Міністерство освіти і науки України

Одеський національний політехнічний університет

Навчально-науковий інститут комп’ютерних систем

Кафедра системного програмного забезпечення

*Станіславик Ярослав Георгійович*,

студент групи АС-151

**КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА**

*Програмний аналізатор для фільтрації спаму*

*у коментарях публікацій мережі Facebook*

Спеціальність:

121 – Інженерія програмного забезпечення

Спеціалізація:

Інженерія програмного забезпечення

Керівник:

*Комлева Наталія Олегівна,*

*канд. техн. наук, доцент*

Одеса – 2020

**ЗМІСТ**

|  |  |
| --- | --- |
| ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ | 4 |
| ЗАВДАННЯ НА РОЗРОБКУ РОЗДІЛУ «ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ» | 6 |
| АНОТАЦІЯ | 7 |
| ВСТУП | 8 |
| РОЗДІЛ 1 КРИТИЧНИЙ АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ ДЛЯ ФІЛЬТРАЦІЇ СПАМУ | 10 |
| 1.1 Класифікація спаму у соціальних мережах | 10 |
| 1.2 Основні механізми фільтрації спаму | 11 |
| 1.3 Управління сторінками соціальної мережі Facebook | 12 |
| 1.4 Огляд існуючих програмних аналогів для фільтрації спаму | 16 |
| РОЗДІЛ 2 РОЗРОБКА ПРОГРАМНОГО АНАЛІЗАТОРА ДЛЯ ФІЛЬТРАЦІЇ СПАМУ | 19 |
| 2.1 Призначення програмного аналізатора | 19 |
| 2.2 Модель Facebook-публікацій | 21 |
| 2.3 Визначення принципів роботи програмного класифікатору | 23 |
| 2.4 Верифікація роботи програмного класифікатору | 25 |
| РОЗДІЛ 3 СПЕЦИФІКАЦІЯ ВИМОГ ДО ПРОГРАМНОЇ СИСТЕМИ | 28 |
| 3.1 Вимоги до функціональних характеристик | 28 |
| 3.2 Нефункціональні вимоги до програмного засобу | 35 |
| 3.3 Технічні вимоги | 37 |
| РОЗДІЛ 4 ПРОЕКТУВАННЯ ПРОГРАМНОГО АНАЛІЗАТОРУ ДЛЯ ФІЛЬТРАЦІЇ СПАМУ | 38 |
| 4.1 Проектування архітектури аналізатору | 38 |
| 4.2 Опис програмних модулів | 39 |
| 4.3 Опис функціональної поведінки системи | 46 |
| 4.4 Патерни проектування у системі | 50 |
| 4.5 Сховище даних | 50 |
| РОЗДІЛ 5 ПРОГРАМНА РЕАЛІЗАЦІЯ РОЗРОБЛЮВАНОЇ СИСТЕМИ | 53 |
| 5.1 Особливості створення програмних модулів з урахуванням  мови програмування | 53 |
| 5.2 Розробка програмного інтерфейсу | 56 |
| 5.3 Особливості розробки моделі класифікації спаму у коментарях | 58 |
| РОЗДІЛ 6 ВИЗНАЧЕННЯ ВЛАСТИВОСТЕЙ АНАЛІЗАТОРА СПАМА | 65 |
| 6.1 Тестування основних класів | 65 |
| 6.2 Контрольні приклади роботи системи | 68 |
| РОЗДІЛ 7 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА У НАДЗВИЧАЙНИХ  СИТУАЦІЯХ | 71 |
| ВИСНОВКИ | 73 |
| ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ | 74 |
| Додаток А. МАТЕРІАЛИ ТЕЗИСІВ | 76 |
| Додаток Б. ЛІСТИНГ ПРОГРАМИ | 78 |
| Додаток В. ПИТАННЯ З ОХОРОНИ ПРАЦІ ТА БЕЗПЕКИ У  НАДЗВИЧАЙНИХ СИТУАЦІЯХ | 99 |
| В.1 Організація та управління охороною праці на робочому місці  інженера програмного забезпечення | 99 |
| В.2 Обґрунтування заходів з покращення умов охорони праці | 100 |
| В.3 Розрахунок захисного занулення | 103 |
| В.4 Надзвичайні ситуації та шляхи їх запобігання | 105 |
| В.5 Заходи з охорони праці та безпеки у надзвичайних ситуаціях | 110 |

Міністерство освіти і науки України

Одеський національний політехнічний університет

Навчально-науковий інститут комп’ютерних систем

Кафедра системного програмного забезпечення

Рівень вищої освіти: другий (магістерський)

Спеціальність: 121 – Інженерія програмного забезпечення

Спеціалізація: Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ

Завідувач кафедри

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Крісілов В.А.

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_ р.

**ЗАВДАННЯ**

**НА КВАЛІФІКАЦІЙНУ РОБОТУ**

*Станіславика Ярослава Георгійовича*, група АС-151

1. Тема роботи: *Програмний аналізатор для фільтрації спаму у коментарях публікацій мережі Facebook*

Керівник роботи: *Комлева Наталія Олегівна, канд. техн. наук, доцент*

затверджені наказом ректора від «29» жовтня 2020 р. № 412-в.

1. Зміст роботи: проведення критичного аналізу існуючих рішень для фільтрації спам у соцмережах; розробка програмного аналізатору для фільтрації спаму, вибір алгоритмів; специфікація вимог до програмного аналізатору; проектування програмного аналізатору; програмна реалізація та визначення властивостей розроблюваного продукту; заходи з охорони праці та безпеки у надзвичайних ситуаціях.
2. Перелік ілюстративного матеріалу: згідно зі слайдами презентації.
3. Консультанти розділів роботи

|  |  |  |  |
| --- | --- | --- | --- |
| Розділ | Прізвище, ініціали та посада консультанта | Підпис, дата | |
| завдання видав | завдання прийняв |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

1. Дата видачі завдання: «\_02\_\_» \_\_\_\_\_\_09\_\_\_\_\_\_\_\_ 2020 \_\_\_\_ р.

**КАЛЕНДАРНИЙ ПЛАН**

|  |  |  |  |
| --- | --- | --- | --- |
| № з/п | Назва етапів кваліфікаційної роботи | Строк виконання | Примітка |
| 1 | Аналіз існуючих рішень | 2.09.2020 – 16.09.2020 | вик. |
| 2 | Специфікація вимог до програми | 17.09.2020 – 30.09.2020 | вик. |
| 3 | Проєктування програмної системи | 01.10.2020 – 17.10.2020 | вик. |
| 4 | Програмна реалізація та тестування системи | 18.10.2020 – 11.11.2020 | вик. |
| 5 | Завдання з охорони праці та БНС | 14.11.2020 – 28.11.2020 | вик. |
| 6 | Оформлення пояснювальної записки та графічного матеріалу | 29.11.2020 – 10.12.2020 | вик. |

**Здобувач вищої освіти** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ *Я. Г. Станіславік*

**Керівник роботи** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ *Н. О. Комлева*

**ЗАВДАННЯ**

на розробку розділу «Охорона праці та безпека в надзвичайних ситуаціях»

*Станіславика Ярослава Георгійовича*, група АС-151

Навчально-науковий інститут комп’ютерних систем

Кафедра системного програмного забезпечення

Тема роботи: *Програмний аналізатор для фільтрації спаму у коментарях публікацій мережі Facebook*

Зміст розділу:

1. Аналіз умов праці і вибір заходів і засобів захисту від небезпечних і шкідливих виробничих факторів.
2. Аналіз техногенних небезпек і вибір заходів і засобів забезпечення безпеки у надзвичайних ситуаціях
3. Індивідуальне завдання

Керівник роботи Консультант з охорони праці та БНЗ

\_\_\_\_\_\_\_\_\_\_\_\_\_ *Н. О. Комлева* \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А. Ю. Москалюк

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 20 \_\_\_ р. «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 20 \_\_\_ р.

# АНОТАЦІЯ

Метою роботи є збільшення кількості корисної інформації, що знаходиться у коментарях публікацій Facebook, за рахунок розробки та програмної реалізації відповідної інформаційної технології, що здійснює класифікацію та подальше видалення спаму з коментарів. Технологіями розробки є Facebook API Pages, як джерело даних, мова програмування C++, за допомогою якої здійснюється робота з інтерфейсом системи, мова програмування Python, за допомогою якої здійснюються запроси до Facebook API Pages, бібліотека sklearn, яка допомагає у класифікації спаму, фреймворк QT, що використовується для розробки інтерфейсу системи. Як результат роботи виконана програмна реалізація системи, яка надає можливість видаляти спам з коментарів публікацій Facebook.

Ключові слова: Facebook API Pages, класифікація спаму, токен користувача, маркер доступу, коментарі публікацій, інтелектуальній аналіз даних, sclearn.

**ABSTRACT**

The aim of the work is to increase the amount of useful information contained in the comments of Facebook posts, through the development and software implementation of appropriate information technology that classifies and further removes spam from comments. Development technologies are Facebook API Pages as a data source, C ++ programming language, which is used to work with the system interface, Python programming language, which is used to query Facebook API Pages, sklearn library, which helps to classify spam, QT framework, used to develop the system interface. As a result of the work, a software implementation of the system was made, which provides an opportunity to remove spam from comments on Facebook posts.

Keywords: Facebook API Pages, spam classification, user token, access token, post comments, data mining, sclearn.

**ВСТУП**

В даний час методи боротьби зі спамом, які засновані на правилах фільтрації повідомлень, стають все менш ефективними. Це відбувається через вимоги до збільшення витрат на працю фахівців із захисту від спаму, а також на підтримку правил фільтрації в актуальному стані [1]. З цього випливає, що для методів боротьби зі спамом необхідно постійної участі людини. Це дозволяє ефективно аналізувати текст. В іншому випадку методи не здатні самонавчатися і самостійно виробляти нові фільтраційні правила [2]. Коли при боротьбі зі спамом задіяна людина, вона ґрунтується на досвіді роботи з мережею Facebook та власних перевагах щодо коментарів, і тому має здатність правильно виявляти ознаки спаму та самонавчатись.

Автоматизація процедур фільтрації дозволяє економити час та інші ресурси і є набагато більш привабливою ніж ручна робота. При цьому процес фільтрації не зводиться до шаблонів і є більш ефективним. Саме тому завдання створення інструменту для боротьби зі спамом зводиться до наділення цього інструменту такими властивостями, як: здатність до навчання, використання системи переваг і винятків, аналіз контексту коментарів Facebook, застосування системи прийняття рішень [3, 4].

Метою роботи є збільшення кількості корисної інформації, що знаходиться у коментарях публікацій Facebook, за рахунок розробки та програмної реалізації відповідної інформаційної технології, що здійснює класифікацію та подальше видалення спаму з коментарів. Під терміном «одиниця корисної інформації» буде вважатися коментар, що не є спамом.

Для досягнення цієї мети вирішено наступні завдання:

* розглянуто принципи роботи мережі Facebook та інформаційно-аналітичний склад структури сторінки на публікації Facebook;
* виконано аналіз методів класифікації з визначенням особливостей застосування кожного алгоритму;
* сформовано перелік параметрів Facebook-публікації для можливості контролю корисності її коментарів, та запропоновано спосіб визначення цієї корисності;
* визначено вимоги до функціональних та нефункціональних характеристик розроблюваного програмного застосування;
* виконано проектування, програмна реалізація та тестування програми.

Наукова новизна одержаних результатів:

* вперше отримано модель Facebook-публікацій з врахуванням взяємозв’язку контенту публікації та ступені корисності коментарів;
* одержала подальший розвиток методика класифікації спаму та алгоритм видалення спаму з коментарів Facebook-публікацій.

Практична цінність одержаних результатів. Виконана програмна реалізація системи, яка надає можливість видаляти спам з коментарів публікацій Facebook. Використання програми дозволило знизити кількість некорисних коментарів приблизно на 11,9%.

Був проведений аналіз структури Facebook-публікацій, результати опубліковано у збірнику доповідей XIIІ міжнародної науково-практичної конференція «Інформаційні технології і автоматизація – 2020» [5].

Даний звіт є результатом виконання дипломного проектування. Він містить вступ, сім розділів, висновки, список літератури та додатки.

### **КРИТИЧНИЙ АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ**

### **ДЛЯ ФІЛЬТРАЦІЇ СПАМУ**

**1.1 Класифікація спаму у соціальних мережах**

Під спамом розуміється відправлення кореспонденції рекламного чи іншого характеру людям, які не висловили бажання її одержувати. Спочатку термін «спам» стосувався рекламних електронних листів, але потім його значення стало більш широким.

Популярність соціальних мереж стрімко зростала. Вони опинилися під пильною увагою спамерів, які здійснювали спам-розсилки. Спочатку спам-посилання надходили з власних адресів, але цьому було легко запобігти, модератори мали засоби для боротьби з цим явищем. Тому пізніше спамери почали використовувати технологію крадіжки профілів інших користувачів та надсилати спам-повідомлення саме з них. З таким способом боротись трудніше, тому він застосовується спамерами й досі.

Спам у соціальних мережах може бути класифікованим за наступними типами:

* рекламні банери та push-повідомлення;
* фейкові публікації (пости), що найчастіше розсилаються спам-ботами;
* спам-коментарі під публікаціями.

Одним з різновидів повідомлень є push-повідомлення. Вони періодично з’являється на екрані, мають невеличкий розмір та можуть мати посилання. Найчастіше вони містять оголошення про події чи елементи рекламної кампанії. Push-повідомлення можуть розміщуватись як справжніми інформаційними джерелами у рамках маркетингових кроків, так і підставними особами, метою яких є розповсюдження спаму. Потрібно вміти розрізняти комерційне рекламування марки чи бренду, надання користувачам корисної інформації про появу нових технологій, інформаційних майданчиків, комерційного контенту та ін. від спаму та клікбейт-повідомлень.

Спамери є активними користувачами багатьох форумів, на якіх створюють профілі та розміщують у них різноманітні спам-посилання. Такі ж посилання вони залишають і в коментарях інших користувачів форумів та блогів. Проблема є в тому, що виявити та заблокувати таких спамерів не завжди вдається. У деяких випадках спам-посилання переводять на зовнішні ресурси, що не мають ніякого відношення до тематиці форуму чи блогу. В інших випадках рекламують власні продукти, сервіси та послуги.

Поширення спаму в коментах є серйозною проблемою, і саме вирішенню цієї проблеми для соціальної мережі Facebook присвячена дана робота.

* 1. **Основні механізми фільтрації спаму**

Існує багато різних способів, за допомогою яких можливо боротися з різноманітним спамом у відповідних коментарях:

* застосування та використання премодерації;
* бан розсильників (спамерів) через IP-адресу: зазвичай бан полягає в тому, що виконується певне обмеження відповідних конкретних прав користувача (наприклад, це може бути створення/відправлення якихось повідомлень або створення якихось нових тем на [певних](https://uk.wikipedia.org/wiki/%D0%A4%D0%BE%D1%80%D1%83%D0%BC_(%D1%96%D0%BD%D1%82%D0%B5%D1%80%D0%BD%D0%B5%D1%82-%D1%80%D0%B5%D1%81%D1%83%D1%80%D1%81)) інтернет-форумах, відправлення певних повідомлень в групі ([чаті](https://uk.wikipedia.org/wiki/%D0%A7%D0%B0%D1%82)), на коментування на певну тематику в [блогах](https://uk.wikipedia.org/wiki/%D0%91%D0%BB%D0%BE%D0%B3) та інше). Можливість було введено з метою захисту будь-яких інтернет-сайтів від інтернет-[тролів,](https://uk.wikipedia.org/wiki/%D0%A2%D1%80%D0%BE%D0%BB%D1%96%D0%BD%D0%B3) спамерів та будь-яких інших осіб, чиї повідомлення несуть загрозу продуктивній роботі сайту);
* прикріплення якоїсь капчі в відповідну форму для написання коментарів: Ще може виглядати так - CAPTCHA  – це свого роду тест Тюринга, який являється повністю автоматизованим та використовується для розрізнення роботи комп'ютерів від роботи людей. У самому поширеному варіанті CAPTCHA потребується від користувача ввести певні символи, які зображені, зазвичай, у спотвореному вигляді та розташовані на якомусь малюнку, іноді на цій малюнок додається шум або напівпрозорість. Також (але рідше) застосовують CAPTCHA, яка є заснована на розпізнаванні певної мови (зазвичай це використовується для людей з певними порушеннями зору), або яка заснована на інших варіантах завдань з наявністю штучного інтелекту. CAPTCHA частіше за все використовується коли необхідно запобігти використанню будь-яких інтернет-сервісів певними ботами (наприклад - запобігання автоматичній реєстрації на сайті, викачуванню з сайту файлів, масовим розсилкам тощо);
* видалення з усіх форм для написання коментарів такої позиції як «сайт» (іншими словами, це - заборона наявності будь-яких посилань на інші ресурси в полі для коментарів).

**1.3 Управління сторінками соціальної мережі Facebook**

API Pages надає можливість програмам здійснювати доступ до певної Facebook-сторінки та виконувати з нею наступні дії:

* отримувати та оновлювати налаштування обраної сторінки;
* створювати і отримувати публікації;
* отримувати коментарі до публікацій;
* отримувати та оновлювати статистику сторінки та ін.

