонентів програмного забезпечення, зосереджену як на технічній стабільності, так і на відповідності очікуваному функціоналу. Для ручного тестування REST API використовувався інструмент Postman, який дозволив наочно перевірити валідність відповідей на запити до серверної частини системи.

Першим етапом тестування була авторизація користувача. Запит за ендпоінтом GET /api/user/log-in імітує введення верифікаційного коду, що вказаний у полі x-token, який користувач отримує через Telegram-бота. У відповідь система надсилає JSON-об’єкт із полем isOk: true, що свідчить про успішний вхід до системи, як зображено на рисунку 4.1.

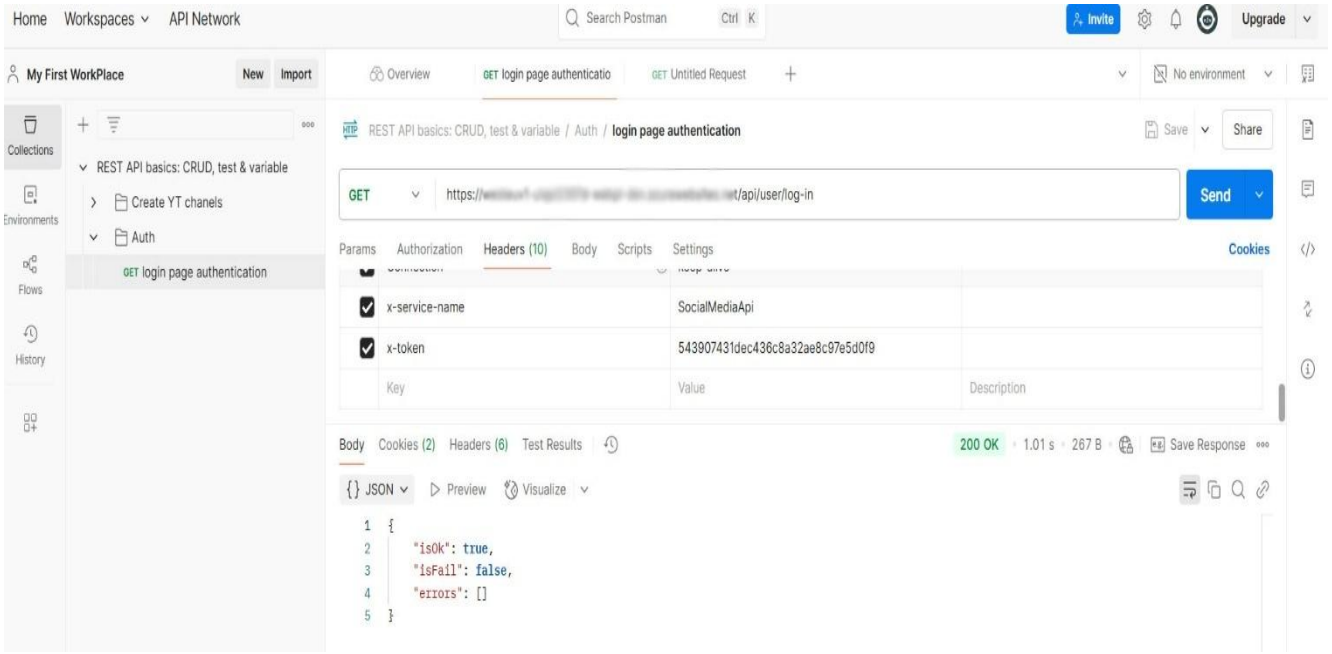


Рисунок 4.1 – Результат успішної авторизації користувача

Далі перевірялася можливість додавання YouTube-каналу за псевдонімом за допомогою запиту POST /api/youtube/add-channel/{channelUsername}, результат зображено на рисунку 4.2. При успішному виконанні користувач отримує підтвердження isOk: true, або повідомлення про наявні помилки у полі errors. Це дозволяє зрозуміти, чи існує канал і чи його можна додати.

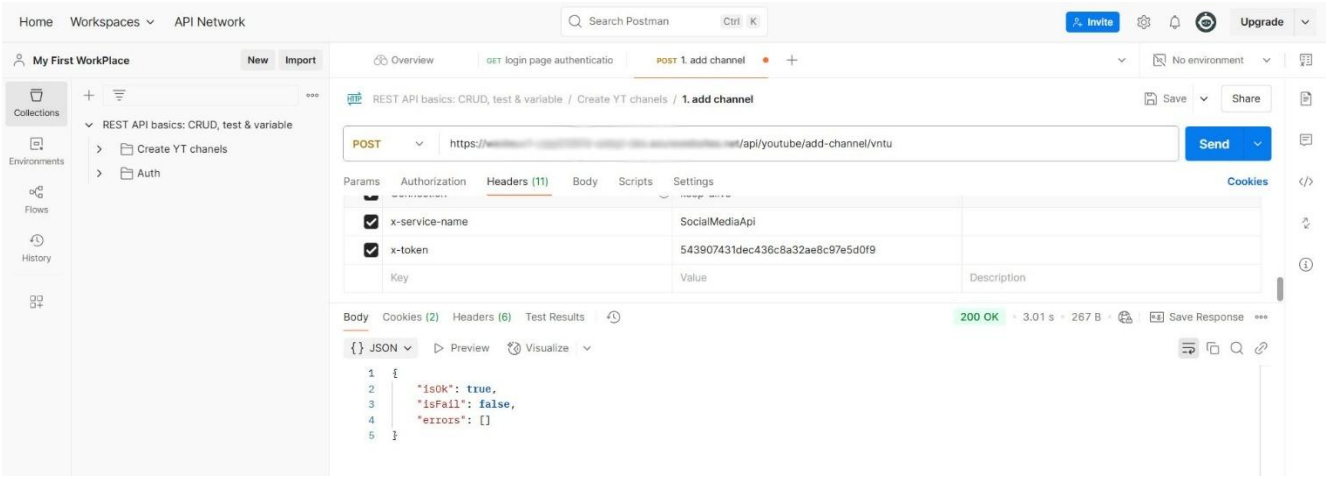


Рисунок 4.2 – Результат успішного додавання каналу

Після того, як канал було додано, тестувалася функціональність отримання списку підключених каналів через запит GET /api/youtube/get-channel-list. Цей запит повертає список всіх каналів, на які користувач оформив підписку, разом із метаданими про кожен канал, такими як id, title, imageUrl, aiPrompt та promtPropetries. Це дозволяє перевірити, чи правильно відображаються всі підключені канали та їх параметри.

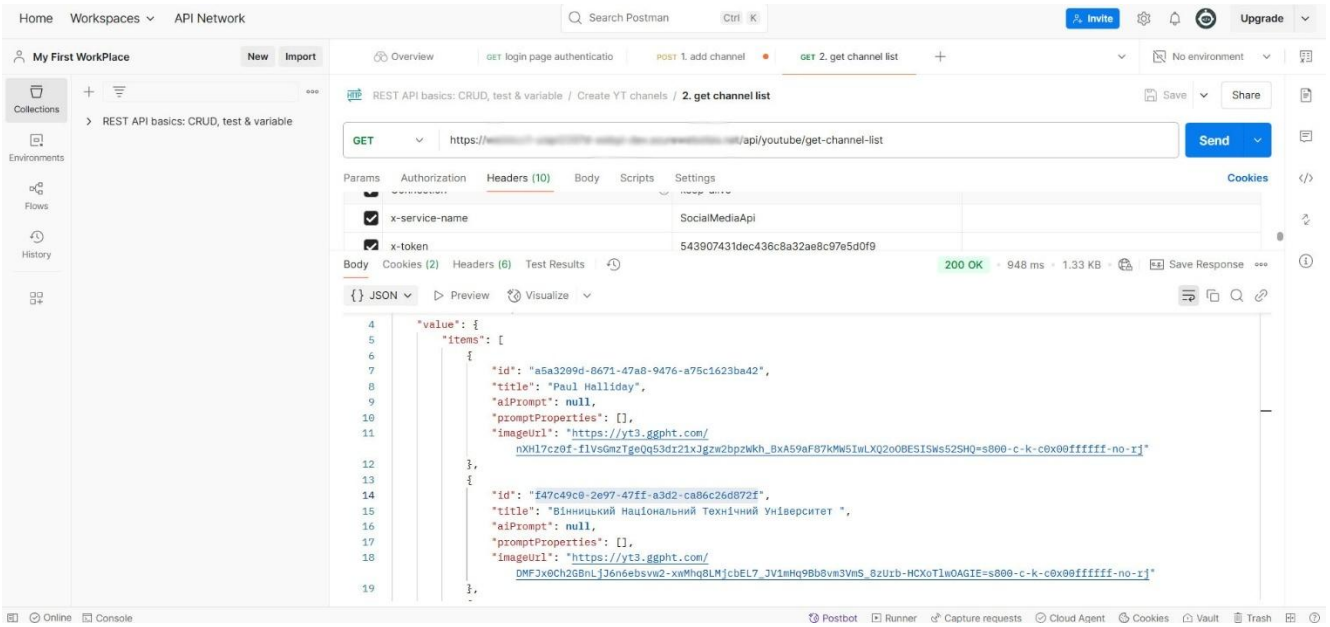


Рисунок 4.3 – Результат отримання списку каналів

Наступним кроком стало тестування запиту GET /api/youtube/get-channel-videos/{channelId} з метою отримання списку відео певного каналу. У відповіді повертається масив відео, кожен елемент якого містить id, title, externalId, recognitionState. У тестуванні перевірялася як структура, так і наявність контенту в полі items.

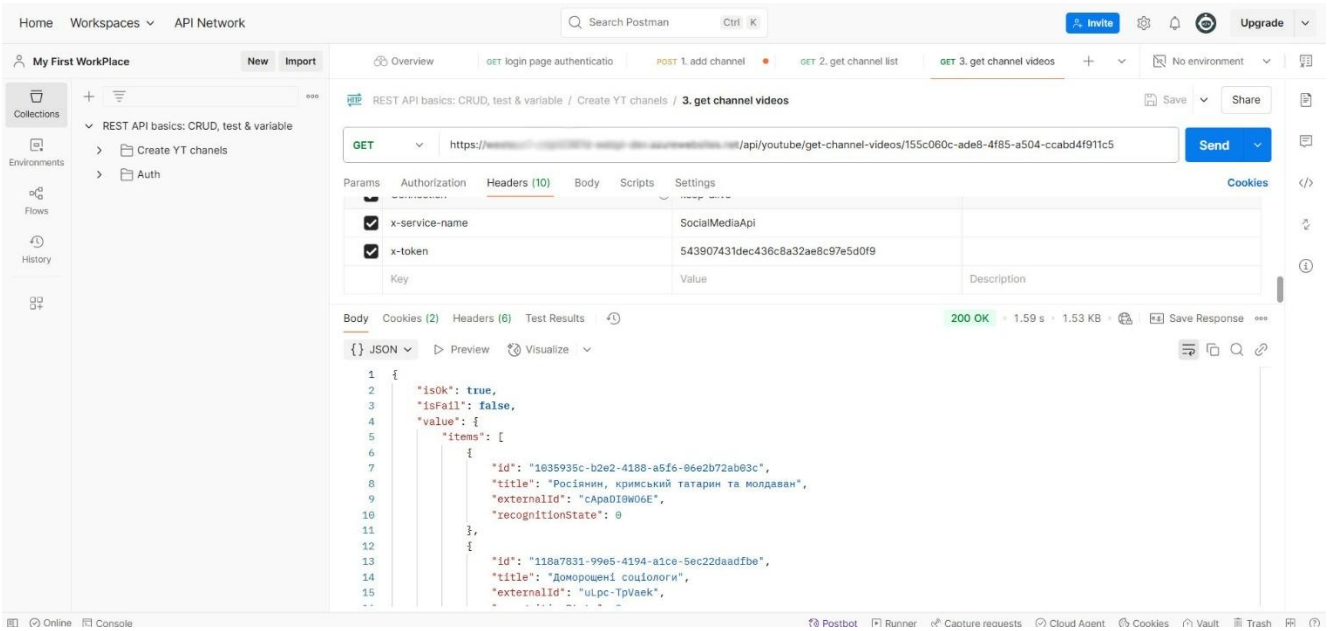


Рисунок 4.4 – Результат отримання списку відео з певного каналу

Усі тести підтвердили стабільність та узгодженість функціональності бекенду. Postman дозволив швидко виявити і виправити незначні помилки на етапі розробки, а також переконатися, що обробка запитів і повернення даних відбуваються згідно з очікуваними сценаріями.

Загалом, результати тестування підтвердили стабільну роботу основних функціональних модулів системи. Виявлені незначні недоліки були оперативно усунені на етапі тестування, що забезпечило підготовку якісного, надійного та зручного у використанні продукту.

4.3 Розробка інструкції користувача ПЗ

Інструкція користувача призначена для забезпечення зрозумілої та ефективної взаємодії користувача з вебзастосунком для персоналізованого

аналізу відео з YouTube на основі нейронних мереж. У цьому підрозділі наведено покроковий опис дій, які повинен виконати користувач для повноцінного використання функціональності програмного забезпечення.

Після відкриття вебзастосунку користувач бачить екран авторизації, як зображено на рисунку 4.5. Щоб увійти, користувачу необхідно ввести верифікаційний код, який надсилає Telegram-бот.

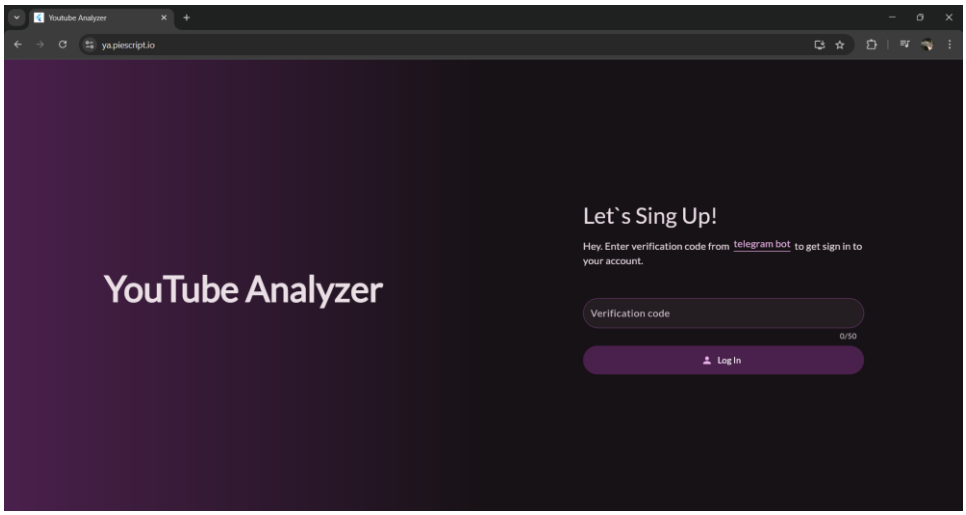


Рисунок 4.5 – Сторінка авторизації користувача

Якщо верифікаційний код неправильно записаний або є застарілим, вебсайт повідомить користувача про це, як зображено на рисунку 4.6.

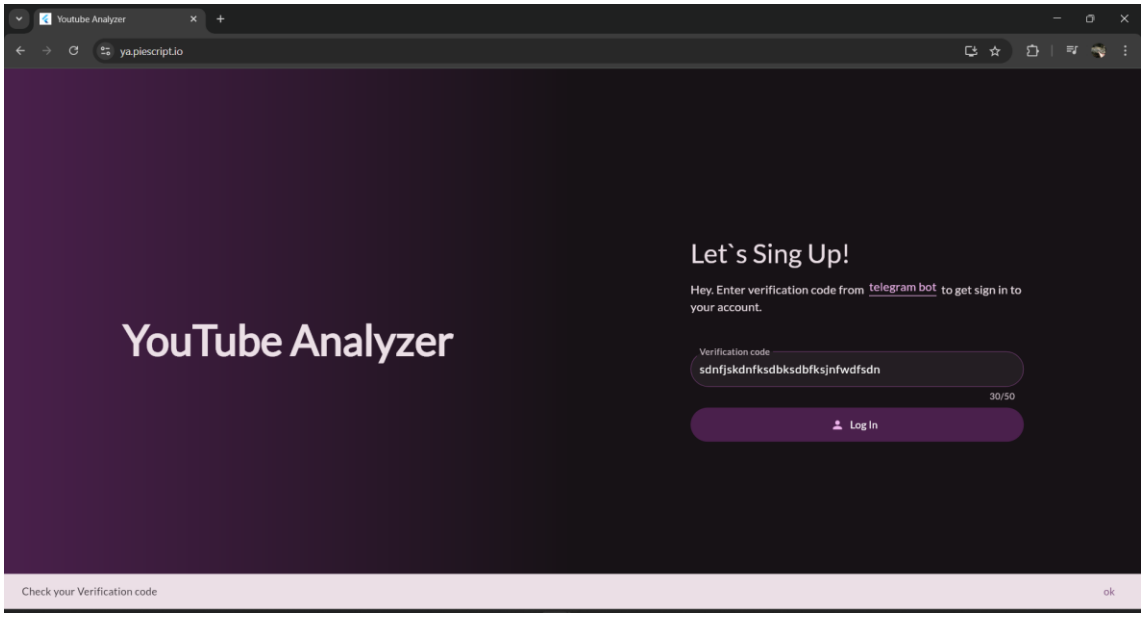


Рисунок 4.6 – Результат введення недійсного верифікаційного коду

Після введення правильного коду, користувач автоматично перенаправляється до головного інтерфейсу програми, де відображаються підписані канали.

