Державний вищий навчальний заклад
“Прикарпатський національний університет імені Василя Стефаника”
Кафедра інформаційних технологій

УДК 004

ДИПЛОМНИЙ ПРОЕКТ

Тема: Розробка сервісу для автопостингу в
соціальних мережах. Розробка бота

Спеціальність: 121 Інженерія програмного забезпечення

ПОЯСНЮВАЛЬНА ЗАПИСКА

ДП.ІПЗ-28.ПЗ
(позначення)

Рецензент
зав. кафедри Козленко М.І.
(посада) (підпис) (дата) (розшифровка підпису)

Студент
ІПЗ-41 Харун Ю.Ю.
(шифр групи) (підпис) (дата) (розшифровка підпису)

Нормоконтролер
зав. кафедри Козленко М.І.
(посада) (підпис) (дата) (розшифровка підпису)

Керівник дипломного проекту
професор Кузь М.В.
(посада) (підпис) (дата) (розшифровка підпису)

Допускається до захисту

Завідувач кафедри

Козленко М.І.
(посада) (підпис) (дата) (розшифровка підпису)

2020
(рік)

Державний вищий навчальний заклад
«Прикарпатський національний університет імені Василя Стефаника»
Факультет математики та інформатики Кафедра інформаційних технологій
Спеціальність Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ:
Завідувач кафедри Козленко М.І.
_____
„_____” _____ 20___р.

ЗАВДАННЯ
НА ВИКОНАННЯ ДИПЛОМНОГО ПРОЕКТУ

Студенту Харуну Юрію Юрійовичу
(прізвище, ім’я, по батькові студента)
1. Тема проекту Розробка сервісу для автопостингу в соціальних мережах. Розробка бота
_____
затверджена розпорядженням по факультету математики та інформатики від „11” вересня 2019 р. №7
2. Термін здачі студентом закінченого проекту 22 травня 2020р.
3. Вихідні дані до дипломного проекту Стандарт кафедри інформаційних технологій, технологія програмування серверів – Selenium Webdriver, технологія розгортання програмного забезпечення – Heroku
4. Зміст пояснювальної записки (перелік питань, що їх належить опрацювати)
1. Розгляд предметної області та аналіз ринку для сервісу автопосингу
2. Опис основних вимог та характеристик бота для сервісу автопостингу
3. Використані інструменти та розробка функціоналу бота для сервісу автопостингу
4. Бізнес план розробки бота для автопостингу
5. Перелік графічного матеріалу (з точним забезпеченням обов’язкових креслень) 1. Титульний аркуш; 2. Загальний опис і мета. 3. Головні завдання автопостингу; 4. Актуальність, переваги та недоліки; 5. Формулювання завдання; 6. Покроковий алгоритм роботи сервісу; 7. ER-Diagram бази даних; 8. Use Case Diagram; 9. Створення вебдодатку; 10. Створення бота; 11. Бізнес план розробки автопостингу.
_____
_____

6. Дата видачі завдання 11.09.2019 р.

Керівник _____ Кузь М.В.
(підпис) (розшифровка підпису)
Завдання прийняв до виконання _____ Харун Ю.Ю.
(підпис) (розшифровка підпису)

КАЛЕНДАРНИЙ ПЛАН

Номер і назва етапів дипломного проекту	Термін виконання етапів проекту	Примітка
1. Розгляд предметної області та аналіз ринку для сервісу автопостингу	12.12.2019	Виконав
2. Опис основних вимог та характеристик бота	15.02.2020	Виконав
3. Використані інструменти та розробка функціоналу бота	17.04.2020	Виконав
4. Бізнес план розробки автопостингу	11.05.2020	Виконав
5. Оформлення пояснювальної записки	18.05.2020	Виконав

Студент _____ Харун Ю.Ю
(підпис) (розшифровка підпису)

Керівник проекту _____ Кузь М.В.
(підпис) (розшифровка підпису)

РЕФЕРАТ

Пояснювальна записка: 73 сторінки (без додатків), 31 рисунок, 3 таблиці, 21 джерело, 1 додаток на 8 сторінках.

Ключові слова: СЕРВІС АВТОПОСТИНГУ, БОТ, SELENIUM WEBDRIVER, PYTHON, HEROKU, GOOGLE CHROME, ДРАЙВЕР.

Об`єктом дослідження є бот сервісу автопостингу для соціальних мереж.

Мета роботи: проектування та розробка бота сервісу автопостингу для соціальних мереж.

Стислий опис тексту пояснювальної записки:

У даному дипломному проекті розглядається предметна область та проаналізовано ринок для сервісу автопостингу. Також було описано основні вимоги та характеристики бота для автопостингу, його проектування та реалізацію та обґрунтовано його актуальність та цінність.

ABSTRACT

Explanatory note: 73 pages (without appendix), 31 figures, 3 tables, 21 references, 1 appendix on 8 pages.

Key words: AUTOPOSTING SERVICE, BOT, SELENIUM WEBDRIVER, PYTHON, HEROKU, GOOGLE CHROME, DRIVER.

The object of research is the bot of the autoposting service for social networks.

The purpose of the work is design and development of an autoposting service bot for social networks is the purpose of the work.

Brief description of the explanatory note:

In this diploma project the subject area is considered and the market for autoposting service is analyzed. The main requirements and characteristics of the bot for auto-posting, also its design and implementation were described and its relevance and value were substantiated.

ЗМІСТ

ВСТУП 8
1 РОЗГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ ТА АНАЛІЗ РИНКУ ДЛЯ СЕРВІСУ АВТОПОСТИНГУ 10
1.1 Аналіз предметної області для сервісу автопостингу 10
1.2 Аналіз ринку для сервісу автопостингу 14
1.3 Постановка задачі дипломного проекту 22
2 ОПИС ОСНОВНИХ ВИМОГ ТА ХАРАКТЕРИСТИК БОТА ДЛЯ СЕРВІСУ АВТОПОСТИНГУ 24
2.1 Функціональні вимоги бота для сервісу автопостингу 24
2.2 Нефункціональні вимоги бота для сервісу автопостингу 27
2.3 Короткий аналіз основних характеристик бота для сервісу автопостингу 31
2.4 Timeline diagram бота для сервісу автопостингу 34
2.5 Sequence diagram бота для сервісу автопостингу 35
2.6 Activity diagram бота для сервісу автопостингу 39
3 ВИКОРИСТАНІ ІНСТРУМЕНТИ ТА РОЗРОБКА ФУНКЦІОНАЛУ БОТА ДЛЯ СЕРВІСУ АВТОПОСТИНГУ 42
3.1 Використані інструменти для розробки бота для сервісу автопостингу та їх підключення 42
3.2 Функції Webdriver та входу в профіль 46
3.3 Функції переходу на профіль користувача та публікації 48
3.4 Функції отримання публікацій та їх видалення 50
3.5 Функції публікації та видалення для соціальної мережі Фейсбук 54
4 БІЗНЕС ПЛАН РОЗРОБКИ БОТА ДЛЯ АВТОПОСТИНГУ 61
4.1 Резюме 61
4.2 Маркетинг 61
4.3 Обґрунтування фінансових вкладів 66
4.4 Нормативно-правові нюанси 68

4.5 Складання фінансового бюджету 69
4.6 Оцінка можливих ризиків 70
ВИСНОВКИ 71
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 72
ДОДАТКИ 74

ВСТУП

Автопостинг – це публікація матеріалу, яка здійснюється у автоматичному режимі із застосуванням стороннього програмного забезпечення або скриптів, та з можливістю вибирати час, зручний для публікації того чи іншого матеріалу [1].

Зараз він набирає особливу популярність, хоча багато хто ще не знає, як ним користуватись. Коли передивляєшся картинки або читаєш коментарі у випадковій або улюбленій групі, не звертаєш увагу, ким та як зроблена публікація. Це влаштовує як і автора, так і простого користувача. Автору не потрібно хвилюватись, щоб матеріал опублікувався вчасно, а користувач завжди буде отримувати контент вчасно.

Простий користувач у теперішніх реаліях достатньо велику частину часу проводить у соціальних мережах. Це не завжди використовується для просування сервісів автопостингу на можливий ринок збуту. Так, як в цьому є великий потенціал росту прибутку, його даремно не враховують.

Актуальність теми:

Багато хто використовує такі ресурси для демонстрування товару або послуги. Можливо дехто знає, що існує автоматичне розміщення реклами в пабліках, групах та інших місць з великою базою користувачів [2].

Поки що тільки частина заінтересованих людей використовує ці сервіси, які можуть значно збільшити число людей, зацікавлених їхнім товаром. Але за цей рахунок вони виграють та мають змогу обійти конкурентів.

Виконується публікація матеріалу у заздалегідь відібраних відомих користувачу соціальних мережах. Звичайна розсилка повідомлень на пошту тут не передбачена, так як її тепер часто приймають за спам. Даний метод дає дуже хорошу підтримку на майбутнє та прекрасний старт.

Використання автопостингу значно збільшує ефективність реклами та економить час, тому автоматизація процесу хороша у будь якій сфері. Цілеспрямований підбір певного матеріалу під простих користувачів, тобто головну аудиторію, та розбиття цієї аудиторії за їхніми інтересами, дає дуже хорошу віддачу.

Об’єкт дослідження: популярні сервіси автопостингу у соціальних мережах.

Предмет дослідження: функціонал вибраних сервісів автопостингу.

Методи дослідження: Проведення аналізу та дослідження популярних сьогодні сервісів автопостингу за допомогою ресурсів в мережі Інтернет та опрацювання самих сервісів.

Мета дипломної роботи: розробка бота для сервісу автопостингу в соціальних мережах.

Для досягнення мети дипломної роботи поставлено такі завдання:

- Перегляд популярних сьогодні сервісів автопостингу та їх функціоналу.
- Виявлення особливостей окремих сервісів.
- Аналіз їхнього функціоналу, розцінок та особливостей.
- Створення власного сервісу на основі проведеного аналізу.

Завдання роботи: розробка сервісу автопостингу для соціальних мереж. Розробка бота.

1 РОЗГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ ТА АНАЛІЗ РИНКУ ДЛЯ СЕРВІСУ АВТОПОСТИНГУ

1.1 Аналіз предметної області для сервісу автопостингу

Сам метод автопостингу по впливу на звичайних користувачів та, можливо, майбутніх клієнтів, значно перевершує метод розсилки повідомлень [2].

Отримувач може здійснити будь які маніпуляції:

прочитати;

- видалити;
- закинути у папку зі спамом;
- просто проігнорувати;
- зовсім забути про конкретну поштову скриньку.

За будь яких умов відбудеться контакт «продавець – заінтересований або незаінтересований (поки що) клієнт» [2]. Рекламні пости та звичайний контент соціальних мереж неможливо буде уникнути у своїй стрічці новин.

Тепер стає ясно, кому ж потрібен автопостинг. Звичайному підприємцю чи компанії будь яких розмірів, які хочуть якісно, дешево та масово розрекламувати продукт, який хочуть продати або новини, які хочуть розповсюдити, відомим блогерам та простим людям, які хочуть поширити інформацію у дві-три свої соціальні мережі.

Користуючись можливостями автопостингу простий користувач значно спрощує для себе задачу: може виставити настройки таким чином, щоб в задану дату в конкретний час сервіс розмістив бажаний контент, тобто текст, заголовки, хештеги, картинки та фото по раніше заданих соціальних мережах та групах. Все дуже гнучко налаштовується під конкретну задачу.

Також очевидним є те, в чому ж автопостинг є вигідним: набагато більше охоплення аудиторії, що для простих розсилок повідомлень, які налаштовуються вручну, є непосильною задачею. Якісно організований та налаштований автопостинг не залишить шансів великій кількісті найнятих помічників, асистентів та модераторів. Вони не зможуть осилити і малої долі від цього об’єму роботи.

В теперішній час існує велика кількість різних соціальних мереж. Сам автопостинг тепер є доступним в будь якій з них, але для власної вигоди, тобто масової публікації матеріалу, потрібно орієнтуватись на найбільш популярні соціальні мережі. Звичайний користувач бачить принцип роботи тільки поверхнево, але для максимальної віддачі потрібно знати багато деталей та прийомів. Матеріал повинен бути заздалегідь підготований. Тоді він передається сервісу та буде виставлятися згідно певного, заздалегідь вибраного розкладу на тиждень, місяць, рік чи більше.

Очевидним є зручність таких сервісів.

Саме налаштування автопостингу, особливості кожного матеріалу та адаптація під задані цілі є дуже серйозною задачею [2]. Це все вже було вивчено та оброблено спеціалістами, які займалися питаннями в цій сфері.

Соціальна мережа, яка має чи не найбільшу базу користувачів, Фейсбук (Facebook) також має в собі велику аудиторію ділових людей [3]. Тому автопостинг в цій соціальній мережі є достатньо ефективним.

Але взаємодіяти можна не тільки з однією соціальною мережею, а з багатьма ресурсами. Можна розміщати матеріал та пости в інших мережах.

Ніяких проблем не складе і розміщувати один і той же самий матеріал у різні соціальні мережі, або навпаки, потрібне число проектів чи матеріалу в одну соціальну мережу.

Також є можливість налаштувати розсилку матеріалу по різних групах чи соціальних мережах зразу з декількох акаунтів. І все це можна буде гнучко

налаштовувати під конкретні потреби: періодичність надсилання, час, коли це буде відправлено, саме відправлення матеріалу.

Рекламний матеріал автопостинг відправить у спеціально призначені для цього тематичні групи. Існує декілька напрямків, по яких їх можна буде направити: по активності групах та соціальних мережах, по кількості користувачів та по темах матеріалу. Це можна буде протестувати, подивитись на відгук аудиторії та пізніше прийняти рішення про включення цього в автопостинг. І після необхідного налагодження цей процес зможе працювати в автоматичному режимі.

Також існує досить велика соціальна мережа Інстаграм (Instagram) [4]. В останні роки він набрав не аби яку популярність, хоча в першу чергу задумувався, як місце для розміщення за збереження фотографій та картинок. Тут автопостинг працює на повну, адже ця соціальна мережа дуже швидко набрала свою багатомільйонну аудиторію та стала однією з найбільш прибуткових та головних рекламних платформ.

Так як простий користувач зацікавлений в розкрутці свого продукту, використання функціоналу автопостингу забезпечить великий приріст аудиторії, що має задовільнити бажання користувача.

Також інтересною середою для спілкування та, в особливості, автопостингу, виступає Твітер (Twitter) [5]. Тут люди дуже активно спілкуються, обмінюються інформацією та відвідують дану соціальну мережу регулярно.

Неповні знання про автопостинг можуть бути часто неперевіреними та суперечливими, і, як правило, шкодити користувачу. Автопостинг – це добре вивчена, налагоджена та комплексна система [1]. Довіряти варто тільки великим компаніям або спеціалістам, так як конкуренція висока.

Правильний вибір соціальних мереж та змістовне наповнення розсилок – від цього напряму залежить результативність роботи автопостингу. Потрібно направляти розсилки в потрібне для користувача русло. Правильно підібраний

та навчений колектив врахує інтереси тих споживачів, що будуть бачити рекламні пости та їхню кількість.

В теперішніх реаліях соціальні мережі дуже добре адаптувались під онлайн бізнес. Аудиторія з самого початку вже є розділеною за своїми бажаннями. Створились ідеальні умови. Залишається тільки врахувати всі потреби споживача, та налагодити під них інструменти автопостингу.

Соціальні мережі проводили експерименти з алгоритмами показу контенту [2]. І вони є більш ніж виправдані. Соціальні мережі тепер показують найбільш підходящий матеріал для просто користувача, відфільтрувавши його перед цим. Якби цього фільтрування не було, прості користувачі не встигали б бачити перед собою саме важливий контент, інтересні пости т матеріал, призначений саме для них. Стрічка новин була б просто перевантажена інформацією, як потрібною, так і зайвою. Кожна соціальна мережа по різному фільтрує свою стрічку новин, адже кожна з них переважно має свою аудиторію та розрахована на певний напрямок трохи більше, ніж інші.

Багато хто тепер фільтрує стрічку новин, але першими це зробила соціальна мережа Фейсбук [2]. Тут проблема зменшення охоплення сервісами автопостингу розглядається найбільш глобально.

Оброблюючи контент користувачів, Фейсбук в основному орієнтується на:

- коментарі;
- пости друзів та груп певного користувача;
- тривалість перегляду певного матеріалу;
- кліків на сторінках.

Саме тому часткову втрату деякої частини аудиторії чи контенту можна віднести до самих соціальних мереж. Аналіз усіх вище перерахованих і не тільки даних має певний вплив на те, що користувач буде бачити у свій стрічці новин.

Також потрібно підігрівати інтерес користувачів саме для вас, давати корисну та цікаву інформацію, потрібну для них. Не потрібно публікувати тільки товарний матеріал або рекламні пости.