«Для того, щоб отримати доступ до певної сторінки, по-перше - потрібно отримати певний доступ до Token-у користувача. Цей Token можна отримувати та користуватися ним протягом двох або трьох місяців.

Для отримання такого Токену, користувач повинен відправити наступне:

-curl- -i -X –GET- "https://graph.face-book.com/oauth/access\_token?- grant\_type=fb-\_exchange\_token- & -client\_id={-app-id-} & client\_secret={-app-secret-} & fb\_exchange\_token={-short-lived-user-access-token-}"

Якщо операція проведена успішно, то програма виведе певний результат з "access\_token" = "{long-lived-user-access-token}"» [6]

Далі користувачу необхідно надати дозвіл до певної сторінки. Користувач може налаштувати права доступу для кожній сторінки так, як він захоче. Так само він може обрати особу, якою ці дії будуть виконуватися. Функції особи можуть бути різними: це може бути і модерація сторінок, і створення постів і так далі. Головна умова для виконання цих дій – особа повинна мати на це певні права. Для отримання прав особа повинна отримати певний Токен для доступу до сторінки. Цей токен обов’язково повинен мати ідентифікатор «page-id». Щоб отримати такий ідентифікатор, потрібно надіслати певний запит «GET» на кінцеву точку, яка має назву {page-id}. У цьому запиті система зчитує маркер доступу користувача:

curl -i -X –GET- "https://graph.facebook.com/{page-id}? -fields-=access\_token& access\_token={user-access-token}"

Якщо виконання проходить успішно, то ми отримуємо результат: "id" = "{page-id}" та "access\_token" = "{page-access-token}". Якщо потрібен маркер с короткочасною тривалістю роботи, то треба мати на увазі, що він буде діяти лише протягом однієї години. Якщо використовувати довгостроковий маркер, то обмеження в часі для користування ним буде необмеженим.

Якщо користувач лише керує сторінкою, не будучи при цьому її власником, то він може отримати маркери сторінок для того, щоб можна було їми керувати. Для отримання списку «сторінка –маркер доступу», необхідно використовувати власний маркер та обов’язково необхідно визвати спец. дозвіл «pages\_show\_list». Особа, яка хоче отримати маркер доступу, повинен бути наділена можливістю виконувати на цій сторінці певні завдання, тобто мати на це відповідні права. Щоб отримати такі права, користувачу необхідно відправити запит «GET» / {user-id} / account:

curl -i -X GET "https://graph.facebook.com/{user-id} / accounts ? fields=name, access\_token & access\_token={user-accesss-token}"

Розглянемо приклад, коли користувач зробив запит щодо доступу до декількох сторінок. У разі успіху повертається наступне [7]: «{

"data": [

{

"name": - "Facebook Page 1",

"access\_token": - "{page-access-token-for-Page-1}",

"id": - "{page-1-id}"

},

{

"name": - "Facebook Page 2",

"access\_token": - "{page-access-token-for-Page-2}",

"id": - "{page-2-id}"

},

{

"name": - "Facebook Page 3",

"access\_token": - "{page-access-token-for-Page-3}",

"id": - "{page-3-id}"

},»

...

Усі такі маркери доступу в конкретних обставинах можуть іноді стати недійсними. Наприклад, якщо пароль певної сторінки зміниться, або якщо користувач, який запитує маркер, вже не має ролі на цій сторінці (видалення, корегування, додавання, модерація та інше), чи коли виявлена якась проблема з теми безпеки, тоді маркер доступу до цієї сторінці, як правило, стане недійсним.

Для отримання таких постів якоїсь певної сторінки зазвичай потрібен дозвіл на читання – «pages\_read\_engagement». Такий маркер доступу до певної сторінки повинен надаватися особою, яка має певні права на модерацію конкретної сторінки: MODERATE.

Якщо ця персона, яка надсилає спам-message, визначена – вона може бути моментально заблокована. Перш за все, треба затребувати певний дозвіл: pages\_manage\_engagement – (MANAGE (від англ.) – це управління), цей дозвіл – pages\_read\_user\_content (MODERATE (від англ.) – надання дозволу) та певний маркер доступу до сторінки, запитуваний тією особою, яка зможе виконати операцію [MANAGE або MODERATE] певного завдання для сторінці, саме на яку виконується цей запит.

Якщо нам відомий ідентифікатор певної особи, то потрібно надіслати певний запит – POST на {page-id}, так звану заблоковану кінцеву точку, у якої наявний ідентифікатор особи, що потрібно заблокувати:

curl -i -X POST - "https://graph.facebook.com/{page-id}/blocked?, user={user-id-to-block} &access\_token={«page-access-token»}"

При успіху блокування ми отримуємо значення - "{user-id-to-block}", яке є «true».

Для отримання певних показників публікації конкретної сторінки нам потрібно відправити запит «GET» до певної кінцевої точки – {page-post-id} із полями певних показників:

curl -i - X - GET "{«page-post-id»} / metric = post\_reactions\_like\_total, post\_reactions\_love-total, post\_reactions\_wow-total & access\_token = {«page-access-token»}". Після успіху ця програма вже отримує відповідь, за допомогою якої можна визначити кількість конкретних реакцій кожного типу, поданого на публікацію.

При використанні програми API Pages вже потрібно буде враховувати обмеження кількості усіх звернень. Як правило, це – кількість викликів «API», яку програма або користувач може виконати за якійсь певний проміжок часу. Якщо це обмеження (або це може бути обмеження на певне завантаження ЦП та його загальний час) перевищено, то для користувача або програми вже починає діяти певне регулювання, тоді подальші запити до «API» буде завершено помилкою. Як правило, обмеження кількості звернень зазвичай поширюється на всі можливі виклики «API».

Для усіх запитів до «API Pages» діє обмеження платформи, та (або) обмеження для певних комерційних програм. Це, як правило, залежить від типу маркера у цьому запиті. При використанні маркеру доступу користувача або програми діють певні обмеження платформи. Та якщо це – маркер доступу сторінки або системного користувача – то це вже обмеження для комерційних програм.

**1.4 Огляд існуючих програмних аналогів для фільтрації спаму**

Найближчим і найвідомішим аналогом для програмного продукту, який розробляється, є CPA.Rip Cleaner. Зазвичай, для коректної роботи сервісу треба обрати певний токен на конкретній сторінці, де відбувається налаштування цього платежу (<https://cpa.rip/facebook/api-token-facebook/>). Після додавання токену потрібно обрати певний пост, в якому треба почистити спам (зайві коментарі) (це може бути одна публікація чи набір публікацій). Завдання, як правило, ставиться на фоні, а коментарі зазвичай чистяться. Також, для кращої роботи, можна використовувати свої юзерагенти та/або проксі. Token додається, як правило, зі свого кабінету, який відповідає за рекламу. Зазвичай, Proxy вводиться у слідуючому форматі:

-IP-: -ПОРТ- або

-IP-: -ПОРТ-: -ЛОГІН-: -ПАРОЛЬ-,

тип кожного Proxy визначається автоматично. Але якщо поле для вводу залишається порожнім, то використовують системні Proxy.

У якості -User-Agent- можна також використовувати -User-Agent- (лише свій). Наприклад, для того, щоб звернутися до «API Facebook» можна з бази використовувати «User-Agent». Після додавання токену, та при потребі заповнення якихось інших полів, треба обрати потрібний «РА» (або Adset) , або можна обрати якийсь конкретний пост або оголошення.

Далі, зазвичай, обираються конкретні параметри для виконання цих завданнь, тривалість їх чищення, певні операції над цими коментарями – наприклад, можна «Сховати» або «Видаляти» певні коментарі, або можно по різному відобразити вимоги до ціх коментарів – як приклад – відобразити всі існуючі або відобразити тільки нові. Після такого запуску це завдання вже буде працювати у так званому фоновому режимі, та буде працювати 6, 12 або 24 години (в залежності від налаштування). Після цього часу треба буде запускати чистку заново. Далі треба перевірити наявність нових коментарів. Це, як правило, відбувається раз у 4/8/16 хвилин за так званою прогресивною шкалою. У ситуації, коли за перші 4-и хвилини система не знайде ніякого коментаря, то вже наступна перевірка буде відбуватися на 8-й хвилині. Якщо і на 8 хвилині не буде нічого знайдено, то наступна перевірка, аналогічно буде відбуватися вже через 16 хв. У випадку, коли всі необхідні коментарі було знайдено ще при першій перевірці, то вже наступна перевірка буде так само виконуватися, але вже через 10 хв.

Сервіси Society Checker та JustClean являються менш близькими аналогами, які відрізняються від сервісу CPA.Rip Cleaner, який є достатньо недорогого, більш високою ціною та, окрім цього, мають обмеження по деяким характеристикам.

Порівняльна характеристика аналогів наведена у табл. 1.1. На перетині строки і стовпця стоїть «+», якщо ця програмна система вирішує певну проблему, і «-», якщо не вирішує.

Таблиця 1.1 – Порівняльна характеристика програмних продуктів

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Назва продукту | Властивості продукту | | | | |
| Класифікація спаму | Зберігання спаму у журналі | Відобра­ження статис­тики | Видалення спаму вручну | Персоналіза­ція моделі класифікації спаму |
| CPA.Rip Cleaner | + | - | + | - | + |
| Society Checker | + | - | - | + | + |
| JustClean | + | - | - | + | - |
| Розроблюваний програмна система | + | + | + | + | + |

Аналогі проаналізовано за наступними властивостями програмних продуктів:

* можливість класифікації певного спаму;
* можливість зберігання певного спаму у журналі (протягом часу, який зазначено користувачем);
* існує доступ до системи «Facebook»;
* відображення в системі статистики з кількості певного спаму;
* можливість видалення конкретного спаму вручну;
* персоналізація моделі класифікації спаму;
* можливість авторизації у системі.

Цей проведений аналіз доказує необхідність розробки такої системи, яка буде надавати можливість видаляти певний спам з коментарів публікацій «Facebook», що зазвичай усуває виявлені недоліки у порівнянні з його аналогічними програмними продуктами.

1. **РОЗРОБКА ПРОГРАМНОГО АНАЛІЗАТОРА ДЛЯ ФІЛЬТРАЦІЇ СПАМУ**

**2.1 Призначення програмного аналізатора**

Ця програмна система розроблена для того, щоб можна було видаляти певну інформацію, наприклад, якщо вона виглядає як спам. Для цього використано спам-фільтр, який є самонавчальним. Цей фільтр є найкращою частиною цієї програмної розробки, бо він розуміє потреби кожного користувача та відповідно підлаштовується під ці потреби.

“Користувач” є основним актором у цій предметної області. Йому доступні такі види діяльності:

1. Класифікація спаму

Користувач може аналізувати певні коментарі під будь-якими постами та перевіряти, чи наявний у них спам. Якщо процедуру аналізу вже їм розпочинено, то після завершення цієї процедури система виводить на екран користувачу список з усіх коментарів, які мають відмітку «спам».

1. Зберігання спаму на протязі того часу, який від самого початку обрано користувачем.

Також, для того, щоб перенавчити в майбутньому цю модель класифікації, цей весь спам, який був видалений, все одно зберігається у цій системі. Також користувачеві дозволяється якось відмітити коментар у журналі, ця відмітка буде впливати на врахування цього видаленого коментаря при перенавчанні класифікатора.

Усі коментарі, які були помічені як спам (але користувач їх не видалив) під час цього аналізу, так само зберігаються у журналі. Все одно пізніше у Користувача все одно є можливість видалення коментаря, це існує для того, щоб можна було інформувати систему про певний збій (помилку) у класифікації. У такому випадку цей видалений з журналу коментар буде значитися як коментар, який не є спамом і знову буде брати участь у перенавчанні цій системи,.

1. Відображення статистики з наявності спаму (і його кількості) в системі.

Для статистиці у журналі існує окремий розділ. Головне – показати у цьому розділу наскільки забруднена спамом система та показати користь від отримання цього продукту.

У статистиці присутні наступні основні величини:

* кількість спаму (у вигляді коментарів), який було видалено;
* кількість спаму, який було видалено, але його показано як співвідношення цього спаму до інформації, яка виявилася корисною;
* діаграма розподілу (згідно з конкретними датами) всього спаму;
* графік зростання у всіх коментарях усієї корисної інформації а також під окремими постами. Також можна буде порівняти з тим періодом, коли користувач ще не використовував фільтр-систему.

1. Видалення коментарів вручну

Користувач також може видаляти будь-які коментарі вручну, або з використанням класифікацію усього спаму. Це дуже облегшує роботу при впізнаванні усіх потреб користувача, коли він намагається видалити усю некорисну, згідно з його інтересами, інформацію.

1. Персоналізація моделі, що відповідає за класифікацію спаму

Модель може перенавчатися опираючись на дії користувача. Але не усі дії користувача впливають на цю модель. Серед них є такі дії:

* видалення вручну певних коментарів;
* видалення з журналу та системи певних коментарів;
* видалення лише з журналу певних коментарів.

1. Авторизація користувача у системі

Обов`язковим правилом є авторизація кожного користувача через аккаунт Google. Також користувач обов’язково повинен надати системі доступ до усіх його рекламних кабінетів та до Facebook-сторінок, якщо вони існують. Видалення коментарів можливе тільки завдяки цьому. Так як система є персоналізованою під дії кожного авторизованого користувача, то Facebook-акаунт автоматично підтянеться у нашу систему.

**2.2 Модель Facebook-публікацій**

Визначимо у загальному вигляді модель Facebook-публікації з врахуванням взаємозв’язку контенту публікації та ступені корисності коментарів.

Нехай маємо публікацію *P*, що належить певному користувачеві, та розташована на однієї з його сторінок. Тоді маємо:

*P* = <*User*, *Page*, *С, Comm, Info, Status* >

де *User* – акаунт користувача;

*Page* – сторінка, до якої належить публікація, *Page* ϵ *Pages*;

*C* – контент публікації;

*Comm* – набір коментарів до публікації;

*Info* – системна інформація щодо публікації;

*Status* – статус публікації та рівня корисної інформації у її коментарях.

Акаунт користувача визначається як

*User* = < *Pages, Token, UserInfo>,*

*де Pages = <Page, AccessToken>* – набір Facebook-сторінок, доступних користувачеві *User*, *Page* – назва сторінки, *AccessToken* – рівень доступу до сторінки.

Рівні доступу *AccessToken* визначаються згідно з можливими ролями користувача – для кожної ролі є визначений перелік можливий дій. Тоді множина *AccessToken* визначається як:

*AccessToken =* {*Administrator*, *Advertiser*, *Analyst*, *Editor*, *Moderator*},

де *Administrator* – адміністратор (рекламування, аналіз, создание контенту, управління, модерація);

*Advertiser* – рекламодавець (рекламування, аналіз);

*Analyst* – аналітик (аналіз);

*Editor* – редактор (рекламування, аналіз, создание контенту, модерація);

*Moderator* – модератор (рекламування, аналіз, модерація).

*Token* – це свого роду маркер, який існує для того, щоб користувач міг отримати доступ. Як правило, цей рядок є прихованим, його використовують для того, щоб ідентифікувати певного користувача, або якусь сторінку, або прикладну програму, зазвичай його використовують для того, щоб викликати API Graph. При вході певної особи, яка використовує Facebook і надає необхідні дозволи, які запитуються, у програму, то програма отримує певний маркер доступу, який у свою чергу гарантує захищений доступ (але тимчасовий) до якихось різних Facebook API. Також завдяки маркеру доступу програмі забезпечена перевірка на її конфіденційність.

*UserInfo* – персональная інформація щодо користувача: ПІБ, місто, країна, стать, вік і т.д.

Контент публікації може бути описаний через матрицю TF-IDF (Term Frequency – Inverse Document Frequency) – матриця частотністі термінів та зворотної частотності документів:

*C* = < *Links*, *TF*, *IDF* >,

де *Links* – набір елементів множини { *LinkPost*, *LinkUser* }, *LinkPost* – посилання у поточній публікації на іншу публікацію, *LinkUser* – посилання у поточній публікації на людину, що не є автором публікації;

***TF***  — це частотність терміна, яка вимірює, наскільки часто термін (певне слово) зустрічається в документі. Логічно припустити, що в довгих документах термін може зустрітися частіше кількостях, ніж в менш змістових документах, тому абсолютні числа не зможуть пройти. Саме тому краще використати відносні. Для цього та кількість разів, коли термін, який нам необхіден, був помічен у тексті, поділяють на всю кількість слів, яка існує в даному тексті;

*IDF* – це теж частотність документів, але вона зворотня. Вона вимірює безпосередньо важливість терміна. Це робиться, щоб мати можливість змінювати рівень важливості термінів один відносно іншого. *IDF* обчислюється як логарифм від загальної кількості документів, поділеній на кількість документів, в яких зустрічається певний термін.