На вкладці «Subscriptions» користувач вводить псевдонім (username) YouTube-каналу, який хоче додати, у поле що зображене на рисунку 4.7. Після натискання кнопки підтвердження, відправляється запит до серверу для перевірки існування каналу. У разі успіху канал додається до списку підписок, як зображено на рисунку.

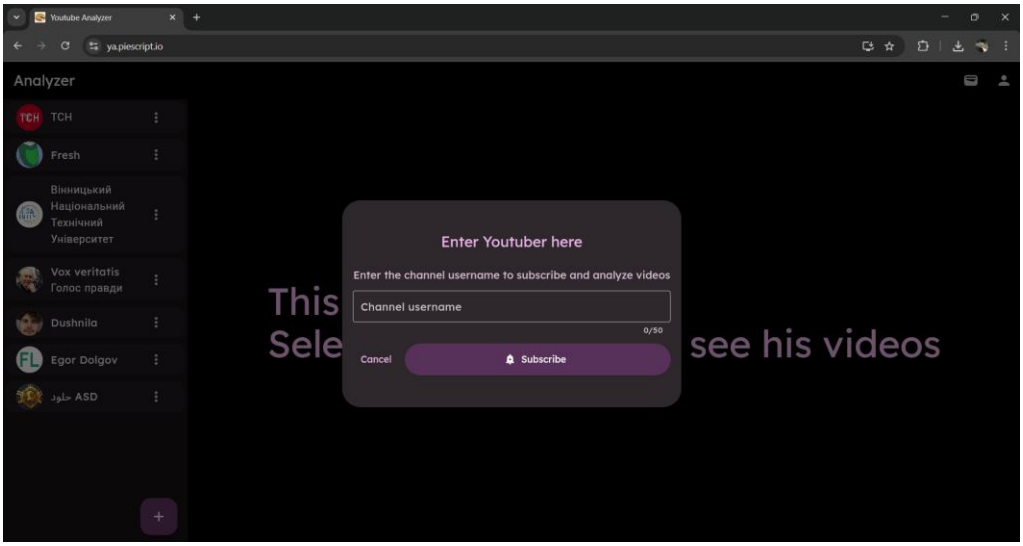


Рисунок 4.7 – Поле для оформлення підписки на канал за псевдонімом

Після додавання каналу він відобразиться у вкладці «Subscription», що знаходиться у лівій частині екрану, як зображено на рисунку 4.8.

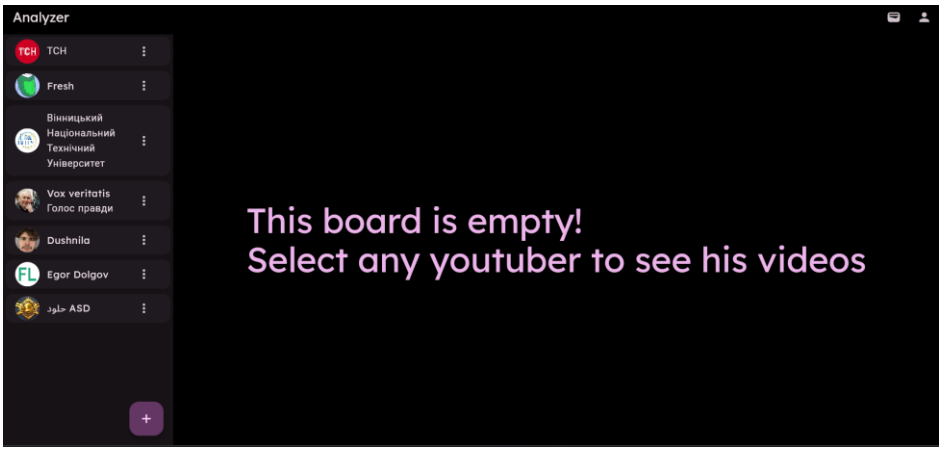


Рисунок 4.8 – Список каналі, на яких підписаний користувач

Після додавання каналу, користувач може перейти на вкладку «Content» де відображаються всі доступні відео обраного каналу. Для цього достатньо натиснути на картку потрібного каналу у вкладці підписок, і система автоматично отримає перелік відео, як зображено на рисунку 4.9.

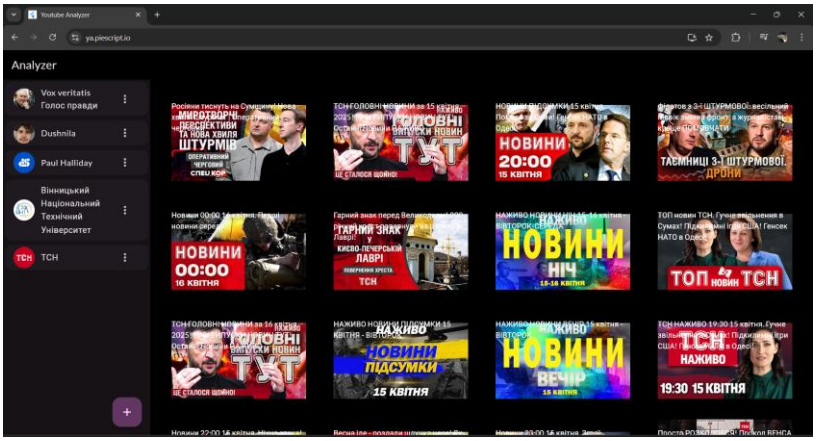


Рисунок 4.9 – Всі доступні відео обраного каналу ТСН

Щоб переконатись, що відео дійсно належать цьому каналу, на рисунку 4.10 зображено список останніх відео ТСН.

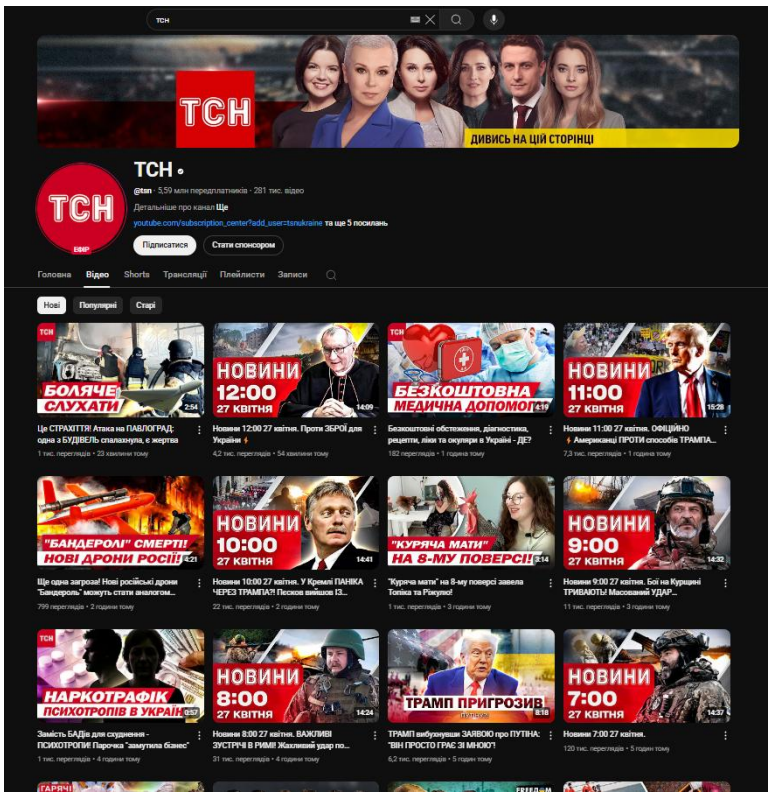


Рисунок 4.10 – Список відео каналу ТСН на YouTube

На рисунку 4.11 зображено інтерфейс реалізованого криптовалютного гаманця, що дозволяє кожному користувачу зберігати та поповнювати кошти в межах системи. Гаманець є частиною модулю монетизації функціоналу і слугує основою для обмеження кількості запитів до нейромережевого аналізу залежно від балансу. У користувача є змога переглядати поточний баланс, історію транзакцій, а також ініціювати операції переказу коштів іншому користувачу за адресою його гаманця. Розробка даного модуля реалізована із використанням криптографічних бібліотек та алгоритмів для забезпечення безпеки транзакцій та збереження даних.

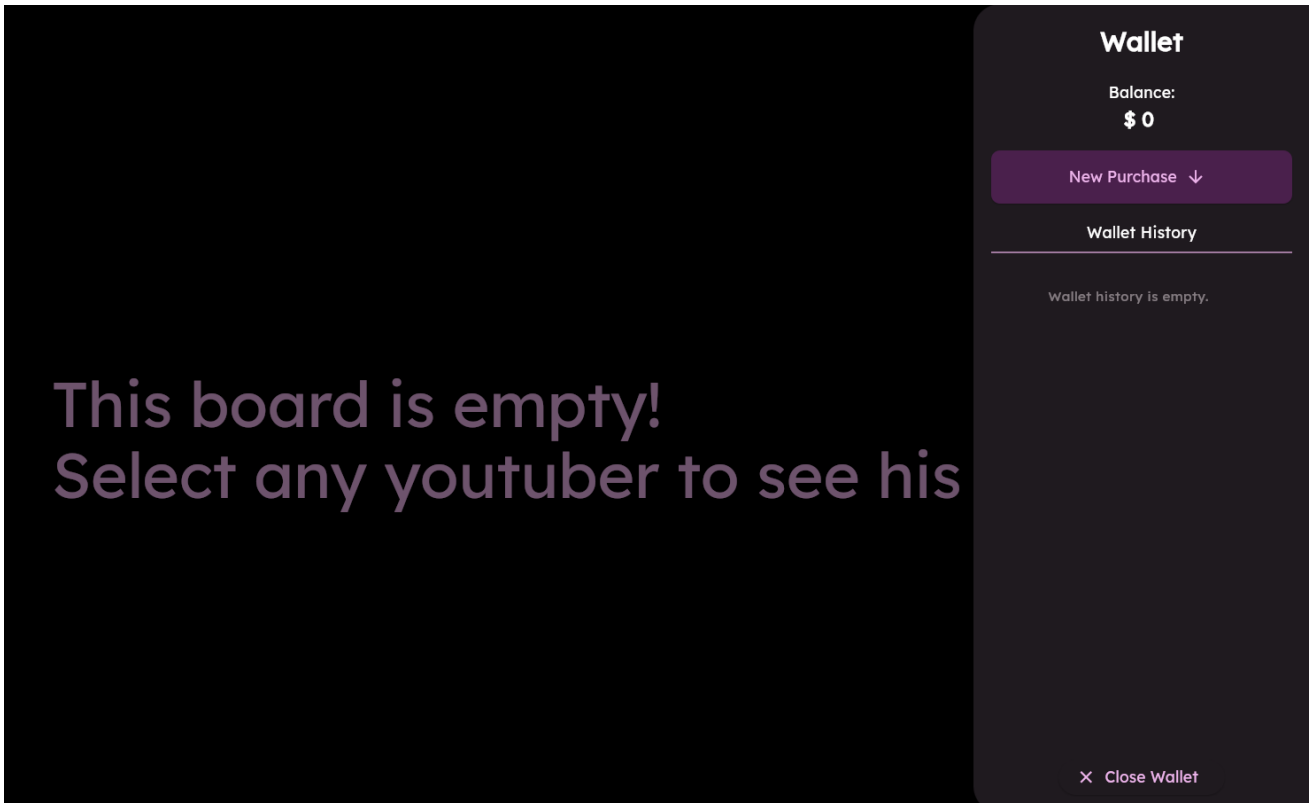


Рисунок 4.11 – Інтерфейс реалізованого криптовалютного гаманця

Після натискання кнопки «New Purchase» користувачу пропонується вибрати криптовалютний токен та мережу для здійснення поповнення, як зображено на рисунку 4.12. Також вводиться сума поповнення у доларах США, яка буде автоматично конвертована у відповідну кількість токенів згідно з

актуальним курсом. Цей етап дозволяє швидко визначити параметри транзакції перед її підтвердженням.

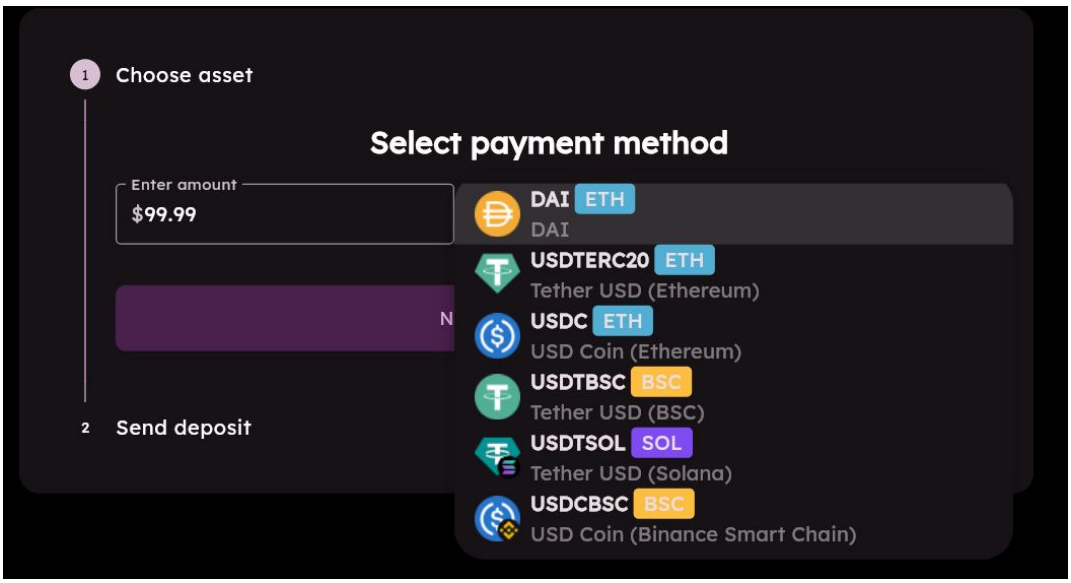


Рисунок 4.12 – Введення суми поповнення на вибрану мережу для поповнення гаманця.

На наступному екрані поповнення відображається згенерована адреса гаманця для переказу, сума в обраних токенах, еквівалентна введеній сумі в доларах, а також QR-код для зручного сканування та здійснення транзакції з таймером скасування ордеру (див. рисунок 4.13). Такий підхід спрощує процес оплати й мінімізує ймовірність помилки при ручному введенні адреси.

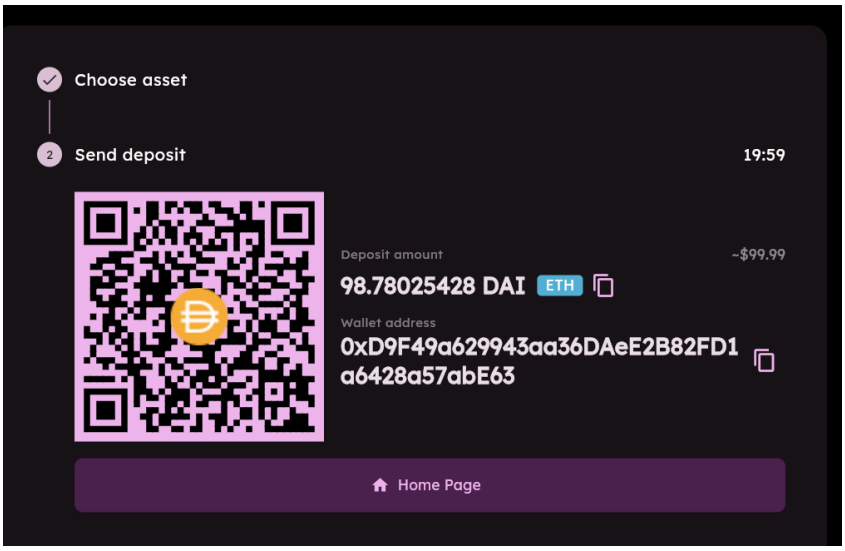


Рисунок 4.13 – Екран з реквізитами для поповнення гаманця

Після повернення на головний екран, у розділі історії гаманця з’являється новий ордер із зазначенням суми поповнення та його поточним статусом. Кнопка для створення нового поповнення стає тимчасово заблокованою на 2 хвилини – це обмеження встановлено системою, щоб запобігти надто частим запитам. Якщо користувач вирішить створити новий ордер, не закривши поточний, то попередній буде автоматично скасовано системою для уникнення дублювання транзакцій, як зображено на рисунку 4.14.