Для просто того посту зазвичай не стає простого тексту. Він завжди буде збирати менше уваги. Для цього потрібно використовувати картинки, але і їх потрібно підбирати. Також для зручності та аналізу можна відстежувати публікації, успішніші за інші [6]. Деякі з них можна буде використати знову для привернення уваги аудиторії.

Існує дуже багато всього, для чого можуть бути корисні сервіси автопостингу. Але саме економія часу простого користувача є головною його перевагою.

Можливість підібрати один матеріал, та автоматично публікувати його на всіх доступних чи вибраних соціальних мережах, розкрутити свій проект чи пост на всіх доступних ресурсах [6].

Також корисною є статистика. Багато теперішніх сервісів автопостингу надають можливість подивитись звіт по вже зробленій роботі [6]. Є доступниими вік та стать цільової аудиторії, чому вона надає перевагу та час найбільшої її активності.

Тепер існує багато додатків, які мають тільки мобільну версію. Однією з таких соціальних мереж є Інстаграм. Сервіси автопостингу дозволяють працювати з нею через комп’ютер, використовуючи для цього спеціальні утиліти, без необхідності у смертфоні.

1.2 Аналіз ринку для сервісу автопостингу

Таких компаній, які б зараз не перебували в мережі Інтернет, тепер вже немає. Зараз усі пробують просуватись вверх по кар’єрних сходах через соціальні мережі.

Це все сталось через те, що ніхто не хоче втрачати таке хороше джерело доходу та трафіку, тому що зараз більшість популярних соціальних мереж стали повноцінними платформами з маркетинговими каналами.

Зараз буде розглянуто статистику по сервісах автопостингу:

- 3.5 мільярдів людей використовують зараз соціальні мережі;
- 95 відсотків цієї аудиторії віком від 14 до 35 років слідкують за брендами у соціальних мережах;
- 90 відсотків споживачів використовують соціальні мережі для перегляду та оцінки брендів;
- 90 відсотків брендів є у соціальних мережах, та, частіше за все, ведуть декілька сторінок, переважно від 4 до 10;
- в середньому на соціальні мережі та месенджери тратиться приблизно 2.5 години;
- 70 відсотків споживачів, чи потенційних клієнтів, які взаємодіяли з аккаунтами брендів та залишились задоволені, будуть рекомендувати їх друзям [7].

Так, як потрібно публікувати якісний заздалегідь підготований матеріал і слідкувати та обробляти коментарі, відповідати на вхідні листи, мотивувати простих підписників на активність і просто красиво та грамотно оформити свої сторінки чи профілі груп, ринок в даній області величезний.

Дуже часто звертаються до спеціалістів, щоб вони допомогли з сервісами автопостингу і ті взяли на себе частину роботи, так як для одної людини це складає велику проблему.

Великою популярністю користуються сервіси автопостингу, які дозволяють працювати з декількома соціальними мережами одразу, так як комплексне рішення проблеми завжди є кращим.

Функціонал комплексного сервісу автопостингу є достатньо широким. У особливих випадках це маже бути навіть автоматична генерація контенту для соціальної мережі, але в більшості випадків це публікування вибраного

матеріалу, його імпорт з зовнішніх ресурсів, планування публікації поста у вибраних соціальних мережах [2].

Дані сервіси автопостингу збирають всі потрібні функції та настройки в одному місці, чим значно спрощують задачу спеціалістам та економлять їхній час.

Так як ринок даної продукції зараз достатньо великий та активно розвивається, я порівняю кілька найбільш популярних з них.

Першим з даних сервісів автопостингу я розгляну NovaPress Publisher [8].

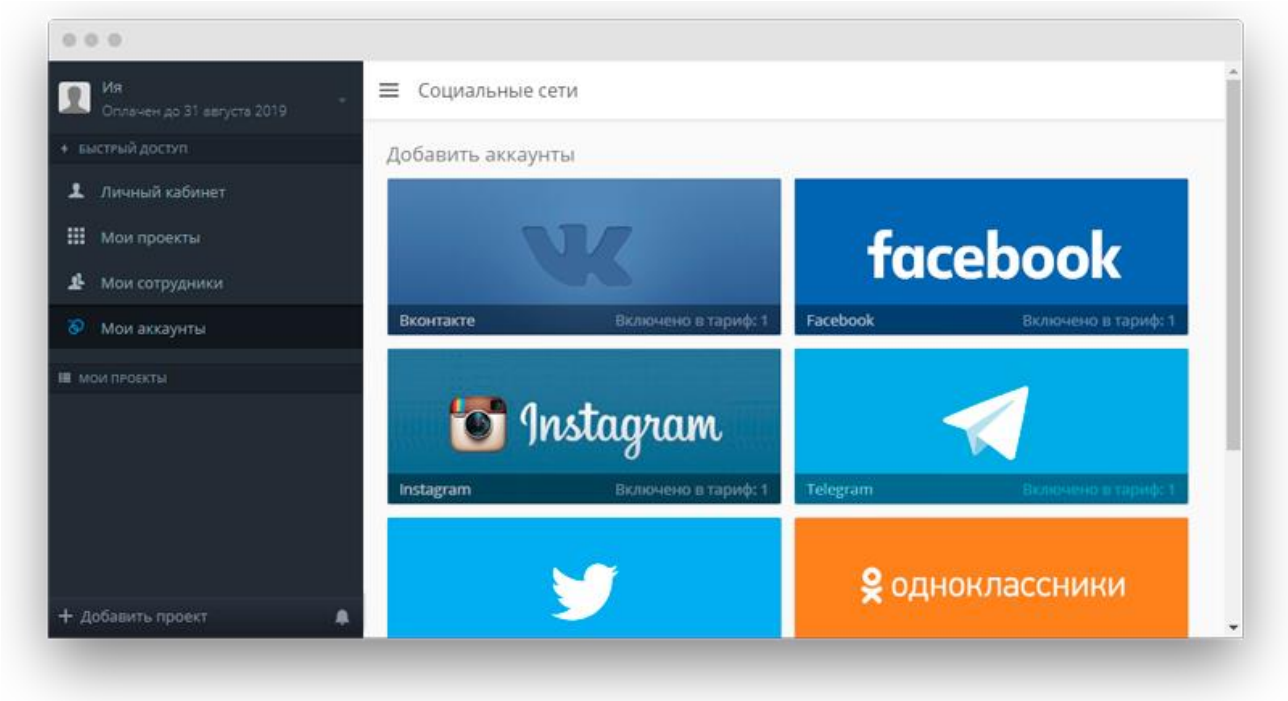

Рисунок 1.1 – Головна сторінка автопостингового сервісу NovaPress Publisher

Даний сервіс є дуже зручним. Його зручність полягає в тому, що він створений для тих, хто створює контент для груп в соціальних мережах та черпає інформацію з різних сайтів чи блогів. Функціонал у даного сервісу не дуже великий, але він компенсує це тим, що дозволяє повністю автоматизувати у себе розміщення контенту від брендів. Наприклад:

- автоматичне виставлення хештегів;
- автоматичне виставлення водяних знаків на фото;
- імпортування записів у соціальні мережі;
- комплексна публікація матеріалу;
- гнучке налаштування розкладу публікацій.

Даний сервіс автопостингу є повністю автоматизованим. Реєстрація проходить стандартно, через заповнення конкретних форм. Самі пости виходять в групи однаковими, але створюється враження, що даний контент є брендовим.

Також даний сервіс підтримує досить багато соціальних мереж, серед який є такі, як: Інстаграм, Фейсбук та інші.

Ціна підписки на даний сервіс є близько 130 гривень за місяць на одну сторінку. Також є можливість додавати соціальні мережі за приблизно 30 гривень. Для випробовування сервісу новим користувачам дається безплатний пробний період у вигляді 10 днів користування сервісом [9].

Наступним сервісом автопостингу, який я розгляну, буде SMM Aero [10].

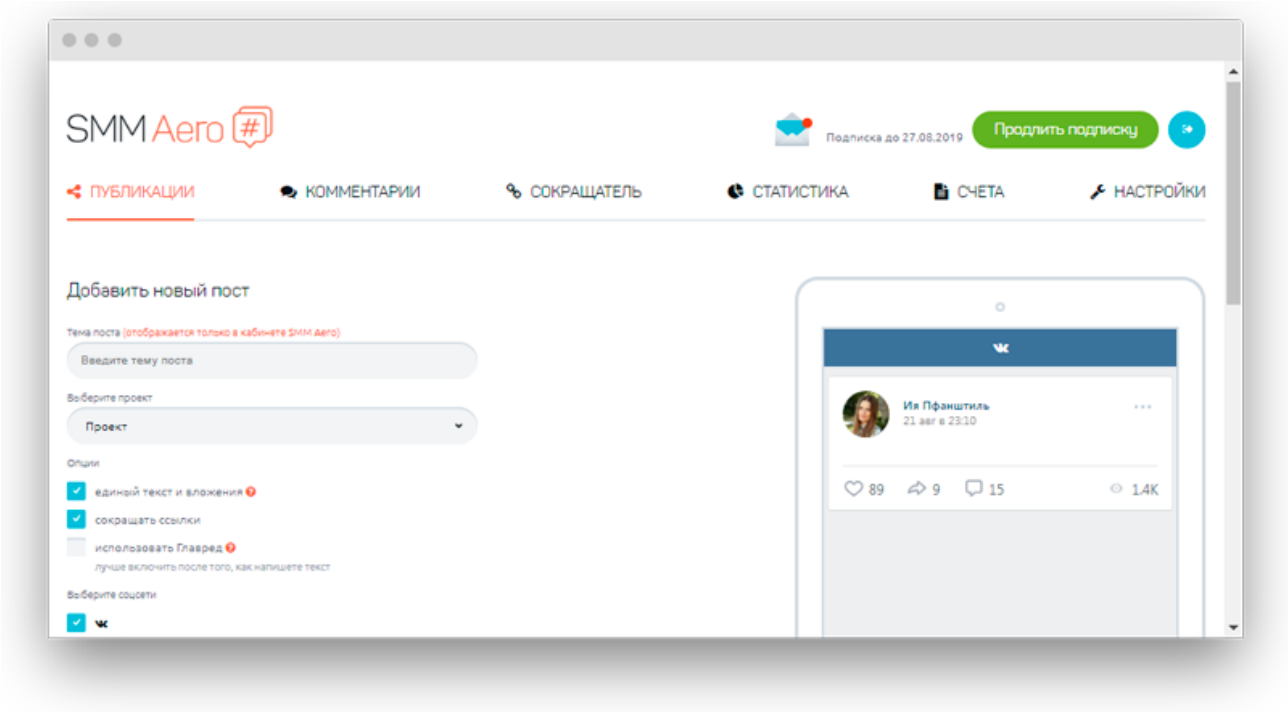

Рисунок 1.2 – Головна сторінка автопостингового сервісу SMM Aero

Даний сервіс автопостингу буде зручний для тих, хто активно відповідає на питання своїх підписників та активно взаємодіє з аудиторією, тобто маркетологам, бізнесменам та простим блогерам, які в теперішній час активно набувають популярність. У нього є такі функції:

- створювати пости, які будуть публікуватись миттєво, чи пости, які будуть публікуватись в певний вибраний заздалегідь час;
- перевіряти текст свого поста у редакторі;
- автоматичне скорочення посилань на сторонні ресурси;
- зручне SMM-планування;
- налаштування видалення постів;
- дуже корисна функція «чорновика»;
- рекомендація найкращого часу для обраної публцікації;
- об’єднання вибраного матеріалу у проекти для зручності.

Даний сервіс автопостингу підтримує тільки чотири соціальні мережі: Вконтакті, Інстаграм, Однокласники та Фейсбук.

Також даний сервіс є одним з найдешевших — всього 90 гривень на місяць за один проект. За ці гроші можна вести по одному аккаунту у кожній соціальній мережі, що пропонує даний сервіс. Як і всюди, тут є безплатний пробний період, тривалістю сім днів, які не обмежують функціонал даного сайту [9].

Наступним сервісом буде Ампліфер [11]. Це дуже авторитетний сервіс, який користується попитом багатьох відомих компаній.

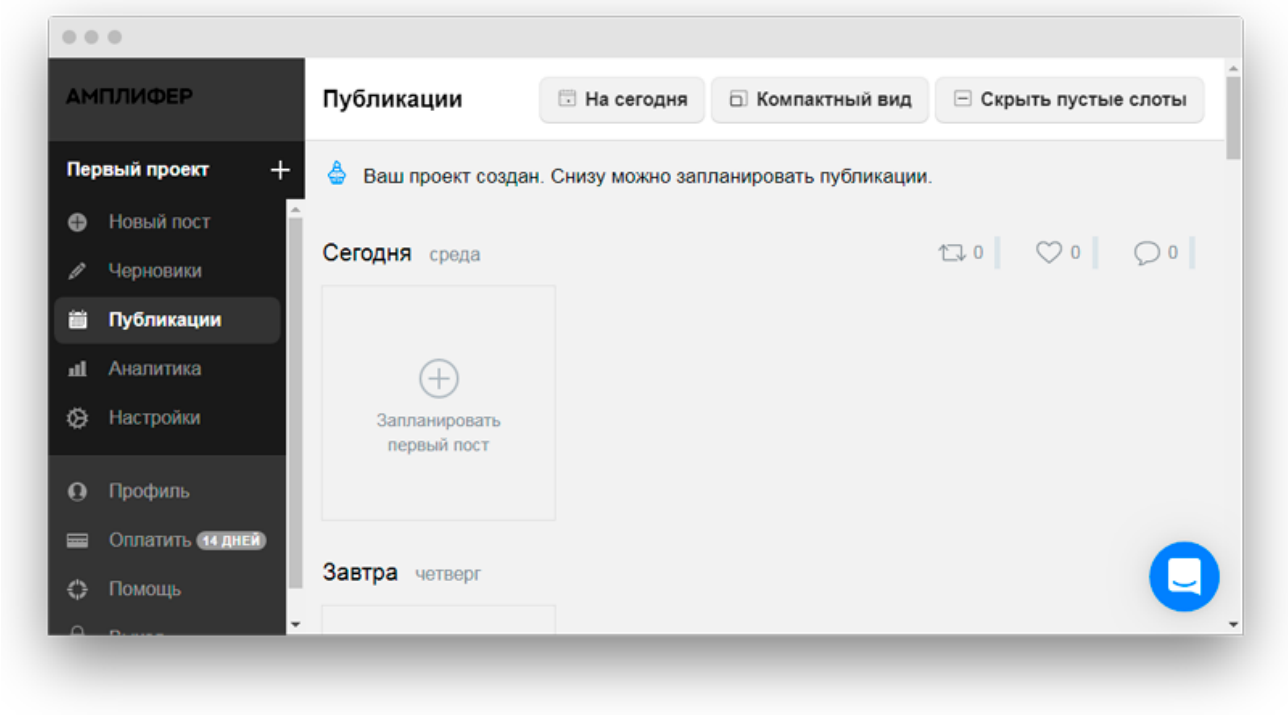

Рисунок 1.3 — Головна сторінка автопостингового сервісу Ампліфер

Даний сервіс автопостингу підходить тим, хто має кілька окремих сторінок в соціальних мережах чи веде різні проекти. Головною особливістю Ампліфера є те, що його вбудований алгоритм дозволяє підібрати найкращий час для публікації вашого матеріалу. Сервіс аналізує, коли краще розміщувати пости, а також ваш розклад їх розміщення.

Функції в даного сервісу такі:

- присутній чорновик;
- зручне SMM-планування;
- також тут присутній унікальний інструмент, який носить назву автопілот, та може доповнювати ваш контент-план контентом, який буде найбільш перспективний.
- присутні шаблони;
- імпорт з зовнішніх ресурсів;
- можливість працювати, як самому, так і з командою, також надаючи їм певні права доступу;

- об’єднання в проекти;
- скорочення силок на інші ресурси;
- аналіз конкурентів;
- написання різного тексту для різних соціальних мереж.

Даний сервіс підтримує цілих дев’ять соціальних мереж. Це такі основні мережі, як Вконтакті, Фейсбук, Інстаграм та інші.

Але даний сервіс є також одним з найдорожчих. Для того, щоб обслуговувати одну сторінку в соціальній мережі, потрібно біля п’яти доларів, тобто близько 110 гривень. Варто перемножити це на кількість сторінок та соціальних мереж, щоб отримати суму оплати на місяць. Тут також присутній безплатний пробний період у вигляді двох тижнів повного користування [9].

Наступний сервіс, який буде розглянуто, носить назву Kuku.io [12].