Набір коментарів визначається як множина *Comm* ={*Commi*}, де кожний коментар

*Commi* = <*ComState*, *ComInfo*, *Com*\_*TF*, *Com*\_*IDF* >,

де *ComState* – статус коментаря, *ComState* = {*Active*, *Deleted\_by\_Filter*, *Deleted\_by\_Hand*}, Active – активний, тобто доданий та доступний для читання, *Deleted\_by\_Filter* – видалений розроблюваним програмним спам-фільтром, *Deleted\_by\_Hand* – видалений користувачем вручну;

*ComInfo* – інформація про автора, дату та час додавання коментаря;

*Com*\_*TF*, *Com*\_*IDF* – відповідні значення для *TF* та *IDF* поточного коментаря відносно усіх *Active*-коментарів публікації *P*.

Системная інформація щодо публікації *Info* включає ступінь доступу до публікації: усім користувачам Facebook, «друзям» автора публікації, обмеженому колу читачів.

Значення *Status* для публікації може визначатись для певної ітерації *j* (моменту часу):

*Statusj* = < *SpamLevel*, *State* >,

де *SpamLevel* – рівень заспамленості публікації, визначається як відношення кількості спамних коментарів до загальної кількості коментарів для публікації *P*:

*SpamLevel* = *spam(Comm)* / *all(Comm)*;

*State* = {*Active*, *Planned*, *Deleted*}, де *Active* – публікація доступна для читання, *Planned* – публікація стане доступною у майбутньому, *Deleted* – публікацію видалено.

**2.3 Визначення принципів роботи програмного класифікатору**

Першим етапом класифікації текстового спама є представлення текста у числовому форматі, як вектор ознак. Є три способи векторизації тексту за якими будується вектор: Count, TF, TF-idf.

Count – підрахунок кількості слів або груп слів у тексті, а результатом є вектор “Слово – кількість”. Але цей спосіб не є дуже гарним для документів різного розміру, бо в великому документі частота вживання одних й тих самих слів буде на порядок більше ніж у маленькому, навіть якщо вони написані на одну й туж тему одним користувачем.

TF - підрахунок частот вживання слів або груп слів у тексті, результатом є вектор “Слово - частота вживання”. Саме частота вживання вирішує проблему великих та маленьких текстів. Частота формується завдяки поділу кількості певних слів на загальну кількість слів в документі. Але знову є проблема - слова які виконують роль зв'язка між іншими словами. Їх може бути дуже багато, та це може повліяти на працездатність класифікатора [8].

TF-IDF - підрахунок частот вживання слів або груп слів у тексті зі зниженням ваги слова, яке з'являється у багатому кількості документів та є менш інформативним. Цей спосіб є оптимальним для аналізу будь-якого тексту.

Другий і останній крок аналізу тексту є вибір способу аналізу векторізованого текста. Для аналізатору тексту на спам дослідним шляхом було обрано класифікацію  One-Vs-The-Rest logistic regression.

One-Vs-The-Rest класифікація позначає, що для кожного класу, який визначено розробником створюється окрема модель класифікації. А logistic regression це метод класифікації. Результатом алгоритму логістичної регресії є 1 або 0. Тобто тільки два класи. Якщо представити логістичну регресію як функцію, то ми отримаємо рис. 2.1.

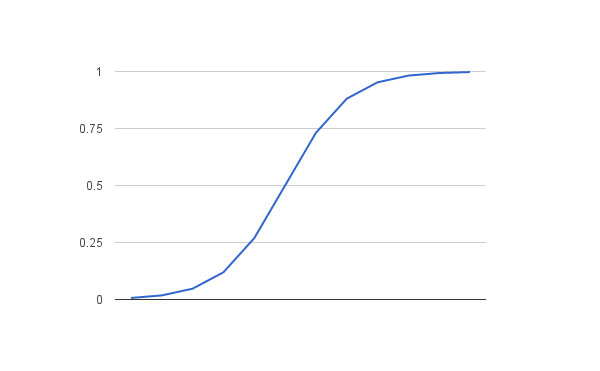


Рисунок 2.1 – Логістична регресія

По осі y йдуть значення від 0 до 1, по x – числові параметри які треба класифікувати. Модель навчається на тренувальних даних та сам вигляд функції змінюється в залежності від тестових даних. На рисунку 2.2 показан приклад розмежування даних.

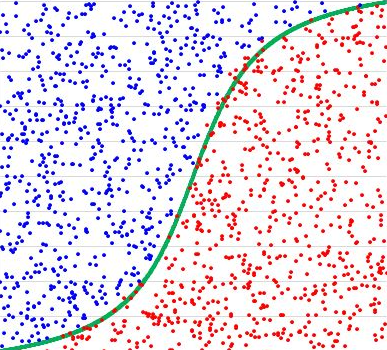


Рисунок 2.2 – Розмежування даних

Аби получити результат у навченої моделі береться точка на осі x, та обчислюється за функцією, за допомогою якої будується графік (рис. 2.3).

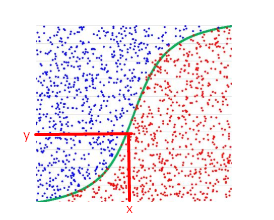


Рисунок 2.3 – Отримання результату

Якщо значення по осі y менше ніж 0.5, то результатом праці моделі є 0, якщо більше, то 1.

**2.4 Верифікація роботи програмного класифікатору**

Як правило, роботи будь-якого спам-фільтру треба веріфіковувати. Для цього використають помилки I і II роду (а саме їх значення): зазвичай це певні ключові поняття, які частіше за все використовують у математичній статистиці та аналізі даних. В аналізі даних ці значення помилок I і II роду взагалі використовують для того, щоб можна було оцінювати точність певних моделей (а саме – моделей бінарної класифікації). Помилки I і II роду використовують і у математичній статистиці, як правило – при перевірці статистичних гіпотез.

Кожний класифікатор може зробити помилку, це нормально. Помилок може бути навіть дві (у нашому випадку):

* можливе виникнення так званої «помилкової тривоги», коли коментар, який насправді не є спамом, розпізнається системою як спам;
* можлива ймовірність виникнення «пропуску цілі», це розпізнавання спаму як самий звичайний коментар.

Як правило, «пропуск цілі» є більш серйозною помилкою, ніж «помилкова тривога». У нас буде два результати класифікації: позитивним та негативний. Позитивний, коли коментар є спамом, а негативний – коли коментар звичайного змісту. Є декілька можливих результатів класифікації:

* спам був розпізнан як спам (як позитивний) і отримав назву істинно-позитивний (TP або true positive)
* коментар, який не є спамом, був вірно розпізнан(як негативний), тоді він отримує назву істинно негативний (TN або true negative);
* коментар, який не є спамом розпізнан як спам, тобто допущена помилка і негативна класифікація помилково розпізнена як позитивна. Такий результат має назву помилково-позитивна класифікація (FP або false positive). Помилка у такій ситуації являється помилкою I-го роду;
* коментар, який є спамом, розпізнано як коментар без спаму, тобто допущена помилка у розпізнаванні і позитивна класифікація помилково розпізнена як негативна. Такий результат має назву помилково-негативна класифікація (FN або false negative). Помилка у такій ситуації вже являється помилкою II-го роду.

Тобто, якщо негативні дії розпізнано як позитивні, то це помилка першого роду. А якщо позитивні дії розпізнано як негативні, то це вже помилка другого роду.

При завершенні класифікації йде підрахунок помилок першого і другого роду, а також чисельних правильних класифікацій коментарів. Після цього, зазвичай, треба обчислити специфічність та чутливість кожного з класифікаторів. На основі цих обчислень за допомогою ROC-кривої можна буде оцінити точність цієї бінарної моделі. Рівень значущості в статистиці – це ймовірність помилки першого роду. Площа, яка розташована під цією ROC-кривою має назву Area Under Curve (або AUC). Вона не залежить від будь-якого співвідношення всіх помилок та є так званою агрегированою характеристикою (тобто чим вище значення Area Under Curve, тим «якісніше» ця модель класифікації)

1. **СПЕЦИФІКАЦІЯ ВИМОГ ДО** **ПРОГРАМНОЇ СИСТЕМИ**

**3.1 Вимоги до функціональних характеристик**

Як результат роботи повинна бути виконана програмна реалізація програмного фільтру, що аналізує та видаляє спам у коментарях публікацій мережі Facebook. Видалення може виконуватись у автоматичному та ручному режимах.

Програмний фільтр є вебзастосуванням, призначеним для підвищення якості роботи користувача мережі Facebook.

На рисунку 3.1. можна побачити діаграму варіантів використання [9, 10].

Діючою особою виступає “Користувач”, який може виконувати в системі наступні дії:

1. Авторизація в системі.

а) Вибір аккаунта “Google” для роботи з мережею Facebook.

б) Надання дозволу на редагування контенту.

в) Підтвердження дозволу.

1. Перегляд списку публікацій з можливістю перегляду детальної інформації щодо окремій публікації
2. Перегляд списку коментарів.
3. Видалення коментарів вручну: включає перенавчання моделі для класифікації спаму
4. Запуск класифікації спаму.

6. Перегляд журналу.

а) Перегляд загальної статистики щодо спама на Facebook-сторінці.

б) Перегляд видалених коментарів.

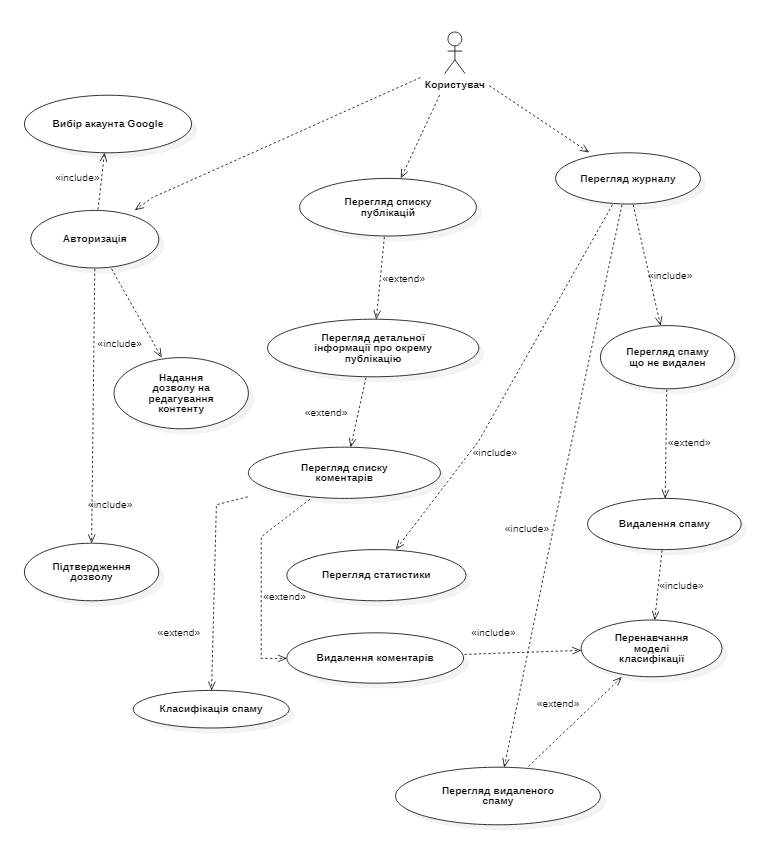


Рисунок 3.1 – Діаграма варіантів використання

Сценарій використання для прецеденту №1

Назва прецеденту: Авторизація в системі

Основна діюча особа: Користувач

Учасники та інтереси:

Користувач – запускає Систему, бажає увійти до свого аккаунта.

Передумова: Користувач запустив Систему

Мінімальні гарантії: Користувач увійшов у Систему

Гарантія успіху: Вхід у Систему був виконан менш ніж за 10 секунд.

Тригер: була натиснута кнопка “Розпочати авторизацію”

Основний сценарій:

а) Користувач натискає кнопку “Розпочати авторизацію”, Система відчиняє вікно браузера.

б) Система відображає список доступних аккаунтів, Користувач обирає один із списку.

в) Система запрошує дозвіл на редагування контенту на сторінці Facebook, Користувач надає дозвіл.

г) Система запрошує підтвердити, що Користувач повністю довіряє продукту, Користувач підтверджує.

д) Система підключається до аккаунту Користувач.

Розширення:

б1) Система пропонує створити новий аккаунт, якщо Користувач не має свого, Користувач створює аккаунт.

б2) Система відображає список доступних аккаунтів, Користувач відмовляється обирати аккаунт, Система відображає вікно з помилкою.

в1) Система запрошує дозвіл на редагування контенту на сторінці Facebook, Користувач не надає дозвіл, Система відображає вікно з помилкою.

г1) Система запрошує підтвердити, що Користувач повністю довіряє продукту, Користувач не підтверджує, Система відображає вікно з помилкою.

д1) Проблема з інтернет з'єднанням, Система відображає вікно з помилкою.

Сценарій використання для прецеденту №2

Назва прецеденту: Перегляд списку публікацій.

Основна діюча особа: Користувач

Учасники та інтереси:

Користувач – бажає подивитися публікаії, що містяться на сторінці.

Передумова: Користувач виконав авторізацію.

Мінімальні гарантії: Система відобразила список публікацій.

Гарантія успіху: Список публікацій був відображений не більш ніж за 3 секунди.

Тригер: була натиснута кнопка “Список публікацій”.

Основний сценарій:

 а) Користувач натискає кнопку “Список публікацій”, Система відчиняє сторінку зі списком.

б) Система відображає інформацію о завантаженні списку, Система посилає запит до сторінки Facebook щодо отримання списку публікацій.

в) Система отримала список публікацій, Система закриває інформацію о завантажені списку.

г) Система відображає список публікацій, Користувач бачить список.

д) Користувач натискає на одну публікацію, Система відображає детальну інформацію щодо обраної публікації.

Розширення:

б1) Система відображає інформацію о завантажені списку, Система помічає відсутність інтернет з'єднання, Система відображає вікно з помилкою “Інтернет з'єднання відсутнє”.

в1) Система не отримала список публікацій протягом 20 секунд, Система відображає вікно з помилкою “Запит триває більш ніж 20 секунд. Спробуйте пізніше”.

г1) Система отримала 0 публікацій, Система відображає інформацію “На сторінку не було завантажено жодній публікації”.

Сценарій використання для прецеденту №3

Назва прецеденту: Перегляд списку коментарів.

Основна діюча особа: Користувач

Учасники та інтереси:

Користувач – бажає подивитися коментарі до публікації.

Передумова: Користувач відчинив детальну інформацію щодо будь-якої публікації.

Мінімальні гарантії: Система відобразила список коментарів.

Гарантія успіху: Список коментарів був відображений не більш ніж за 10 секунд.

Тригер: була натиснута кнопка “Список коментарів”.

Основний сценарій:

а) Користувач натискає кнопку “Список коментарів”, Система відчиняє сторінку зі списком.

б) Система відображає інформацію о завантажені списку, Система посилає запит до Facebook-сторінці щодо отримання списку коментарів.

в) Система отримала список коментарів, Система закриває інформацію о завантажені списку.

г) Система відображає список коментарів, Користувач бачить список.

Розширення:

б1) Система відображає інформацію о завантажені списку, Система помічає відсутність інтернет з'єднання, Система відображає вікно з помилкою “Інтернет з'єднання відсутнє”.

в1) Система не отримала список публікацій протягом 20 секунд, Система відображає вікно з помилкою “Запит триває більш ніж 20 секунд. Спробуйте пізніше”.

г1) Система отримала 0 коментарів, Система відображає інформацію “Під публікацією ще не було написано жодного коментарю”.

Сценарій використання для прецеденту №4

Назва прецеденту: Видалення коментарів вручну.

Основна діюча особа: Користувач

Учасники та інтереси:

Користувач – бажає видалити деякі коментарі.

Передумова: Користувач відчинив список коментарів до будь-якій публікації.

Мінімальні гарантії: Коментарі були видалені з публікації.

Гарантія успіху: Процесс видалення тривав не більш ніж 2 секунди.

Тригер: була натиснута кнопка “Видалити коментарі”.

Основний сценарій:

а) Користувач натискає кнопку “Видалити коментарі”, Система відчиняє вікно “Чи дійсно ви бажаєте видалити N коментарів?”.

б) Користувач натискає “Так”, Система починає процес видалення.

в) Система тренує модель для класифікації на видалених коментарях.

г) Система отримала сигнал про завершення видалення, Система інформує Користувача щодо вдалого видалення.

Розширення:

а1) Користувач натискає кнопку “Видалити коментарі”, Система дізнається що було обрано 0 коментарів, Система блокує натискання кнопки.

б1) Користувач натискає “Ні”, Система закриває вікно.

г1) Система отримала сигнал про невдале завершення видалення, Система відображає вікно “Один або декілька коментарів не було видалено у зв'язку з непередбаченою помилкою”.

Сценарій використання для прецеденту №5

Назва прецеденту: Запуск класифікації спаму.

Основна діюча особа: Користувач

Учасники та інтереси:

Користувач – бажає видалити спам з коментарів публікації.

Передумова: Користувач відчинив список коментарів до будь-якій публікації, або детальну інформацію про публікацію.