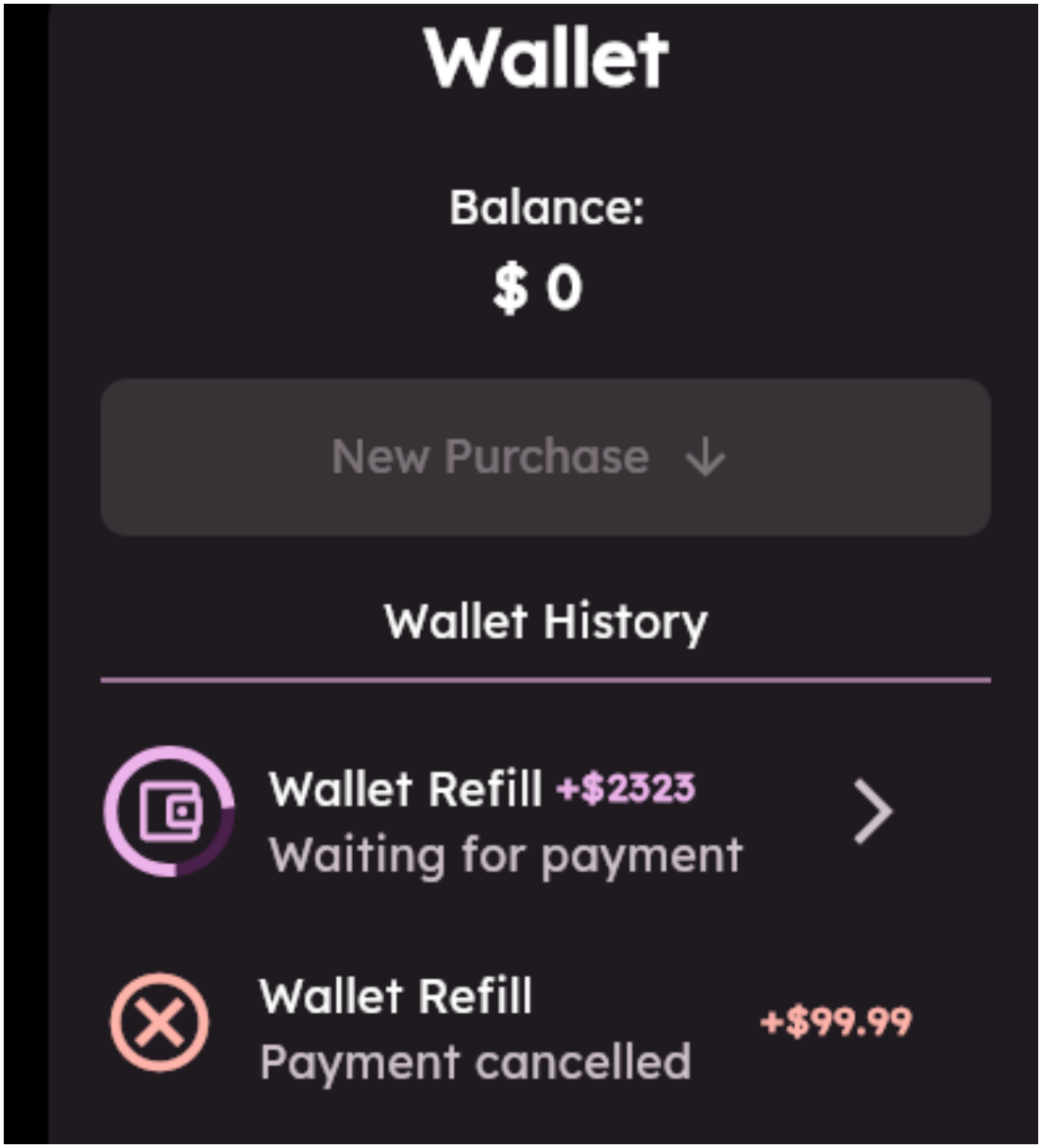


Рисунок 4.14 – Відображення нового ордеру в історії гаманця

У системі передбачено 5 статусів ордерів, які відображають поточний стан поповнення:

Waiting — очікується надходження коштів на вказану адресу;

In Progress — транзакція в обробці мережею;

Done — кошти успішно зараховані на рахунок користувача;

Fail — кошти успішно зараховані, але сталась помилка на серверній частині;

Cancel — ордер скасовано системою.

Обробка кожного із цих статусів в історії гаманця зображена на рисунку 4.15.

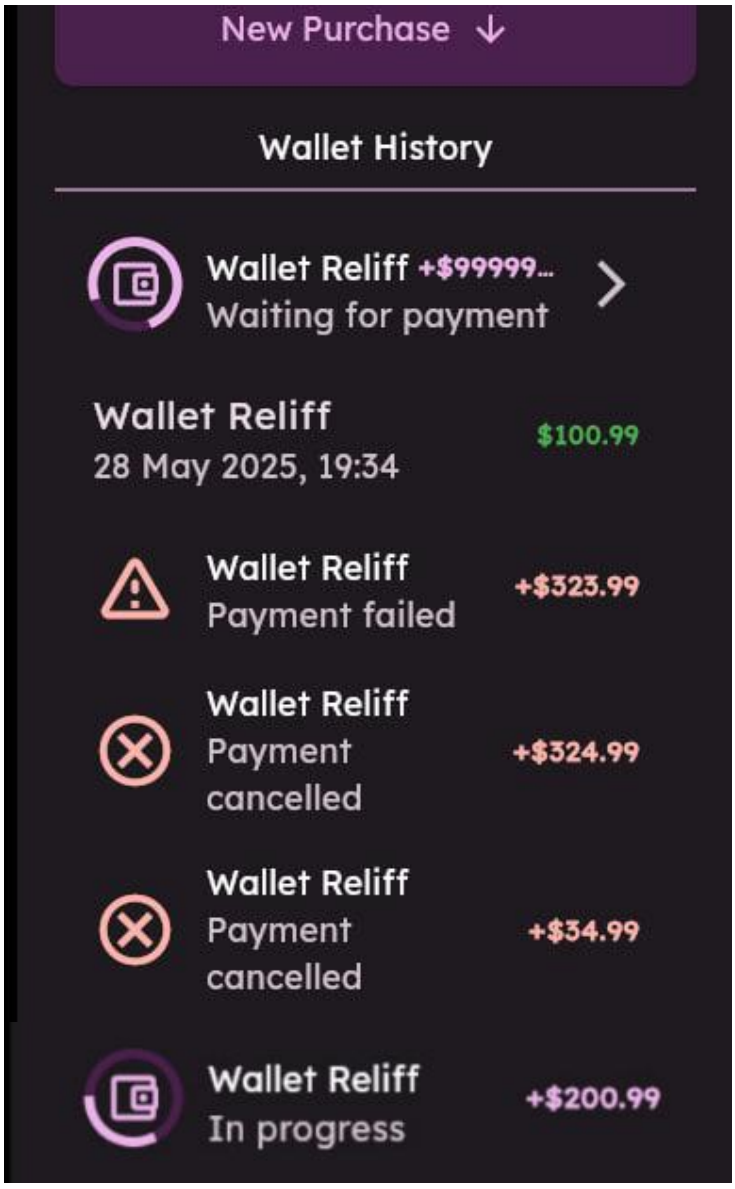


Рисунок 4.15 – Обробка кожного статусу в історії гаманця

Користувач також має змогу задати власний AI-промпт для будь-якого доданого каналу, натиснувши відповідну кнопку на картці каналу. У вікні, що відкриється, можна ввести текст запиту, який буде застосовуватись до кожного відео цього каналу для формування модифікованого тексту (див. рисунок 4.16). У разі потреби користувач може видалити будь-який канал із підписок, натиснувши відповідну кнопку на його картці. Після підтвердження канал буде видалено, а доступ до пов’язаних з ним відео заблоковано.

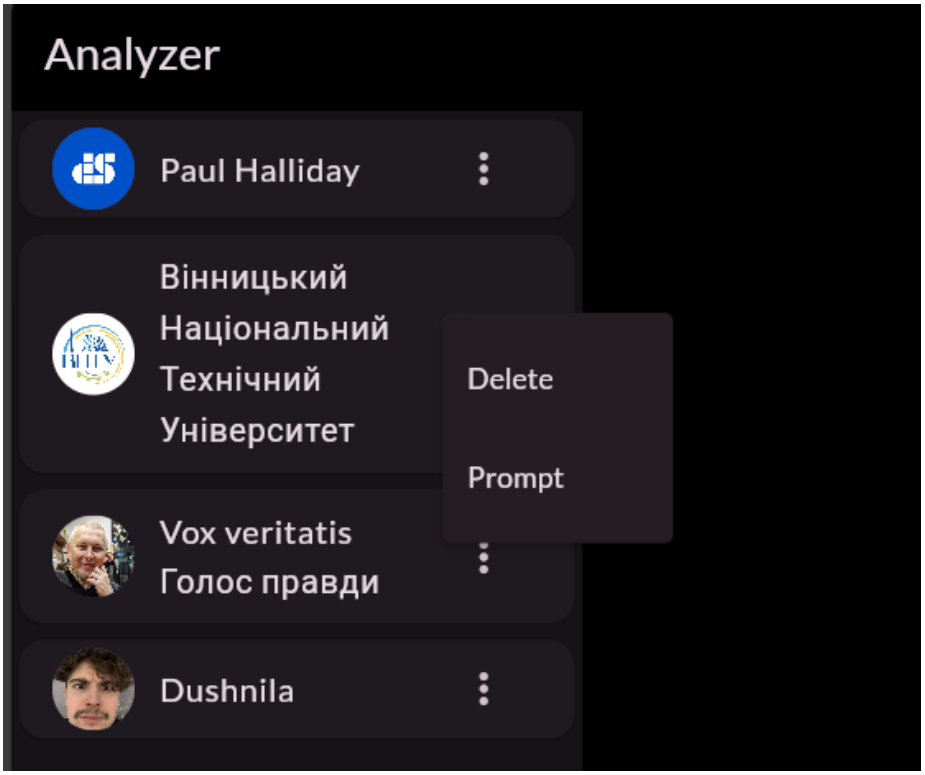


Рисунок 4.16 – Функції для встановлення промпту та видалення каналу

Після додавання YouTube-каналу та завдання промпту користувач має можливість переглянути модифікований текст, сформований на основі оригінальної транскрипції відео, як зображено на рисунку 4.17. Цей текст створюється автоматично за допомогою нейромережевої моделі, яка враховує інструкцію, задану користувачем у промпті. Наприклад, це може бути завдання на кшталт: «Сформулюй головні тези відео», «Зроби короткий конспект», тощо.

Рисунок 4.17 – Вивід модифікованого тексту

Модифікований текст дозволяє отримати адаптований зміст відео, що значно економить час на перегляд і допомагає краще зосередитися на ключовій інформації. Кожен користувач може встановити унікальний промпт для окремого каналу, що робить персоналізований аналіз більш гнучким і точним. Якщо промпт не встановлено, модифікований текст не формується і замінюється відповідним повідомленням у вікні.

Оригінальний текст є повною транскрипцією відео, згенерованою за допомогою моделі OpenAI Whisper. Він точно передає мовлення автора, включаючи усі слова, фрази та інтонаційні паузи. Цей текст є незмінним і надає користувачу змогу ознайомитися з усім вмістом відео без редагування чи інтерпретації (див. рисунок 4.18). Транскрипція створюється автоматично під час обробки відео на сервері, а збереження відбувається у базі даних для подальшого використання.

Користувач може переглянути оригінальний текст у діалоговому вікні при натисканні на відео у вкладці Content. Такий підхід дає змогу мати повний доступ

до первинного джерела, що особливо корисно для аналітики, створення власних конспектів або перевірки точності модифікованого варіанту.

Рисунок 4.18 – Вивід оригінального тексту

4.4 Висновки

У процесі розробки та тестування програмного забезпечення для персоналізованого аналізу YouTube-відео на основі нейронних мереж були досягнуті значні результати. Перш за все, було створено інтуїтивно зрозумілий інтерфейс, що дозволяє користувачам без складнощів додавати канали, переглядати відео, а також працювати з модифікованими текстами на основі власних AI-промптів.

Програмне забезпечення реалізує ключові функції: додавання каналів через псевдоніми, отримання списку відео, перегляд оригінального і модифікованого тексту відео, поповнення криптогаманцю, а також задавання спеціальних промптів для каналів.

Тестування системи засвідчило її ефективність та стабільність при роботі з основними функціями. Всі основні API-запити, зокрема отримання списку криптоактивів, додавання каналів, отримання відео та встановлення промптів працюють коректно.

Розробка інструкції користувача сприяє зручності взаємодії з програмним продуктом, забезпечуючи покрокове керівництво для кожного етапу роботи. Це дозволяє користувачам швидко освоїти основні можливості програми.

ВИСНОВКИ

У процесі виконання бакалавської кваліфікаційної роботи була розроблена вебсистема для персоналізованого аналізу YouTube-відео з використанням нейронних мереж. Реалізація бкр відповідає методичним вказівкам [25].

Для досягнення поставленої мети було вирішено такі завдання:

– проведено аналіз існуючих рішень у сфері автоматизованого аналізу відеоконтенту та засобів візуалізації фінансової активності користувача;

– визначено способи інтеграції результатів нейромережевої обробки відео в інтерфейс користувача;

– розроблено архітектуру вебсервісу та його основні компоненти, враховуючи нові функції;

– реалізовано механізми відображення результатів аналізу текстового контенту та інтеграцію криптогаманця в інтерфейс користувача;

– протестовано роботу клієнтської частини системи, оцінити зручність інтерфейсу та коректність виводу фінансової інформації.

Аналіз задачі та постановка проблеми дозволили чітко визначити функціональні вимоги до системи та обґрунтувати необхідність її створення. Було визначено ключові завдання, які включають інтеграцію з YouTube API для роботи з каналами і відео, а також використання нейронних мереж для аналізу текстів. Це забезпечило основу для подальшої розробки програмного продукту.

Моделювання структури системи дало змогу визначити архітектуру, що базується на принципах Clean Architecture та DDD. Це дозволило чітко структуризувати програмні модулі та їх взаємодії. В результаті було створено логічну та зручну для подальшого розширення систему з чітким розподілом відповідальності між компонентами.

Розробка програми включала в себе варіантний аналіз вибору технологій для реалізації продукту. Завдяки вибору технологій, таких як .NET, RabbitMQ,

PostgreSQL, Hive та Azure, вдалося забезпечити стабільну та масштабовану платформу для роботи системи. Створені API-запити, що дозволяють додавати канали, отримувати відео та працювати з текстами, відповідають усім вимогам до функціональності.

Розробка інтерфейсу з урахуванням принципів UX/UI дозволила створити зручне середовище для користувачів. Візуальний дизайн був розроблений з урахуванням принципів колористики, що робить взаємодію з програмою приємною та інтуїтивно зрозумілою. Розроблювана вебсистема для персоналізованого аналізу відео на YouTube [26].

На завершення було створено інструкцію для користувачів, що дозволяє швидко освоїти основні функції програми та ефективно налаштувати її для використання.

У підсумку, розроблена система відповідає поставленим вимогам і є готовою до використання. Основні можливості, такі як додавання каналів, перегляд відео та аналіз текстів, успішно реалізовані. Результати виконання підтверджують ефективність запропонованого підходу та технологій.

ЛІТЕРАТУРА

1. Статистика YouTube за 2025 рік : вебсайт URL: https://affmaven.com/uk/youtube-statistics/

2. ТОП 10 інструментів для Аналітиків Даних - Web Academy Media : вебсайт URL: https://web-academy.ua/blog/junior/top-10-analytics-tools

3. Персоналізація контенту сучасних онлайн-медіа: виклики та перспективи : вебсайт URL: https://www.researchgate.net/publication/382859887_PERSONALIZACIA_KONTENTU_SUCASNIH_ONLAJN-MEDIA_VIKLIKI_TA_PERSPEKTIVI

4. Кучерявий І. В., Романюк О. В. Аналіз використання технологій штучного інтелекту у вивченні правильної вимови // Матеріали XVI міжнародної науково-практичної конференції «Інформаційні технології і автоматизація – 2023», Одеса, 19–20 жовтня 2023 р. Одеса : Видавництво ОНТУ, 2023. С. 354–355.

5. Telegram став другим найпопулярнішим месенджером — статистика за березень 2025 року / NV : вебсайт URL: Telegram став другим найпопулярнішим месенджером — статистика за березень 2025 року / NV

6. Що таке гаманці Web3 — ITstatti.in.ua : вебсайт URL: https://itstatti.in.ua/crypto/1138-shcho-take-gamantsi-web3.html

7. Столяр В.В., Ракитянська Г.Б. Розробка вебсайту для персоналізованого аналізу youtube-відео: обробка та аналітика на основі нейромереж / Матеріали LIV Всеукраїнської науково-технічної конференції підрозділів Вінницького національного технічного університету (НТКП ВНТУ–2025) : збірник доповідей : вебсайт URL: https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2025/paper/view/24251/85357

8. Як працюють алгоритми YouTube: як потрапити у рекомендації? : вебсайт URL: https://gyre.pro/ua/blog/what-we-know-about-youtubes-algorithms-in-early-2022

9. Катков Ю. І., Зінченко О. В., Прокопов С. В., Боутьонок А. Г. Дослідження засобів підвищення ефективності створення відеоконтенту високої якості // Наукові записки УНДІЗ. 2020. № 3(59).