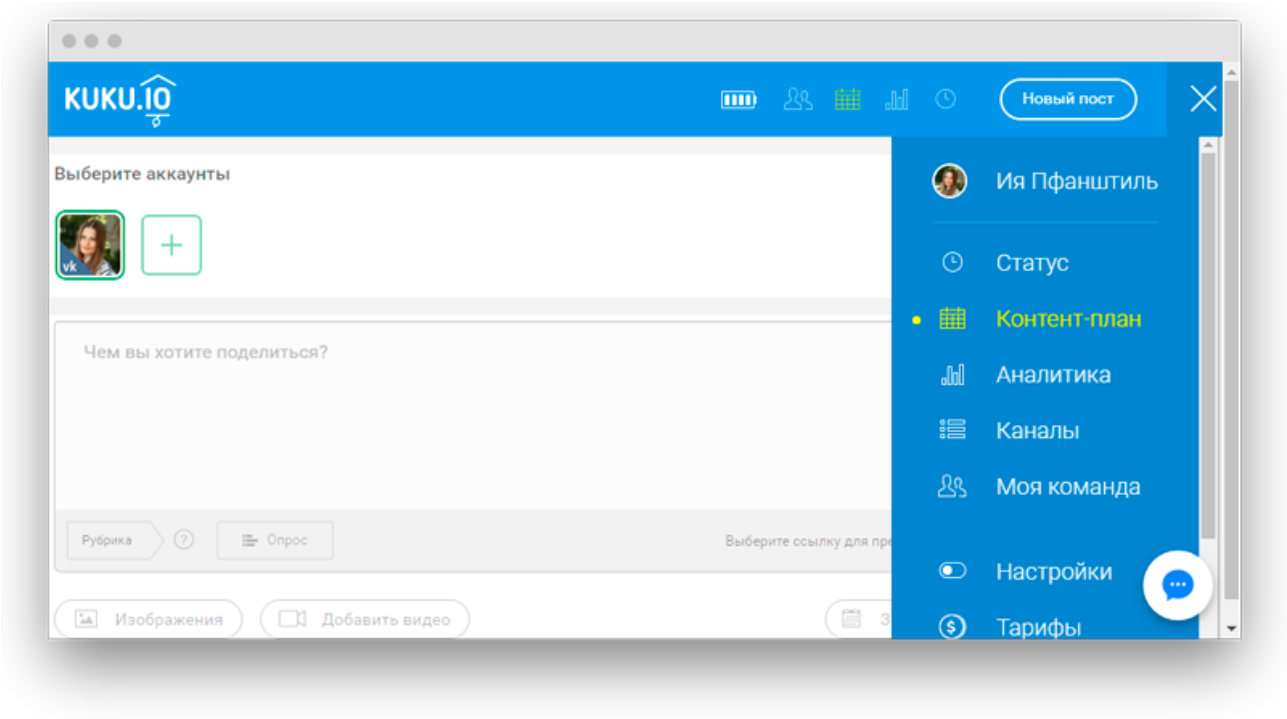

Рисунок 1.4 – Головна сторінка автопостингового сервісу Kuku.io

Даний сервіс є міжнародним інструментом, який буде допомагати планувати публікації у соціальних мережах.

У даного сервісу достатньо велика інтеграція з соціальними мережами. Його функціоналом є:

- скорочення силок;
- адаптація під вигляд соціальних мереж;
- показ статистики;
- кроспостинг.

Даний сервіс є достатньо дорогим. Він обійдеться у 7 доларів щомісяця. Також присутній безплатний пробний режим у вигляді надання можливості 30 публікувань протягом двох тижнів користування [9].

Останнім розглянутим сервісом буде Publbox [13].

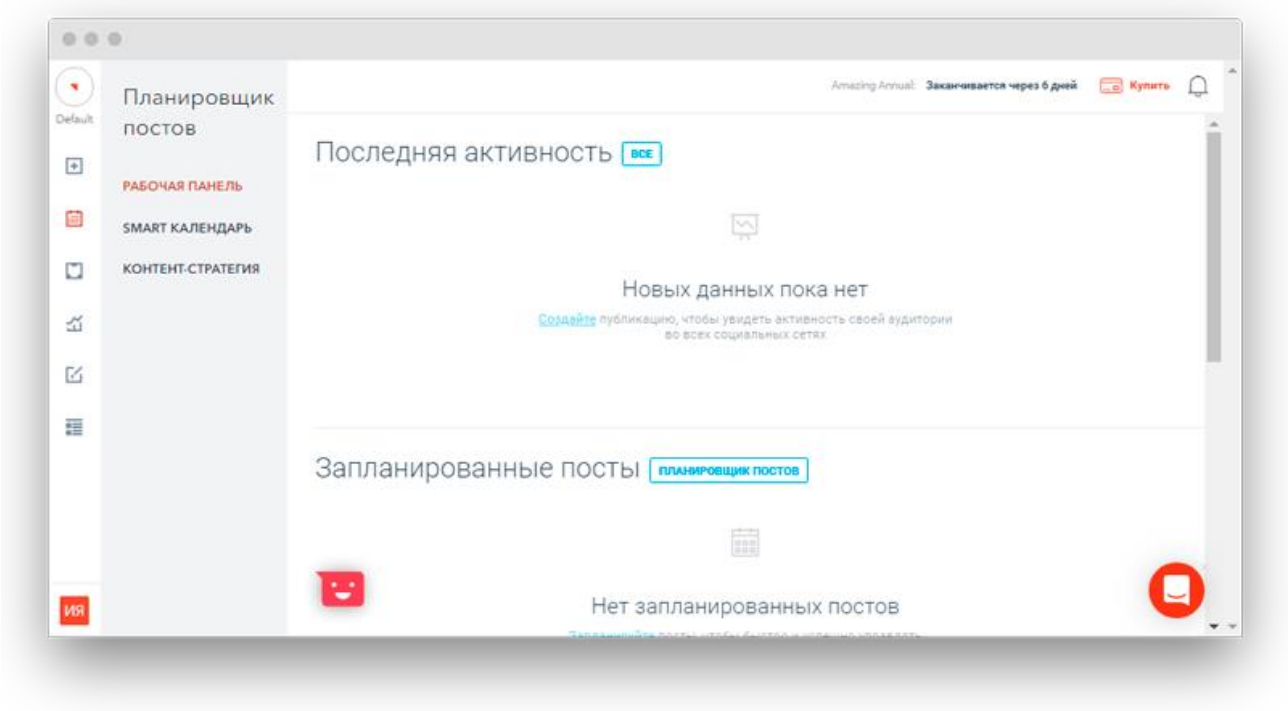

Рисунок 1.5 – Головна сторінка автопостингового сервісу Publbox

Даний сервіс пропонує навчальні курси по SMM. Також він є зручним для тих, хто працює з багатьма клієнтами та вимушений часто їх змінювати. Даний сервіс розроблений спеціально для малого бізнесу. Також особливістю даного проекту є те, що у нього присутні особливі умови співпраці з громацькими або благодійними компаніями та організаціями.

Функціонал у даного сервісу такий:

- присутня статистика;
- зручне планування;
- скорочення силок;
- присутні шаблони;
- відкладений та миттєвий постинг.

Підтримує сумісність з достатньою кількістю соціальних мереж. Серед них Фейсбук, Інтаграм та інші.

Даний сервіс буде обходитись у 9 доларів щомісячно. Існує можливість підключити тариф на рік. Тоді ціна складатиме 5 доларів у місяць. Присутній тестовий період, який триває один тиждень. Також існує можливість безплатної публікації 2 постів в тиждень [9].

1.3 Постановка задачі дипломного проекту

Після проаналізованих даних можна зробити висновок, що більшість якісних сервісів є дуже дорогими і є не по кишені простому користувачеві. Плюс, майже усі з них є англомовними, трохи менше російськомовними.

У даної дипломної роботи основною задачею є розробка та створення бота, тобто скрипта для полегшення публікації матеріалу для університету та для себе.

Для того, щоб все відбувалось коректно, вхідними даними повинні бути логіни та паролі від усіх соціальних мереж, для яких буде публікуватись контент, та сам матеріал: текст, картинки, фото, силки та хештеги.

Даний бот опублікує підготований матеріал у задані соціальні мережі.

Вихідними даними буде повідомлення про те, що даний пост чи матеріал в цілому був успішно викладений у соціальну мережу.

Щоб створити даного бота для автопостингу, потрібно:

- розробити архітектуру самого бота;
- реалізувати усі необхідні для нас функції;
- налаштувати його для правильної роботи та безперебійного функціонування.

З плюсів даного бота можна буде відмітити такі, як:

- економія часу;
- простота використання;
- зручність використання;
- безплатне користування, на відмінну від більшості його аналогів.

З мінусів можна відмітити тільки відносно менший функціонал, в порівнянні з дорогими аналогами.

2 ОПИС ОСНОВНИХ ВИМОГ ТА ХАРАКТЕРИСТИК БОТА ДЛЯ СЕРВІСУ АВТОПОСТИНГУ

2.1 Функціональні вимоги бота для сервісу автопостингу

Для того, щоб розпочати проектування потрібного для нас сервісу автопостингу, а в особливості, бота, потрібно визначитись, які будуть вимоги до нього та початкові дані, які він буде приймати. На основі аналізу цих пунктів можна отримати структурні схеми та діаграми даного продукту, а також його текстовий опис. Також при розробці архітектури буде загальна модель поведінки даного програмного продукту при контакті з зовнішніми впливами, та будуть визначені його головні функції.

Сам етап постановки задачі є чи не найголовнішим та найвідповідальнішим при створенні даного сервісу та і взагалі в цілому. Саме на цьому етапі обговорюються функціонал даного бота, та обмеження, які можуть чи будуть на нього накладені. Також вибирається середовище розробки скрипта, вигляд його архітектури та інтерфейсу. Саме від цього буде залежати кінцева вартість та якість, з якою продукт буде виготовлений.

Вимоги для програмного забезпечення є дуже важливою річчю. Це може бути набір певних вимог стосовно властивостей та функціоналу для програмного продукту, який є у розробці чи буде розроблятись [14].

Вимоги для даного бота будуть визначені при їх аналізі, та будуть зафіксовані в специфікації вимог. Також будуть визначені діаграми, які допоможуть з розробкою бота.

Розроблення вимог для даного проекту буде розділена на кулька певних етапів:

- визначення самих вимог, тобто те, який вигляд даний проект мав би мати;

- аналіз даних вимог, тобто перевірка їх на правильність, закінченість та цілісність;
- специфікація вимог, тобто документування визначених вимог для даного проекту;
- тестування вимог [14].

Функціональні вимоги є дуже зручними, адже вони описують те, що повинно надаватись програмним середовищем. Також описують його типову поведінку та поведінку у певних ситуаціях, реакцію на різні вхідні дані. Ще тут повинна бути описана реакція системи на деякі з боку користувача та дії, які вона дозволить йому виконувати [14]. Сюди може бути додана також інформація про те, як система вести себе не повинна.

Усі програмні проекти повинні виконувати певні заздалегідь визначені функції. Потрібно мати чітку критерії, за допомогою яких можна буде правильно визначити, чи підходить даний програмний продукт для розв’язання тієї чи іншої задачі.

Для написання функціональних вимог для даного бота потрібно врахувати те, що чим більш детально їх буде описано, тим точнішою буде оцінка роботи по термінах розробки та вартості для розроблення програмного забезпечення перед виставленням завдань та створенням самого бота, щоб на наступному етапі розробки не потрібна була додаткова розробка.

Таблиця 2.1 – Функціональні вимоги для бота

Функціональна можливість	Опис
Виклик функції з веб додатку	Можливість виклику вебсервісом необхідної функції для початку процесу публцікації
Отримання даних для входу	Отримання певних даних для можливості здійснити вхід у вибрану соціальну мережу на вибраний профіль

Кінець таблиці 2.1

Отримання даних для публікації	Отримання певного матеріалу, тексту та фото для можливості здійснити публікацію у вибраній соціальній мережі на вибраному профілі
Здійснення входу на акаунт	Можливість здійснити вхід у вибраний обліковий запис певної соціальної мережі
Перехід на стіну профіля	Можливість після входу на акаунт перейти на стіну профіля для здійснення публікації
Перехід на стіну групи	Можливість після здійснення входу на вибраний акаунт переходу на стіну вибраної групи для здійснення публікації
Публікація даних	Можливість використати отримані раніше дані для вставлення їх у відповідні поля та створення публікації
Вихід з акаунту	Здійснення виходу з акаунту користувача після завершення усіх необхідних завдань
Сповіщення про успішну публікацію чи помилку	Сповіщення веб додатку про успішну публікацію чи серію публікацій на вибраних сторінках та завершення роботи на них

2.2 Нефункціональні вимоги бота для сервісу автопостингу

В даному розділі описані нефункціональні вимоги для розробленого скрипта.

Не функціональні вимоги – це такі вимоги для програмних продуктів, які будуть визначати, як буде оцінюватись якість роботи цих продуктів. Ці вимоги визначають, яким даний продукт мав би бути, а функціональні вимоги – що він повинен робити. Ці вимоги повинні бути визначені ще на першій стадії розробки програмного забезпечення, а саме на стадії аналізу вимог [16].

Нефункціональні вимоги є описом для певних обмежень та залежностей, у яких повинні виконуватись функції.

Дані вимоги поділяються на:

- вимоги до продукту, тобто , наприклад можуть бути використані тільки певні пристрої;
- вимоги до процесу, тобто, наприклад даний проект повинен працювати тільки за певними стандартами;
- різні зовнішні вимоги, тобто будь які зовнішні фактори, які впливатимуть на дієздатність програмного забезпечення, такі як використання тільки певної бази даних та інші [16].

Для даних вимог не підходить формулювання «ефективний», «зручний» чи «безпечний» так, як дані вимоги не можливо перевірити. Має існувати метод їхньої перевірки, тобто визначення того, що вони виконуються. Тому вони повинні бути кількісними.

Для програмних продуктів можна виділити такі нефункціональні вимоги, як:

- правильність отриманих даних – даний програмний продукт повинен на виході давати правильні результати для будь яких вхідних даних з наперед визначеного діапазону допустимих значень. Якщо програмний продукт є достатньо великим, хороший програміст навіть з першого

разу не зможе написати його без помилок. Дані помилки є різного характеру: від механічних (неправильний синтаксис, банально забув покласти крапку з комою, дрібні помилки) до неправильно побудованого алгоритму виконання чи неправильного використання функціоналу програмного продукту;

- ефективність використання – розроблений програмний продукт має бути не дуже ресурсо-затратним та видавати результати своєї роботи за прийнятний час. Різні проекти, є призначеними для вирішення однакових завдань, можуть справлятись з ними по різному. Потрібно надавати перевагу тим, хто за однакових умов ефективніше виконує свою роботу;
- надійність використання – програмний продукт повинен бути надійним. Простий користувач не повинен боятись використовувати його. Він має довіряти програмному продукту. Також дане програмне забезпечення повинне бути правильним. Навіть якщо помилки залишились, вони не мають бути серйозними, приводити до серйозних збоїв. Але і цього замало. Ще програма повинна реагувати на ввід неправильних даних з боку користувача та повідомляти його про це, тому що якщо програма прийме ці дані, та почне з ними працювати, це може вилитись у ще гірші наслідки відносно того, що програма б взагалі не працювала. Використання програмного продукту не повинно призводити до будь яких поганих наслідків;
- універсальність – програмний продукт повинен мати широкий спектр вхідних даних та, по можливості, мати широкий функціонал ля полегшення роботи;
- функціональність програмного продукту – дане програмне забезпечення повинно мати весь необхідний функціонал, щоб задовольняти усі головні потреби. Також для програмний продукт не повинен містити в собі тих функцій, які користувачу не потрібні;

- зручність – програмне забезпечення має бути зручним у використанні. Також інтерфейс програмного продукту повинен бути зрозумілим на інтуїтивному рівні, так як за останній час цей напрямок дуже сильно розвинувся;
- стандартизованість – розроблене програмне забезпечення повинно бути схожим до аналогічних, тобто мати схожий інтерфейс та подібні засоби керування. Це повинно бути зроблено для того, щоб певні користувачі, які вже користувались схожим програмним забезпеченням могли з найбільшою зручністю перейти до роботи з новим;
- переносимість – можливість переносу програмного забезпечення з одної машини чи хостингу на інші, при цьому змін не повинно бути взагалі чи вони повинні бути мінімальними;
- читабельність програмного забезпечення – код програмного забезпечення повинен бути написаний максимально просто та читабельно, щоб інші спеціалісти могли при потребі в цьому розібратись;
- модифікованість програмного забезпечення – програмний продукт повинен бути розроблений таким чином, щоб в ньому передбачалась можливість розширення функціоналу, загальних змін та деяких доповнень при потребі;
- документованість програмного забезпечення – кожне програмне забезпечення повинно мати інструкції стосовно свого використання. Також дані інструкції мають бути зрозумілими. Сам код повинен бути описаний коментарями для кращого розуміння та доступності. Вони повинні описувати суть функціоналу та основних елементів [16].