Мінімальні гарантії: більш ніж 50% спаму було видалено

Гарантія успіху: більш ніж 80% спаму було видалено.

Тригер: була натиснута кнопка “Розпочати видалення спаму”.

Основний сценарій:

а) Користувач натискає кнопку “Розпочати видалення спаму”, Система відчиняє вікно “Процес видалення може тривати до 5 хвилин, чи дійсно ви бажаєте розпочати видалення спаму?”.

б) Користувач натискає “Так”, Система починає процес класифікації спаму.

в) Система отримала сигнал о завершенні класифікації, Система відображає список коментарів, які можуть бути видалені.

г) Користувач обирає коментарі для видалення, Система виконує процес видалення.

Розширення:

а1) Користувач натискає кнопку “Розпочати видалення спаму”, Система дізнається що до публікації немає жодного коментаря, Система блокує натискання кнопки.

в1) Система отримала сигнал о завершенні класифікації, усі коментарі класифіковані як “не спам”, Система відображає вікно “Спам не було знайдено”.

Сценарій використання для прецеденту №6

Назва прецеденту: Перегляд журналу.

Основна діюча особа: Користувач

Учасники та інтереси:

Користувач – бажає переглянути статистику або видалені коментарі.

Передумова: Користувач виконав авторізацію.

Мінімальні гарантії: Журнал завантажився.

Гарантія успіху: Журнал завантажився менш ніж за 10 секунд.

Тригер: була натиснута кнопка “Журнал”.

Основний сценарій:

а) Користувач натискає кнопку “Журнал”, Система відчиняє сторінку журналу.

б) Система пропонує клієнту переглянути загальну статистику, або сховище видалених коментарів.

в) Користувач обирає “Загальна статистика”, Система відчиняє сторінку статистики.

г) Користувач обирає “Сховище видалених коментарів”, Система відчиняє сторінку з видаленими коментарями.

Розширення:

б1) Система пропонує клієнту переглянути загальну статистику, Система не бачить видалених коментарів, Система робить кнопку “Сховище видалених коментарів” не доступною.

## 3.2 Нефункціональні вимоги до програмного засобу

Якісні вимоги:

а) Надійність

Якщо користувач видаляє коментар, то система відображає повідомлення про результат не більш ніж через 2 секунди.

Якщо користувач запрошує список публікацій, то система відображає повідомлення про результат не більш ніж через 3 секунди.

Якщо користувач запрошує список коментарів, то система відображає повідомлення про результат не більш ніж через 5 секунд.

Якщо користувач почав виконання класифікації спаму, то система відображає результат не більш ніж через 5 хвилин.

Якщо користувач запрошує перегляд журналу, то система відображає інформацію не більш ніж за 2 секунди.

б) Ефективність

Якщо система виконує запит на видалення коментаря, то сервер повертає результат не більш ніж за 1,5 секунди.

Якщо система виконує запит на отримання списку публікацій, то сервер повертає результат не більш ніж за 2,5 секунди.

Якщо система виконує запит на отримання списку коментарів, то сервер повертає результат не більш ніж за 4,5 секунди.

Якщо система виконує запит на отримання списку спаму у коментарях, то класифікатор повертає результат не більш ніж за 4 хвилини, 50 секунд.

Якщо система виконує запит на отримання списку спаму у коментарях, то класифікатор обробляє не менш ніж 5 коментарів у секунду.

в) Функціональність

Якщо система виконує запит на видалення коментаря, то сервер повертає результат у цілісності не менш ніж у 95% випадків.

Якщо система виконує запит на отримання списку публікацій, то ймовірність повернення списку публікацій у правильному порядку є 0,9.

Якщо система виконує запит на отримання списку коментарів, то сервер повертає результат у цілісності не менш ніж у 90% випадків.

Якщо система виконує запит на отримання списку спаму у коментарях, то результат класифікації є невизначеним не більш ніж в одному випадку із 100.

г) Зручність використання

Якщо користувач виконує авторизацію, то йому це вдається не більш ніж за 1 хвилину.

Якщо користувач бажає видалити окремий коментар, то йому це вдається не більш ніж за 40 секунд.

Якщо користувач бажає розпочати класифікацію спаму, то йому це вдається не більш ніж за 15 секунд.

Якщо користувач хоче переглянути розширену інформацію про окрему публікацію, то йому це вдається не більш ніж за 20 секунд.

д) Супроводжуваність

Якщо треба запровадити новий функціонал, то система почне працювати з ним не більш ніж через 1 місяць.

Якщо треба видалити частину функціоналу, то система почне працювати без нього не більш ніж через 2 тижня.

Якщо треба змінити частину функціоналу, то система почне працювати з ним не більш ніж через 3 тижня.

е) Портативність

Якщо систему треба буде перенести на іншу операційну систему (Linux, Android, IOS. MacOS), то процес переносу триватиме не більш ніж 3 місяці.

Якщо систему треба перенести на інший пристрій з такою ж самою операційною системою, то процес переносу триватиме не більш ніж 1 годину.

## 3.3 Технічні вимоги

а) Мінімальні вимоги до апаратного середовища

Операційна система: Windows 10.

Кількість оперативної пам'яті: 4гб.

Процесор: Intel Core i3, AMD Ryzen 3.

б) Задовольні вимоги до апаратного середовища

Операційна система: Windows 10

Кількість оперативної пам'яті: 8гб.

Процесор: Intel Core i5, AMD Ryzen 5.

1. **ПРОЕКТУВАННЯ ПРОГРАМНОГО АНАЛІЗАТОРУ**

**ДЛЯ ФІЛЬТРАЦІЇ СПАМУ**

* 1. **Проектування архітектури аналізатору**

На рис. 4.1 наведено архітектуру проекту у загальному вигляді.

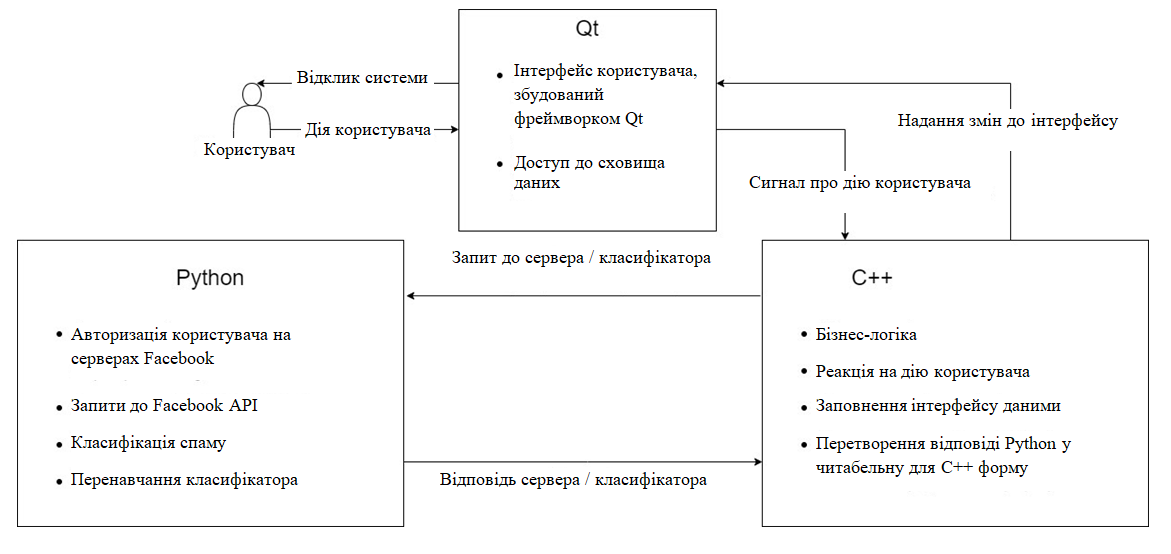


Рисунок 4.1 – Узагальнена архітектура розроблюваної системи

В описаній вище архітектурі використовується три модулі. Далі буде приведена розширена інформація про кожний з них.

1. Модуль Qt

Цей модуль є зв'язком між користувачем, та бізнес логікою інформаційної системи. Саме за допомогою Qt будується інтерфейс користувача, та обробляються всі натиски та інші дії [11].

Також фреймворк Qt допомагає у будуванні бази даних, та обробляє усі запроси до неї.

1. Модуль C++

Цей модуль виконує усю бізнес-логіку системи. Він реагує на сигнали від Qt, та обробляє усі дії користувача. Як результат – заповнює інтерфейс користувача інформацією.

Також цей модуль виступає у ролі проксі між Qt та C++, бо він передає бажання користувача, які пов'язані з доступом до авторизації, Facebook та класифікації до наступного модуля – Python.

1. Модуль Python

Цей модуль є кінцевим модулем у системі. Він виконує функціональність двох типів – серверні функції, функції класифікації. А саме авторизацію користувача, запроси до Facebook API Pages, класифікацію спаму, перенавчання класифікатора.

## 4.2 Опис програмних модулів

Розроблювана система складається з модулів двох типів: Python, C++.

Далі представлено список Python модулей:

Клас SpamClassifier

Це допоміжний модуль, який займається класифікацією спама, та перенавчанням класифікатора [12].

Основними методами цього модулю є:

initialize\_classifier – ініціалізація об'єкту класифікатора за допомогою  раніше збережених файлів;

make\_prediction – класифікація спаму на основі взятого тексту;

retrain\_classifier – додає до навчальної вибірки групу коментарів, та виконує процес перенавчання моделі класифікації.

Клас FacebookHelper

Це допоміжний модуль, який виконує операції, пов'язані з доступом до платформ Facebook, а також виконує авторизацію користувача.

Основними методами цього модулю є:

start\_connection – стартує з'єднання із серверами Google та Facebook, використовуючи ключ розробника;

get\_page\_posts – повертає список публікацій до окремій Facebook сторінки;

get\_posts\_info – обробляє список публікацій, та повертає його, відокремлюючи зайву інформацію;

get\_all\_comments – повертає список коментарів до окремій публікації;

get\_comments\_info – обробляє список коментарів, та повертає його, відокремлюючи зайву інформацію;

get\_my\_page\_id – повертає ID Facebook сторінки вторизованого користувача;

delete\_comment – видаляє коментар.

Клас Handler

Це модуль який включає в себе минулі два, та реалізує методи, які потрібні C++ частині.

Основними методами цього модулю є:

login – авторизація користувача;

get\_my\_page\_id – повторює функціонал модуля FacebookHelper;

get\_post\_list – повертає читабельний список публікації;

get\_comments\_list – повертає читабельний список коментарів;

start\_comments\_classification – стартує процес класифікації коментарів під окремою публікацією. Повертає список коментарів для видалення;

delete\_comments\_group – видаляє группу коментарів, використовуючи їх ID.

Тепер перейдемо до опису модулів С++. Зв'язок  між деякими С++ модулями представлений у вигляді діаграми класів на рисунку 4.2. Далі наведемо опис та призначення кожного класу окремо.

Клас CPyInstance

Цей клас має два призначення:

* ініціалізація інтерпретатора Python у середовищі C++;
* скасування роботи інтерпретатора Python у середовищі C++.

Клас CPyObject

Цей клас виконує роль обгортки для вбудованого класу PyObject, усі методи та оператори, які є публічними у класі PyObject, переписані у класі CpyObject.

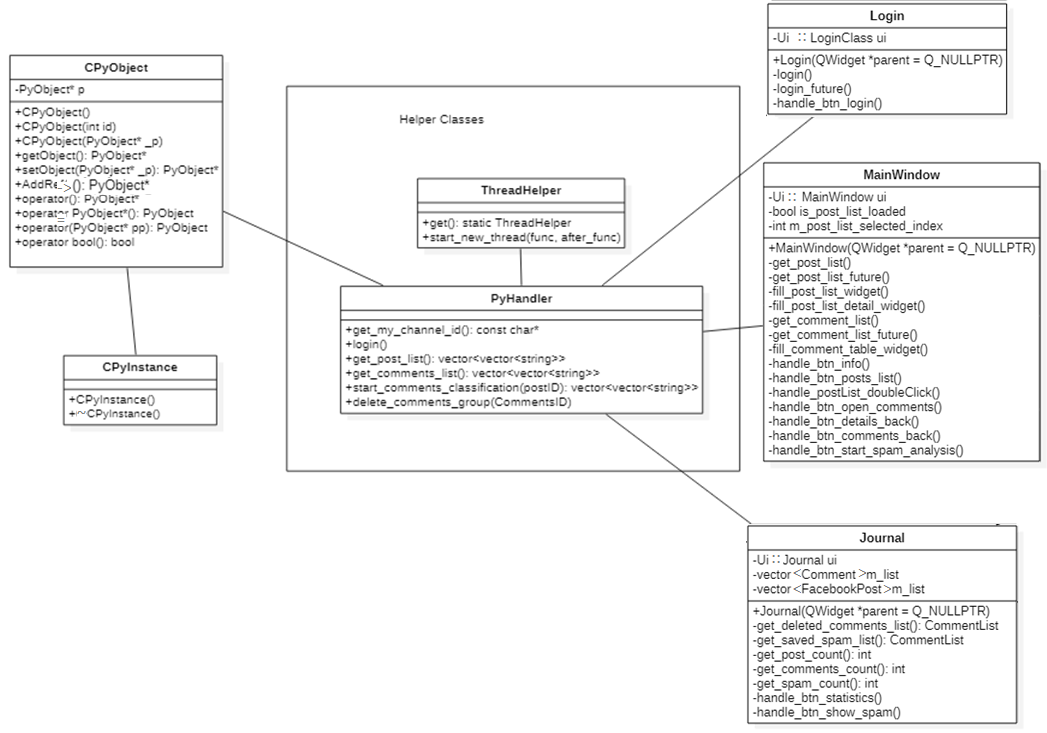


Рисунок 4.2 – Діаграма класів модулів С++. Зв’язок з Python

Така обгортка потрібна для того, щоб цей клас дотримувався принципам RAII (Resource Acquisition Is Initialization) – захват ресурсу є ініціалізація. Тобто, ресурсом у нашому випадку є покажчик на об'єкт типу PyObject. Обгортка захоплює цей ресурс, виконуючи ініціалізацію всіх його полей, та переписуючи усі його публічні методи. Після цього клас CPyObject може використовуватися всюди, де використовується ресурс. Але тепер не потрібно дбати про очищення пам'яті. CpyObject автоматично очищує пам’ять, на яку посилається ресурс.

CPyObject  використовує клас CPyInstance для ініціалізації та скасування роботи Python.

Клас PyHandler

Цей клас використовується для налагодження зв'язку з Python-частиною розробки.

Всі методи класу дають команду вбудованому інтерпретатору Python виконати тот або інший метод, та перетворюють повернений результат, якщо він є, у тип, який може бути прочитаний C++.

Клас ThreadHelper

Цей клас має завдання допомогти запустити паралельно декілька важливих операцій всередині програми.

Тепер розглянемо частину діаграми класів, що безпосередньо пов’язана з обробкою коментарів публікацій (рис. 4.3).

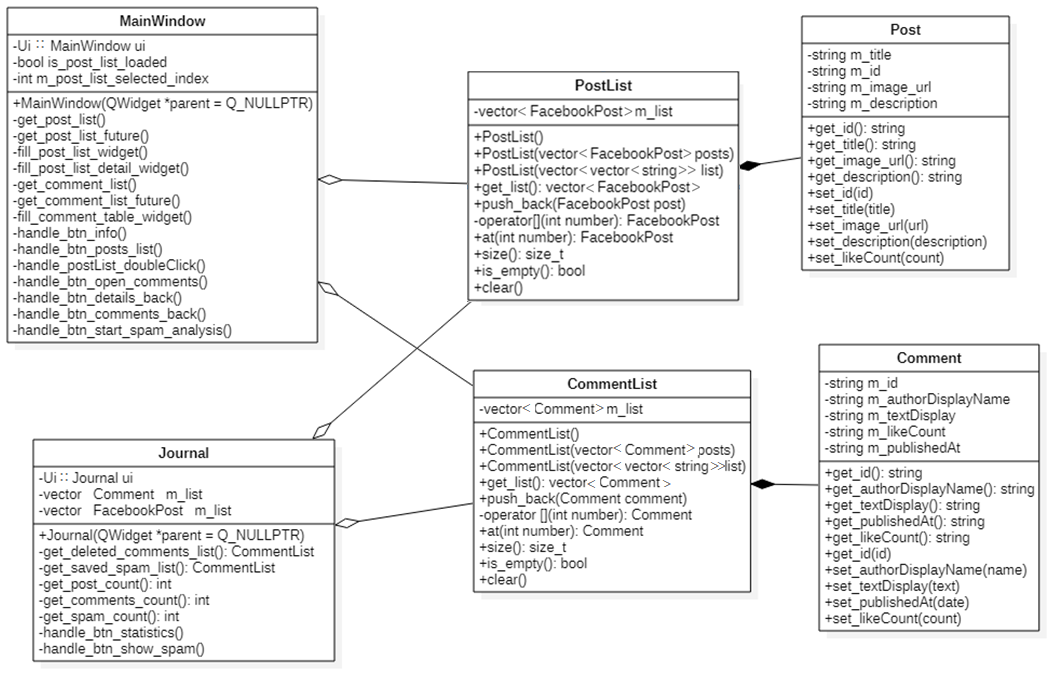


Рисунок 4.3 – Діаграма класів модулів С++. Обробка коментарів у публікаціях Facebook

Клас Comment

Цей клас виступає сховищем для об'єкту типу «коментар». Він  надає читабельний доступ до атрибутів коментарю, що полегшує читабельність коду.