10. Романюк О. М., Курінний М. С., Романюк С. О., Коробейнікова Т. І., Романюк О. В. Модифікація методу оцінювальної функції для антиаліайзингу векторів // На шляху до індустрії 4.0: інформаційні технології, моделювання, штучний інтелект, автоматизація: монографія. Одеса, 2021. С. 409–422.

11. YouTube Analytics and Reporting APIs | Google for Developers : вебсайт URL: https://developers.google.com/youtube/analytics?

12. Vidooly - Crunchbase Company Profile & Funding : вебсайт URL: https://www.crunchbase.com/organization/vidooly

13. TubeBuddy Extension - TubeBuddy : вебсайт URL: https://www.tubebuddy.com/tubebuddy-extension/

14. Sonix - Easy With AI : вебсайт URL: https://easywithai.com/ai-transcription-tools/sonix/

15. IBM Watson Speech to Text : вебсайт URL: https://www.ibm.com/products/speech-to-text

16. Hidden Markov Models and Gaussian Mixture Models : вебсайт URL: https://www.inf.ed.ac.uk/teaching/courses/asr/2016-17/asr03-hmmgmm-handout.pdf

17. TF-IDF — Term Frequency-Inverse Document Frequency – LearnDataSci : вебсайт URL: https://www.learndatasci.com/glossary/tf-idf-term-frequency-inverse-document-frequency/

18. Статистика YouTube за 2025 рік : вебсайт URL: https://affmaven.com/uk/youtube-statistics/

19. Романюк О. Н. , Ковальова Ю. О. , Романюк О. В. , Майданюк В. П.. Концепції інтерактивного дизайну // Глобалізація та трансформація: національний та світовий виміри : колективна монографія. Харків : СГ НТМ «Новий курс», 2025. Розд. 1.2. С. 20-30.

20. Правильний спосіб використання Draw.io у створенні діаграм : вебсайт URL: https://www.mindonmap.com/uk/blog/drawio-timeline/

21. Мікросервісна архітектура: плюси та мінуси - IT Education Blog : вебсайт URL: https://itedu.center/ua/blog/articles/microservices-architecture-advantages-and-disadventages/

22. Microsoft Azure Portal | Microsoft Azure : вебсайт URL: https://azure.microsoft.com/en-us/get-started/azure-portal/?msockid=1e42adeaed306df52a4ab9cdec9a6c6c

23. Як імплементувати Clean Architecture та Domain Driven Design на базі Nest.js | DOU : вебсайт URL: https://dou.ua/forums/topic/47721/

24. Модульний тест проти інтеграційного тесту – різниця між ними : вебсайт URL: https://www.guru99.com/uk/unit-test-vs-integration-test.html

25. Методичні вказівки до виконання бакалаврських кваліфікаційних робіт для студентів спеціальності 121 «Інженерія програмного забезпечення» / уклад. : О. Н. Романюк, Г. О. Черноволик. – Вінниця : ВНТУ, 2022. – 51 с.

26. Вебзастосунок Youtube Analyzer | Власна розробка: вебсайт URL: https://ya.piescript.io/

ДОДАТКИ

Додаток А – Технічне завдання

Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії

ЗАТВЕРДЖУЮ
д.т.н., проф. О. Н. Романюк
"25" березня 2025 р.

Технічне завдання
на бакалаврську кваліфікаційну роботу «Розробка вебсайту для персоналізованого аналізу YouTube-відео: обробка та аналітика на основі нейромереж» за спеціальністю 121 Інженерія програмного забезпечення

Керівник бакалаврської кваліфікаційної роботи:
_____к.т.н., доцент. Г.Б. Ракитянська
"24" березня 2025 р.
Виконав:
_____студент гр.2ПІ-21б В.В. Столяр
"24" березня 2025 р.

Вінниця – 2025 року

1. Найменування та галузь застосування

Бакалаврська кваліфікаційна робота: «Розробка веб-сайту для персоналізованого аналізу YouTube-відео: обробка та аналітика на основі нейромереж».

Галузь застосування - інформаційні системи, штучний інтелект, вебтехнології.

2. Підстава для розробки.

Підставою для виконання бакалаврської кваліфікаційної роботи (БКР) є індивідуальне завдання на БКР та наказ №97 від 20 березня 2025 р. ректора по ВНТУ про закріплення тем БКР.

3. Мета та призначення розробки.

Метою роботи є процес покращення якості персонального аналізу YouTube-відео шляхом створення функціонального вебзастосунку з використанням нейромережевих технологій.

Призначення роботи – розробка програмного рішення, яке забезпечує персоналізований збір, обробку та аналітичну інтерпретацію відеоконтенту на базі даних із платформи YouTube.

3 Вихідні дані для проведення НДР

Перелік основних літературних джерел, на основі яких буде виконуватись БКР.

1. Дослідження засобів підвищення ефективності створення відеоконтенту високої якості (2020) | Катков Ю.І., к.т.н. Зінченко О.В., к.т.н. Прокопов С.В., к.т.н. Боутьонок А.Г.| Наукові записки УНДІЗ. –2020. –No3(59)

2. Романюк О. Н. , Ковальова Ю. О. , Романюк О. В. , Майданюк В. П.. Концепції інтерактивного дизайну . Романюк О. Н., Ковальова Ю. О., Романюк О. В., Майданюк В. П. Концепції інтерактивного дизайну // Глобалізація та трансформація: національний та світовий виміри : колективна монографія. Харків : СГ НТМ «Новий курс», 2025. Розд. 1.2. С. 20-30.

3. Романюк Олександр , Романюк Сергій , Курінний Михайло , Романюк Оксана , Коробейнікова Татяна. Модифікація методу оцінювальної функції для антиаліайзингу векторів . Романюк О. Н., Курінний М. С., Романюк С. О., Коробейнікова Т. І., Романюк О.В. Модифікація методу оцінювальної функції для антиаліайзингу векторів /На шляху до індустрії 4.0:інформаційні технології, моделювання, штучний інтелект, автоматизація. Монографія.Одеса, 2021. -С.409-422.

4. Кучерявий Ігор , Романюк Оксана. Аналіз використання технологій штучного інтелекту у вивченні правильної вимови . Кучерявий І. В., Романюк О. В. Аналіз використання технологій штучного інтелекту у вивченні правильної вимови. Матеріали XVI міжнародної науково-практичної конференції «Інформаційні технології і автоматизація - 2023» ,Одеса, 19-20 жовтня 2023 р. Одеса, Видавництво ОНТУ, 2023 р. -С. 354-355.

5. Технічні вимоги

Вебзастосунок має включати: модуль авторизації користувача, модуль підключення до YouTube API для отримання інформації про відео, модуль аналітики (на основі нейромереж), адаптивний інтерфейс, реалізований на Flutter. Для бекенду використовується ASP.NET Core з REST-архітектурою. Продукт має забезпечувати відповідність загальноприйнятим вимогам до безпеки, UX/UI-дизайну, продуктивності та масштабованості. Вихідні дані – візуалізовані аналітичні звіти про активність користувача та контент.

6. Конструктивні вимоги.

Інтерфейс вебсайту має бути інтуїтивно зрозумілим, адаптованим до різних пристроїв, відповідати принципам сучасного UX/UI-дизайну.

Уся документація має бути оформлена згідно з діючими стандартами України.

7. Перелік технічної документації, що пред'являється по закінченню робіт:

- пояснювальна записка до БКР;
- технічне завдання;
- лістинги програми.

8. Вимоги до рівня уніфікації та стандартизації

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ.

9. Стадії та етапи розробки:

№ з/п	Назва етапів бакалаврської кваліфікаційної роботи	Строк виконання етапів роботи
1	Аналіз предметної області, постановка задач	25.03.25 – 03.04.25
2	Розробка архітектури системи та вибір технологій	04.04.25 – 11.04.25
3	Проєктування інтерфейсу користувача	12.04.25 – 19.04.25
4	Реалізація основних програмних модулів	20.04.25 – 10.05.25
5	Тестування та розробка інструкції користувача	11.05.25 – 20.05.25
6	Оформлення матеріалів до захисту БКР	21.05.25 – 30.05.25

10. Порядок контролю та прийняття.

Виконання етапів бакалаврської кваліфікаційної роботи контролюється керівником згідно з графіком виконання роботи. Захист бакалаврської кваліфікаційної роботи здійснюється ДЕК, затвердженою зав. кафедрою згідно з графіком

Додаток Б – Протокол перевірки на плагіат

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: Розробка вебсайту для персоналізованого аналізу YouTube-відео: обробка та аналітика на основі нейромереж

Тип роботи: бакалаврська кваліфікаційна робота
(бакалаврська кваліфікаційна робота / магістерська кваліфікаційна робота)

Підрозділ кафедра програмного забезпечення, ФІТКІ, 2ПІ-21б
(кафедра, факультет, навчальна група)

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism 3.8 %

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне)

- ■ Запозичення, виявлені у роботі, є законними і не містять ознак плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту
- ☐ У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.
- ☐ У роботі виявлено ознаки плагіату та/або текстових маніпуляцій як спроб укриття плагіату, фабрикації, фальсифікації, що суперечить вимогам законодавства та нормам академічної доброчесності. Робота до захисту не приймається.

Експертна комісія:

_____ _____
(прізвище, ініціали, посада) (підпис)

_____ _____
(прізвище, ініціали, посада) (підпис)

Особа, відповідальна за перевірку _____ Черноволик Г. О.

З висновком експертної комісії ознайомлений(-на)

Керівник _____ Ракитянська Г.Б.
(підпис) (прізвище, ініціали, посада)

Здобувач _____ Столяр В.В.