Також є різні програмні помилки, такі як:

- синтаксичні помилки – програмні помилки, які виникли після порушення синтаксису певної мови. Дані помилки легко

виправляється, тому що їх виправленням займаються на етапі компіляції;

- непередбачуване завершення роботи програми – програмне забезпечення виявивши операції, що не можливо виконати, такі як ділення на нуль подібні, припиняє свою роботу;
- зависання програмного забезпечення – програмний продукт не може дійти до кінця, заходить в нескінченний цикл або просто зависає;
- видача неправильних результатів роботи програми [16].

Перераховані помилки мають бути виявлені в процесі відлагодження та виправлені після цього. Також потрібно переконатись, що розроблене програмне забезпечення справно працює. Для цього його тестують, а цей процес називається тестуванням. Саме ці два процеси складають велику частину роботи над програмним забезпеченням.

На основі цієї інформації було описано нефункціональні вимоги для розробленого скрипта:

- даний бот отримує дані від веб додатку з бази даних. Після чого входить в профіль та публікує дану інформацію. Якщо ж дані не вірні, він припинить свою роботу, а веб додаток попросить ввести коректні дані для входу;
- бот працює достатньо швидко та не є сильно ресурсо-затратним;
- так як даний скрипт є частиною веб додатку, він є надійним, так як працює тільки з тими даними, які постачає йому веб додаток. Якщо ж дані неправильні, скрипт припиняє свою роботу, а веб додаток в свою чергу видасть помилку та прохання ввести правильну інформацію;
- розроблений бот може приймати в себе всю необхідну для своєї роботи інформацію, таку як текст, фото, хештеги;
- функціонал розробленого скрипта повністю задовільняє всі потреби користувача. Там є всі необхідні для сервісу функції. Також відсутніми є функції, які для публікації матеріалу не потрібні;

- сам скрипт не буде доступний користувачу, але веб додаток розроблений у відповідності з схожими сервісами для більшої зручності та для того, щоб користувач інтуїтивно міг в ньому розібратись. Інтерфейс та засоби керування є схожими до відповідних сервісів автопостингу;
- розробленого бота легко перенести на будь який інший хостинг. Для цього потрібні лише мінімальні налаштування сервісу;
- читабельність коду є достатньо високою. Змінні названі коротко та правильно;
- даний скрипт можна легко модифікувати в майбутньому. Ніяких проблем з цим виникнути не повинно;
- дане програмне забезпечення не є документованим. Але вибрані назви функцій передають головну їх суть, тому розібратись в коді буде легше.

2.3 Короткий аналіз основних характеристик бота для сервісу автопостингу

Для того, щоб проаналізувати, які головні вимоги існують для програмного забезпечення та якими характеристиками він має мати є достатньо багато джерел. Основними з них є:

- спостереження – достатньо ефективне джерело. Спостерігаючи за схожим програмним забезпеченням можна отримати необхідні дані про його основні характеристики, функціонал та необхідні моделі поведінки. На основі цих даних можна побудувати свої моделі та скласти власний функціонал або покращити з вже наявні;
- анкетування – опитування певної аудиторії, на яку розраховане програмне забезпечення, з метою покращити його роботу. Дане

джерело є одним з найменш достовірних та працюючих, так як в теперішніх реаліях аудиторія є часто не заінтересованою в заповнені анкети, або, що ще гірше, в правильному її заповненні. Також часто аудиторія є просто не компетентною;

- інтерв'ю – цей метод використовується при роботі з замовником для того, щоб дізнатись, яке програмне забезпечення йому потрібно. Це обговорення моделей поведінки, можливого функціоналу та інших деталей;
- прототипування – дане джерело є створенням моделей поведінки програмного продукту, його працюючих прототипів, які зможуть в повній мірі виконувати більшість поставлених задач. Також після цього це передбачає редагування програмного забезпечення та доведення до потрібного нам вигляду [17].

Обробивши дану інформацію та прийнявши її до уваги оптимальним рішенням було використати тільки два з вище перерахованих, так як є можливість спостереження за аналогічними програмами та складення на основі цього своїх моделей та функціоналу. Решту методів є неефективними в даній ситуації, так як прямого замовника немає як і цільової аудиторія, яка навіть за наявності не була б компетентною в даному питанні.

Для того, щоб була можливість представити моделі та алгоритми роботи програмного забезпечення є діаграми UML. Це уніфікована мова, яка використовується для моделювання [18]. Вона може запропонувати декілька видів діаграм, серед яких:

- use Case Diagram – діаграма, яка показує, як той чи інший користувач чи об'єкт може використовувати систему. Головної її метою є показати функціонал та можливості програмного забезпечення і дати загальні принципи, як програму можна використовувати [18];

- sequence Diagram – діаграма, яка показує послідовності. Вона відображає взаємодію певних об’єктів, враховуючи при цьому час їх використання, тобто послідовність дій [18];
- activity Diagram – діаграма, яка показує, як може вести себе програмне забезпечення. Візуально показує діяльність, яку може мати програмний продукт [18];
- statechart Diagram – ця діаграма показує стани, у яких може перебувати програмне забезпечення [18];
- class Diagram – є діаграмою класів. Представляє програмне забезпечення як статичну структуру моделей. Прикладом такої діаграми є діаграма моделі бази даних [18];
- timeline Diagram – діаграма, яка відображає, як то чи інше програмне забезпечення повинно вести себе крок за кроком. Також може показувати, як воно буде вести себе з плином часу [18].

Розглянувши дану інформацію було вибрано та реалізовано кілька з цих діаграм для представлення моделі розробленого бота, а саме:

- покрокова діаграма роботи розробленого скрипта для відображення роботи його функціоналу(Timeline Diagram);
- діаграма послідовностей. Вона відображатиме, як даний бот взаємодіє з іншими об’єктами та коли і як має бути застосований(Sequence Diagram);
- діаграма активностей, яка показує, який функціонал буде мати розроблений скрипт(Activity Diagram).

2.4 Timeline diagram бота для сервісу автопостингу

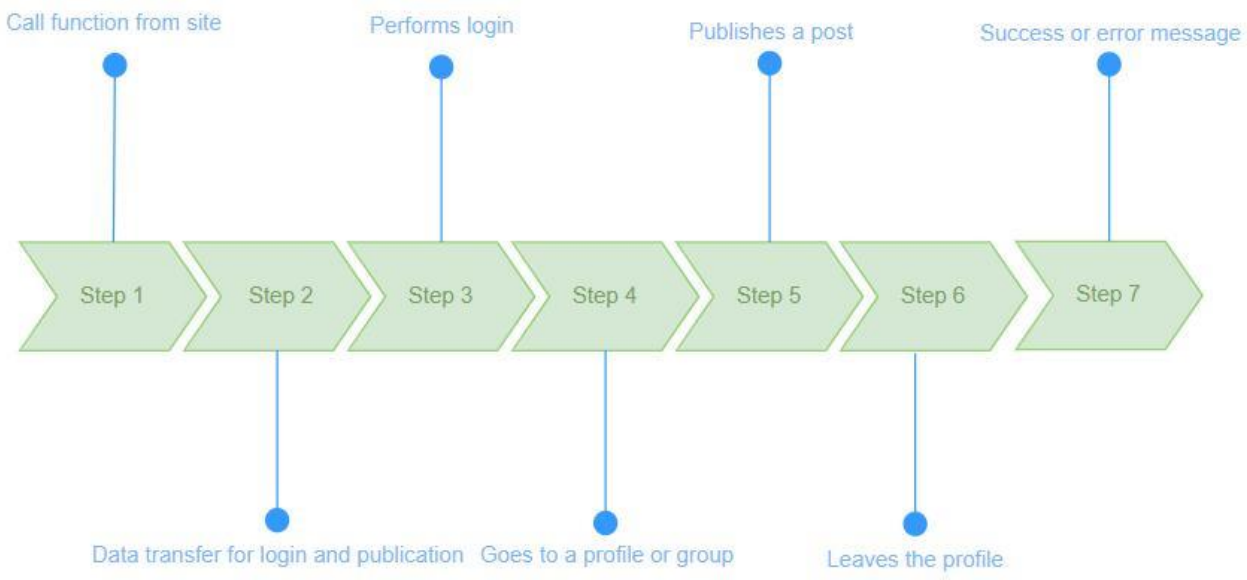

Рисунок 2.1 – Timeline diagram. Покроковий алгоритм взаємодії веб додатку з ботом

Дана діаграма(див. рисунок 2.1) показує, як виглядає покроковий алгоритм взаємодії веб додатку з розробленим скриптом. Даний алгоритм дуже чітко та покроково демонструє, що бот буде робити при своєму виклику.

Дана схема складається з семи кроків, а саме:

1. Call function from site(виклик бота з веб додатку) – даний скрипт має бути викликаним з веб сервісу для початку своєї роботи, інакше процес публікації не почнеться;
2. Data transfer for login and publication(передача інформації для входу та публікації) – ця ланка показує, що з веб додатку буде передана інформація, збережена у базі даних, для того, щоб була можливітьс авторизуватись, перейти на певний профіль чи групу та опублікувати

заздалегідь підготований матеріал. Також, якщо це рекламна публікація, передання силки на пост для його видалення;

3. Performs login(виконання входу) – після отримання даних від веб додатку бот переходить на вибрану соціальну мережу та здійснює вхід в неї;
4. Goes to a profile or group(перехід до стіни профілю чи групи) – після входу даний скрипт повинен перейти на стіну профілю, на який виконано вхід, чи на стіну групи, на якій повинна відбутись процес публікації;
5. Publishes a post(публікація інформації) – процес публікування переданого веб додатком матеріалу у вибраній соціальній мережі на заданому профілі чи групі;
6. Leaves the profile(вихід з профілю) – операція виходу з поточного профілю соціальної мережі після завершення публікації матеріалу для завершення роботи бота;
7. Success or error message(повідомлення про успіх чи помилку) – останній пункт даної діаграми. Він повідомляє веб додаток про успішну публікацію отриманого матеріалу чи помилку на будь якому з ранніх етапів роботи.

2.5 Sequence diagram бота для сервісу автопостингу

Тепер розглянемо діаграму послідовностей(див. рисунок 2.2). Вона відображає взаємодію між певними об’єктами за часом. Також у ній зображуються лише ті об’єкти, які будуть взаємодіяти між собою.

Ця діаграма складається з деяких елементів, таких як:

- об’єкт – це елемент, який безпосередньо прийматиме участь у взаємодії. Має вигляд невеличких прямокутників, що зображені на діаграмі, та всередині має назву самих елементів взаємодії;
- стрілки та повідомлення – ці елементи описують певну взаємодію, що відбувається в певний час та прийом деякими об’єктами певної інформації, що в свою чергу запускає наступну взаємодію у життєвому циклі;
- фокус управління – цей елемент має вигляд вузького прямокутника, що зображений під деякими об’єктами, та проходить вздовж лінії життя цих об’єктів. Верхня частина цього елементу зображує початок дії цього елементу, тобто фокусу управління, а нижня частина є його закінченням. Цей елемент розміщується нижче вибраного об’єкта та замінює лінію життя там де це потрібно, а деколи тягнеться весь час роботи системи. Саме за допомогою цього елемента і можна відрізнити активну фазу життя об’єкта від пасивної;
- лінія життя – цей об’єкт є звичайною вертикальною пунктирною лінією. Ця лінія тягнеться тільки від певного елемента та показує, скільки цей елемент існуватиме в системі та коли братиме участь у її житті. Якщо ж вибраний елемент у системі існує весь час її функціонування, то даний об’єкт повинен продовжуватись по усій лінії життя цього об’єкта [18].

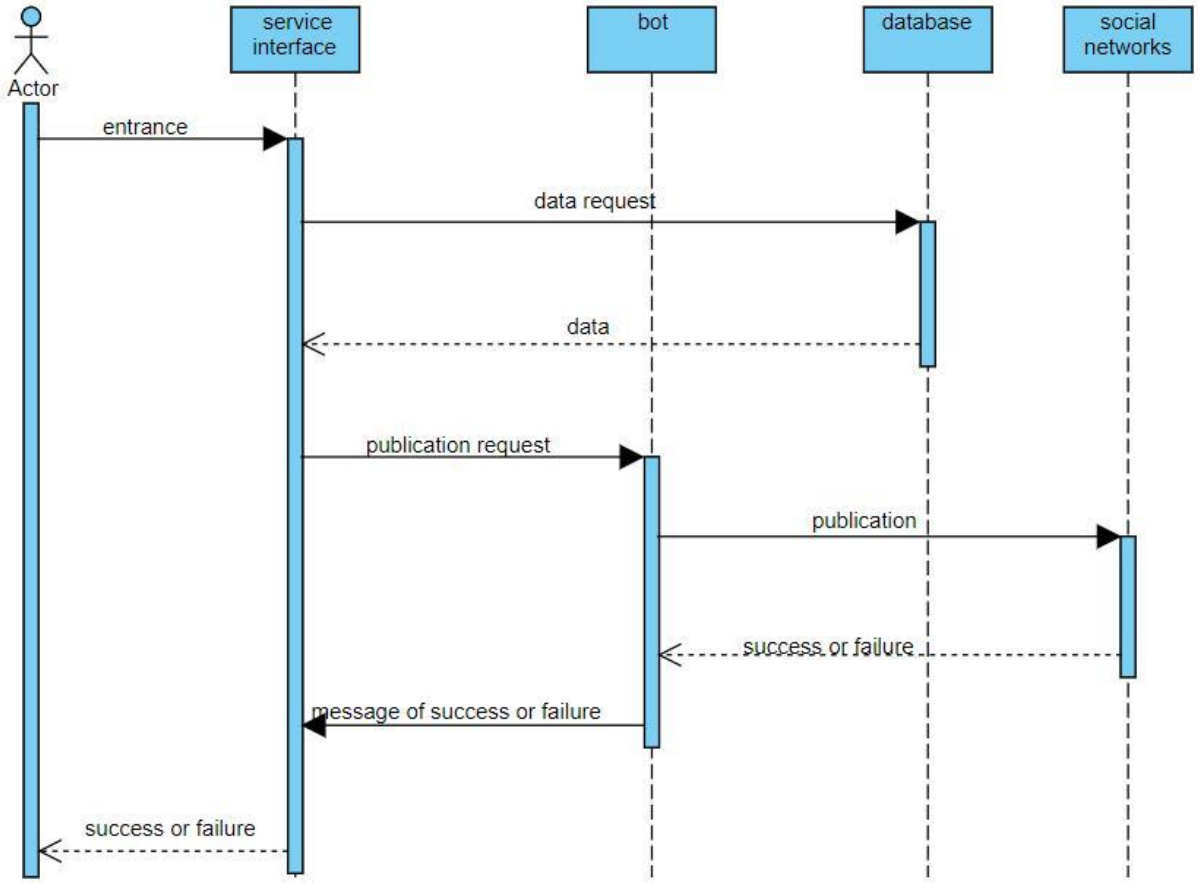

Рисунок 2.2 – Sequence Diagram. Діаграма послідовностей для відображення взаємодії об’єктів за часом

На цій діаграмі показано деякі наступні об’єкти, а саме:

- Service interface(інтерфейс веб додатку) – цей об’єкт є інтерфейсом веб додатку, з яким буде взаємодіяти користувач, а сам інтерфейс буде взаємодіяти з ботом;
- Bot(бот) – даний об’єкт є розробленим скриптом. Він отримуватиме дані від інтерфейсу веб додатку, які будуть введені користувачем та виконуватиме сам процес публікації, для якої і є призначений;
- Database(база даних) – цей об’єкт є базою даних. У ній зберігається уся введена користувачем інформація для подальшого збереження та використання її у майбутньому;

- Social networks(соціальні мережі) – цей об’єкт представляє з собою соціальні мережі, з якими буде взаємодіяти бот та публікувати на них отриманий матеріал.

Об’єкт інтерфейс має найдовшу лінію життя. Вона проходить через весь його життєвий цикл. Вона є такою тому, що користувач при користуванні сервісом в будь якому випадку стикається з інтерфейсом.

Спочатку користувач входить(entrance). Тепер він розпочав взаємодію з інтерфейсом. Для подальшої роботи, такої як додавання соціальних мереж чи створення відкладеної публікації веб додаток мусить взаємодіяти з базою даних. Саме для цього будуть здійснюватися запити до бази даних(data request) через веб додаток з метою створення публікації чи її редагування. Також для самої публікації потрібну інформацію(data) сервіс буде передавати боту з бази даних. Так як база даних більше використовуватись не буде, її життєвий цикл на цьому припиняє своє існування.

Після того, як веб додаток отримав потрібну інформацію, він передає її боту(publication request). Тут життєвий цикл бота розпочинається. Спочатку він публікує отриманий матеріал в соціальні мережі(publication). Після цього до цього повертається результат(success or failure) про успішне завершення публікації чи неможливість опублікування матеріалу.