Клас CommentList

Цей класс являє собою колекцію об'єктів типу Comment.  Він використовується для зручного зберігання таких об'єктів. Одна з причин  створення окремого классу – трансформація інформації, яку надає класс PyHandler о коментарях у колекцію типу Comment всередині самого классу-колекції.

Клас Post

Цей клас виступає сховищем для об'єкту типу post. Він  надає читабельний доступ до атрибутів публікації, що полегшує читабельність коду.

Клас PostList

Цей клас являє собою колекцію об'єктів типу Post.  Він використовується для зручного зберігання таких об'єктів. Одна з причин  створення окремого класу – трансформація інформації, яку надає клас PyHandler о коментарях у колекцію типу Post всередині самого класу-колекції.

Клас LogIn

Цей клас використовується для опису бізнес-логіки авторизації користувача, та спілкування з інтерфейсом авторизації. Поле Ui::LogInClass ui зберігає в собі весь інтерфейс авторизації користувача, цей клас був згенерований фреймворком Qt.

Опис методів:

login() – використовується для виклику метода login() класу PyHandler;

login\_future() – використовується для запису FB-сторінки до бази даних, та переходу до наступної сторінки, тобто для дій, які будуть виконані після завершення методу login();

handle\_btn\_login() – використовується для обробки натискання на кнопку авторизації. Обробка виконується за допомогою класу ThreadHelper паралельно, аби інтерфейс був клікабельний.

Клас MainWindow

Цей клас використовується для опису бізнес-логіки основних дій користувача у розроблюваній системі, а саме: відображення списку публікацій, відображення деталей публікацій, відображення списку коментарів до публікації, видалення окремого коментаря, виконання класифікації коментарів.

Поле Ui::MainWindowClass ui зберігає в собі весь інтерфейс описаних вище функцій системи, цей клас був згенерований фреймворком Qt.

Опис методів:

get\_post\_list() – використовується для виклику метода get\_post\_list() класу  PyHandler;

get\_post\_list\_future() – використовується для виклику fill\_post\_list\_widget(), запису списку публікацій до бази даних, та інших дій, які виконуються після отримання списку публікацій;

fill\_ post\_list\_widget() – заповнює список публікацій, передає його до інтерфейсу користувача;

fill\_ post\_list\_detail\_widget() – заповнює сторінку з деталями до окремої публікації;

get\_comment\_list() – використовується для виклику метода get\_comments\_list() класу  PyHandler;

get\_comment\_list\_future() – використовується для дій, які будуть виконані після отримання списку коментарів;

fill\_comment\_table\_widget() – заповнює таблицю з коментарями;

delete\_comment() – використовується для виклику метода delete\_comments\_group() класу  PyHandler;

delete\_comment\_future() – дії, які будуть виконані після видалення коментарю або коментарів вручну, а саме запис видалених коментарів у базу даних;

start\_spam\_analysis() – використовується для виклику метода start\_comments\_classification() класу  PyHandler;

start\_spam\_analysis\_future() – виклик дій, які будуть виконані після закінчення процесу класифікації, а саме виклик методу delete\_comment() для подальшого видалення одного або декількох коментарів, що є спамом, та їх подальше додавання до бази даних з метою відображення у журналі.

Усі методи з префіксом handle() виконують дії, пов'язані за натиском на кнопки, тобто обробляють ввод користувача. Всі вони використовують об'єкт типу ThreadHelper, з метою паралельного запуску декількох операцій одночасно.

Також цей клас використовує об'єкти типів PostList та CommentList для збереження стану списків публікацій та коментарів.

Клас Journal

Цей клас використовується для відображення журналу спаму, та загальної статистики.

Поле Ui::JournalClass ui зберігає в собі весь інтерфейс журналу спаму та статистики, цей клас був згенерований фреймворком Qt.

Опис методів:

get\_deleted\_comments\_list() – використовується для завантаження списку видалених коментарів з бази даних, та їх подальшого відображення у журналі;

get\_saved\_spam\_list() – використовується для завантаження списку коментарів, які були помічені як спам, але не були видалені з системи;

get\_post\_count() – повернення розміру списку публікацій.

get\_comments\_count() – повернення розміру списку всіх коментарів;

get\_spam\_count() – повернення розміру списку всіх коментарів, що були помічені як спам;

get\_deleted\_spam\_count() – повернення розміру списку всіх коментарів, що були помічені як спам, та були видалені з системи;

delete\_comment\_from\_journal() – видалення коментарів з журналу, тобто з бази даних;

delete\_comment\_from\_journal\_and\_system() – видалення коментарів з журналу, та також з  публікації на певній сторінці, використовується тільки на коментарях, що були помічені як спам, але не були видалені з системи.

Усі методи з префіксом handle() виконують дії, пов'язані за натиском на кнопки, тобто обробляють ввод  користувача. Всі вони використовують об'єкт типу ThreadHelper, з метою паралельного запуску декількох операцій одночасно.

## 4.3 Опис функціональної поведінки системи

Функціональна поведінка системи описана за допомогою діаграм взаємодії системи, які побудовані на основі варіантів використання.

На рис. 4.4 наведено діаграму взаємодії для процесу авторизації у проектованій системі. Середовище Qt посилає сигнал до модуля С++, який в свою чергу встановлює видимість анімації завантажування та передає сигнал до Python. Як результат, Python передає до С++ об'єкт, за допомогою якого здійснюється з'єднання з сервером Google та передача інформації.

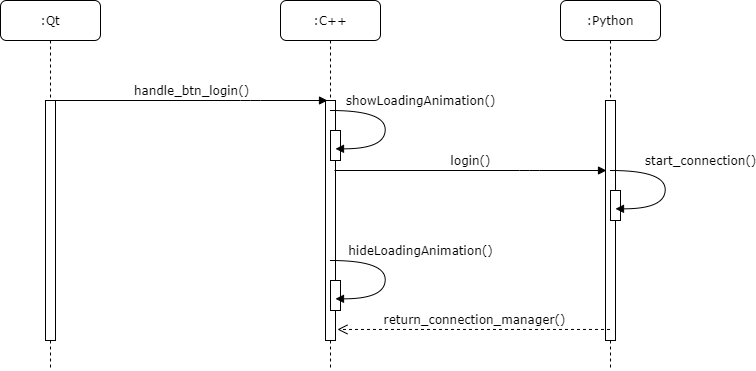


Рисунок 4.4 – Діаграма взаємодії для авторизації

На рис. 4.5 наведено діаграму взаємодії для процесу отримання списку публікацій.

Середовище Qt посилає сигнал до модуля С++, який в свою чергу встановлює видимість анімації завантажування та передає сигнал до Python. Python запрошує ідентифікатор Facebook-сторінки, та, використовуючи його, запрошує список публікацій цієї сторінки. Як результат Python передає до С++ список публікацій. С++-частина завершує анімацію завантаження та заповнює список даними публікацій.

На рис. 4.6 наведено діаграму взаємодії для процесу отримання списку коментарів.

Середовище Qt посилає сигнал handle\_btn\_posts\_list() до модуля С++, який передає сигнал get\_comment\_list() до Python. Python запрошує список коментарів до публікації get\_comments\_from\_post(). Як результат, Python передає до С++ список коментарів comment\_list. С++-частина завершує анімацію завантаження та заповнює список коментарів даними fill\_comment\_list\_widget().

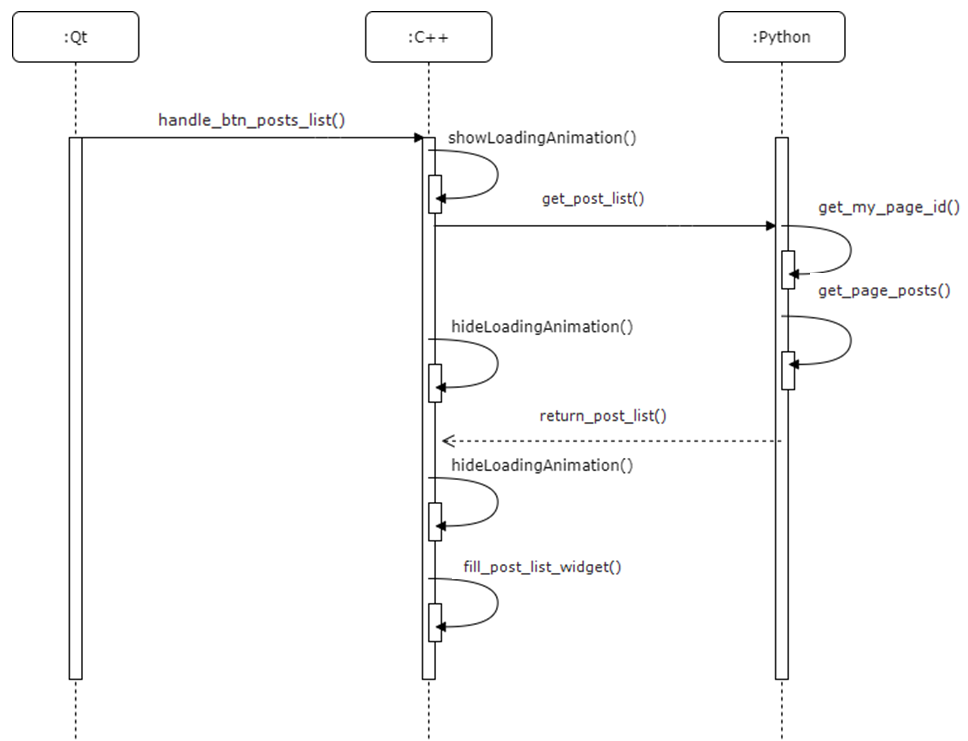


Рисунок 4.5 – Діаграма взаємодії для отримання списку публікацій

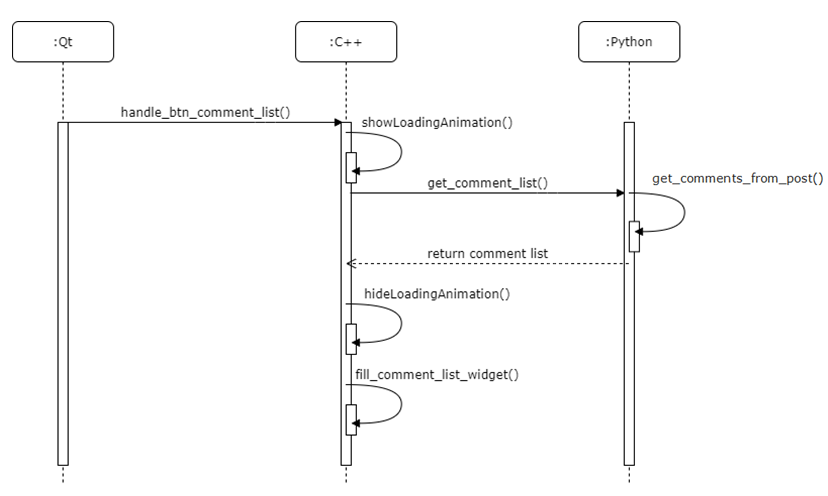


Рисунок 4.6 – Діаграма взаємодії для отримання списку коментарів

Діаграма взаємодії для процедури видалення спам-коментаря наведена на рис. 4.7.

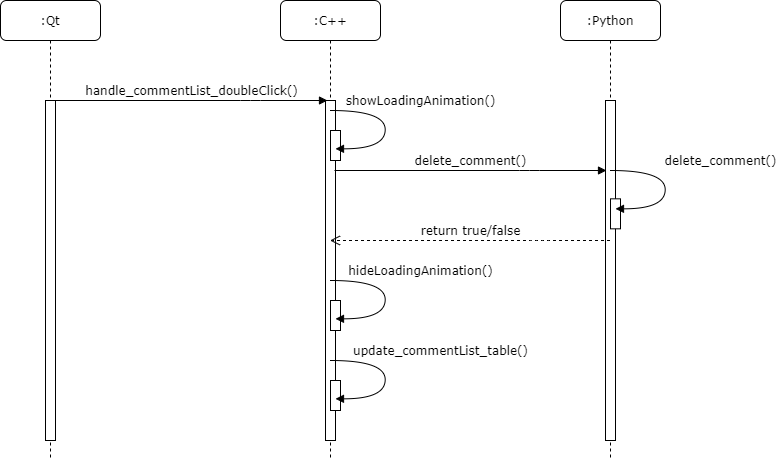


Рисунок 4.7 – Діаграма взаємодії видалення спам-коментаря

Середовище Qt посилає сигнал handle\_commentList\_doubleClick() до модуля С++. Цей модуль передає запит delete\_comment(), який оброблюється модулем Python. Він повертає результат true у випадку успішного видалення коментаря, чи false у разі помилки. Після цього модуль С++ оновлює таблицю коментарів, у разі успішного видалення коментаря її розмір змінюється, тобто зменшується на 1 рядок. Зауважимо, що обсяг таблиці повинен бути достатнім для збереження усіх коментарів.

На рис. 4.8 наведена діаграма взаємодії для процедури класифікації спаму. Середовище Qt посилає сигнал handle\_btn\_start\_spam\_analysis() до модуля С++. Цей модуль передає запит start\_spam\_analysis() до модулю Python. Модуль Python виконує ініціалізацію класифікатору за допомогою методу initialize\_classifier() та виконує класифікацію за допомогою методу make\_prediction(). Він повертає результат у вигляді списку спам-коментарів spam\_list. Завдяки цього списку модуль С++ оновлює список коментарів update\_commentList\_table(), до кожного коментаря додається мітка: спам/неспам.

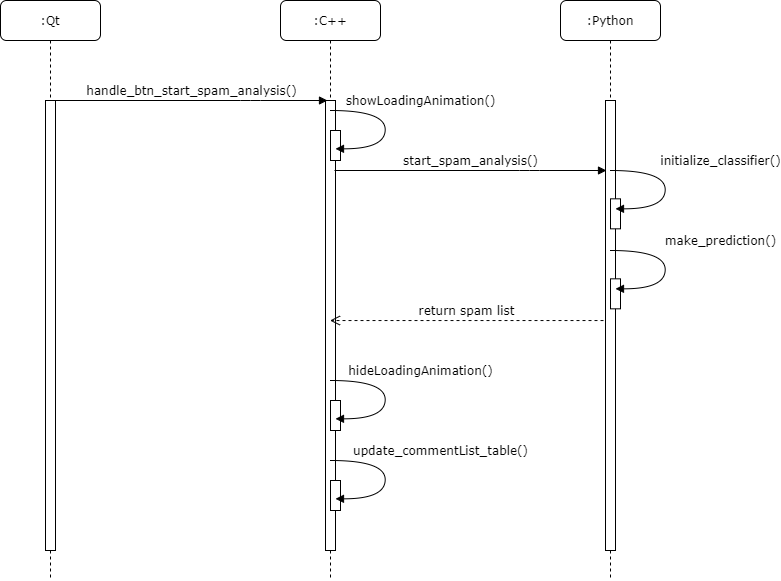


Рисунок 4.8 – Діаграма взаємодії для класифікації спаму

## 4.4 Патерни проектування у системі

Розглянемо патерни проектування та відносні до них класи, що були використані у системі.

Proxy

«Призначенням цього патерну є підстановка спеціальних об'єктів-замінників замість реальних об'єктів. Ці об'єкти-замінники перехоплюють виклики системи до первинного об'єкту та дозволяючи виконати певні дії або до, або після передачі виклику оригінальному об'єкту.» [11]

Класи, які застосовують даний патерн, це: LogIn, MainWindow, Journal. Усі вони є проксі між класом інтерфейсу користувача, та класом PyHandler, адже саме PyHandler виконує виклик вбудованого інтерпретатора Python.

Singleton

Цей патерн гарантує, що у класі є тільки один екземпляр, і надає до нього глобальну точку доступу.

Клас, який застосовує даний патерн є CPyInstance. Об'єкт цього класу має ініціалізуватися та  руйнуватися тільки по одному разу за весь життєвий цикл інформаційної системи. Це потрібно для того, щоб інтерпретатор Python ініціалізувався та згортався тільки один раз, тобто немає сенсу робити цей процес постійно з кожним викликом функції.

## 4.5 Сховище даних

Для збереження даних був використаний фреймворк Qt з вбудованим сховищем типу MySQL. Звертаючи увагу до потреб програмної системи, була спроектована концептуальна модель бази даних (рис. 4.9) [13].