Додаток В – Лістинг програми

```
import 'package:flutter/material.dart';

import 'package:youtube_analyzer/common/database.dart';
import 'package:youtube_analyzer/repositories/subcription_channels/youtube_repository.dart';

import 'package:youtube_analyzer/screensOld/main_page.dart';

class LoginPage extends StatefulWidget {
  const LoginPage({super.key});

  @override
  State<LoginPage> createState() => _LoginPageState();
}

class _LoginPageState extends State<LoginPage> {
  bool isLoading = false;
  final _verificationCode = TextEditingController();

  @override
  void dispose() {
    _verificationCode.dispose();
    super.dispose();
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Theme.of(context).colorScheme.surface,
      body: Row(
```

```
children: [
  Expanded(
    flex: 5,
    child: Container(
      decoration: BoxDecoration(
        gradient: LinearGradient(
          colors: [
            Theme.of(context).colorScheme.onPrimary,
            Theme.of(context).colorScheme.surface,
          ],
        ),
      ),
      child: Column(
        mainAxisAlignment: MainAxisAlignment.center,
        crossAxisAlignment: CrossAxisAlignment.center,
        children: [
          Text(
            'YouTube Analyzer',
            style: Theme.of(context)
                .textTheme
                .displayLarge!
                .copyWith(fontWeight: FontWeight.bold),
          ),
        ],
      ),
    ),
  ),
  Expanded(
    flex: 5,
    child: Center(
```

```
child: SizedBox(
  height: 500,
  width: 450,
  //color: Theme.of(context).colorScheme.surfaceContainer,

  child: Column(
    crossAxisAlignment: CrossAxisAlignment.start,
    mainAxisAlignment: MainAxisAlignment.center,
    children: [
      Text(
        'Let`s Sing Up!',
        style: Theme.of(context).textTheme.displaySmall,
      ),
      const SizedBox(height: 12),
      RichText(
        text: TextSpan(
          text: 'Hey. Enter verification code from  ',
          style: Theme.of(context).textTheme.titleMedium,
          children: [
            WidgetSpan(
              child: MouseRegion(
                cursor: SystemMouseCursors.click,
                child: GestureDetector(
                  onTap: () {
                    print('You pressed');
                  },
                  child: Container(
                    decoration: BoxDecoration(
                      border: Border(
                        bottom: BorderSide(
```

```
                color: Theme.of(context)
                    .colorScheme
                    .primary,
                ),
              ),
            ),
            child: Text(
              'telegram bot',
              style: Theme.of(context)
                  .textTheme
                  .titleMedium!
                  .copyWith(
                    color: Theme.of(context)
                        .colorScheme
                        .primary,
                  ),
            ),
          ),
        ),
      ),
      const TextSpan(
        text: '  to get sign in to your account.',
      ),
    ],
  ),
),
const SizedBox(height: 48),
TextField(
  controller: _verificationCode,
```

```
  keyboardType: TextInputType.text,
  maxLength: 50,
  decoration: InputDecoration(
    label: const Text('Verification code'),
    filled: true,
    fillColor:
        Theme.of(context).colorScheme.surfaceContainer,
    enabledBorder: OutlineInputBorder(
        borderRadius: BorderRadius.circular(24),
        borderSide: BorderSide(
            color: Theme.of(context)
                .colorScheme
                .primaryContainer)),
    focusedBorder: OutlineInputBorder(
        borderRadius: BorderRadius.circular(12),
        borderSide: BorderSide(
            color: Theme.of(context).colorScheme.primary)),
  ),
),
const SizedBox(
  height: 8,
),
SizedBox(
  height: 46,
  width: double.infinity,
  child: FilledButton.icon(
    onPressed: isLoading
        ? null
        : () async {
            if (mounted) {
```

```
  setState(() {
    isLoading = true;
  });
}
// Future.delayed(const Duration(seconds: 2), () {
//   // for testing circular indicator
//   setState(() {
//     isLoading = false;
//   });
// });

if (_verificationCode.text.length != 32) {
  setState(() {
    isLoading = false;
  });
  ScaffoldMessenger.of(context).showSnackBar(
    SnackBar(
      content: const Text(
          'Check your Verification code'),
      action: SnackBarAction(
        label: 'Ok',
        onPressed: () {
          // Code to execute.
        },
      ),
    ),
  );
  return;
}
```

```
                    bool isLoadUser = await
YoutubeRepository().checkPersonAuthTokenKey(_verificationCode.text);
                    if (isLoadUser) {
                      Database.set(Database.personAuthTokenKey,
                          _verificationCode.text);
                      setState(() {
                        isLoading = false;
                      });
                      Navigator.pushReplacement(
                        context,
                        MaterialPageRoute(
                          builder: (context) => const MainPage(),
                        ),
                      );
                    } else {
                      ScaffoldMessenger.of(context).showSnackBar(
                        SnackBar(
                          content: const Text(
                              'This user unauthorized! Check your Verification code'),
                          action: SnackBarAction(
                            label: 'ok',
                            onPressed: () {
                              // Code to execute.
                            },
                          ),
                        ),
                      );
                      setState(() {
                        isLoading = false;
                      });
                    }
```

```
                    },
                icon: isLoading
                    ? null
                    : Icon(
                        Icons.person,
                        color: Theme.of(context).colorScheme.primary,
                      ),
                label: isLoading
                    ? CircularProgressIndicator(
                        valueColor: AlwaysStoppedAnimation<Color>(
                            Theme.of(context).colorScheme.primary),
                      )
                    : const Text('Log In'),
                style: FilledButton.styleFrom(
                  backgroundColor:
                      Theme.of(context).colorScheme.onPrimary,
                  foregroundColor: Colors.white,
                ),
              ),
            ),
          ],
        ),
      ),
    );
  }
}
```

```
import 'package:flutter/material.dart';
// import 'dart:convert';
// import 'package:http/http.dart' as http;
import 'package:cached_network_image/cached_network_image.dart';

// import 'package:youtube_analyzer/data/dummy_data.dart'; // import basic url and token
import 'package:youtube_analyzer/modelsOld/youtube.dart';
import 'package:youtube_analyzer/repositories/subcription_channels/youtube_repository.dart';
import 'package:youtube_analyzer/widgetsOld/add_new_youtuber.dart';
import 'package:youtube_analyzer/widgetsOld/dialog_prompt.dart';

class SubscriptionsYtScreen extends StatefulWidget {
  const SubscriptionsYtScreen({super.key, required this.onSelectedYoutuber});
  final void Function(String author) onSelectedYoutuber;

  @override
  State<SubscriptionsYtScreen> createState() => _SubscriptionsYtScreenState();
}

class _SubscriptionsYtScreenState extends State<SubscriptionsYtScreen> {
  List<Youtuber> subscrtiptionYT = [];

  void _deleteYoutuber(String youtuberId) async {
    final bool isDeleted = await YoutubeRepository().deleteChannel(youtuberId);

    if (isDeleted) {
      debugPrint("Youtuber is deleted Succsessfully");
      setState(() {});
    } else {
      debugPrint("something went wrong");
    }
  }
```

```
void _addYoutuber(Youtuber youtuber) {
  setState(
    () {
      subscrtiptionYT.add(youtuber);
    },
  );
}

@override
void initState() {
  super.initState();
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    backgroundColor: Theme.of(context).colorScheme.surface,
    floatingActionButton: FloatingActionButton(
      onPressed: () {
        showDialog(
          context: context,
          builder: (ctx) => Dialog(
            child: SizedBox(
              width: 550,
              height: 250,
              child: AddNewYoutuber(
                onAddYoutuber: _addYoutuber,
              ),
            ),
          ),
        );
      },
      tooltip: 'Add new youtuber',
```

```
  child: const Icon(Icons.add),
),
body: subscrtiptionYT.isEmpty
    ? Center(
        child: Text(
          'Add any Youtuber to see their content',
          style: Theme.of(context)
              .textTheme
              .titleLarge!
              .copyWith(color: Theme.of(context).colorScheme.secondary),
          textAlign: TextAlign.center,
        ),
      )
    : ListView.builder(
        itemCount: subscrtiptionYT.length,
        itemBuilder: (ctx, index) {
          return Card(
            child: ListTile(
              leading: CachedNetworkImage(
                imageUrl: subscrtiptionYT[index].logo,
                imageBuilder: (context, imageProvider) => CircleAvatar(
                  backgroundImage: imageProvider,
                ),
                placeholder: (context, url) => const CircleAvatar(
                  child: CircularProgressIndicator(strokeWidth: 2),
                ),
                errorWidget: (context, url, error) => const CircleAvatar(
                  backgroundImage: AssetImage('lib/assets/image1.jpg'),
                ),
              ),
              trailing: PopupMenuButton(
                itemBuilder: (ctx) => [
                  PopupMenuItem(
```

```
                value: 'Delete',
                onTap: () {
                  debugPrint(
                      "channel  ${subscrtiptionYT[index].name} \n Id: ${subscrtiptionYT[index].id}\n
imageURL:  ${subscrtiptionYT[index].logo}");
                  _deleteYoutuber(subscrtiptionYT[index].id);
                  debugPrint(subscrtiptionYT[index].logo);
                  //removeYouTuber(index);
                  setState(() {
                    //widget.onSelectedYoutuber('');
                  });
                },
                child: const Text('Delete'),
              ),
              PopupMenuItem(
                value: 'Prompt',
                onTap: () {
                  debugPrint(
                      'channel  ${subscrtiptionYT[index].name} \n Id: ${subscrtiptionYT[index].id}\n
imageURL:  ${subscrtiptionYT[index].logo}');
                  showDialog(
                    context: context,
                    builder: (ctx) => Dialog(
                      child: DialogPrompt(
                        channelId: subscrtiptionYT[index].id,
                      ),
                    ),
                  );
                },
                child: const Text('Prompt'),
              ),
            ],
          ),
```

```
                title: Text(subscrtiptionYT[index].name),
                onTap: () {
                  setState(() {
                    widget.onSelectedYoutuber(subscrtiptionYT[index].id);
                  });
                },
              ),
            );
          },
        ),
    );
import 'package:flutter/material.dart';
import 'package:flutter_dotenv/flutter_dotenv.dart';

import 'package:youtube_analyzer/common/database.dart';
import 'package:youtube_analyzer/repositories/subcription_channels/models/environment.dart';
import 'package:youtube_analyzer/youtube_analizer_app.dart';

Future<void> main() async {
  WidgetsFlutterBinding.ensureInitialized(); //magic widget
  await dotenv.load(fileName: Environment.fileName);
  await Database.init();

  runApp(const YoutubeAnalizerApp());
}
import 'package:flutter/material.dart';
import 'package:youtube_analyzer/repositories/models/wallet.dart';

class WalletHistoryList extends StatelessWidget {
  const WalletHistoryList({
    super.key,
    required this.isLoadingHistory,
    required this.historyOrders,
```

```
  required this.openSecondaryPurchaseScreen,
});
final bool isLoadingHistory;
final List<OrderHistory> historyOrders;
final void Function() openSecondaryPurchaseScreen;

@override
Widget build(BuildContext context) {
  final ColorScheme colorTheme = Theme.of(context).colorScheme;
  final TextTheme textTheme = Theme.of(context).textTheme;

  return ListView(
    padding: const EdgeInsets.only(right: 24),
    // padding: EdgeInsets.zero,
    children: [
      isLoadingHistory
          ? Center(
              heightFactor: 8,
              child: CircularProgressIndicator(
                valueColor: AlwaysStoppedAnimation<Color>(colorTheme.primary),
              ),
            )
          : historyOrders.isNotEmpty && !isLoadingHistory
              ? ListView.builder(
                  shrinkWrap: true,
                  physics: const NeverScrollableScrollPhysics(),
                  itemCount: historyOrders.length,
                  itemBuilder: (context, index) {
                    String name = historyOrders[index].name;
                    String status = historyOrders[index].status;
                    String itemType = historyOrders[index].itemType;
                    String amount = historyOrders[index].amount.toString();
                    String updateAt = historyOrders[index].formattedData;
```

```
if (status != 'Done') {
  return InkWell(
    onTap: status == 'InProgress' ||
           status == 'Fail' ||
           status == 'Cancel'
        ? null
        : openSecondaryPurchaseScreen,
    child: ListTile(
        contentPadding: (status == 'New' ||
               status == 'Waiting' ||
               status == 'InProgress')
            ? const EdgeInsets.only(left: 3)
            : const EdgeInsets.only(left: 0),
        leading: (status == 'New' ||
               status == 'Waiting' ||
               status == 'InProgress')
            ? Stack(
               alignment: Alignment.center,
               children: [
                 CircularProgressIndicator(
                   value: 1.0, // 100%
                   valueColor:
                       AlwaysStoppedAnimation<Color>(
                           colorTheme.onPrimary),
                 ),
                 CircularProgressIndicator(
                   valueColor:
                       AlwaysStoppedAnimation<Color>(
                           colorTheme.primary),
                 ),
                 Icon(
                     Icons
```

```
                .account_balance_wallet_outlined,
            color: colorTheme.primary),
        ],
      )
    : status == 'Fail'
        ? Icon(
            Icons.warning_amber_rounded,
            color: colorTheme.error,
            size: 36,
          )
        : status == 'Cancel'
            ? Icon(
                Icons.cancel_outlined,
                color: colorTheme.error,
                size: 36,
              )
            :
            // Icon(Icons.check_circle_outline_outlined);
            Icon(
                Icons
                    .account_balance_wallet_outlined,
                color: colorTheme.primary),
title: Row(
  children: [
    Text(name, style: textTheme.bodyMedium),
    (status == 'Fail' ||
            status == 'Cancel' ||
            status == 'InProgress')
        ? const SizedBox.shrink()
        : Expanded(
            child: Text(
              ' +\$$amount',
              style: textTheme.bodySmall!
```

```
                    .copyWith(
                        fontWeight: FontWeight.bold,
                        color: colorTheme.primary),
                  overflow: TextOverflow.ellipsis,
                ),
              ),
            ],
          ),
          subtitle: status == 'Waiting' || status == 'New'
              ? const Text('Waiting for payment')
              : status == 'InProgress'
                  ? const Text('In progress')
                  : status == 'Fail'
                      ? const Text('Payment failed')
                      : status == 'Cancel'
                          ? const Text('Payment cancelled')
                          : Text(updateAt),
          trailing: (status == 'InProgress' ||
                  status == 'Fail' ||
                  status == 'Cancel')
              ? Text(
                  ' +\$$amount',
                  style: textTheme.bodySmall!.copyWith(
                      fontWeight: FontWeight.bold,
                      color: (status == 'Fail' ||
                              status == 'Cancel')
                          ? colorTheme.error
                          : colorTheme.primary),
                  overflow: TextOverflow.ellipsis,
                )
              : const Icon(Icons.arrow_forward_ios)),
    );
  } else {
```

```
                return ListTile(
                  title: Text(itemType),
                  subtitle: Text(updateAt),
                  trailing: Text(
                    '\$$amount',
                    style: const TextStyle(
                        fontWeight: FontWeight.bold,
                        color: Colors.green),
                  ),
                );
              }
            },
          )
        : Opacity(
            opacity: 0.5,
            child: Center(
              child: Padding(
                padding: const EdgeInsets.all(24),
                child: Text("Wallet history is empty.",
                    style: textTheme.bodySmall),
              ),
            ),
          ),
    ],
  );
 }
}
import 'dart:async';

import 'package:cached_network_image/cached_network_image.dart';
import 'package:flutter/material.dart';
import 'package:flutter/services.dart';
import 'package:youtube_analyzer/common/database.dart';
```

```
import 'package:youtube_analyzer/repositories/models/wallet.dart';
import 'package:youtube_analyzer/repositories/payment_repository.dart';
import 'package:youtube_analyzer/repositories/widgets/handle_verified_auth_token.dart';
import 'package:youtube_analyzer/w_dummy_test/dummy_test_data.dart';
import 'package:youtube_analyzer/features/main_page/view/wallet/view/payment_screen.dart';
import
'package:youtube_analyzer/features/main_page/view/wallet/view/select_payment_screen.dart';

import 'package:qr_flutter/qr_flutter.dart';

class PurchaseScreen extends StatefulWidget {
  const PurchaseScreen({
    super.key,
    required this.avaliableCurrency,
  });
  final List<PaymentCurrency> avaliableCurrency;

  @override
  State<PurchaseScreen> createState() => _PurchaseScreenState();
}

class _PurchaseScreenState extends State<PurchaseScreen> {
  final _amountController = TextEditingController();
  final _formKey = GlobalKey<FormState>();

  List<PaymentCurrency> _avaliableCurrency = [];

  int? _value = 0;
  int _currentStep = 0;

  // int _remainingSeconds = 20; // 20 minutes in seconds
  int _remainingSeconds = 1200; // 20 minutes in seconds
  Timer? _timer;
```

```
bool _isTimerStarted = false;
Payment? _paymentOrder;
bool _isSetPaymentOrder = false;

void _startTimer() {
  _timer = Timer.periodic(const Duration(seconds: 1), (timer) {
    if (_remainingSeconds > 0) {
      setState(() {
        _remainingSeconds--;
        Database.set(Database.timerSeconds, _remainingSeconds);
      });
    } else {
      timer.cancel();
      setState(() {
        _isTimerStarted = false; // Додаємо цей рядок!
      });
      _showTimeUpDialog();
    }
  });
}

void _showTimeUpDialog() {
  showDialog(
    context: context,
    builder: (_) => AlertDialog(
      title: const Text("Payment Time Expired"),
      content: const Text(
          "The allowed time for payment has ended. Your order can no longer be processed."),
      actions: [
        TextButton(
          onPressed: () => Navigator.pop(context),
          child: const Text("OK"),
        ),
```

```
      ],
    ),
  );
}

Future<void> handlePurchase(
    {required bool isLoading,
    required String payCurrency,
    required String priceUsd}) async {
  setState(() {
    isLoading = true;
    _isSetPaymentOrder = false;
  });

  final Payment? payment = await PaymentRepository().postPayment(
    payCurrency,
    priceUsd,
  );
  debugPrint('Payment from handlePurchase: $payment');

  if (payment != null) {
    _paymentOrder = payment;
    setState(() {
      _isSetPaymentOrder = true;
      isLoading = false;
    });
    _onChanged();
  }
}

Future<void> _copyToClipboard(String text) async {
  await Clipboard.setData(ClipboardData(text: text));
  if (mounted) {
```

```
    ScaffoldMessenger.of(context).clearSnackBars();
    ScaffoldMessenger.of(context).showSnackBar(
      SnackBar(
        content: const Text('The element is copied to clipboard'),
        action: SnackBarAction(
          label: 'ok',
          onPressed: () {
            debugPrint('Vladus is the best');
            // Code to execute.
          },
        ),
      ),
    );
  }
}

Future<void> _onChanged() async {
  if (_value != null) {
    setState(() {
      _currentStep = 1;

      if (!_isTimerStarted) {
        _startTimer();
        _isTimerStarted = true;
      }
    });
  }
}

_onSelected(int selected) {
  setState(() {
    _value = selected;
  });
```

```
  // Validate the form after the state has been updated
  // This is necessary to ensure that the validation runs after the state change
  WidgetsBinding.instance.addPostFrameCallback((_) {
    _formKey.currentState?.validate();
  });
}

String get _formattedTime {
  final minutes = (_remainingSeconds ~/ 60).toString().padLeft(2, '0');
  final seconds = (_remainingSeconds % 60).toString().padLeft(2, '0');
  return "$minutes:$seconds";
}

@override
void dispose() {
  _timer?.cancel();
  _amountController.dispose();
  super.dispose();
}

@override
void initState() {
  super.initState();
  _avaliableCurrency = widget.avaliableCurrency;
}

@override
Widget build(BuildContext context) {
  final ColorScheme clrScreme = Theme.of(context).colorScheme;
  return Scaffold(
    appBar: AppBar(
      automaticallyImplyLeading: false,
      title: const Text('Purchase Wallet'),
```

```
),
body: Center(
  child: Container(
    width: 800,
    padding: const EdgeInsets.all(16),
    decoration: BoxDecoration(
      color: clrScreme.surface,
      borderRadius: BorderRadius.circular(16),
    ),
    child: Stepper(
      currentStep: _currentStep,
      onStepContinue: () {
        if (_currentStep == 0 && _value != null) {
          setState(() {
            _currentStep = 1;
          });
        } else if (_currentStep == 1) {
          Navigator.pop(context, _paymentOrder);
          // Navigator.pop(context, _avaliableCurrency[_value!]);
          //переконатись чи дійсно нам потрібно передавати значення
        }
      },
      onStepCancel: () {
        if (_currentStep > 0) {
          setState(() {
            _currentStep = 0;
          });
        }
      },
      steps: [
        Step(
          title: const Text('Choose asset'),
          isActive: _currentStep >= 0,
```

```
    state:
        _currentStep > 0 ? StepState.complete : StepState.indexed,
    content: SelectPaymentScreen(
      handlePurchase: handlePurchase,
      value: _value,
      avaliableCurrency: _avaliableCurrency,
      // onChanged: _onChanged,
      onSelected: _onSelected,
      minAmountController: _amountController,
      formKey: _formKey,
      // postPayment: _postPayment,
    ),
  ),
  Step(
      title: Row(
        children: [
          const Text('Send deposit'),
          const Spacer(),
          if (_currentStep != 0) Text(_formattedTime),
        ],
      ),
      isActive: _currentStep >= 1,
      state: StepState.indexed,
      content: _isSetPaymentOrder
          ? PaymentScreen(
              paymentOrder: _paymentOrder,
              copyToClipboard: _copyToClipboard,
            )
          : const SizedBox.shrink()),
],
controlsBuilder: (context, details) {
  // hide the default buttons
  return const SizedBox.shrink();
```