Тут варто підмітити, що життєвий цикл соціальних мереж є дуже малим. Він існує тільки тоді, коли бот проводить публікацію. Також і життєвий цикл бота, після того як він отримав сповіщення, підходить до кінця. Він передає ці результати назад до інтерфейсу(message of (success or failure).

Після передачі результатів від бота до веб додатку користувач бачить результат. Він може бути успішний або може виникнути помилка при публікації. Це значить що на цьому життєвий цикл веб додатка закінчився, як закінчився і життєвий цикл системи загалом.

2.6 Activity diagram бота для сервісу автопостингу

У даному розділі буде розглянуто діаграму дій. На відмінну від інших діаграм, які описують роботу системи в загальних рисах, дана діаграма навпаки показує окремий варіант розвитку подій та кілька його можливих сценаріїв.

Основними елементами такої діаграми є:

- символи початку та кінця роботи програмного забезпечення;
- функції, тобто дії, що буде виконувати система;
- потоки управління;
- розгалуження, тобто кілька можливих варіантів розвитку ситуації;
- лінійки синхронізації [18].

Дана діаграма(див. рисунок 2.3) показує, яку активність буде проявляти система. Вона дозволяє показати, як система буде використовуватись з достатньо хорошою деталізацією [18]. Через те, що використання виконується поступово, використовується діаграма з лінійним потоком.

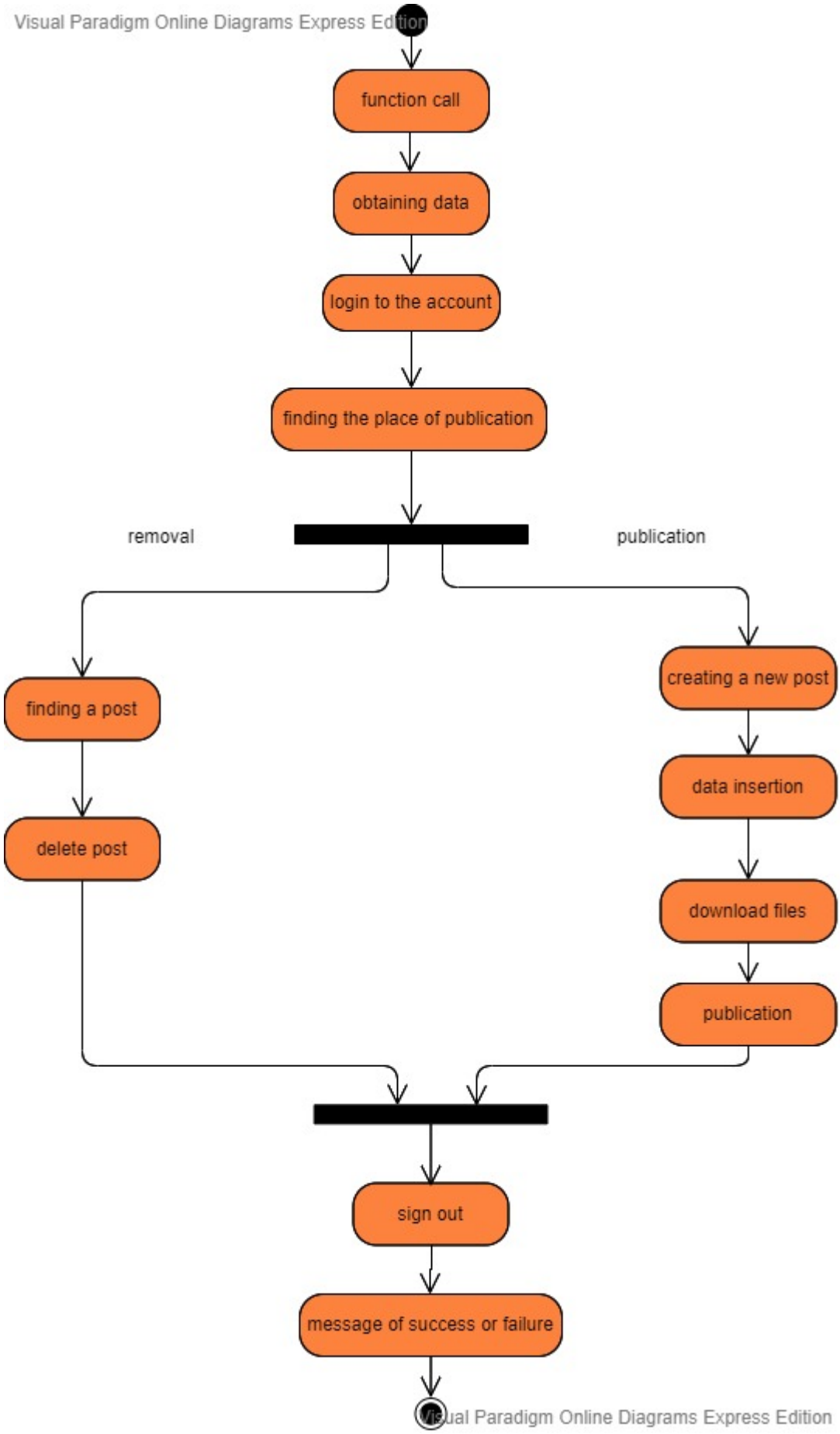

Рисунок 2.3 – Activity diagram. Діаграма дій для розробленого бота

Все починається з символу початку. Після цього скрипт викликається з веб додатку(function call) для початку своєї роботи. Отримує від нього дані на вхід та матеріал на публікацію(obtaining data). Після чого починає виконувати вхід у заданий профіль(login to the account). Виконавши вхід, скрипт мусить перейти на місце проведення публікації. Для цього він шукає стіну профілю або вибраної групи(finding the place of publication).

Після цього йде розгалуження, адже скрипт може як публікувати матеріал(publication), так і видаляти рекламні пости(removal).

Якщо мова йде про публікацію, бот створює новий пост у відповідному вікні(). Після чого вставляє всю необхідну інформацію, таку як текст, фото(download files) та хештеги у відповідні поля заповнення(data insertion). Після того, як все загружено та вставлено, відбувається сама публікація матеріалу(publication).

Якщо ж мова йде про видалення рекламного поста, бот шукає необхідну рекламну публікацію(finding a post) та після її знаходження видаляє її(delete post).

Після виконання своєї роботи бот покидає профіль, на якому знаходився(sing out), та сповіщає веб додаток про успішне завершення чи невдачу в публікації матеріалу(message of success or failure).

3 ВИКОРИСТАНІ ІНСТРУМЕНТИ ТА РОЗРОБКА ФУНКЦІОНАЛУ БОТА ДЛЯ СЕРВІСУ АВТОПОСТИНГУ

3.1 Використані інструменти для розробки бота для сервісу автопостингу та їх підключення

Для розробки даного бота було використано Selenium Webdriver [19] на мові програмування Python [20]. Це простий інструмент, який є драйвером браузерів. По суті, він просто максимально наближено імітує дії, які робив би користувач при використанні браузера.

Спочатку буде розглянуто драйвер. Драйвер – це по суті маленька програмна бібліотека, яка дозволяє різним програмам взаємодіяти між собою [19]. Тоді Webdriver є драйвером браузера, тобто програмною бібліотекою, яка не має UI інтерфейсу(користувацького інтерфейсу) та дозволяє керувати ним, його поведінкою, заставляти виконувати певні команди та отримувати від нього необхідні дані [19]. У нашому випадку браузером є Google Chrome.

Для роботи з Webdriver потрібно кілька ключових елементів. Такими елементами є:

- браузер – це справжній браузер, чию роботу потрібно автоматизувати. У нашому випадку це Google Chrome;
- Driver – він є найважливішою сутністю, але попри це також є веб сервером. Він може як запускати браузер, так і закривати його. Також є можливість відправлення команд для браузера;
- Webelement – також є важливою сутністю. По суті є абстракцією різних веб елементів, таких як посилання та кнопки;
- скрипт – є деяким набором команд для драйвера браузера, написаний певною мовою;

- By – дана сутність по суті є абстракцією над локаторами деяких веб елементів [19].

Розроблений бот розміщується на хмарній платформі Heroku [21], яка є хорошим вибором для даного проекту.

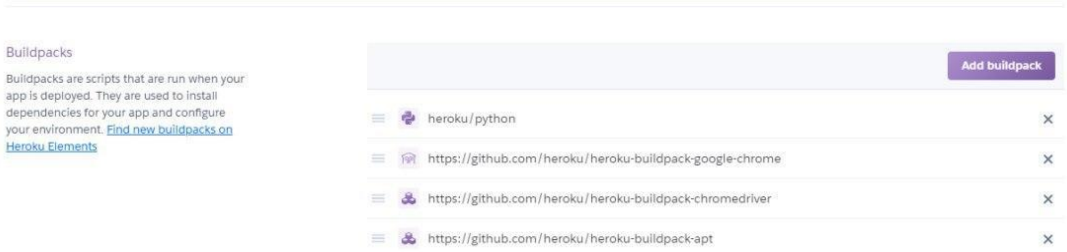

Рисунок 3.1 – Buildpacks. Підключення необхідних елементів

На рис. 3.1 показано підключення ключових елементів, потрібний для роботи бота. Першим з них є підключення самої мови Python [20], на якій буде написаний бот. Другим є підключеня браузера Google Chrome, так як він потрібен нам для автоматизації роботи. І останнім важливим для нас є підключення бібліотеки Webdriver [19] для вибраного браузера, щоб мати змогу з ним працювати.

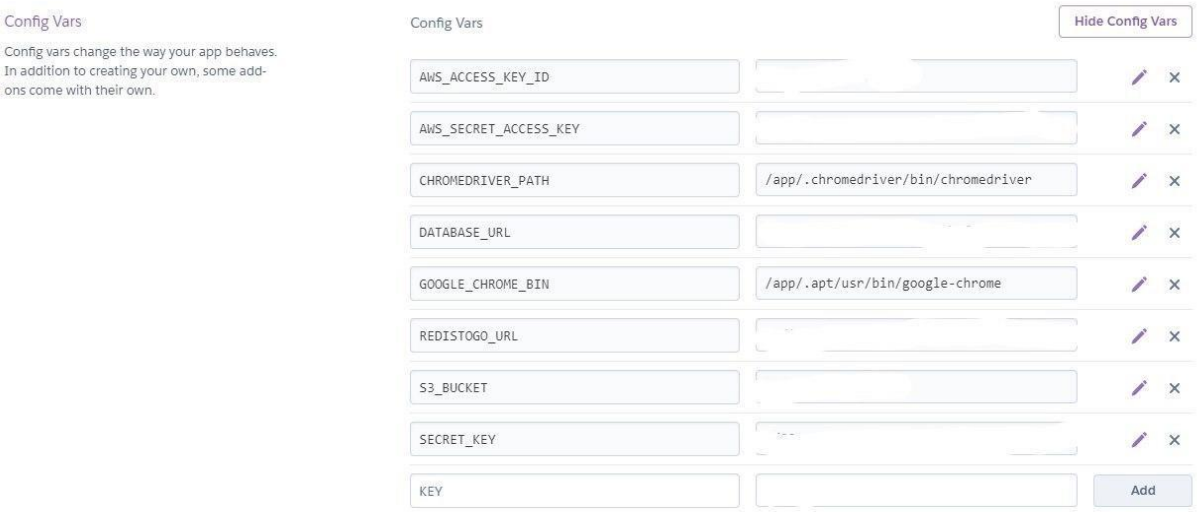

Рисунок 3.2 – Config Vars. Змінні середовища та шляхи

На рис. 3.2 зображені змінні середовища для даного проекту. Першою з них є CHROMEDRIVER_PATH. Це шлях до драйвера браузера Google Chrome. Наступною є GOOGLE_CHROME_BIN. Це є сам браузер, який буде використовуватись.

Ці змінні будуть потрібні для того, щоб в майбутньому створити об'єкт driver, так як один з вихідних параметрів є шляхом до запуску браузера та драйвера.

Після підключення необхідних модулів та створення змінних середовища потрібно підключити їх у скрипт.

```
chrome_options = webdriver.ChromeOptions()
chrome_options.binary_location = os.environ.get("GOOGLE_CHROME_BIN")
chrome_options.add_argument("--headless")
chrome_options.add_argument("--disable-dev-shm-usage")
chrome_options.add_argument("--no-sandbox")
prefs = {"profile.default_content_setting_values.notifications" : 2}
chrome_options.add_experimental_option("prefs",prefs)
driver = webdriver.Chrome(executable_path=os.environ.get("CHROMEDRIVER_PATH"), chrome_options=chrome_options)
```

Рисунок 3.3 – Підключення змінних середовища у проект

Для того, щоб мати змогу користуватись драйвером, спочатку потрібно підключити його та сам браузер до проекту, а після цього і до бота. На рис. 3.3 зображено таке підключення. Спочатку для нашого драйверу створюються опції. Після цього знаходиться бінарна локація браузера. Потім задається деякі настройки.

Дані настройки задаються для коректної роботи в так званому «безголовому» режимі браузера. У нього немає реального відображення його наповнення, тобто він все складає в пам'яті [19]. Це дуже корисно, так як за цей рахунок він використовує менше пам'яті, та, як наслідок, швидше працює. Це значно оптимізує систему, так як користувач не бачить сам процес публікації. Йому не потрібно, щоб це відображалось, так як сам процес відбувається на сервері.

Після цього відбувається відключення сповіщень. Створюється спеціальна змінна prefs з параметром, який відповідає за сповіщення та значенням 2. Дане значення відключає сповіщення. Після цього це налаштування додається до опцій драйвера.

І останньою лінійкою коду є створення самого драйверу. Першим параметром тут є адреса до самого запуску веб драйверу, а другим є ті опції, які були визначені раніше, тобто ті, які ми задаєм самостійно.

Після підключення драйверу до проекту потрібно підключити його безпосередньо до самого бота для використання функціоналу.

```
from selenium import webdriver
from selenium.webdriver.support.ui import WebDriverWait,Select
from selenium.webdriver.support import expected_conditions as EC
from selenium.webdriver.common.by import By
from selenium.webdriver.remote.file_detector import *
from selenium.webdriver import ActionChains
from selenium.webdriver.common.keys import Keys
from autopost import driver
from webdriver_manager.chrome import ChromeDriverManager
from time import sleep
import time
from bs4 import BeautifulSoup
import os
from autopost.resources import *
```

Рисунок 3.4 – Підключення бібліотек

На рис. 3.4 зображено підключення бібліотек для розробленого бота. Першою підключеною з них є бібліотека безпосередньо самого драйвера. Це підносить нам весь функціонал веб драйвера.

Серед них достатньо інтересним є модуль webdriver.common.by. даний модуль дозволяє ефективно шукати необхідні нам елементи. У нього є кілька способів знаходження необхідних веб елементів, що в майбутньому будуть використовуватись, а саме:

- пошук за ідентифікатором;
- пошук за назвою класу;

- пошук по імені;
- пошук за посиланням;
- пошук по частині посилання [19].

3.2 Функції Webdriver та входу в профіль

Зараз потроху буде описуватись функціонал бота. Починається все з створення самого драйвера.

```
facebook_email = './/*[@id="email"]'
facebook_password_field = './/*[@id="pass"]'
facebook_login_button = './/*[@id="loginbutton"]'

def get_driver():
    get_driv=driver
    return get_driv

def exit_driver(driver):
    time.sleep(1)
    driver.quit()
```

Рисунок 3.5 – Функції входу та виходу драйвера

З самого початку для управління та пошуку елементів буде потрібен драйвер. Він створюється в функції get_driver, та буде використовуватись на протязі всієї роботи.

Також, для зручності та швидкості роботи в майбутньому, id деяких елементів, які будуть розшукуватись, поміщено в змінні. Це такі елементи, як id email, паролю та кнопка входу. Вони будуть потрібні в майбутньому для входу в акаунт користувача та завжди є однаковими.

Ще є присутньою функція виходу з драйвера exit_driver. Це зроблено для того, щоб коли робота буде виконана, сервер не загружався, так як його потужність є обмеженою.

```
def facebook_login_fun(driver,login,password):
    status = False
    driver.get('https://www.facebook.com')
    facebook_email_element = driver.find_element_by_xpath(facebook_email)
    facebook_email_element.send_keys(login)
    time.sleep(1.5)
    facebook_password_element = driver.find_element_by_xpath(facebook_password_field)
    facebook_password_element.send_keys(password)
    time.sleep(1.5)
    facebook_login_element = driver.find_element_by_xpath(facebook_login_button)
    facebook_login_element.click()
    time.sleep(1.5)
    status = True
    return status
```

Рисунок 3.6 – Функція входу на акаунт користувача

На рис. 3.6 зображено код для функції входу розробленого бота. Для зручності тут і в наступних функціях буде розглянуто варіант з соціальною мережею Фейсбук [3]. Спочатку дана функція приймає дані на вхід. Після чого встановлюється булева змінна status, яка повертає тільки 2 значення: True або False. Це зроблено для того, щоб було зрозуміло, виконалась функція чи ні.