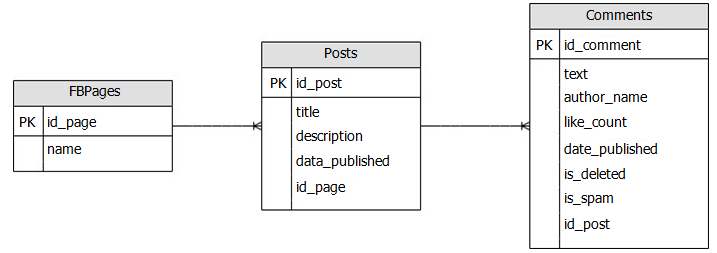


Рисунок 4.9 – Концептуальна модель бази даних

Основні сутності концептуальної моделі та їх опис:

Сутність FBPages

У цій таблиці зберігаються Facebook-сторінки, які були задіяні користувачем у процесі роботи з розроблюваною програмною системою.

У кожного запису є такі атрибути:

* id\_page – ідентифікатор сторінки у форматі VARCHAR(48);
* name – назва сторінки у форматі VARCHAR(48).

Сутність Posts

У цій таблиці зберігаються публікації, які були завантажені у систему під час роботи з нею.

У кожного запису є такі атрибути:

* id\_post – ідентифікатор публікації у форматі VARCHAR(48);
* title – заголовок публікації у форматі VARCHAR(48);
* description – опис публікації у форматі VARCHAR(560);
* date\_published – дата створення публікації у форматі DATETIME;
* id\_page – ідентифікатор сторінки, до якої ця публікація відноситься – у форматі VARCHAR(48).

Сутність Comments

У цій таблиці зберігаються коментарі, які були коли-небудь завантажені в інформаційну систему.

У кожного запису є такі атрибути:

* id\_comment – ідентифікатор коментарю у форматі VARCHAR(48);
* text – текстовий зміст коментарю у форматі VARCHAR(310);
* author\_name – ім'я автору VARCHAR(48);
* like\_count – кількість лайків під публікацією INTEGER;
* date\_published – дата публікації коментарю DATETIME;
* is\_deleted – відповідає за видалення коментарю з FB-сторінки (0 – не видалений, 1 – видалений) BOOLEAN;
* is\_spam – відповідає за маркировку коментарю, як спаму (0 – не спам, 1 – спам) BOOLEAN;
* id\_post – ідентифікатор публікації, до якої відноситься даний коментар, VARCHAR(48).

1. **ПРОГРАМНА РЕАЛІЗАЦІЯ РОЗРОБЛЮВАНОЇ СИСТЕМИ**

**5.1 Особливості створення програмних модулів з урахуванням мови програмування**

Розроблювана система містить модулі на двох мовах програмування – С++ та Python. Розглянемо використання цих мов більш детально.

Мова С++ має складний синтаксис, що забезпечує велику гнучкість при його використанні у різноманітних програмних проектах. Програми, створені на С++, мають дуже велику швидкість виконання, що є важливим аспектом при виборі мові програмування. C++ за допомогою фреймворку Qt став основною мовою для створення інтерфейсу користувача. Також на стороні C++ реалізован доступ до бази даних.

Python є мовою загального призначення, легкою у застосуванні. Python має набір бібліотек з машинного навчання, що дозволяє використовувати функції цих бібліотек для побудови класифікаторів спаму. На стороні Python був реалізований доступ до функціоналу, який пов'язаний з Facebook-системою.

Python-реалізація використовує наступні технології:

*Pandas*

Бібліотека, яка допомагає видозмінювати набори даних, вирізаючи неважливі частини, або додаючи нові. Майже всі операції можна виконати у декілька строк, що дуже спрощує розробку. Використання спеціальних операцій з DataFrame дозволяє не використовувати зайві цикли, що прискорює виконання операцій на великих масивах даних.

*sklearn*

Бібліотека, яка допомагає у вирішенні задач класифікації та векторизації текстової інформації. Вона допомагає будувати модель класифікації спаму, та перенавчати її.

*pickle*

Бібліотека для виконання збереження даних будь-якого виду, будь то прості вбудовані типи даних, чи складні користувацькі типи.

*google\_auth\_oauthlib*

За допомогою цієї бібліотеки здійснюється авторизація користувача, використовуючи ключ розробника, згенерований на ресурсах Google. Вхід виконується або через сервер користувача, або через вікно браузера.

*googleapiclient*

Бібліотека надає дозвіл опрацьовувати дані з системи Facebook. Вона здійснює передачу повідомлень між серверами Google, та об'єктами розробляємої інформаційної системи.

*API Pages*

Дозволяє звертатись до певних Facebook-сторінок та їх контенту особам, що мають права на це звернення, та виконувати дії з читання, запису, корегування контенту та ін.

C++ реалізація використовує наступні технології:

*Python.h*

Файл заголовку, який надає доступ для поєднання C++ та Python частин розробки. Інтерпретатор Python вбудовується у систему C++, використовуючи раніше написані функції та модулі пітона.

*Qt*

Сучасний фреймворк для побудови інтерфейсу користувача та багатьох інших задач. Наприклад за допомогою Qt реалізована база даних розроблюваної системи. Також Qt має зручні засоби для створення коду, тестування його та відлагодження.

Тепер визначимо особливості обраних структур даних.

На С++ частині розроблюваної системи майже всі структури даних були реалізовані за допомогою стандартного контейнеру std::vector.

«Вектор в C++ – це заміна стандартному динамічному масиву, пам'ять для якого виділяється вручну, за допомогою оператора new.

Розробники мови рекомендують використовувати саме vector замість ручного виділення пам'яті для масиву. Це дозволяє уникнути витоків пам'яті і полегшує роботу програмісту.

На відміну від std::array, який недалеко відходить від базового функціоналу звичайних фіксованих масивів, std::vector йде в комплекті з додатковими можливостями, які роблять його одним з найбільш корисних і універсальних інструментів в мові C++.

І при неініціалізованому, і при ініціалізуваному векторах не потрібно явно вказувати довжину масивів. Це пов'язано з тим, що std :: vector динамічно виділяє пам'ять для свого вмісту за запитом.

Подібно std :: array, доступ до елементів масиву може виконуватися як через оператор [] (який не виконує перевірку діапазону), так і через функцію at () (яка виконує перевірку діапазону)» [13].

Таким чином, є декілька важливих причин для застосування стандартного контейнеру std::vector:

1. Найчастіша операція з даними – отримання елементу з індексом n.

Vector є контейнером з лінійним сховищем пам'яті, отже всі операції добування будуть виконуватися зі складністю O(1), тобто за константний час. Це можливо, бо значення розміщуються у памяті послідовно. Отже кожна операція добування еквівалентна виразу (\*v) + n, де v - назва змінної типу std::vector, а n - індекс елементу, починаючи з нуля.

1. Кількість елементів заздалегідь не відома.

Наприклад, система не знає скільки буде публікацій чи коментарів під публікацією у окремого користувача. Тому потрібна структура даних, що має можливість розширюватись під час роботи системи. std::vector є саме такою структурою, з можливістю динамічного росту числа елементів.

1. Збереження у кеші.

Використовуючи послідовну структуру даних, більш вірогідно, що елементи потраплять у кеш, бо процесор зчитує інформацію у кеш використовуючи довжину машинного слова. Якщо використовувати не послідовну структуру, то до кеша може потрапити один, у кращому випадку 2-3 елемента, бо вони лежать у пам'яті системи не рівномірно.

## 5.2 Розробка програмного інтерфейсу

Програмний інтерфейс містить декілька форм відповідно до дій, що потрібно виконати. На рис. 5.1 наведено первинну форму, на якій можна увійти у систему (Sign In), зареєструвати нового користувача (Sign Up) та змінити акаунт, якщо користувач знаходиться в одному акаунті та хоче змінити його на існуючий інший (Change account).

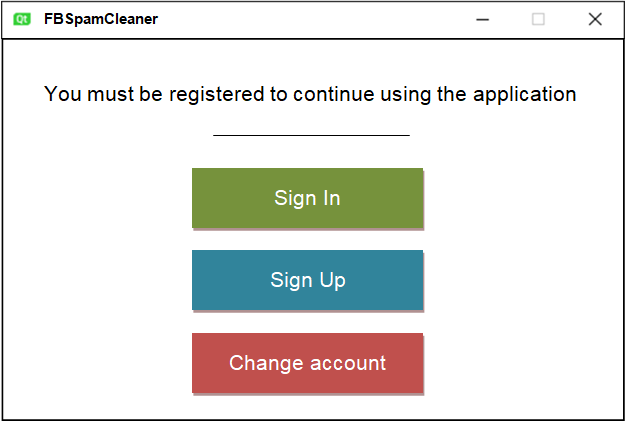


Рисунок 5.1 – Форма реєстрації та авторизації

На рис. 5.2 показано інформаційне вікно програми з роз’ясненням основного призначення розроблюваної системи.

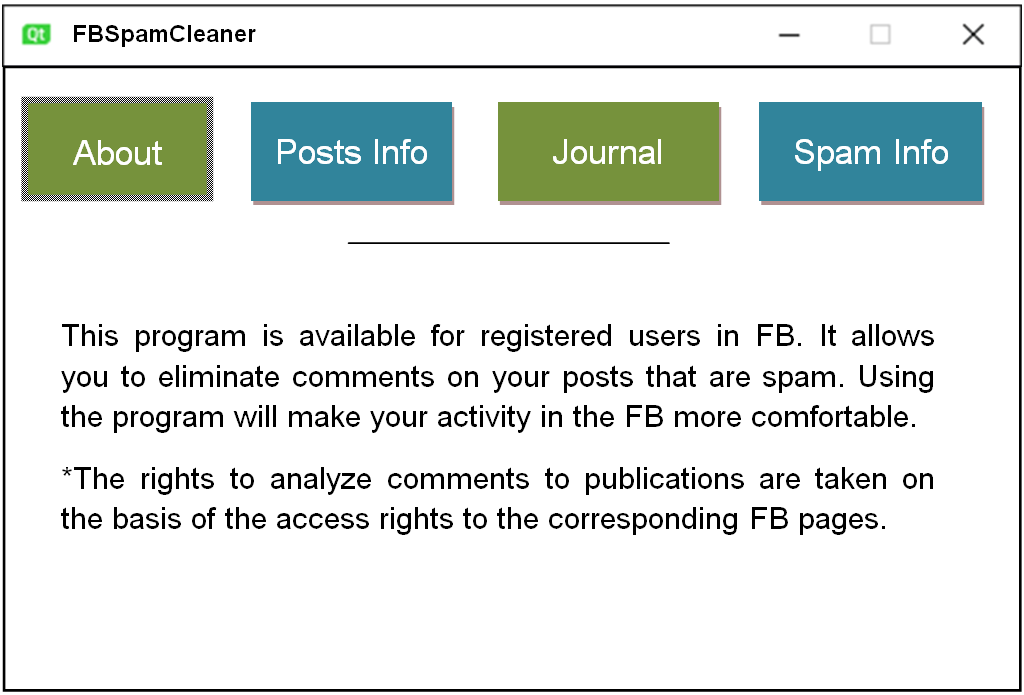


Рисунок 5.2 – Інформаційна форма

Натискання на кнопку «Posts info» веде до перегляду публікацій. Спочатку потрібно обрати необхідну сторінку на той випадок, коли у користувача їх декілька, а потом переглядати превью публікацій цієї сторінки (рис. 5.3).

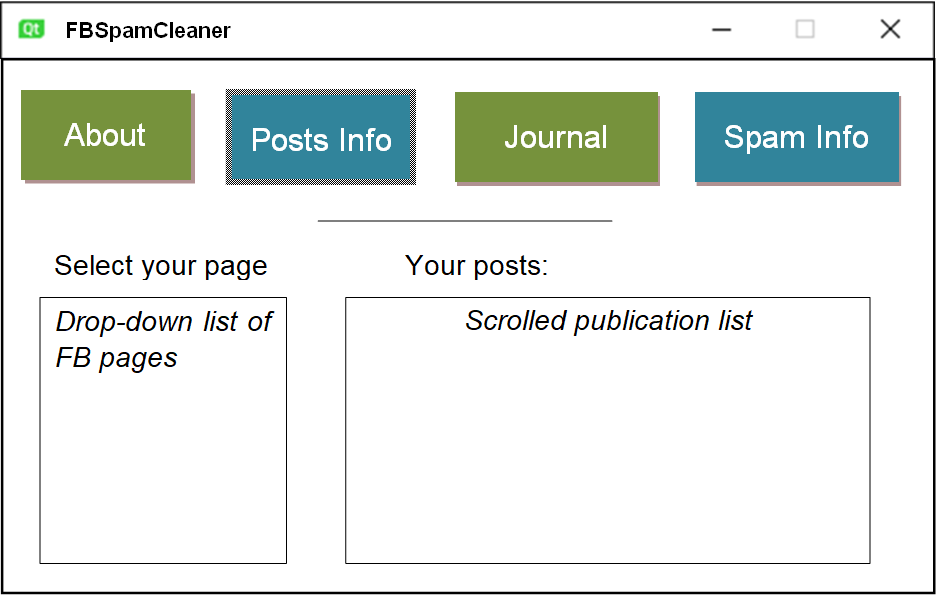


Рисунок 5.3 – Прототип форми для перегляду публікацій

Після обирання певної публікації із списку потрібно надавати користувачеві список коментарів до неї. Прототип відповідної форми наведено на рис. 5.4.

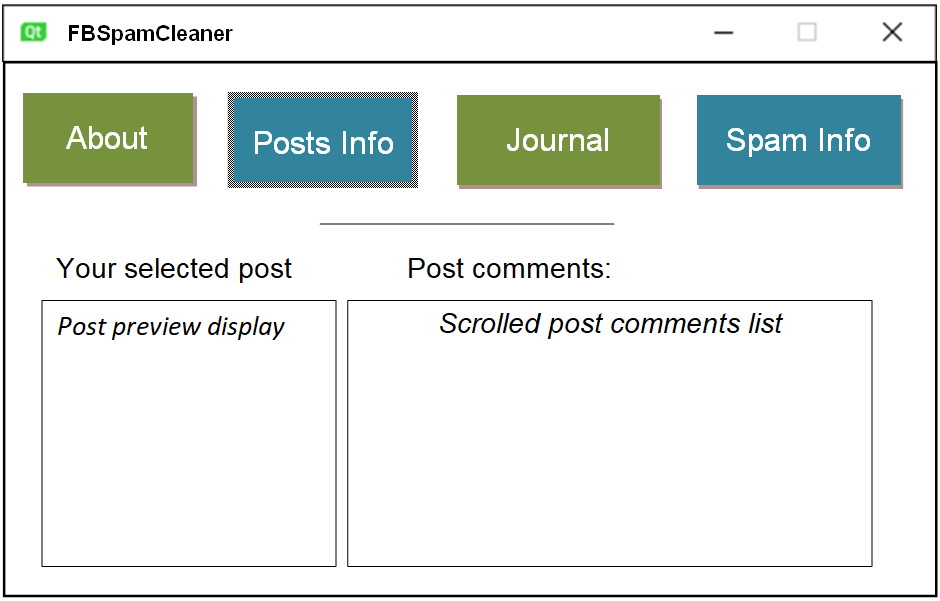


Рисунок 5.4 – Прототип форми для перегляду коментарів до публікацій

Для перегляду інформації щодо наявності спаму у коментарях необхідно натиснути «Spam Info». При цьому завантажуються дані щодо автору коментарю, його зміст, кількість лайків, дата створення коментарю та позначки чи є коментар перевереним та чи є він спамом. Прототип відповідної форми наведено на рис. 5.5.

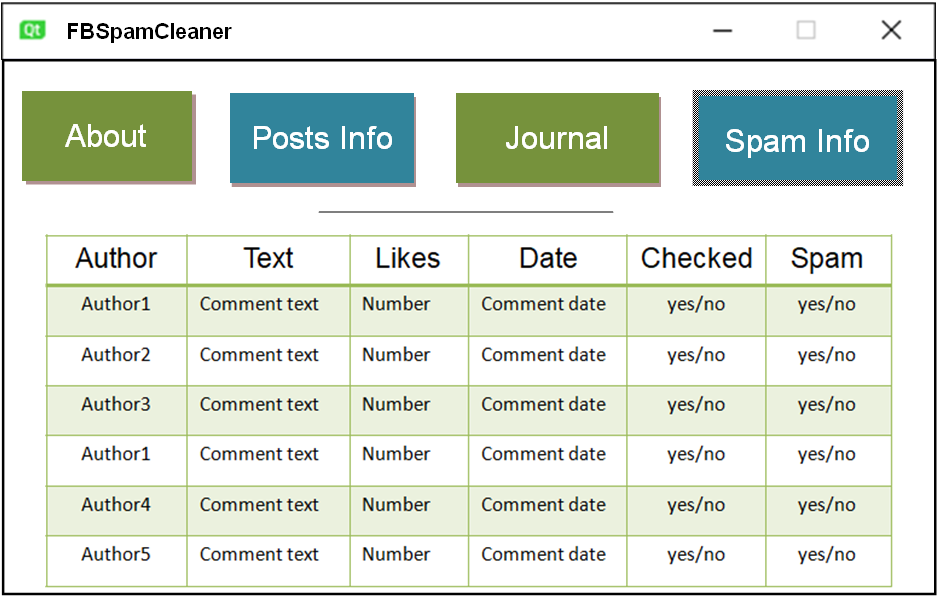


Рисунок 5.5 – Прототип форми для перегляду інформації щодо спаму у коментарях

## 5.3 Особливості розробки моделі класифікації спаму у коментарях

Для початку розробки треба мати модель для класифікації спаму, адже це є основним функціоналом системи.