},

Додаток Г – Ілюстративні матеріали

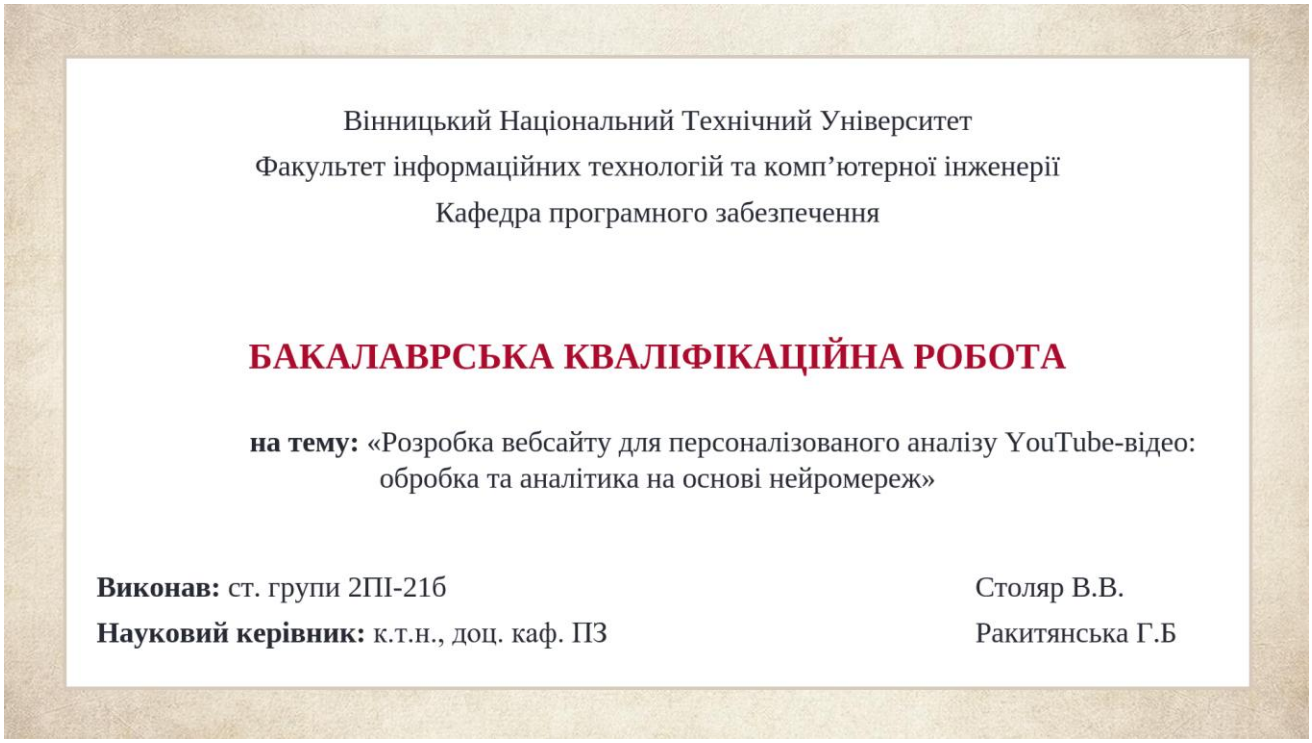


Рисунок Г.1 – Титульний слайд

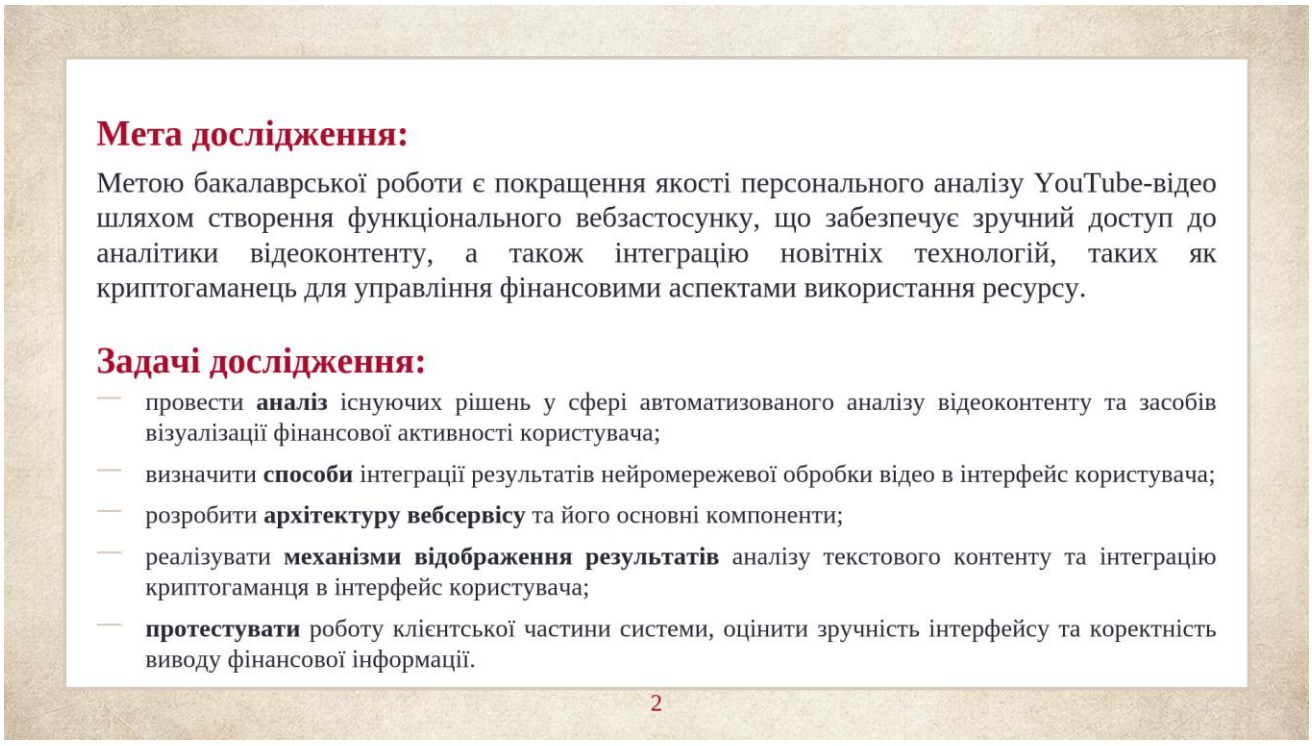


Рисунок Г.2 – Мета та задачі дослідження

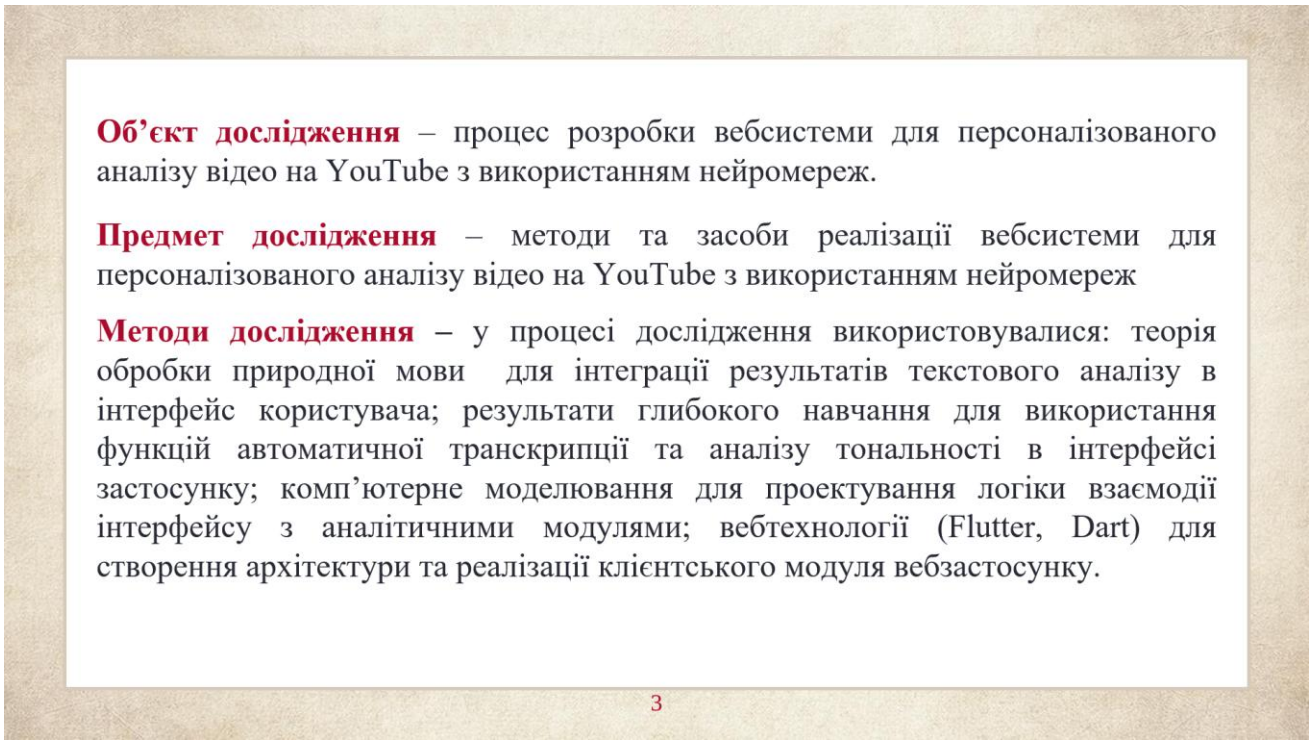


Рисунок Г.3 – Об'єкт, предмет та методи дослідження

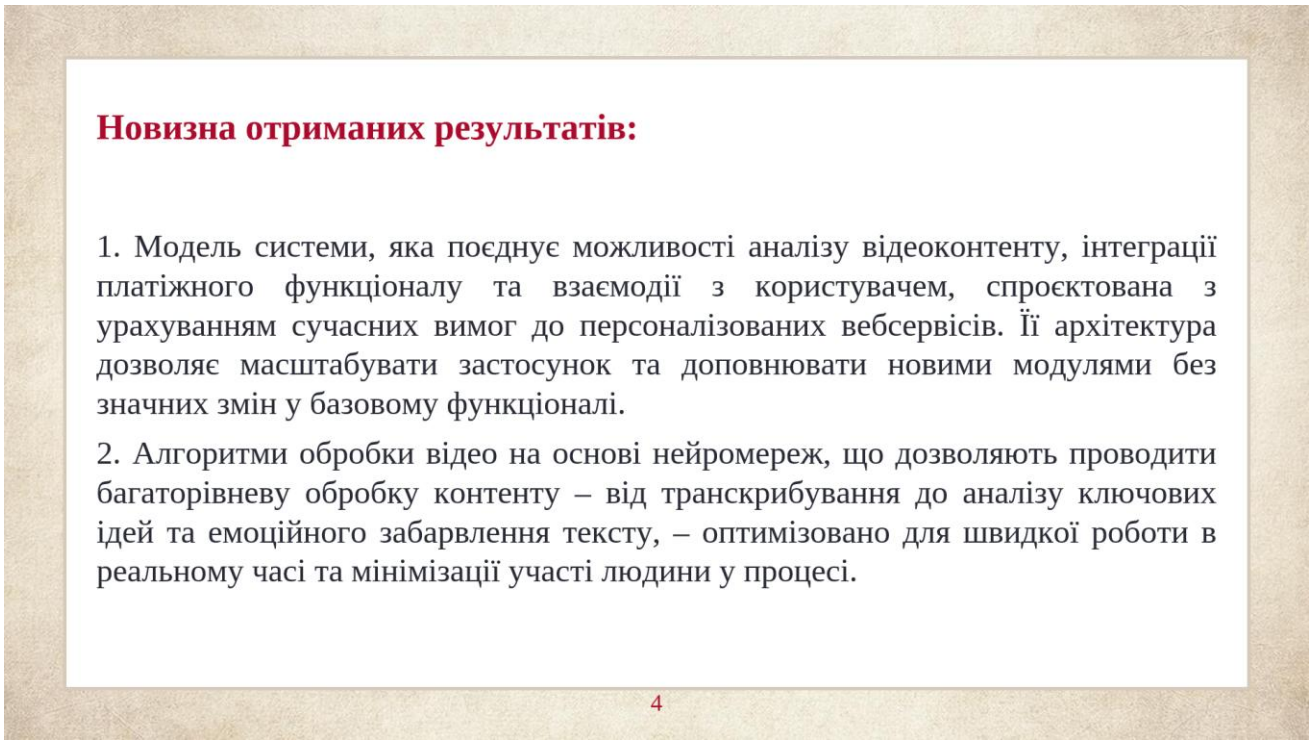


Рисунок Г.4 – Новизна отриманих результатів

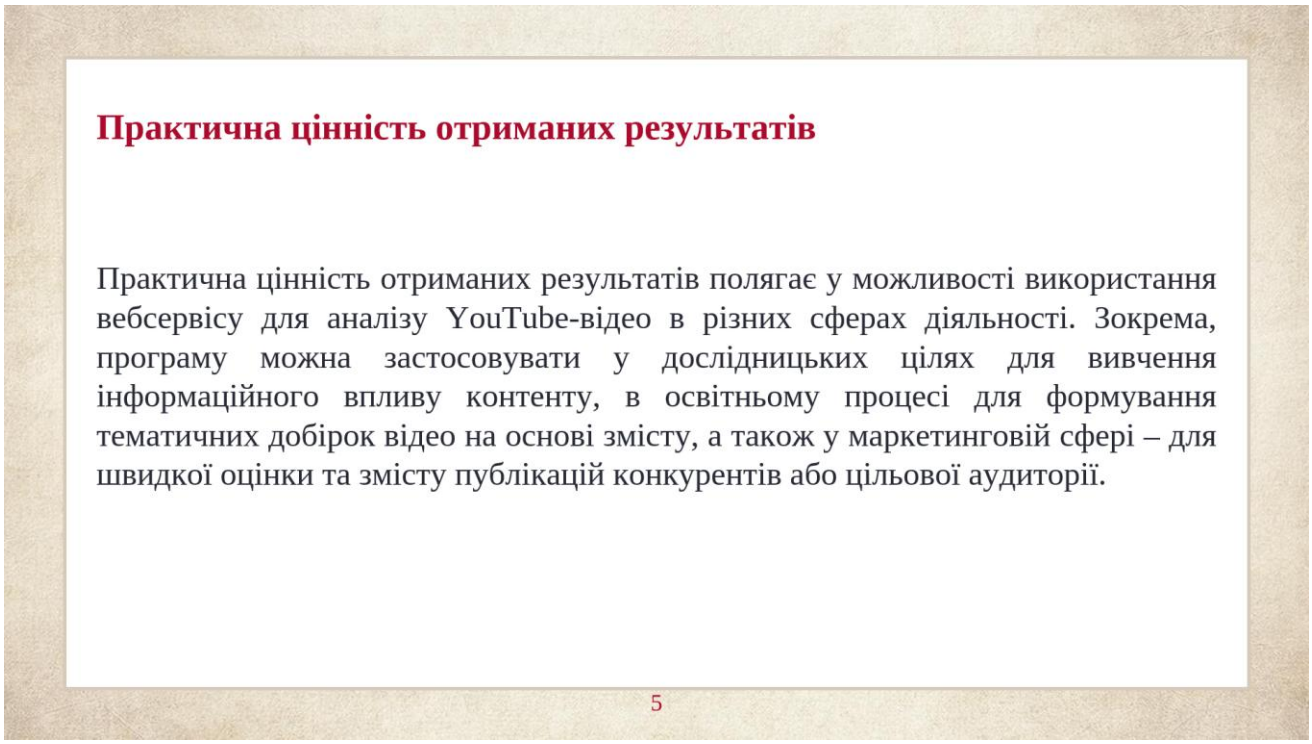


Рисунок Г.5 – Практична цінність отриманих результатів

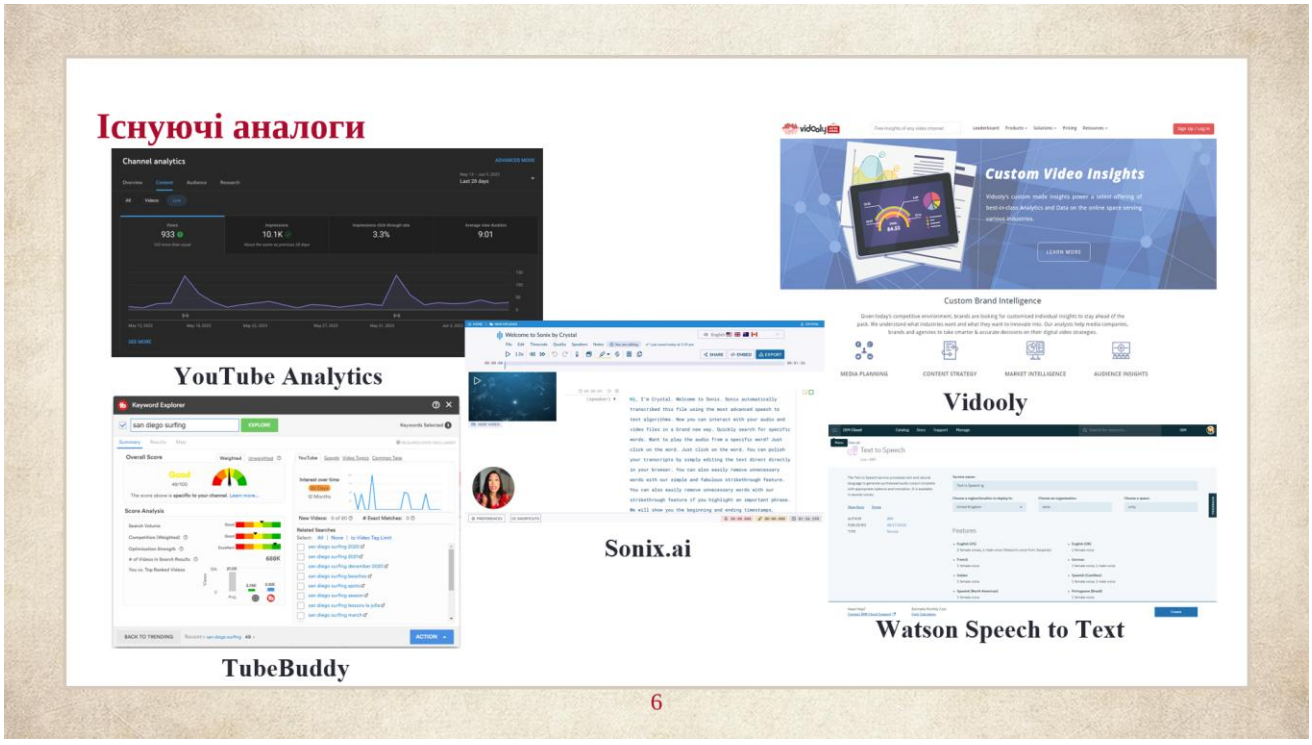


Рисунок Г.6 – Перелік існуючих аналогів

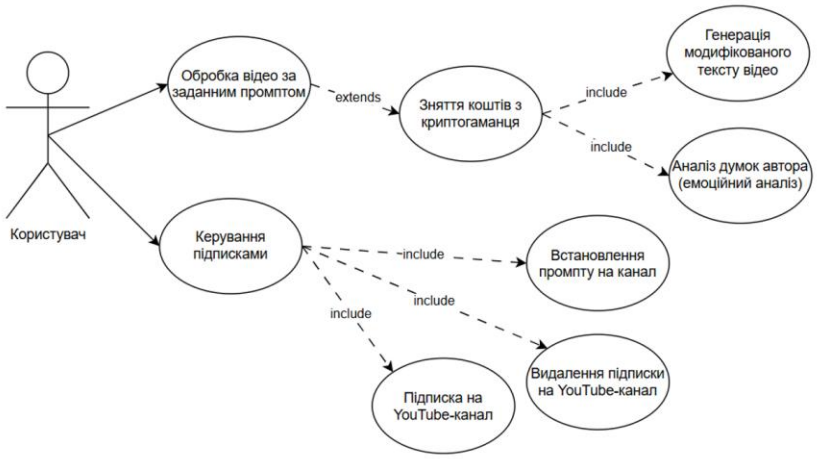
Порівняльний аналіз аналогів

Функціонал	YouTube Analytics	Vidooly	TubeBuddy	Sonix.ai	IBM Watson STT	Власна розробка
1	2	3	4	5	6	7
Транскрипція відео	-	-	-	+	+	+
Аналіз змісту	-	-	-	-	+	+
Персоналізований аналіз	-	-	-	-	-	+
Аналіз емоцій та тональності	-	-	+	-	-	+
Аналітика аудиторії	+	+	-	+	+	+
Загальна оцінка	20%	20%	20%	40%	60%	100%