Після цього драйвер переходить на саму соціальну мережу Фейсбук. Потім виконується пошук елементу, в який потрібно вписати email для входу. Він виконується за допомогою мови XPath [19]. Ця мова використовується для того, щоб знайти необхідні вузли дерева XML-документа. В даному випадку пошук виконується по відносному шляху, так як задано id шуканого елемента.

Після того, як елемент знайдений, в нього вставляється логін користувача. Після цього йде невелика затримка. Вона потрібна для прогруження серверу та буде використовуватись на протязі всієї роботи.

Після невеличкої затримки, по аналогії з логіном такі ж дії провертаються відносно вікна для паролю та його вставлення в нього. Тоді відповідним пошуком знаходить кнопка для входу, після чого драйвер на неї клацає та виконує вхід. Тоді змінна status змінює своє значення на True, що означає, що вхід успішно виконано.

3.3 Функції переходу на профіль користувача та публікації

В цьому підрозділі буде розглянуто функції переходу на вибраний профіль користувача та самої публікації матеріалу. Для початку розглянемо функцію переходу на профіль.

```
def go_to_profile(driver):
    profile = WebDriverWait(driver, 5).until(EC.element_to_be_clickable((By.XPATH, "//a[@accesskey='2']")))
    time.sleep(1)
    profile.click()
```

Рисунок 3.7 – Функція переходу на профіль користувача

На рис. 3.7 зображена функція, яка переходить на вибраний профіль користувача.

Після переходу на місце проведення публікації можна зайнятись самим процесом публікації, тому давайте розглянемо функцію публікації в профілі. Дана функція спершу чекає, поки необхідний елемент стане клікабельним, тобто коли на нього можна буде натиснути. Це елемент, який відповідає за перехід на профіль користувача. Після невеликої затримки для прогруження натискає на нього та переходить на сам профіль.

Після переходу на місце проведення публікації необхідно провести саму публікацію. Для цього існує наступна функція.

```python
def publish_post(driver,status_message,url_image=None):
    time.sleep(3)
    test_frame = WebDriverWait(driver, 2).until(EC.element_to_be_clickable((By.XPATH, "//div[starts-with(@id, 'u_0_')]//textarea[@name='xhpc_message']")))

    if url_image!= None:
        sleep(1.5)
        input = driver.find_element_by_xpath("//div[starts-with(@id, 'u_0_')]//textarea[@name='xhpc_message']").send_keys(url_image)
        time.sleep(2)
        elem = driver.switch_to_active_element()
        elem.send_keys(Keys.ENTER)
        elem.send_keys(Keys.ENTER)
        url_image_len2 = len(url_image)
        while url_image_len2>-10:
            try:
                elem.send_keys(Keys.BACKSPACE)
                print('delete')
                url_image_len2 = url_image_len2-1
            except:
                print('Somethi wrong2')
        time.sleep(3)
        elem.send_keys(status_message)
        print('send message')
        time.sleep(3)
    else:
        ext_to_put_to_clipboard = driver.find_element_by_xpath("//div[starts-with(@id, 'u_0_')]//textarea[@name='xhpc_message']").send_keys(status_message)
    time.sleep(3)
    buttons = driver.find_elements_by_tag_name('button')
    time.sleep(3)
    for button in buttons:
        if button.text=='Опублікувати':
            button.click()
            print('button pressed')
            break
        elif button.text=='Post':
            button.click()
            print('button pressed')
            break
        elif button.text=='Отправить':
            button.click()
            print('button pressed')
            break
    time.sleep(7)
    url = get_post(driver,0)
    time.sleep(2)

    return url
```

Рисунок 3.8 – Функція публікації

На рис. 3.8 зображена функція для публікування інформації, переданої боту з веб додатку. Спершу скрипт приймає дані, після чого чекає поки потрібне текстове поле для публікації стане активним. Спершу відбувається завантаження, якщо воно завантажене користувачем. Після цього у текстове поле вставляється сама інформація, заголовок, текст та хештеги, які були записані. Якщо ж картинки не має, вставляється тільки текст.

Після невеликої затримки для підгруження фотографії знаходитить елемент «кнопка». Незалежно від мови шукається потрібна кнопка та натискається, після чого публікується матеріал, а сама функція повертає силку на нього.

Публікація в групі відбується за схожим алгоритмом, але там є декілька змін.

```
def publish_post_public(driver,url,status_message,url_image=None):
    driver.get(url)
```

Рисунок 3.9 – Відмінність публікації в групі від простої публікації

На рис. 3.9 зображено те, чим публікація в групі відрізняється від публікації на стіні користувача. Додатково дана функція приймає силку на групу, після чого зразу переходить на неї. В загальних рисах алгоритм не міняється, але потрібно пам’ятати, що для публікації в групі користувач повинен бути в ній адміністратором, або хоча б мати права на публікацію. Дана функція повертає силку на публікацію в групі.

3.4 Функції отримання публікацій та їх видалення

Після публікації даних розглянемо функції отримання силок даних публікацій. Це буде потрібно, наприклад , для видалення.

```
def get_post(driver,n):
    go_to_profile(driver)
    time.sleep(4)
    WebDriverWait(driver, 5).until(EC.element_to_be_clickable((By.XPATH, "//div[@data-testid='story-subtitle']")))
    driver.find_elements_by_xpath("//div[@data-testid='story-subtitle']")
    posts = driver.find_elements_by_class_name("timestampContent")
    posts[n].click()
    time.sleep(4)
    url = driver.current_url
    return str(url)
```

Рисунок 3.10 – Функція отримання силки публікації

На рис. 3.10 відображено зміст функції, яка повертає силку публікаці. Спочатку скрипт переходить в профіль користувача, після чого чекає, поки потрібний нам елемент стане доступним. Знаходить його, та створює цикл, щоб перейти на n-ний пост. Після цього повертає на нього силку.

```python
def get_post_public(driver,n,url):
    time.sleep(3)
    driver.get(url)
    time.sleep(4)
    WebDriverWait(driver, 5).until(EC.element_to_be_clickable((By.XPATH, "//div[@data-testid='story-subtitle']")))
    driver.find_elements_by_xpath("//div[@data-testid='story-subtitle']")
    posts = driver.find_elements_by_class_name("timestampContent")
    posts[n].click()
    time.sleep(4)
    url_ret = driver.current_url
    time.sleep(2)
    return str(url_ret)
```

Рисунок 3.11 – Функція отримання посилання на публікацію в групі

Для групи все працює дещо інакше. Драйвер отримає силку на групу, після чого переходить на неї. Далі алгоритм є схожим. Він чекає на те, коли потрібне поле стане доступним, знаходить його та через масив шукає n-ний пост. Після чого повертає на нього посилання.

Але на жаль видалити публікацію по цих функціях неможливо, так як її налаштування не є активним елементом. На нього неможливо «натиснути». Для цього були створені наступні функції.

```python
def get_mobile_post(driver,n):
    go_to_profile(driver)
    url = driver.current_url
    url2 = url[12:]
    mobile_url = 'https://m.' + url2
    driver.get(mobile_url)

    WebDriverWait(driver, 5).until(EC.element_to_be_clickable((By.XPATH, "//div[@data-sigil='m-feed-voice-subtitle']")))
    driver.find_elements_by_xpath("//div[@data-sigil='m-feed-voice-subtitle']")
    posts = driver.find_elements_by_tag_name("abbr")
    posts[n].click()
    time.sleep(2)
    time.sleep(4)
    url_ret = driver.current_url
    time.sleep(2)
    return str(url_ret)
```

Рисунок 3.12 – Функція отримання мобільного посилання на публікацію

Спеціально для видалення бути створені ця(див. рис 3.12) та наступна функції(див. рис. 3.13). Ця функція переходить на профіль вибраного користувача. Після цього відсікає непотрібну частину url та вставляє його

мобільну версію. Потім переходить по цій адресі та опиняється в мобільній версії соціальної мережі.

Тут вже потрібні налаштування публікації є доступними для виявлення та використання. Драйвер чекає, поки необхідний елемент стане доступним, після чого знаходить його, та по аналогії з іншими алгоритмами, видає посилання на публікацію.

```
def get_mobile_post_url(driver,url):
    url2 = url[12:]
    mobile_url = 'https://m.' + url2
    driver.get(mobile_url)
    time.sleep(3)
    return str(mobile_url)
```

Рисунок 3.13 – Функція перетворення звичайного посилання в мобільне

Дана функція є зробленою для того, щоб перетворювати теперішнє звичайне посилання в мобільне. Це використовується при видаленні. Вона видаляє перших 12 символів посилання, після чого на перед ставить свою частину, роблячи посилання мобільним. Після цього здійснює перехід по ньому та повертає силку.

Також присутня функція, яка дозволяє повернути список публікацій.

```python
def get_all_posts(driver):
    post_links = []
    n = 0
    temp = True
    while temp:
        try:
            post_links.append(get_post(driver,n))
            time.sleep(2)
            driver.back()
            time.sleep(1)
            n = n + 1
        except:
            "End of list"
            temp = False
    return post_links
```

Рисунок 3.14 – Функція отримання список усіх публікацій

Функція, зображена на рис. 3.14, дозволяє отримати список публікацій на стіні користувача. За допомогою комбінацій цієї та функцій вище можна буде отримати список посилань на публікації. Дана функція створює список, після чого спускається по стіні вниз, від запису до запису, наповнюючи його. В кінці стіни функція зупиняється та після цього повертає список.

Зараз буде розглянуто функцію видалення.

```python
def delete_post(driver,url):
    test = get_mobile_post_url(driver,url)
    driver.get(test)
    print(test)
    time.sleep(3.5)
    print("get mobile irl success")
    driver.find_element_by_xpath('//a[@aria-haspopup="true"]').click()
    print("find popup")
    time.sleep(3.5)
    WebDriverWait(driver, 3).until(EC.element_to_be_clickable((By.XPATH, '//a[@data-sigil="touchable touchable removeStoryButton enabled_action"]'))).click()
    time.sleep(3)
    print("click popup")
    buttons = driver.find_elements_by_xpath('//a[@role="button"]')
    print("find all buutons")
    time.sleep(4)
    for button in buttons:
        if button.text=='Видалити':
            button.click()
            print("click")
            break
        elif button.text=='Delete':
            button.click()
            print("click")
            break
        elif button.text=='Удалить':
            button.click()
            print("click")
            break
    print("all good")
    time.sleep(5)
```

Рисунок 3.15 – Функція видалення публікацій

Однією з головних функцій є функція видалення(див. рис. 3.15). Головною її метою є видалення рекламних публікацій. Спочатку для видалення необхідно перейти на мобільну версію, так як потрібне меню в звичайній версії не доступно.

Після перетворення посилання на мобільне, драйвер переходить по ньому. Після чого на публікації натискає на потрібне меню. Потім чекає, доки воно стане активне та шукає там потрібний елемент. В даному випадку це може бути кнопка «видалити» на бідь які мові. Після знаходження драйвер натискає на цю кнопку і даний запис видаляється.

3.5 Функції публікації та видалення для соціальної мережі Фейсбук

Розглянувши головний функціонал бота перейдемо до безпосередньо функцій публікації в соціальну мережу. Для зручності та наглядності було вибрано соціальну мережу Фейсбук. В цьому підрозділі буде розглянуто такі функції, як публікація в профілі, публікація в групі та видалення з даної соціальної мережі.

Почнемо з публікації в соціальній мережі Фейсбук.

```
def facebook_create_post(facebook_login=None,facebook_password=None,status=None,url_image=None):
    url = None
    try:
        driver = get_driver()

        try:
            facebook_login_fun(driver,facebook_login,facebook_password)
            print('success login')
        except:
            print('login failed')
        try:
            url = publish_post(driver,status,url_image=url_image)
            print('success publish')
        except:
            print('publish failed')

        exit_driver(driver)
    except:
        print("something went wrong")
    return url
```

Рисунок 3.16 – Функція публікації в соціальній мережі Фейсбук

Дана функція являє собою збірку з менших функцій, кожна з яких виконує свою роботу. Вона приймає дані від бази даних, які передає веб додаток. Це дані на вхід, тобто логін та пароль, посилання на фото. Після цього запускається драйвер, який буде керувати процесом. Спочатку бот пробує виконати вхід, так як є потрібні для цього дані. Далі є два можливих варіанти подій. Вхід виконається успішно або виведеться помилка, так як введені користувачем дані є неправильними.

Дана функція побудована таким чином, щоб на кожному з кроків мати змогу припинити процес та вийти, а не продовжувати працювати.

Після успішного входу в користувацький акаунт буде відбуватись публікація переданого матеріалу. Спочатку драйвер перейде на стіну профілю користувача, а потім почне вставляти необхідні дані та завантажувати фото. На цьому етапі також передбачено вихід з програми у разі помилки.

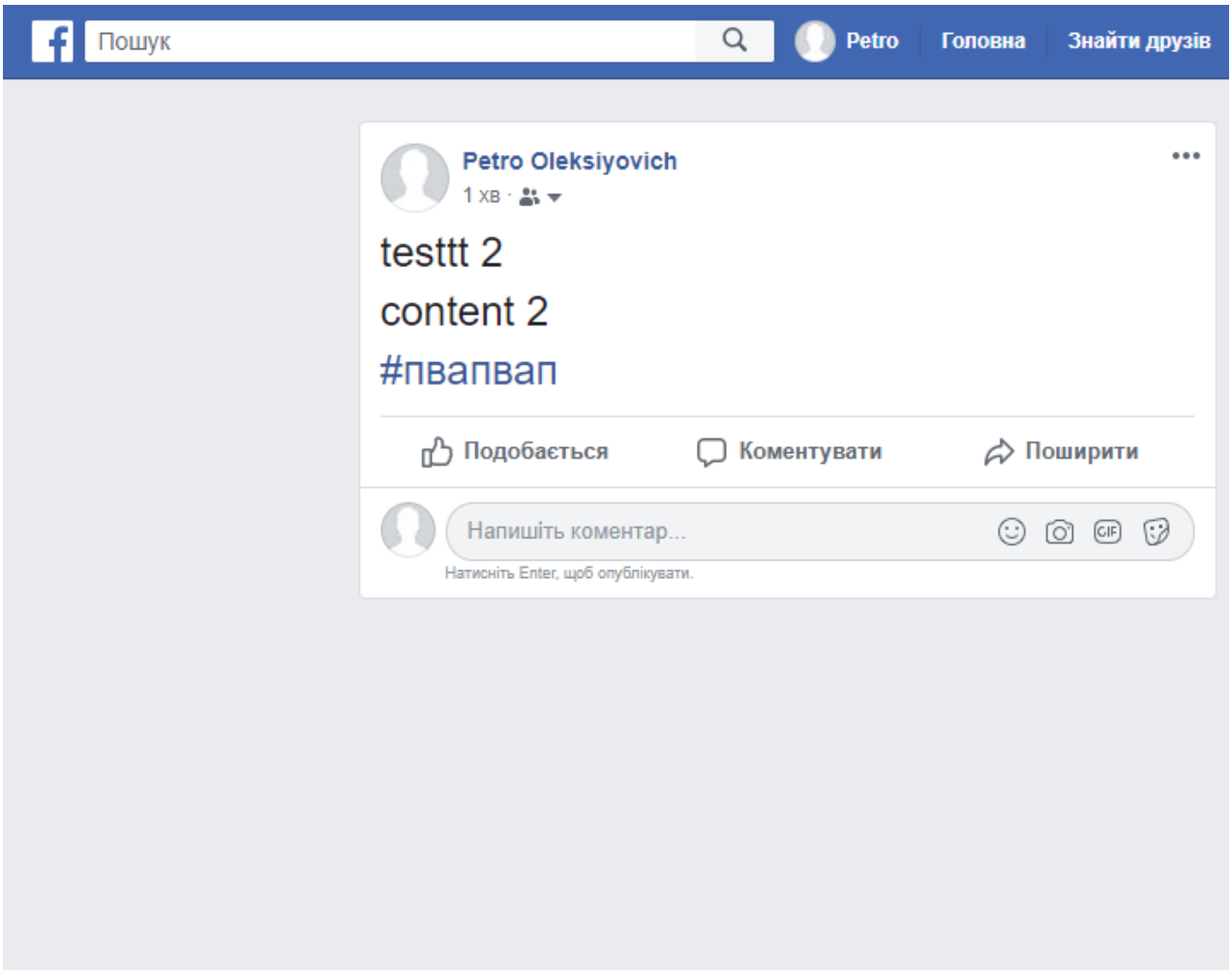

Рисунок 3.17 – Приклад успішної публікації

На рис. 3.17 показано успішний приклад публікації, виконаний розробленим сервісом та, зокрема, ботом.