Спочатку, був знайдений набір даних, в якому присутні коментарі до публікацій в необхідній кількості. Цей набір коментарів був спеціально зібраний для дослідження спаму.

У наведеній нижче таблиці перераховані набори даних, ідентифікатор публікації зі сторінки Facebook, кількість вибірок в кожному класі і загальна кількість зразків в наборі даних. Усі коментарі розмічені, тобто для кожного заздалегідь визначено – є він спамом чи звичайним коментарем.

Усього присутні 5 наборів даних, у кожному наборі різна кількість звичайних та спам-коментарів.

Таблиця 5.1 – Набір даних для дослідження спаму

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Ідентифікатор автору публікації | Ідентифікатор публікації | Кількість спам коментарів | Кількість звичайних коментарів | Загальна кількість коментарів |
| 94HGSwlv37s | 2906538 | 179 | 108 | 287 |
| jshTR76LK09 | 7354800 | 295 | 84 | 379 |
| JfKYW5F6Gj | 3104236 | 180 | 302 | 482 |
| 73GADHks65 | 6118697 | 167 | 185 | 352 |
| pHGt76xHNb | 5482437 | 204 | 206 | 410 |

Кожен коментар описаний у форматі, наведеному у табл. 5.2. Кожен рядок має атрибути, описані наступною таблицею

Таблиця 5.2 – Формат використаних коментарів

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Comment\_id | Author\_id | Date | Content | Class |
| Ідентифікатор коментаря | Ідентифікатор автора | дата створення коментаря | текст коментаря | 1 – це спам  0 – це не спам |

Ідентифікатор коментаря містить цифри, ідентифікатор автора коментаря – цифри та букви латинського алфавіту великі та маленькі (у загальному випадку – різну кількість літер). Формат дати є таким, що спочатку вказується рік, потім – місяць, а далі – день.

На рис. 5.6 показано зчитування набору коментарів за допомогою Python з csv-файлу.

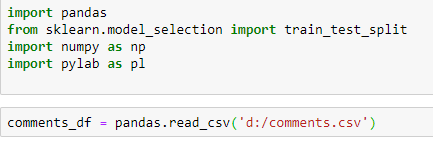


Рисунок 5.6 – Отримання тестового набору коментарів у Python-модулі

Далі приведено рис. 5.7 з прикладом заповнення набору даних.

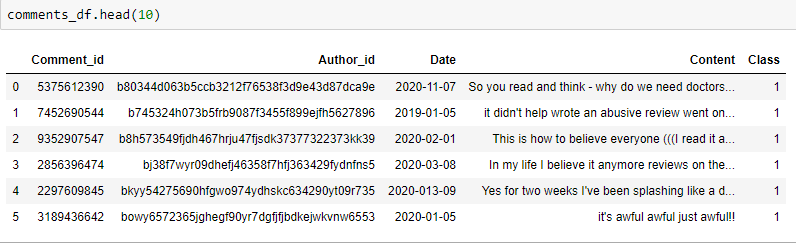


Рисунок 5.7 – Фрагмент тестового набору коментарів

Для максимальної працездатності системи треба обрати один з багатьох алгоритмів класифікації та векторизації тексту. Для цієї задачі була написана функція perform, вхідними параметрами якої є:

* список класифікаторів;
* список векторизаторів тексту;
* тренувальні дані;
* дані для тесту.

Функція видає процент успіху у передбачені класу коментаря, поєднуючи всі доступні класифікатори та векторизатори.

На рис. 5.8 наведений ключовий фрагмент програмної реалізації цієї функції. До списку класифікаторів відноситься vectorizers, успіх визначається з використанням score. Усі дані розділені на тренувальні train\_data та тестові test\_data.

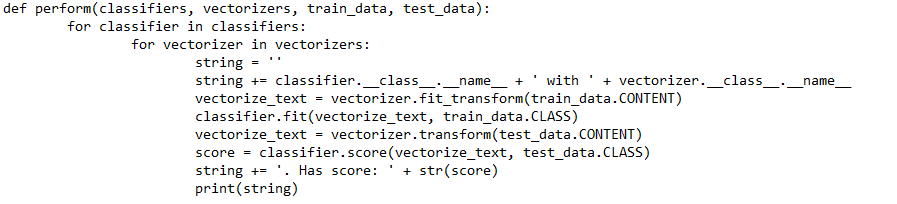


Рисунок 5.8 – Фрагмент методу perform

Список доступних типів класифікаторів classifiers береться з бібліотеки sklearn, список доступних векторизаторів текста vectorizers береться з бібліотеки sklearn, train\_data – частина даних, що використовується для тренування моделі, test\_data – частина даних, що використовується для тестування моделі

Тлумачення роботи методу:

За допомогою двох циклів for, здійснюється комбінація всіх можливих типів класифікаторів та векторизваторів. В основній частині кода формується строка string, яка використовується для виводу результату. Спочатку в цю строку записуються назви класів(типів) класифікатора і векторізатора за допомогою конструкції .\_\_class\_\_.\_\_name\_\_.  Далі будується модель класифікації за допомогою методу fit(), та векторізується текст за допомогою метода transform(). Наприкінці,  виконується підрахунок проценту успішності поточної комбінації за допомогою метода score() та записується до строки. Строка виводиться на екран.

Аби пропустити дані через цей метод, спочатку, треба розділити дані на тренувальні та дані для тесту. На рис. 5.9 показано програмну реалізацію розподілу даних.



Рисунок 5.9 – Розподіл даних на тренувальні та тестові

За допомогою бібліотеки pandas здійснюється завантаження даних у форматі .csv за допомогою метода read\_csv(). Далі дані діляться на тестові та тренувальні за допомогою метода train\_test\_split(). test\_size=0.4 позначає, що 40 відсотків буде зберігатися для тестування, а 60 відсотків для тренування. Випадкове перемішування даних random\_state відсутнє.

Тепер є можливість виконати метод perform(), використовуючи всі доступні класифікатори та векторизатори в бібліотеці sklearn. Результат виконання методу представлено у табл. 5.3. Значення успішності – рівня передбачення – наведені з округленням до сотих.

Таблиця 5.3 – Результат роботи методу perform()

|  |  |  |
| --- | --- | --- |
| Тип класифікатора | Тип векторізатора | Успішність |
| OneVsRestClassifier | TfidfVectorizer | 0.96 |
| RidgeClassifier | CountVectorizer | 0.94 |
| CalibratedClassifierCV | HashingVectorizer | 0.95 |
| KNeighborsClassifier | CountVectorizer | 0.91 |
| DecisionTreeClassifier | CountVectorizer | 0.91 |
| GradientBoostingClassifier | CountVectorizer | 0.89 |
| AdaBoostClassifier | HashingVectorizer | 0.92 |
| RandomForestClassifier | CountVectorizer | 0.93 |
| ExtraTreesClassifier | TfidfVectorizer | 0.93 |

По завершенню виконання методу, було визначено, що пара класифікатор-векторизатор з найбільшим показником передбачення є пара OneVsRestClassifier, TfidfVectorizer з рівнем передбачення у 96.43%.

Коли визначені найбільш відповідні класифікатор та векторизатор для вирішення задачі класифікації спаму, можна починати будування моделі класифікації спаму.

Спочатку, треба відокремити від набору коментарів зайву інформацію, бо для навчання моделі потрібні лише 2 колонки – текст коментарю та клас коментарю.

Для цього потрібно спочатку дані з всіх .csv файлів додати до списку frames. Далі за допомогою бібліотеки pandas з'єднати дані у одну структуру. Це можливо завдяки тому, що кількість та назви колонок однакове у всіх елементів списку. Та наприкінці зайві колонки відокремлюються від даних за допомогою методу iloc. Перший параметр “:” позначає що треба взяти всі строки, другий параметр “3:5” що треба взяти четверту та п'яту колонки.

 Коли дані оброблені, то можна починати процес будування моделі класифікації. На рис. 5.9 показано реалізацію будування моделі.

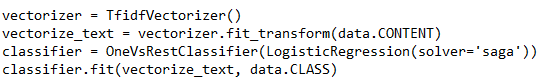


Рисунок 5.9 – Реалізація будування моделі

Тлумачення будування моделі:

На першій строчці будується об'єкт векторізатору. На другій – текстові дані векторізуються у чисельній формат для розпізнавання класифікатором. Далі будується об'єкт класифікатору. Наприкінці розпочинається процес будування моделі на векторизованому тексті за допомогою метода fit().

Тепер треба зберегти побудовану модель класифікації спаму та векторизатор тексту за допомогою коду, що наведений на рис. 5.10.

https://lh3.googleusercontent.com/PuhR3GIRe3pJSyFOggULYszWOOplS87MNsLuENBmhXjg3Xaw3-zE1YA6kkY5iUUKjfh4aU6ZosFsKi6apxQ3TGWCObN4Klsm6ig36OCfLbCzhOurmCnK1yj1oUKSL15Vy9SQHe1y

Рисунок 5.10 - Збереження результатів побудови моделі

При цьому використовується метод dump() з бібліотеки pickle, який приймає два аргументи. Перший – об'єкт який треба зберігати, другий – файл для збереження. Файл для збереження формується методом open(), де перший аргумент – назва файлу, другий – права доступу. Значення параметру “wb” значить, що файл відкрито для запису у бінарному форматі.

Для обчислення середньої абсолютної помилки, середньої квадратичної помилки та помилки кореня середнього квадратичного значення використовуються функції, що наведені на рис. 5.11.

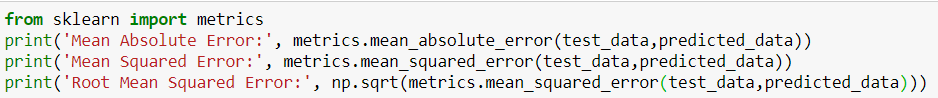


Рисунок 5.11 – Обчислення похибок роботи моделі

Таким чином, основні дії щодо будування моделі класифікації спаму та вектори затору тексту виконано.

# 6 ВИЗНАЧЕННЯ ВЛАСТИВОСТЕЙ АНАЛІЗАТОРА СПАМА

## 6.1 Тестування основних класів

Для тестування С++ модулів системи були задіяні наступні функції з використовуваних бібліотек:

TEST(use\_case\_name, test\_name) – це метод, який буде викликаний автоматично після запуску програми. Аргументи методу: use\_case\_name – ім’я варіанту використання, test\_name – і’мя тесту.

 ASSERT\_STREQ(str1, str2) – це метод котрий здійснює операцію порівняння рядків, переданих до нього. Якщо вони однакові, то тест пройдено, інакше результат вважається помилковим. Аргументи методу: str1 – перший рядок, str2 – другий рядок. Характер порівняння рядків є лексикографічним, тобто пов’язаним з кодами символів.

Було проведено декілька тестів для забезпечення працездатності системи на стороні C++.

Тестування класу “Публікація”

Тест-кейс: Побудувати об'єкт класу “Публікація”, за наступними параметрами description == “description”, id == “123”, image\_url == “ssh//image”, title == title.

Результат: змінні мають бути ініціалізовані у порядку виклику методів.  Далі буде надана програмна реалізація тесту.

TEST(Receive\_items\_from\_post, test\_post\_class) {

FacebookPost post;

post.set\_description("description");

post.set\_id("123");

post.set\_image\_url("ssh//image");

post.set\_title("title");

ASSERT\_STREQ("description", post.get\_description().c\_str());

ASSERT\_STREQ("123", post.get\_id().c\_str());

ASSERT\_STREQ("ssh//image", post.get\_image\_url().c\_str());

ASSERT\_STREQ("title", post.get\_title().c\_str());

}

Тестування прошло успішно.

Тестування класу “Коментар”

Тест-кейс: Побудувати об'єкт класу “Коментар”, за наступними параметрами authorName== “authorName”, id == “123”, likeCount == “4123”, publishedAt== “21.10.2020”, textDisplay == “Good Post”.

Результат: змінні мають бути ініціалізовані у порядку виклику методів.  Далі буде надана програмна реалізація тесту.

TEST(Receive\_items\_from\_comment, test\_comment\_class) {

Comment c;

c.set\_authorDisplayName("authorName");

c.set\_id("123");

c.set\_likeCount("4123");

c.set\_publishedAt("21.10.2020");

c.set\_textDisplay("Good Post");

ASSERT\_STREQ("authorName", c.get\_authorDisplayName().c\_str());

ASSERT\_STREQ("123", c.get\_id().c\_str());

ASSERT\_STREQ("4123", c.get\_likeCount().c\_str());

ASSERT\_STREQ("21.10.2020", c.get\_publishedAt().c\_str());

ASSERT\_STREQ("Good Post", c.get\_textDisplay().c\_str());

}

Тест пройдено успішно.

Тестування класу “Список публікацій”

Тест-кейс: Побудувати об'єкт класу “Список публікацій”.

Результат: дані мають бути не пошкодженими. Далі буде надана програмна реалізація тесту.

TEST(Receive\_post\_list, test\_post\_list\_receive) {

PyHandler pyHandler;

PostList post\_list = pyHandler.get\_post\_list();1

ASSERT\_STREQ("FacebookSpamClassifier", post\_list.at(0).get\_title().c\_str());

ASSERT\_STREQ("Spaaam test #3", post\_list.at(1).get\_title().c\_str());

ASSERT\_STREQ("Post for spam test #2", post\_list.at(2).get\_title().c\_str());

ASSERT\_STREQ("Post test spam #1", post\_list.at(3).get\_title().c\_str());

}

Тестування прошло успішно.

Тестування класу “Список коментарів”

Тест-кейс: Побудувати об'єкт класу “Список коментарів”.

Результат: дані мають бути не пошкодженими. Далі буде надана програмна реалізація тесту.

TEST(Receive\_comments\_list, test\_comments\_list\_receive) {

PyHandler pyHandler;

CommentList comment\_list = pyHandler.get\_comment\_list();

ASSERT\_STREQ("Awesome", comment\_list.at(0).get\_textDisplay().c\_str());

ASSERT\_STREQ("The best post of medicine ever!", comment\_list.at(1).get\_textDisplay().c\_str());

ASSERT\_STREQ("Very good post! I am happy!", comment\_list.at(2).get\_textDisplay().c\_str());

}

Тест пройдено успішно.

Для модулів Python було зроблено тестування моделі класифікації спаму за допомогою мануального тестування.

Тест-кейс: на вхід системи подан коментар з наступним змістом: " It was the worst thing I've seen in my life. Horror is a very mild word"

Очікуваний результат: класифікатор повинен відмітити цей текст, як спам. На рис. 6.1 надана програмна реалізація для тестування моделі класифікації.

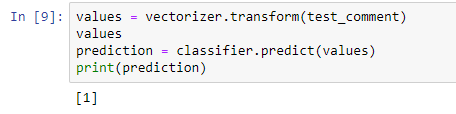


Рисунок 6.1 – Результат тестування класифікатору

Спочатку, текст коментаря векторізується за допомогою TF-idf векторізатору текста  методом transform(), який приймає текст як аргумент.

За допомогою методу predict() з бібліотеки sklearn було зроблено передбачення на основі навченої раніше моделі для класифікації спаму. Результат був відображений методом print().

Результатом виконання цього коду є “1” як значення класу коментарю. Згідно налаштувань моделі класифікації даних, одиниця є класом спама. Отже результат теста задовольняє потребам програмної системи.

## 6.2 Контрольні приклади роботи системи

Розглянемо декілька контрольних прикладів роботи програми. Беручи до уваги, що інтерфейс мережі Facebook є дуже яскравим, було прийнято рішення для розроблюваної програмної системи теж залишити повноколірний інтерфейс. При цьому можливо використовувати превью картинок та фотографій, що є у публікаціях.

На рис. 6.2 показано форму для перегляду публікацій. У лівій частині форми можливо обирати потрібні Facebook-сторінки, у правій частині – публікацій з обраної сторінки. Зауважимо, що можливо обирати тількі ті сторінки, до яких користувачеві надано відповідні права доступу за допомогою API Pages.