7

Рисунок Г.7 – Порівняльний аналіз аналогів

Загальна модель роботи програми



8

Рисунок Г.8 – Загальна модель роботи програми

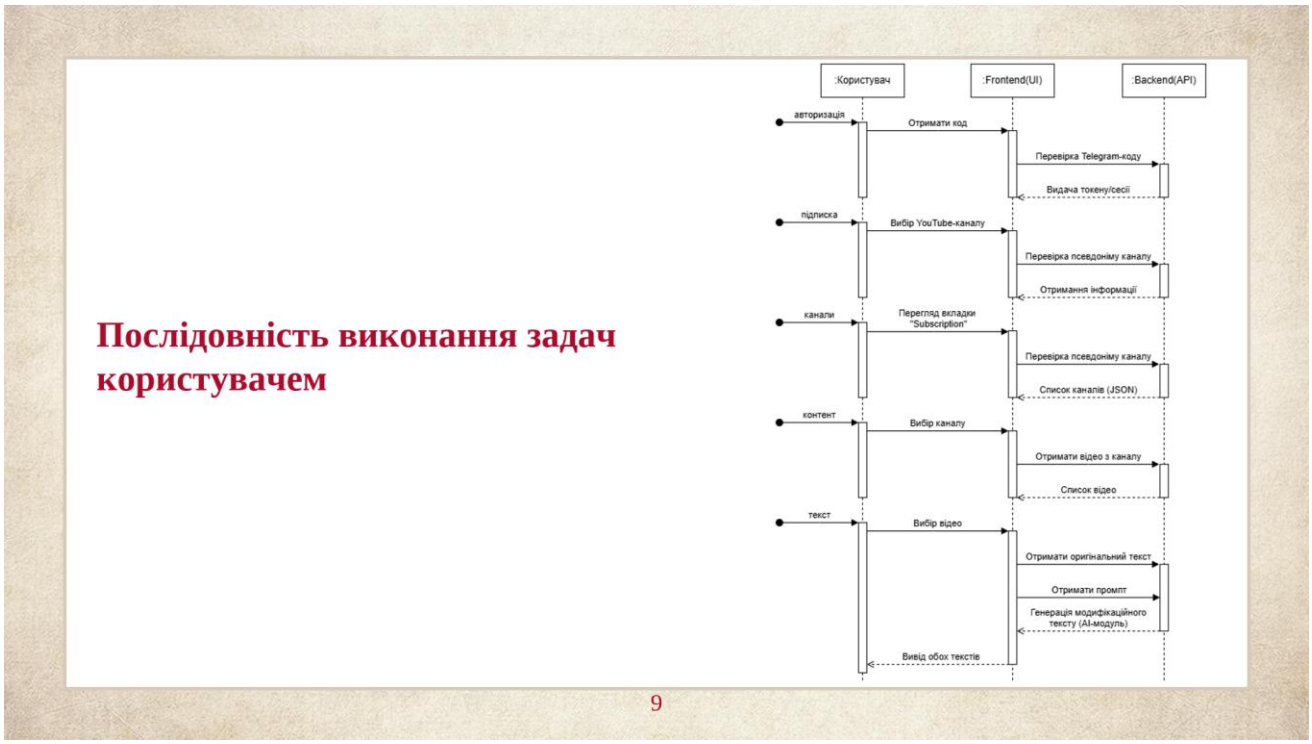


Рисунок Г.9 – Послідовність виконання задач користувачем

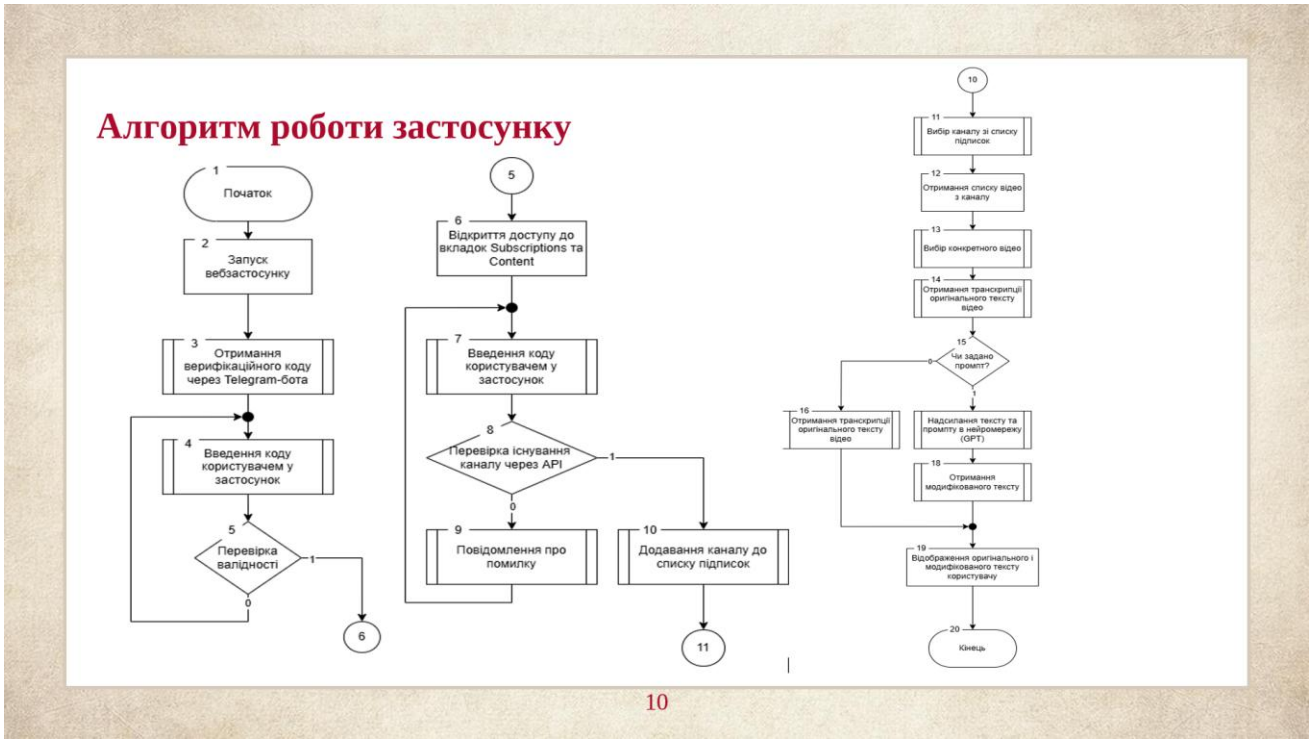


Рисунок Г.10 – Загальний алгоритм роботи застосунку

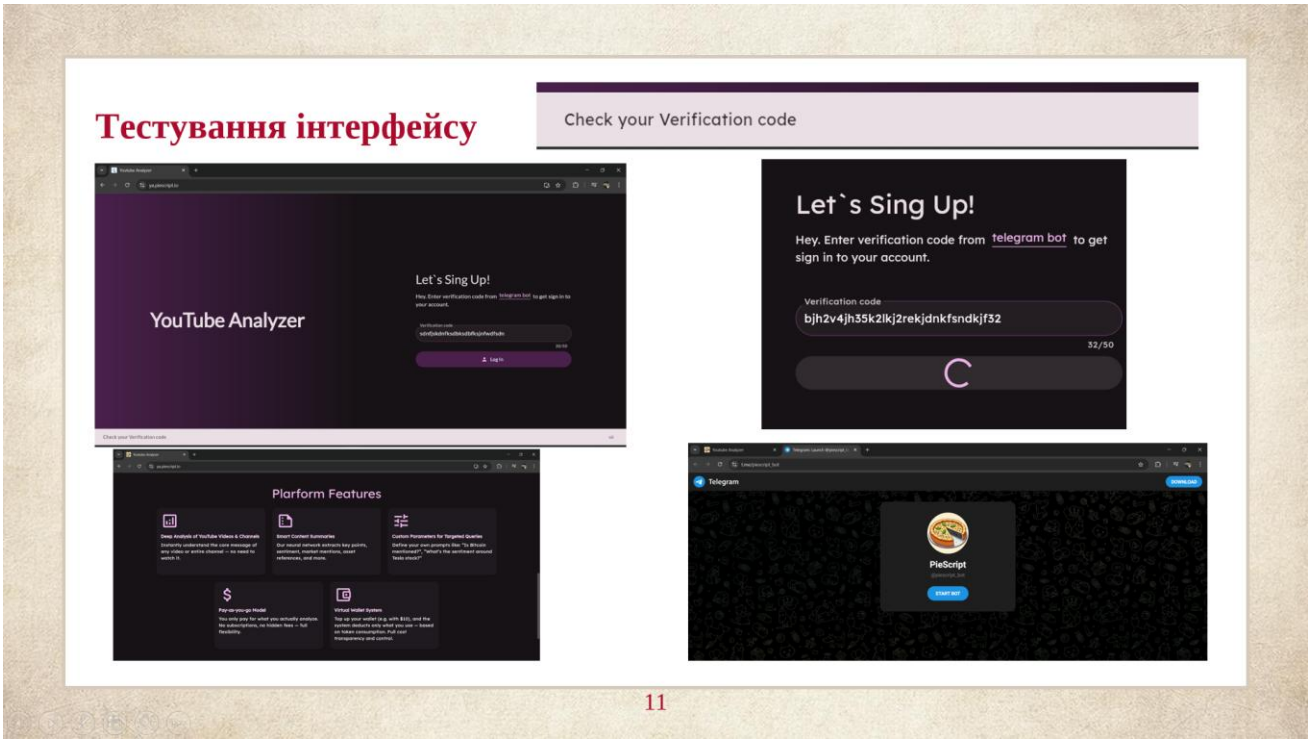
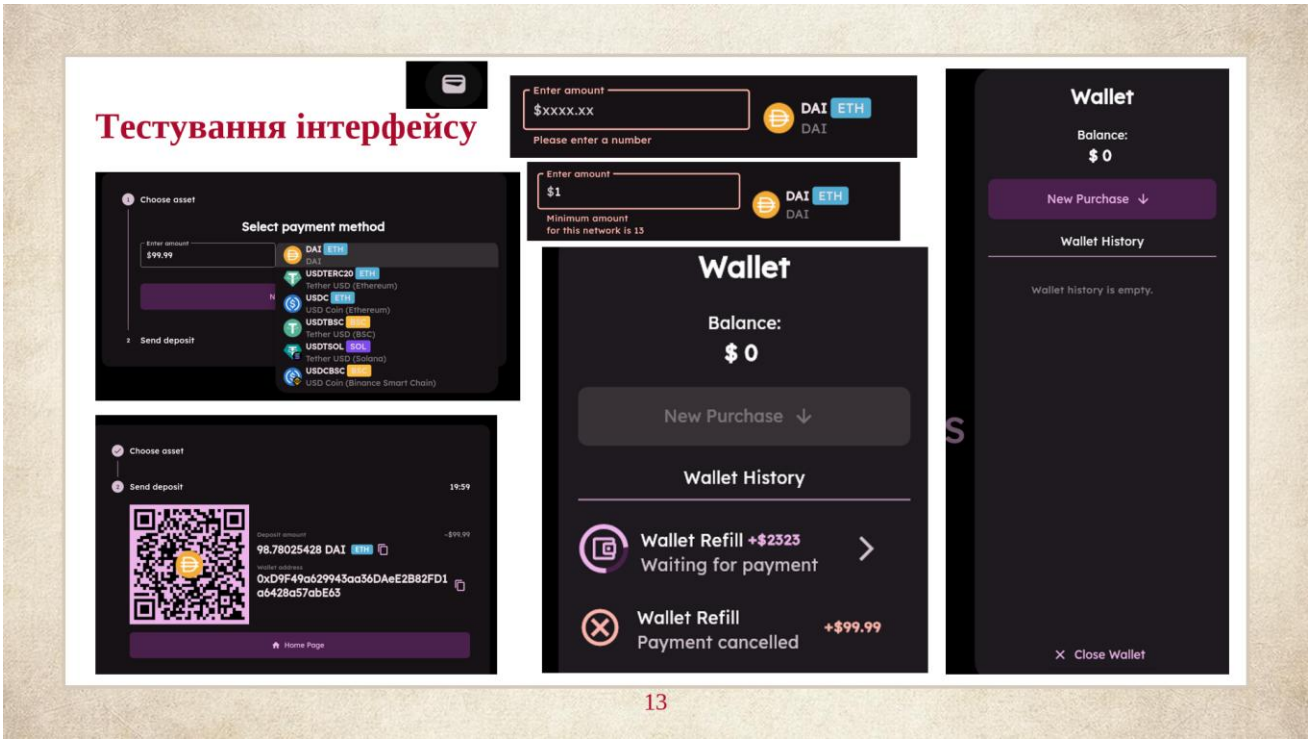


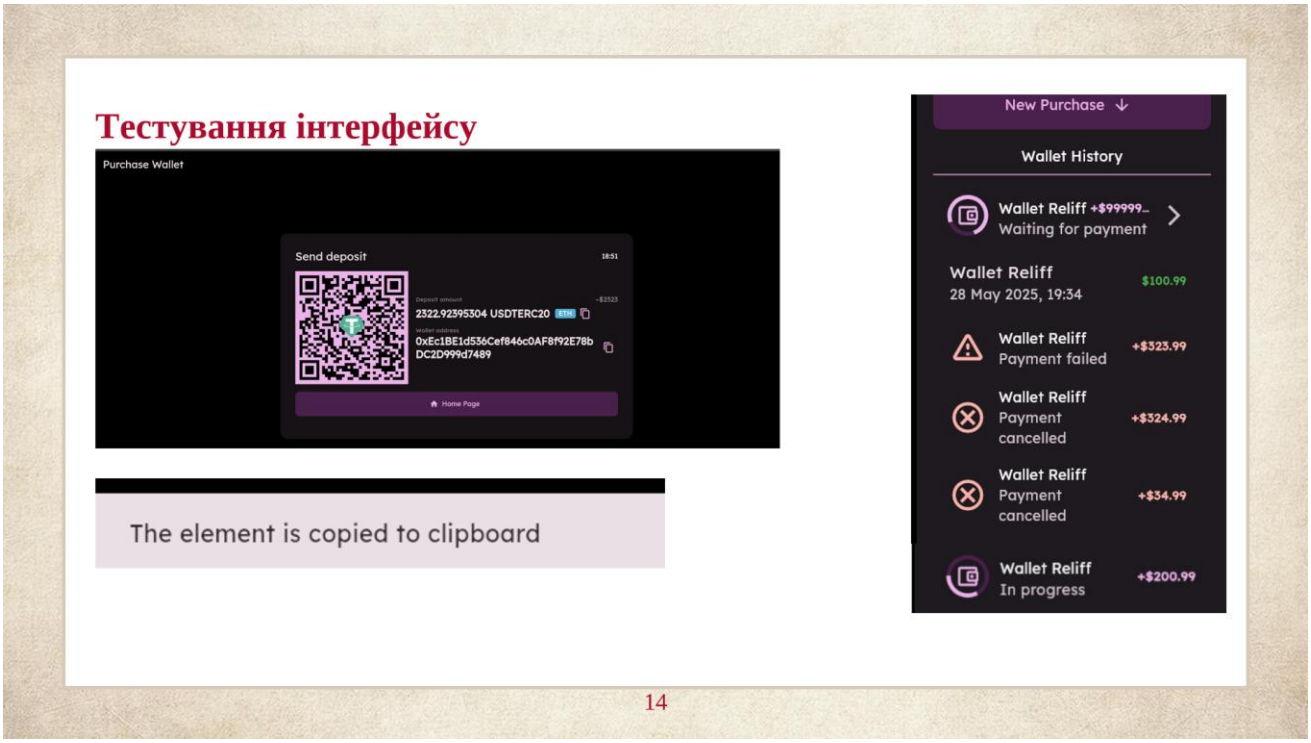
Рисунок Г.11 – Тестування авторизації на вебсайті