Після цього виконується закриття драйверу для того, щоб не загружати сервер.

Дана функція повертає посилання на тільки що створену публікацію. Саме наявність цього посилання говорить веб додатку, що публікація пройшла успішно(див. рис. 3.18).

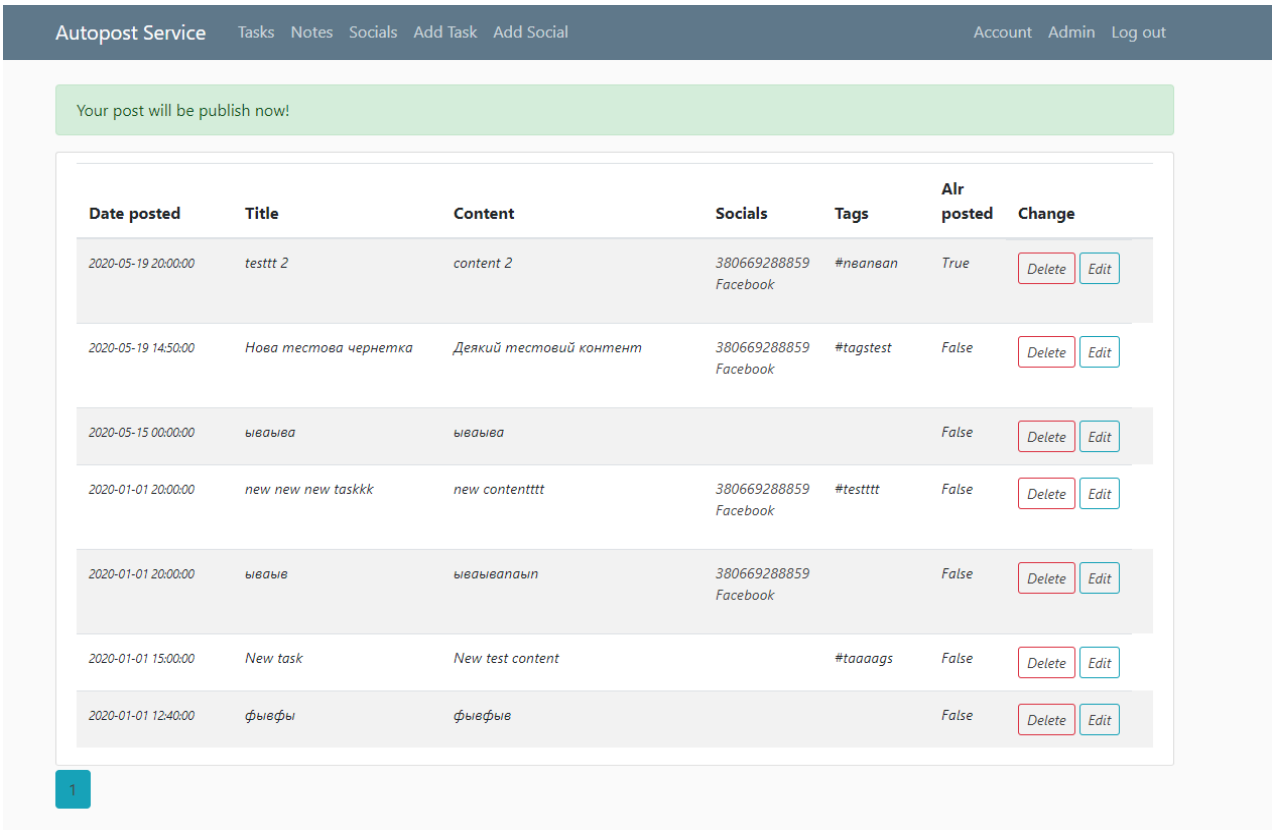

Рисунок 3.18 – Веб додаток сповіщає про успішну публікацію

Наступною є функція публікації матеріалу в групі у соціальній мережі Фейсбук.

```
def facebook_create_post_public(facebook_login,facebook_password,status,url_image=None,url=None):
    url_ret = None
    try:
        driver = get_driver()
        try:
            facebook_login_fun(driver,facebook_login,facebook_password)
            print('success login')
        except:
            print('login failed')
        try:
            url_ret = publish_post_public(driver,url,status,url_image=url_image)
            url_ret = url_ret[:84]
            print('success publish in public')
        except:
            print('publish public failed')
        exit_driver(driver)
    except:
        print("something went wrong")
    return url_ret
```

Рисунок 3.19 – Функція публікації для групи у соціальні мережі Фейсбук

На рис. 3.19 зображено функцію для соціальної мережі Фейсбук, яка призначена для публікації матеріалу в групи. Після прийому необхідних даних у даній функції запускається драйвер, після чого буде стандартна авторизація.

Як можна побачити, дана функція також побудована таким чином, щоб вона мала змогу припинити свою роботу у випадку непередбачуваної помилки.

Після цього драйвер переходить по посиланню на групу, в якій буде публікувати інформацію. Важливо замітити, що для публікації даних в групи потрібно бути її власником, або просто мати достатні права на створення поста.

Після публікації матеріалу драйвер завершує свою роботу, а дана функція повертає силку на опублікований пост. Тоді веб сервіс розуміє, що публікація пройшла успішно та повідомляє про це користувача (див. рис. 3.18).

Останньою функцією буде видалення публікації в соціальній мережі Фейсбук.

```
def facebook_delete_post(facebook_login,facebook_password,url):
    try:
        driver = get_driver()
        try:
            facebook_login_fun(driver,facebook_login,facebook_password)
            print('success login')
        except:
            print('login failed')
        try:
            delete_post(driver,url)
            print('success delete')
        except:
            print('delete failed')
        exit_driver(driver)
    except:
        print("something went wrong, dont delete")
```

Рисунок 3.21 – Функція видалення публікації з соціальної мережі Фейсбук

Дана функція призначена для видалення опублікованого матеріалу з соціальної мережі Фейсбук. Все, що вона приймає, це тільки дані на вхід, тобто логін та пароль, та посилання на публікацію, яку потрібно видалити. У нашому випадку це рекламні публікації, які відстояли свій термін, та потребують видалення.

Після запуску драйверу виконується вхід. Так, як посилання на потрібну публікацію вже є, він переходить зразу на нього. Після цього починається процес видалення. Посилання перетворюється в мобільне, шукається необхідне меню та видаляється сама публікація (див. рис. 3.22).

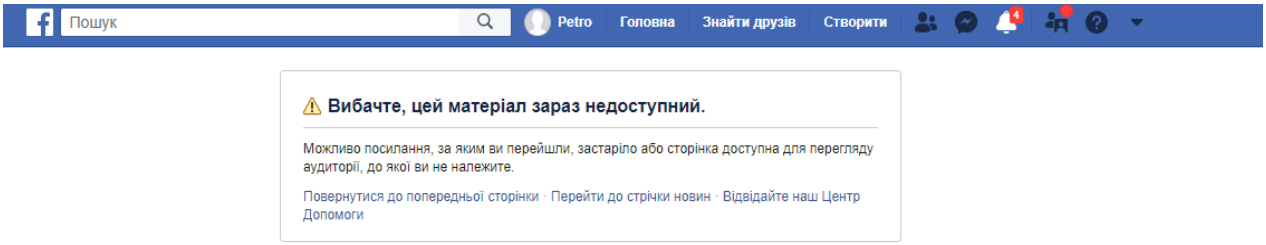

Рисунок 3.22 – Успішне видалення публікації з соціальної мережі Фейсбук

Дана функція теж зроблена таким чином, щоб при будь яких помилках вона завершувала свою роботу на тому ж кроці.

Після успішного видалення веб додаток повертає користувачу відповідне повідомлення (див. рис. 3.23).

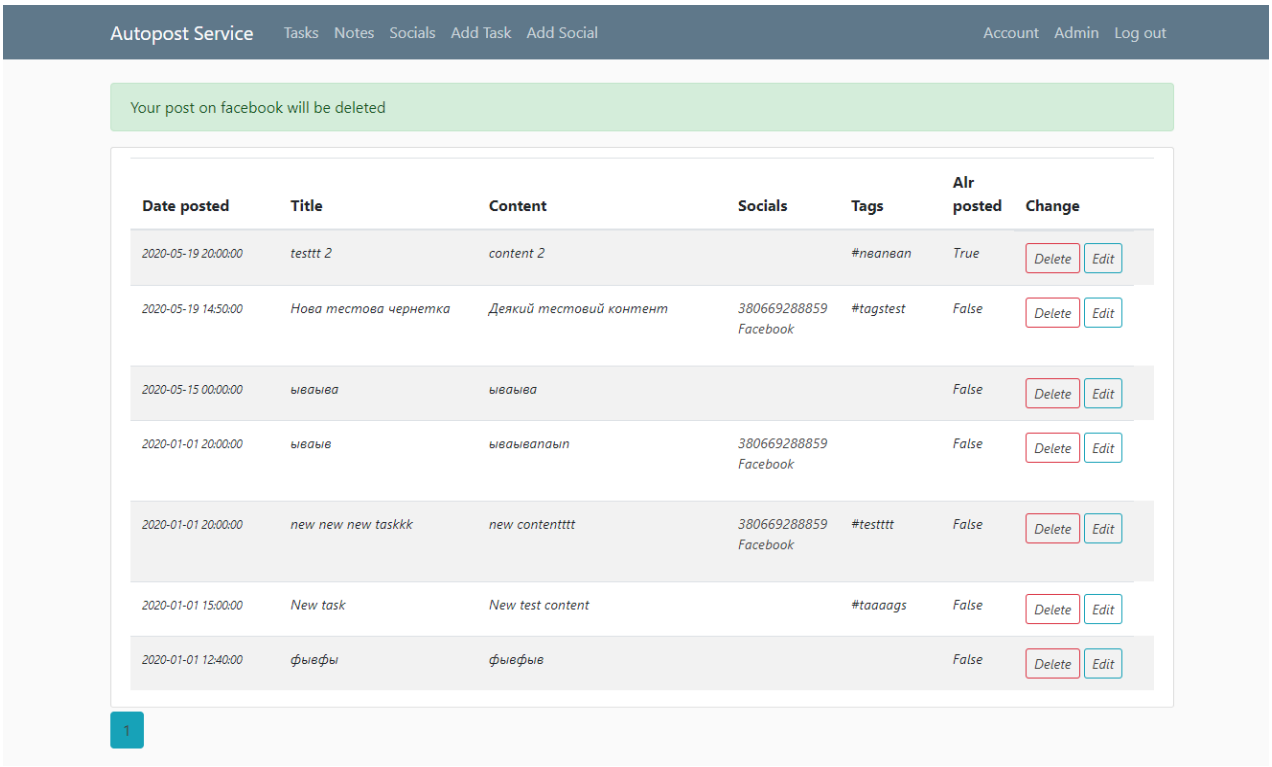

Рисунок 3.23 – Веб сервіс повідомляє користувача про успішне видалення

На цьому етапі розробка бота підійшла до кінця. Детальніше ознайомитись з кодом можна в додатках (див. дод. А).

4 БІЗНЕС ПЛАН РОЗРОБКИ БОТА ДЛЯ АВТОПОСТИНГУ

4.1 Резюме

Розроблений скрипт буде частиною сервісу автопостингу, який буде надавати клієнтам пакет послуг у сфері автоматичної публікації постів.

Цільовою аудиторією даного сервісу будуть невеличкі компанії, навчальні заклади, такі як школи та університети, невеличкі кафе та заклади інфраструктури, політики та взагалі всі, хто вміє користуватись комп'ютером та має кілька соціальних мереж. Вікові рамки цільової аудиторії будуть від 20 до 60 років.

Щоб реалізувати даний проект, потрібна сума від 30 000 до 40 000 грн.

Здійснюватися фінансування буде з особистого прибутку даного сервісу та вкладень інвесторів.

Щоб реалізувати даний проект, до роботи буде залучено двоє співробітників.

Очікуваним місячним доходом буде сума 10 000 грн. чистий прибуток за рік складе приблизно 80 000 грн.

4.2 Маркетинг

Вид послуги

Даний сервіс автопостингу пропонуватиме користувачу послуги автоматичної публікації постів у кілька вибраних соціальних мереж, відкладеного постингу, а також можливість створювати рекламні пости, що буде здійснюватися з одного кабінету за допомогою сайту на комп'ютері чи смартфоні.

Даний сервіс має задовільнити потребу у зручній, швидкій та дешевій публікації матеріалу у різні соціальні мережі з мінімізацією втрати часу та коштів.

Даний проект буде використовуватись у соціальній сфері.

Унікальністю даного проекту буде його відносна безплатність та наявність української мови у майбутньому.

Можливо у майбутньому для більшої зручності буде випущено додаток з представленим проектом.

Недоліками даного сервісу буде відносно менший функціонал стосовно його більш серйозних конкурентів. Але з часом функціонал буде розширюватись, тому це не буде глобальною проблемою.

Джерела вивчення ринку

Щоб вивчити ринок даної продукції, використовувались різні джерела інформації в мережі Інтернет та соціальних мережах, а також були прийняті до уваги особисті спостереження.

Оцінка очікуваного попиту

Клієнтам даного сервісу можуть бути як компанії, так і прості користувачі, тому що вони будуть розосереджені по всій території країни. Єдиною умовою буде наявність хорошого покриття для доступу в мережу інтернет та смартфону чи комп’ютера.

Потенційним клієнтом даного проекту є людина, якій потрібно зручно та малі кошти зробити публікацію у вибрані соціальні мережі, не витрачаючи при цьому на це багато часу.

Мінімальна ціна за аналогічні послуги складає 90-100 грн. за певну кількість публікацій чи підписка на сервіс на місяць. Але і це пропонує тільки дуже мала частина сервісів. Середньою ціною є 200 грн. за таку підписку, що є в рази дорожче, ніж пропонуватиме наш проект.

Дані сервіси зараз дуже активно набувають популярності, адже збільшується вплив цифрових технологій на повсякденне життя. Так як, зараз

великим попитом користується моделі, блогери та Ютуб-канали, попит на дану послугу тільки ростиме.

Конкуренти

У заданій сфері зараз є достатньо велика кількість схожих сервісів. Більшість з них не користується популярністю у нас. Тому конкурувати прийдеться тільки з частиною із них.

Так як даний ресурс розміщений у мережі Інтернет, його покриття розширюється на всю країну. Але, звісно, потрібне підключення до цієї самої мережі.

У конкурентних сервісів середня ціна тут складатиме близько 200 грн. за підписку на місяць чи певну кількість публікацій.

Для свого піару та просування в маси конкуренти використовують рекламу у мережі Інтернет. У більшості випадків все обмежується тільки цією рекламою, так як дозволити собі місце на якійсь значимій технологічній події можуть собі далеко не всі.

Найближчим конкурентом для даного проекту буде сервіс SMM Aero, так як його цінові критерії є дуже близькими, але розроблений проект буде оснащений більшим функціоналом [10].

Даний бізнес зараз добре розвивається, набирається хороша клієнтська база і приріст в доході є сталим.

Також у період сезонного відпочинку у різні пори року є сенс робити акції для залучення більшої аудиторії та збільшення доходу.

Розроблений сервіс матиме кілька переваг над сервісами своєї цінової категорії, а саме буде мати більший функціонал та наявність української мови.

Сценарії розвитку

Існує кілька сценаріїв розвитку даного сервісу. Розглянемо деякі з них.

Першим буде песимістичний сценарій розвитку даного сервісу:

- через те, що дані сервіси є відносно новими у світі, та в Україні в цілому, люди в перший час майже не використовуватимуть дані послуги;
- через деякий час з деякою ймовірністю з’явиться аудиторія, яка користувалась схожими сервісами до цього;
- можлива велика конкуренція з аналогічними сервісами;
- є ймовірність, що даний сервіс буде використаний для спаму, та проявить себе в негативному ключі, після чого відбудеться зменшення аудиторії.

Тепер розглянемо інший сценарій розвитку.

Оптимістичним сценарієм для даного сервісу буде:

- люди, які до цього не користувались подібними послугами, одразу починають активно використовувати їх в повсякденному житті, від чого сильно збільшується аудиторія;
- бачучи великий приріст аудиторії, люди, які до цього користувались аналогічними сервісами, переходять на розроблений продукт, цим самим не тільки серйозно збільшивши платоспроможну аудиторію, а й добре прорекламують наш проект;
- серйозно збільшується попит на даний сервіс та збільшується його поширення у мережі;
- відсутня конкуренція, або ж є зовсім слабкою, що дає привід не хвилюватись про неї;
- Даним сервісом не будуть користуватись для спаму чи інших шкідливих дій.