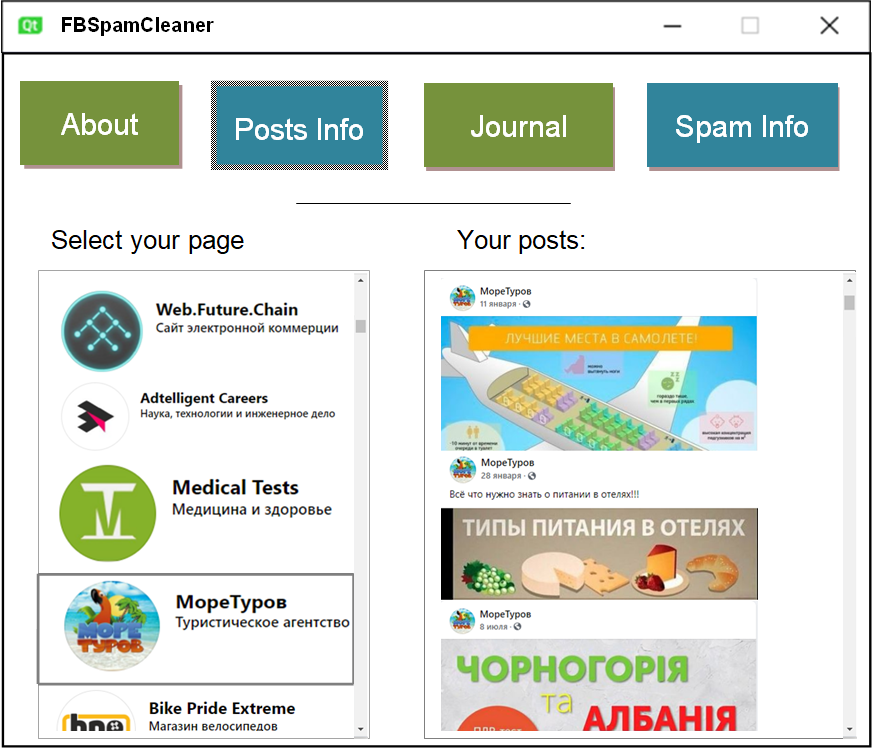


Рисунок 6.2 – Форма для перегляду публікацій

Для обирання певної публікації (посту) треба двічі клацнути по ній мишею після чого відкривається форма для перегляду коментарів до обраної публікації (рис. 6.3).

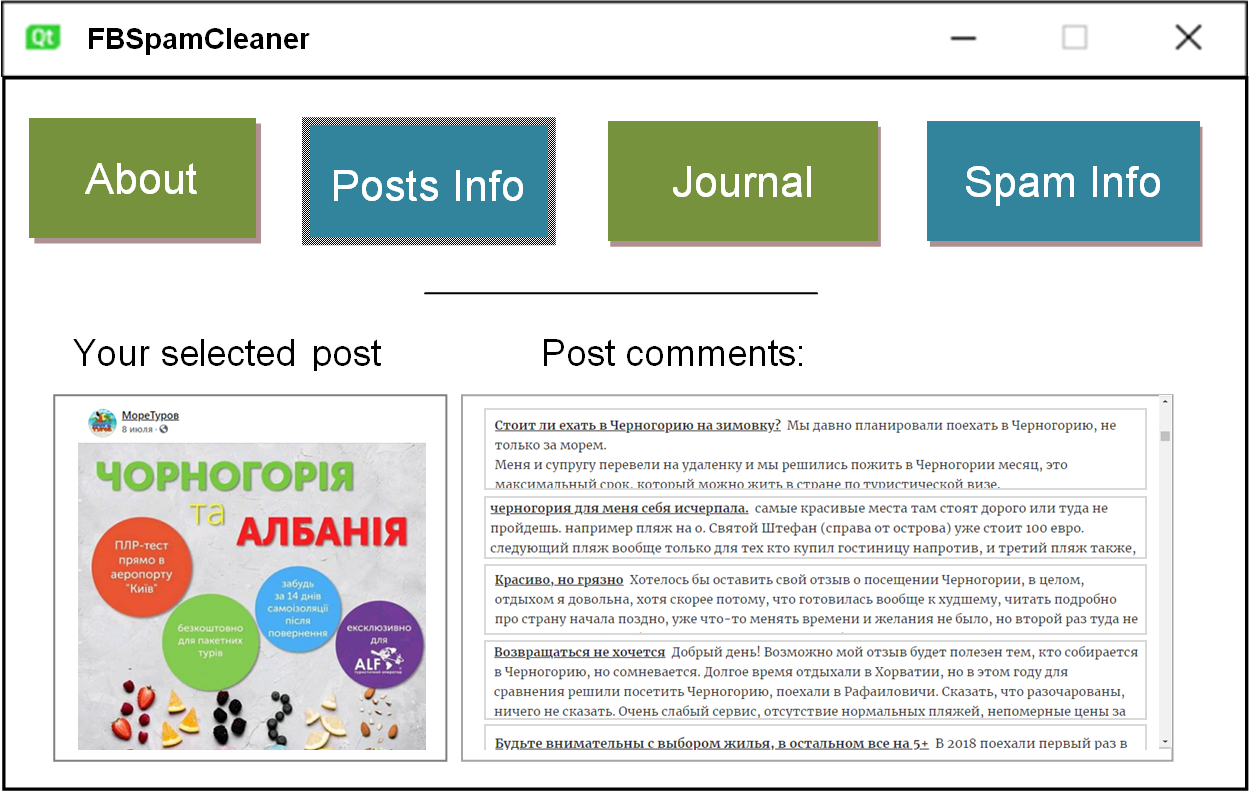


Рисунок 6.3 – Форма для перегляду коментарів до публікацій

Корситувач може скроллити коментарі та при необхідності видаляти їх, двічі клацнувши по коментарю мишею. Заувижимо, що користувач має можливість видаляти коментарі ще до того, як проведена класифікація спаму. Це повністю відповідає ідеології роботи з коментарями до власних публікацій у мережі Facebook. Усі зміни відображаються на відповідній сторінці.

Узагальнені результати роботи системи після перевірки коментарів на спам можна побачити, використовуючи кнопку «Journal». Відповідна форма показана на рис. 6.4. Для перевірки коментаря потрібно двічі клацнувши по ньому мишею. Пысля цього коментар позначається як перевірений (стовпець Checked). Якщо цей коментар є спамом, відповідна комірка у стовпці Spam позначається червоно-коричневим кольором. При бажанні видалити цей коментар по ньому потрібно двічі клацнути мишею. Для автоматичного видалення спам-коментарів потрібно зробити відповідні налаштування у системі, але при невеликій кількісті коментарів більш доцільно видаляти спам вручну.

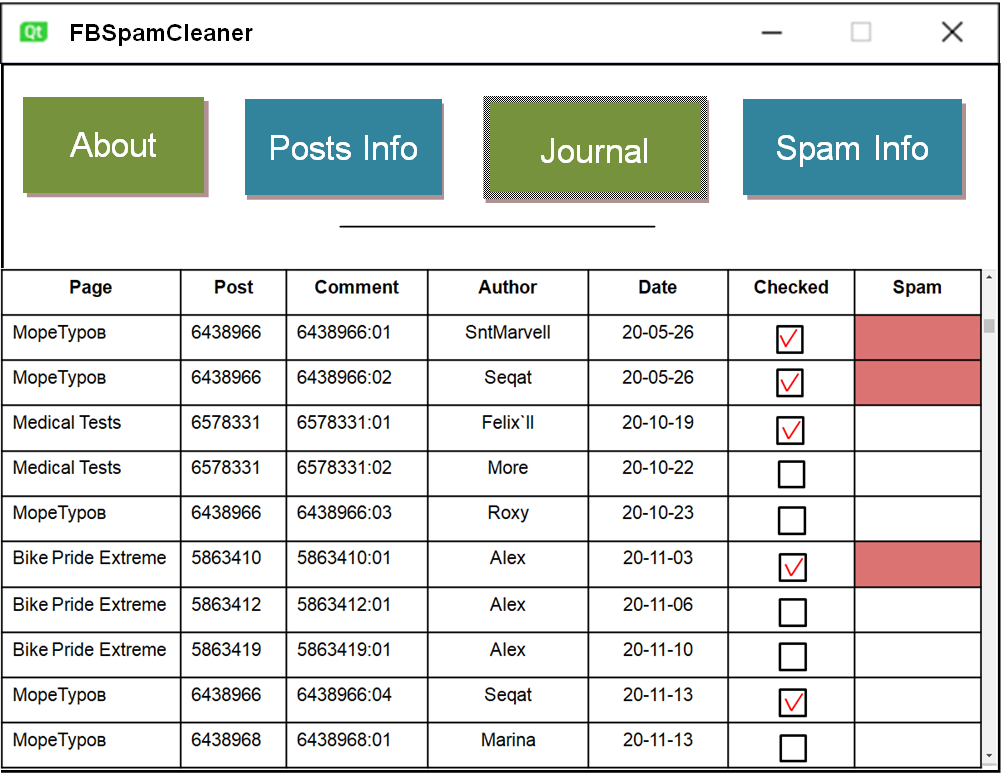


Рисунок 6.4 – Журнал роботи з коментарями

Для перевірки роботи системи було розглянуто набори коментарів до трьох публікацій, результати перевірки наведено у табл. 6.1. Для значень проценту спаму проведено округлення до десятих.

Таблиця 6.1 – Аналіз коментарів до публікацій на спам

|  |  |  |  |
| --- | --- | --- | --- |
| Ідентифікатор публікації | Кількість коментарів | Кількість спаму | Процент спаму |
| 6310527 | 1509 | 217 | 14.4% |
| 6427589 | 1752 | 238 | 13.6% |
| [6427672](https://www.youtube.com/watch?v=eoqR4L5B27M) | 1366 | 106 | 7.8% |

Отже, якщо поєднати результати аналізу коментарів до всіх трьох публікацій, то можна усереднити результати: (14.4 + 13.6 + 7.8) / 3 ≈ 11.9 % спаму в середньому міститься у коментарях розглянутих публікацій. Будемо використовувати термін “одиниця корисної інформації” як синонім коментарю, що не є спамом. Тоді можна вважати, що після видалення спам-коментарів кількість корисної інформації була збільшена приблизно на 11.9 % відносно загальній початковій кількості коментарів.

# 7 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА У НАДЗВИЧАЙНИХ СИТУАЦІЯХ

У будь-якій професії є свої певні вимоги стосовно безпеки експлуатації тих чи інших пристроїв, які використовуються в роботі. У зв'язку з цим була створена і розвивається наука про охорону праці і життєдіяльності людини.

Стан діяльності, за якого з визначеною ймовірністю виключено прояв небезпек або ж відсутня надзвичайна небезпека, визначають як «безпеку». ДСТУ 2293-99 визначає термін «безпека» як стан захищеності особи та суспільства від ризику зазнати шкоди [14].

Дипломна робота розглядає питання фільтрації спаму у коментарях публікацій мережі Facebook, розв’язання яких здійснюється працівниками галузі інженерії програмного забезпечення. Працівники цієї галузі при роботі з комп'ютером наражаються ряду небезпечних і шкідливих виробничих факторів: електромагнітних полів (діапазон радіочастот: ВЧ, УВЧ і СВЧ), інфрачервоного і іонізуючого випромінювань, шуму і вібрації, статичної електрики і ін.[15].

Охорона праці – це регульована предметна область, для якої потрібно визначити види заходів та відповідні умови їх проведення, необхідні для успішної реалізації і завершення кваліфікаційної роботи магістра зі спеціальності інженерія програмного забезпечення.

Організація заходів із забезпечення безпечних умов праці та поведінки у надзвичайних ситуаціях засновується на наступних нормативних документах:

* ДСТУ 2293-99. Охорона праці. Терміни та визначення [16];
* НАПБ А.01.001-2015. Правила пожежної безпеки в Україні [17];
* ДСН 3.3.6.037-99. Санітарні норми виробничого шуму, ультразвуку та інфразвуку;
* НПАОП 0.00-1.28-10. Правила охорони праці при експлуатації ЕОМ [18];
* ДСН 3.3.6.042-99. Санітарні норми мікроклімату виробничих приміщень;
* НАПБ Б.03.001-2204. Типові норми належності вогнегасників;
* ДБН В.2.5-28-2006. Природне і штучне освітлення.

Вивчення найважливіших заходів з охорони праці та безпеки у надзвичайних ситуаціях для працівників галузі інженерія програмного забезпечення наведено у Додатку В та висвітлює наступні питання:

* організація та управління охороною праці на робочому місці інженера програмного забезпечення;

## обґрунтування заходів з покращення умов охорони праці;

## розрахунок захисного занулення;

* надзвичайні ситуації та шляхи їх запобігання.

В ході роботи проаналізовано робоче місце інженера-програміста, вимоги до інтервалів робочих змін, необхідний рівень освітленості при умовах роботи на моніторі високої точності, та режим відпочинку.

Спираючись на проаналізовані дані, було розраховано захисне занулення та визначення категорії пожежонебезпечних приміщень.

В якості рекомендацій щодо поліпшення умов праці, основне – придбання та монтаж засобів сигналізації про порушення нормального функціонування виробничого обладнання, засобів аварійної зупинки, а також пристроїв, що дозволяють виключити виникнення небезпечних ситуацій при повному або частковому припиненні енергопостачання і подальшому його відновленні, а також організація навчання працівників наданню першої допомоги потерпілим на виробництві.

ВИСНОВКИ

В ході роботи була виконана програмна реалізація системи, яка надає можливість видаляти спам з коментарів публікацій Facebook. Для аналізу та класифікації коментарів була розроблена та застосована модель публікації Facebook, що враховує взаємозв’язок контенту публікації та ступені корисності коментарів.

Кількість корисної інформації, що знаходиться у коментарях публікації Facebook, визначається як співвідношення кількості звичайних коментарів до загальної кількості коментарів до публікації. Результати тестування програмної системи показали її працездатність, та спроможність виконувати класифікацію та видалення спаму на окремих публікаціях Facebook. Застосування розробленої системи забезпечило збільшення кількості корисної інформації приблизно на 11,9%.

Для досягнення мети в збільшенні кількості корисної інформації було вжито наступні дії: розробка моделі публікації Facebook та класифікації текстового спаму, розробка модулю для добуття даних зі сторінки мережі Facebook, розробка інтерфейсу програмної системи для взаємодії з користувачем. Також було розроблено зв’язуючий модуль, який зберігає дані, обробляє дії користувача, посилає запити до класифікатору спама.

Технологіями розробки є Facebook API Pages, як джерело даних, мова програмування C++, за допомогою якої здійснюється робота з інтерфейсом системи, мова програмування Python, за допомогою якої здійснюються запроси до Facebook API Pages, бібліотека sklearn, яка допомагає у класифікації спаму, фреймворк QT, що використовується для розробки інтерфейсу системи.

**ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ**

1. Бенгфорт Б. Прикладной анализ текстовых данных на Python. Машинное обучение и создание приложений обработки естественного языка. – М. :  Питер, 2016. - 368 с.
2. Фадеев А.Ю. Сравнительный анализ программного обеспечения. – Наука и перспективы, 2016. – 246 с.
3. Віргес К. Розробка вимог до програмного забезпечення. 2-е вид. / пер. з англ./ – М.: Русская редакция, 2004. – 575 с.
4. Архипенков С. Лекции по управлению программными проектами – М. : Самиздат, 2009. – 128 с.
5. Komleva N.O., Stanislavyk Y. H. Software analyzer for spam filtering in comments on Facebook posts // XIII Міжнародна науково-практична конференція «Інформаційні технології і автоматизація – 2020», Одеса, 22-23 жовтня 2020. С. 272–274.
6. API Pages [Електроний ресурс] Режим доступу: URL: <https://developers.facebook.com/docs/pages/>, Заголовок з екрану. – 10.12.2020.
7. Керування сторінками Facebook [Електроний ресурс] Режим доступу: URL: <https://developers.facebook.com/docs/pages/managing>, Заголовок з екрану. – 10.12.2020.
8. Васильєв О. Програмування мовою Python. – М. : Богдан, 2019. – 180 с.
9. Леоненков А. Самоучитель UML. - БХВ-Петербург, 2007. – 568 с.
10. Кунгурцев О.Б. Об’єктно-орієнтована технологія створення програмних продуктів. – Одеса: ВМВ, 2006. – 186 с.
11. Сидоренко И. Дизайнер интерфейсов. – М. :  Олимп-Бизнес, 2019. – 170 с.
12. Вбудування Python у C++ [Електроний ресурс] Режим доступу: URL: https://www.codeproject.com/Articles/820116/Embedding-Python-program-in-a-C-Cplusplus-code. – 10.12.2020.
13. Морозов Е.А. Анализ предметной области и концептуальное проектирование базы данных – М. :  МИСиС, 2002. – 42 с.
14. Мейерс С. Эффективный и современный С++. – М. :   Диалектика Вильямс, 2020. – 304 с.
15. Гандзюк М.П. Основи охорони праці: Підручник/ М.П. Гандзюк, Є.П. Желібо, М. О. Халімовський. - Львів: Новий світ-2000, 2003. – 408 с.
16. Гогіташвілі Г.Г. Основи охорони праці: Навчальний посібник/ Г.Г. Гогіташвілі, В.М. Лапін. – Львів: Новий світ-2000, 2006. – 232 с.
17. Керб Л.П. Основи охорони праці : Навчальний посібник/ Л. П. Керб. – Вид. 2-ге, без змін. – К.: КНЕУ, 2006. – 216 с.
18. Про затвердження Правил пожежної безпеки в Україні : наказ Міністерства внутрішніх справ України від 30.12.2014 р. № 1417. Офіційний вісник України. 2015 р. № 26. С. 91, Ст. 767.
19. Москальова В.М. Основи охорони праці: Підручник/ В.М. Москальова. – К.: ВД "Професіонал", 2005. – 672 с.