Рисунок Г.12 – Тестування оформлення підписки та отримання відео з каналу



13

Рисунок Г.13 – Тестування першого екрану поповнення криптогаманця



14

Рисунок Г.14 – Тестування другого екрану поповнення з історії криптогаманця

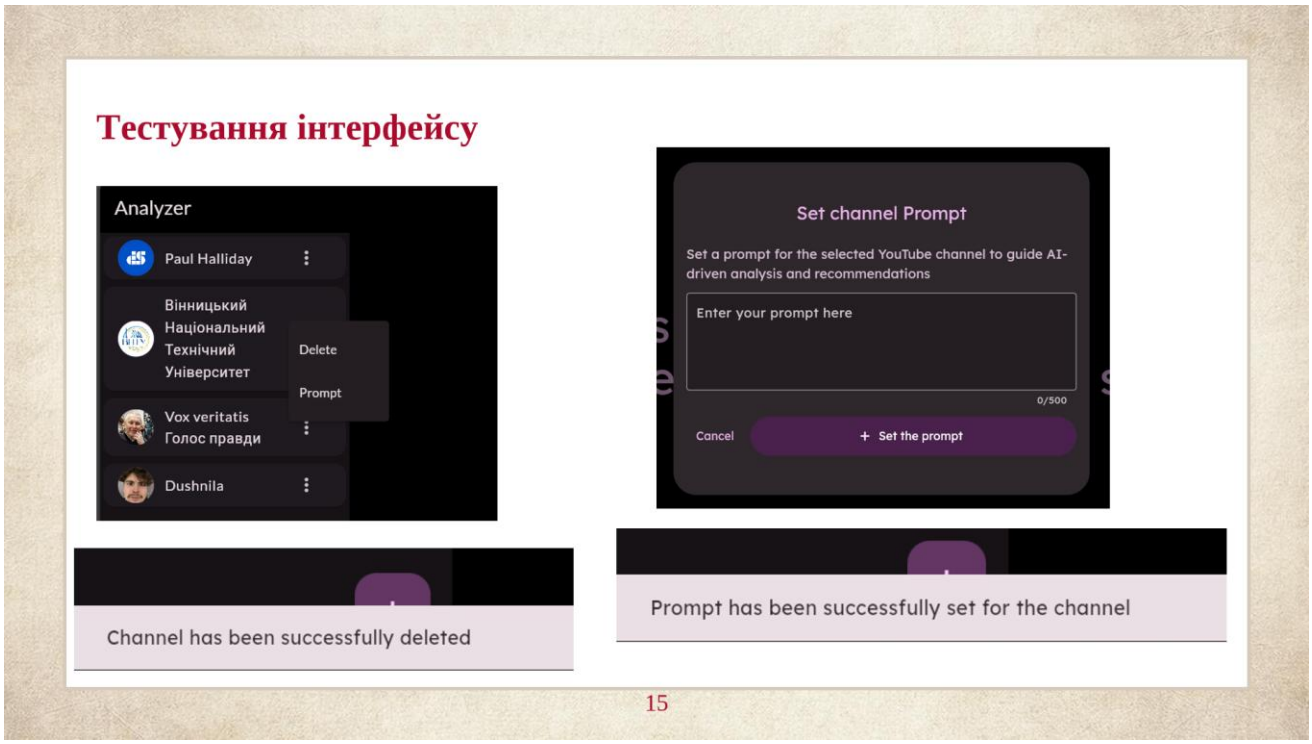


Рисунок Г.15 – Тестування встановлення промпту на канал та видалення каналу

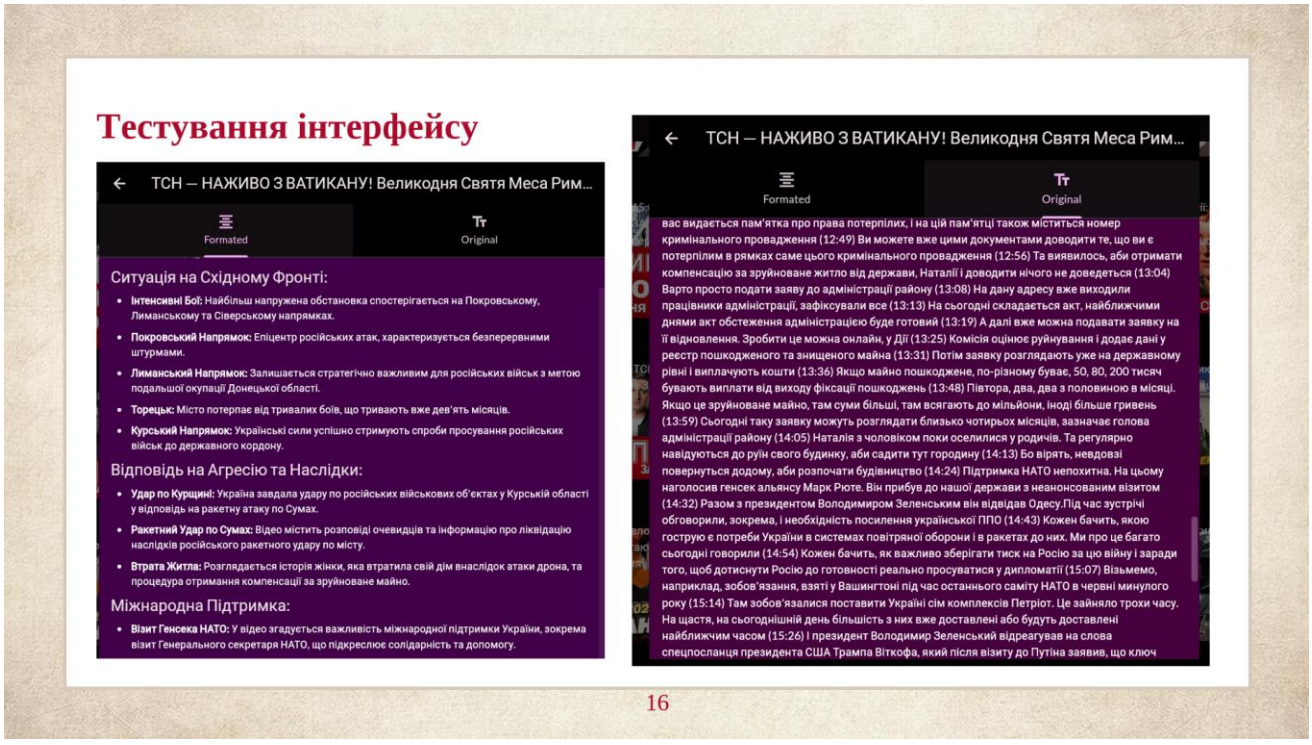


Рисунок Г.16 – Тестування отримання оригінального та модифікованого тексту за встановленим промптом

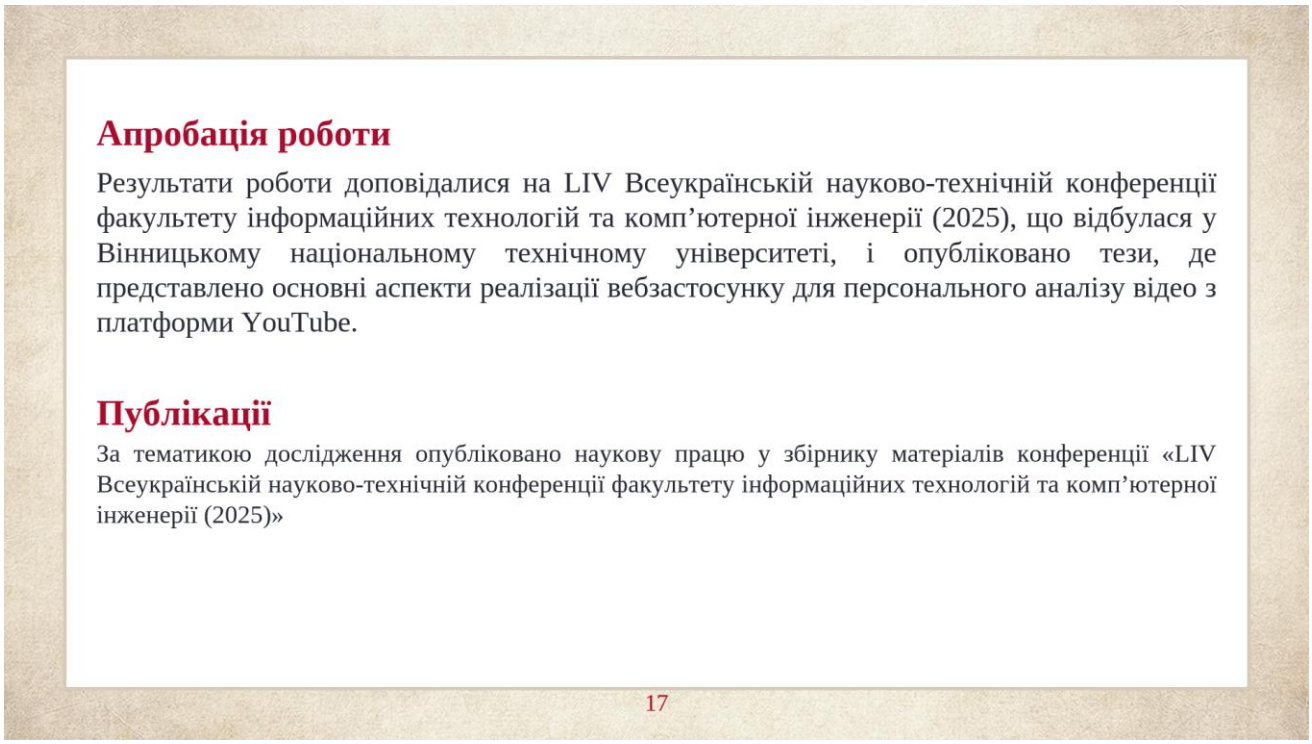


Рисунок Г.17 – Апробація та публікація результатів роботи

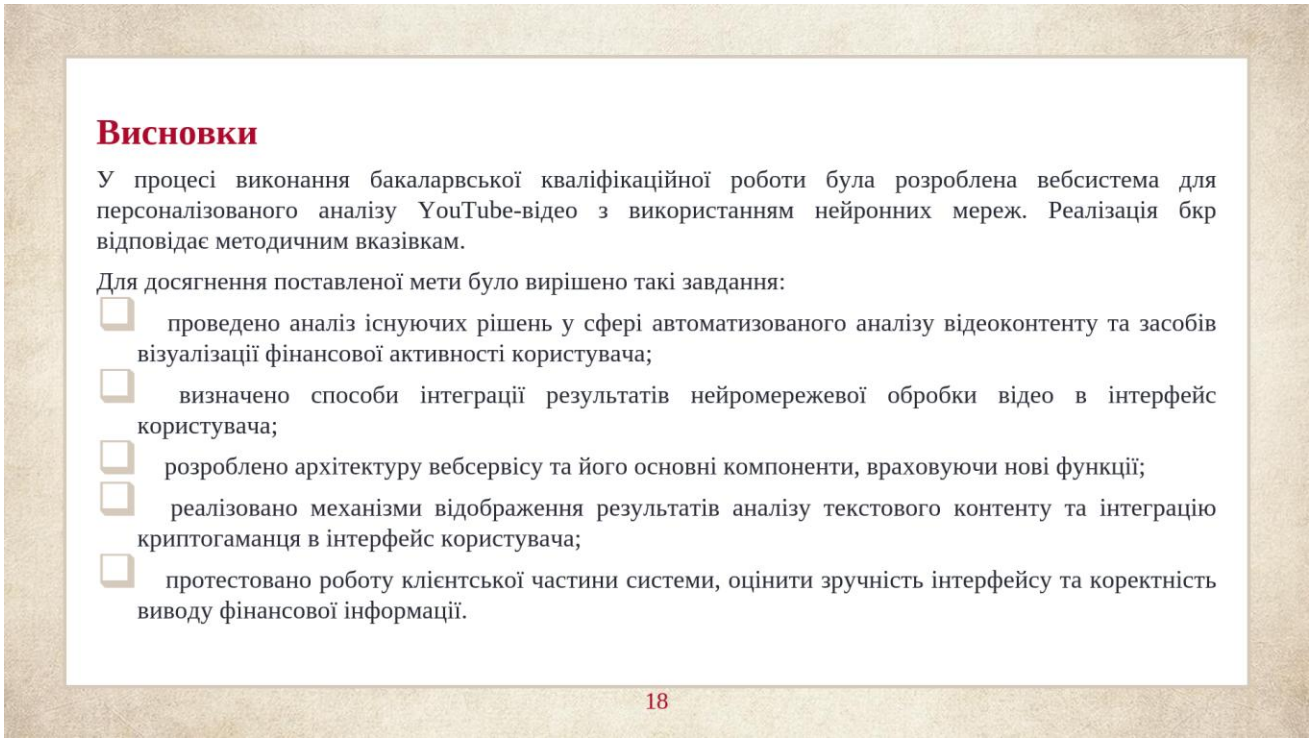


Рисунок Г.18 – Висновки

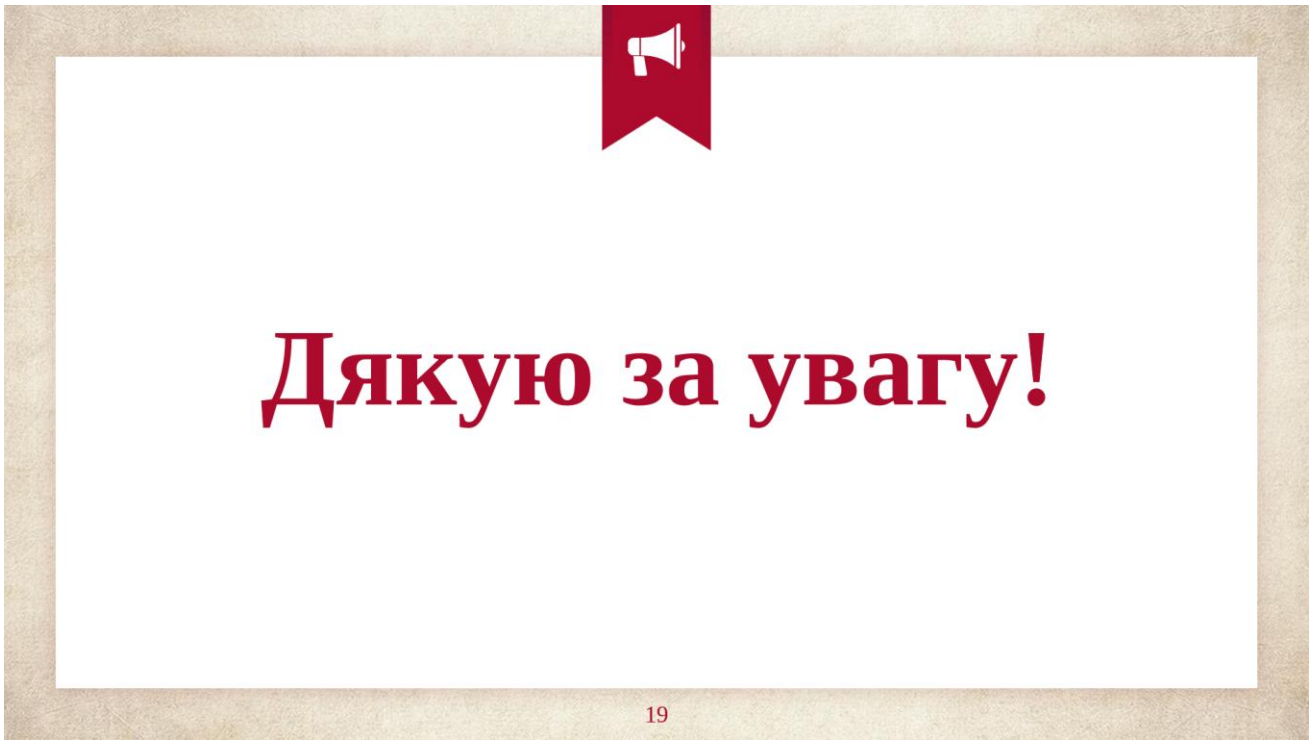


Рисунок Г.19 – Фінальний слайд