Тепер можна розглянути реалістичний сценарій подій, які будуть відбуватись з даним сервісом, а саме:

- з самого початку сервіс не буде користуватись великим попитом, адже є ще новим і більшість аудиторії про нього просто не знає;

- згодом після невеличкої рекламної кампанії попит на даний сервіс починає зростати;
- помірна конкуренція, яка не буде згубно впливати на даний сервіс, а навпаки, тільки дасть мотивацію його розвивати;
- можливі кілька випадків користування даним проектом для спам-розсилки, але тільки на початку його існування, так як надалі сервіс стане більш престижним.

Маркетингова підтримка

Для того, щоб прорекламувати даний сервіс автопостингу, буде закуплена реклама у мережі Інтернет, а саме у різних пабліків, груп та простих блогерів. Це дозволить ефективно та за невеликі кошти прорекламувати даний продукт.

Для цієї реклами буде виділено близько 2000 грн. Їх повинно вистачити для того, щоб привернути увагу початкової аудиторії.

Користуватись даним сервісом можна буде без залучення окремих спеціалістів чи платформ, які пропонують переглянути інструкції використання подібних сервісів. Все буде зрозуміло на інтуїтивному рівні.

Встановлення цін

Діапазон цін буде помітно дешевший, ніж у аналогів з відповідним функціоналом. У сервісів, які мають схожий ціновий діапазон, функціонал буде значно менший, ніж у нашого продукту.

Якщо попит аудиторії буде недостатнім, або вона буде замалою, розумнім рішення буде проведення різних акцій з метою привернення уваги. До таких акцій можуть відноситись як сезонні знижки, так і надання певних льгот за виконання певних умов.

Даний сервіс повинен якщо не повністю, то хоча б на ранніх етапах частково відбити затрати на своє виготовлення та розміщення і мережі Інтернет. А також в найближчий час почати приносити прибуток.

Так як ринок даної послуги зараз великий, про націнку за унікальність не може бути і мови. Це призведе до зменшення аудиторії та антиреклами даного сервісу.

Націнка за унікальність проводитись не буде для збільшення попиту на послугу.

Приблизно сумою за користування даним продуктом буде 100 грн за 4 облікових записи в соціальних мережах.

План збуту

Розроблений проект буде доступний в мережі Інтернет. Потенційний покупець буде мати змогу самостійно користуватись продуктом.

План комплексу просування послуги

Реклама даного сервісу буде організовуватись власними силами, посередники будуть непотрібні. Вона буде розміщуватись у мережі Інтернет на популярних соціальних мережах та міських сайтах, які користуються популярністю.

У даному оголошенні розміщуватимуться дані про зручність, швидкість та якість сервісу. Також буде інформація про відносну дешевизну та своєрідну унікальність, так як продукт буде українським.

Яка сума буде витрачатись на рекламні оголошення? Це все буде залежати від кількості прибутку, яку даватиме сервіс. В середньому на рекламу буде витрачатись третина або чверть доходу.

4.3 Обґрунтування фінансових вкладів

Опис виробничих потужностей

Розташовуватись даний сервіс автопостингу буде в мережі Інтернет на платному хостингу. Буде найнятий спеціаліст, який буде відстежувати стан нашого проекту та піклуватись про його справну роботу.

Можливо в майбутньому буде можливість провести кооперацію з якимось зовнішнім ресурсом чи компанією для привернення більшої аудиторії платоспроможних користувачів, а сервіс в свою чергу представить знижки на свої послуги.

Даний сервіс автопостингу буде працювати цілодобово.

Таблиця 4.1 – Витрати на сервіс атопостингу

Витрати на розробку сервісу автопостингу	20 000 грн.
Витрати на хостинг	500 грн. в місяць
Наймання системного адміністратора	5000 грн в місяць
Разом:	25 500 грн.

Підбір персоналу та оплата праці

На початковому етапі для реалізації даного сервісу не потрібно буде залучати сторонніх спеціалістів. Після того, як сервіс запрацює в автономному режимі, потрібно буде найняти системного адміністратора для відслідковування та коректної роботи проекту.

Підбір працівника на пост системного адміністратора спершу буде проводитись серед студентів, так як це значно зменшить витрати на обслуговування. У випадку успіху сервісу та співробітництва з великими компаніями можна буде найняти консультанта для більш приємної та ефективної співпраці.

Скоріш за все, набір буде проводитись серед знайомих спеціалістів або людей з університету методом співбесіди. Якщо даний варіант не спрацює, можна буде розмістити оголошення в мережі Інтернет або переглянути вже наявні.

Для системного адміністратора робочий день буде складати 8 годин.

Оплата праці буде відбуватись кожного місяця за умови безперебійної роботи нашого сервісу.

Також даний системний адміністратор зможе безплатно використовувати наш сервіс автопостингу в якості приємного бонусу.

Календарний план

Таблиця 4.2 – Календарний план для розробки сервісу автопостингу

Місяць	Заплановані заходи	Особи
1	Розробка архітектури бота та сайту	Навроцький Б. Харун Ю.
2.5	Представлення робочого прототипу сайту та бота та їх тестування	Навроцький Б. Харун Ю.
3	Реліз сервісу	Навроцький Б. Харун Ю.

4.4 Нормативно-правові нюанси

Організаційно-правова форма бізнес-проекту

Даний сервіс автопостингу буде приватним, так як складання договорів з соціальними мережами не складе ніяких проблем.

У майбутньому буде можливий продаж даного сервісу за вигідну ціну, або його інтеграція у більш великі проекти, такі як, наприклад, викуп функціоналу та інтеграція у інші сервіси чи соціальні мережі.

Організаційний план

Керівництвом даного проекту будуть дві особи, у яких будуть права на керування сервісом, прийняття будь яких рішень. Також вони будуть володіти повними правами на власність над цим сервісом.

Після відносного успіху проекту та принесення прибутку буде можливість залучити системного адміністратора та, за необхідності, консультанта.

4.5 Складання фінансового бюджету

Визначення джерел фінансування

Джерелами фінансування будуть власні кошти та, можливо, інвестиції.

Графік початкового етапу реалізації проекту

Тривалість організаційного періоду проекту: 3 місяці.

Очікувана сума загального фінансування: 30 000 – 40 000 грн.

Для ефективної розробки повна сума повинна бути отриманою в період перших двох місяців.

Розрахунок кредитів

Щомісячний податок: від 200 до 4 000 грн. в залежності від прибутку.

План очікуваних прибутків та збитків

Прибуток за надані послуги: від 1 000 грн. до 20 000 за місяць.

Чистий прибуток: від 4 000 грн. за місяць.

Постійні витрати: від 5 000 грн. за місяць

Змінні витрати: від 1 000 грн. за місяць

План руху грошових коштів

З різних доступних джерел фінансування (інвестиції) потрібно отримати разом 30 000-40 000 грн.

Після покупки усього обладнання і сировини буде сплачено 25 500 грн.

Розрахунок показників проекту

Чистий прибуток за рік: 30 000 грн.

Рентабельність: 125 %.

Термін окупності: 9-12 місяців.

4.6 Оцінка можливих ризиків

Наведені нижче ризики є малоймовірними, але все ж мають право на існування. Такими ризиками є:

- непопулярність сервісу або різке її падіння;
- блокування сервісу;
- використовування сервісу для спаму.

ВИСНОВКИ

У процесі розробки даного проекту було розглянуто предметну область для сервісів автопостингу та проаналізовано ринок, після чого порівнювались існуючі рішення для застосування та усунення їх недоліків.

Також для цього був проведений аналіз технологій, що використовувалися для розробки програмного продукту на предмет їхньої доцільності при застосуванні та інші важливі елементи.

В кінцевому результаті було розроблено бота для автопостингу. Він напряму взаємодіє з веб додатком та в сумі вони формують повноцінно функціональний сервіс автопостингу.

Розроблений сервіс має широке застосування. Він може застосовуватись як для простих цілей, таких як просте ведення своєї сторінки в соціальних мережах, так і в комерційних. Це зараз дуже актуально, тому що дозволяє спростити життя компаніям та простим блогерам для більшої зручності.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

REFERENCES

1. Визначення автопостингу. URL: https://umi.ru/blog/avtoposting_v_socseti/ (дата звернення: 28.04.2020)
2. Використання автопостингу. URL: https://www.cossa.ru/276/162274/ (дата звернення: 28.04.2020)
3. Про Фейсбук. URL: https://www.facebook.com/ (дата звернення: 28.04.2020)
4. Про Інстаграм. URL: https://www.instagram.com/?hl=uk (дата звернення: 28.04.2020)
5. Про Твітер. URL: https://twitter.com/ (дата звернення: 28.04.2020)
6. Переваги автопостингу. URL: https://semantica.in/blog/chto-takoe-avtoposting.html (дата звернення: 28.04.2020)
7. Статистика для сервісів автопостингу. URL: https://seoquick.com.ua/autoposting-services/ (дата звернення: 28.04.2020)
8. Про NovaPress Publisher. URL: https://novapress.com/ (дата звернення: 28.04.2020)
9. Про сервіси автопостингу. URL: https://netology.ru/blog/best-autoposting (дата звернення: 28.04.2020)
10. Про SMM Aero. URL: https://smmaero.ru/ (дата звернення: 28.04.2020)
11. Про Ампліфер. URL: https://amplifr.com/ru/ (дата звернення: 28.04.2020)
12. Про Kuku.io. URL: https://kuku.io/ru/ (дата звернення: 28.04.2020)
13. Про Publbox. URL: https://dnative.ru/catalog/obzory/otlozhennyj-posting-v-instagram/publbox/ (дата звернення: 28.04.2020)
14. Кузь М.В., Соловко Я.Т., Андрейко В.М. Методологія формування узагальненого критерію якості програмного забезпечення в умовах невизначеності. Вісник Вінницького політехнічного інституту. Вінниця,

2015. №5. С. 104-107. Про функціональні вимоги. URL: https://studfile.net/preview/5462334/ (дата звернення: 28.04.2020)

15. Кузь М., Новак В., Новак М., Сільва Р. Модель часових параметрів показників якості програмного забезпечення. Прикладні науково-технічні дослідження: матеріали II міжнар. наук.-практ. конф., м. Івано-Франківськ, 3-5 квіт. 2018 р. Івано-Франківськ, 2018. С.32-33.
16. Про нефункціональні вимоги. URL: https://life-prog.ru/ukr/1_7376_nefunktsionalni-vimogi.html (дата звернення: 28.04.2020)
17. Козак О. Л. Аналіз вимог до програмного забезпечення : Опорний конспект лекцій. Тернопіль : ТНЕУ, 2011. 56 с.
18. M. Seidl, M. Scholz, C. Huemer, G. Kappel, UML @ Classroom: An Introduction to Object-Oriented Modeling. Berlin, Germany: Springer, 2015, pp. 218.
19. U. Gundecha, S. Avasarala, Selenium WebDriver 3 Practical Guide: End-to-end automation testing for web and mobile browsers with Selenium WebDriver, 2nd Edition. Birmingham, United Kingdom: Packt Publishing, 2018, pp. 280.
20. L. Ramalho, Fluent Python: Clear, Concise, and Effective Programming. United States: O'Reilly Media, 2015, pp. 792.
21. N. Middleton, Heroku: Up And Running. United States: O'Reilly Media, 2013, pp. 100.

ДОДАТОК А

Код програмного забезпечення

test_bot.py

```
from selenium import webdriver
from selenium.webdriver.support.ui import WebDriverWait,Select
from selenium.webdriver.support import expected_conditions as EC
from selenium.webdriver.common.by import By
from selenium.webdriver.remote.file_detector import *
from selenium.webdriver import ActionChains
from selenium.webdriver.common.keys import Keys
from autopost import driver
from webdriver_manager.chrome import ChromeDriverManager
from time import sleep
import time
from bs4 import BeautifulSoup
import os
from autopost.resources import *

facebook_email = './/*[@id="email"]'
facebook_password_field = './/*[@id="pass"]'
facebook_login_button = './/*[@id="loginbutton"]'

def get_driver():
    get_driv=driver
    return get_driv

def facebook_login_fun(driver,login,password):
    status = False
    driver.get('https://www.facebook.com')

    facebook_email_element =
driver.find_element_by_xpath(facebook_email)
    facebook_email_element.send_keys(login)
    time.sleep(1.5)
    facebook_password_element =
driver.find_element_by_xpath(facebook_password_field)
    facebook_password_element.send_keys(password)
    time.sleep(1.5)
    facebook_login_element =
driver.find_element_by_xpath(facebook_login_button)
    facebook_login_element.click()
    time.sleep(1.5)
    status = True
```

```
    return status

def exit_driver(driver):
    time.sleep(1)
    driver.quit()

def publish_post(driver,status_message,url_image=None):
    time.sleep(3)
    test_frame = WebDriverWait(driver,
2).until(EC.element_to_be_clickable((By.XPATH, "//div[starts-
with(@id, 'u_0_')]//textarea[@name='xhpc_message']")))

    if url_image!= None:
        sleep(1.5)
        input = driver.find_element_by_xpath("//div[starts-
with(@id,
'u_0_')]//textarea[@name='xhpc_message']").send_keys(url_image)
        time.sleep(2)
        elem = driver.switch_to_active_element()
        elem.send_keys(Keys.ENTER)
        elem.send_keys(Keys.ENTER)
        url_image_len2 = len(url_image)
        while url_image_len2>-10:
            try:
                elem.send_keys(Keys.BACKSPACE)
                print('delete')
                url_image_len2 = url_image_len2-1
            except:
                print('Somethi wrong2')
        time.sleep(3)

        #driver.find_element_by_xpath("//div[@aria-
label='Відхилити']").click()
        elem.send_keys(status_message)
        print('send message')
        time.sleep(3)
    else:
        ext_to_put_to_clipboard =
driver.find_element_by_xpath("//div[starts-with(@id,
'u_0_')]//textarea[@name='xhpc_message']").send_keys(status_messag
e)
    time.sleep(3)
    buttons = driver.find_elements_by_tag_name('button')
    time.sleep(3)
    for button in buttons:
        if button.text=='Опублікувати':
            button.click()
            print('button pressed')
            break
```

Продовження додатку А

```
        elif button.text=='Post':
            button.click()
            print('button pressed')
            break
        elif button.text=='Отправить':
            button.click()
            print('button pressed')
            break
    time.sleep(7)
    url = get_post(driver,0)
    time.sleep(2)

    return url

def publish_post_public(driver,url,status_message,url_image=None):
    driver.get(url)
    WebDriverWait(driver,
5).until(EC.element_to_be_clickable((By.XPATH, "//div[starts-
with(@id, 'u_0_')]//textarea[@name='xhpc_message']")))

    if url_image!= None:
        sleep(1.5)
        input = driver.find_element_by_xpath("//div[starts-
with(@id,
'u_0_')]//textarea[@name='xhpc_message']").send_keys(url_image)
        time.sleep(2)
        elem = driver.switch_to_active_element()
        elem.send_keys(Keys.ENTER)
        elem.send_keys(Keys.ENTER)
        url_image_len2 = len(url_image)
        while url_image_len2>-10:
            try:
                elem.send_keys(Keys.BACKSPACE)
                print('delete')
                url_image_len2 = url_image_len2-1
            except:
                print('Somethi wrong2')
        time.sleep(4)

        #driver.find_element_by_xpath("//div[@aria-
label='Відхилити']").click()
        elem.send_keys(status_message)
        print('send message')
        time.sleep(5)
    else:
        ext_to_put_to_clipboard =
driver.find_element_by_xpath("//div[starts-with(@id,
'u_0_')]//textarea[@name='xhpc_message']").send_keys(status_messag
e)
```

Продовження додатку А

```
    time.sleep(7)
    buttons = driver.find_elements_by_tag_name('button')
    time.sleep(4)
    for button in buttons:
        if button.text=='Опублікувати':
            button.click()
            break
        elif button.text=='Post':
            button.click()
            break
        elif button.text=='Отправить':
            button.click()
            break
    time.sleep(7)

    url_ret = get_post_public(driver,0,url)

    return url_ret

def go_to_profile(driver):
    profile = WebDriverWait(driver,
5).until(EC.element_to_be_clickable((By.XPATH,
"//a[@accesskey='2']")))
    time.sleep(1)
    profile.click()

def get_post(driver,n):
    go_to_profile(driver)
    time.sleep(4)
    WebDriverWait(driver,
5).until(EC.element_to_be_clickable((By.XPATH, "//div[@data-
testid='story-subtitle']")))
    driver.find_elements_by_xpath("//div[@data-testid='story-
subtitle']")
    posts = driver.find_elements_by_class_name("timestampContent")
    posts[n].click()
    time.sleep(4)
    url = driver.current_url
    return str(url)

def get_post_public(driver,n,url):
    time.sleep(3)
    driver.get(url)
    time.sleep(4)
    WebDriverWait(driver,
5).until(EC.element_to_be_clickable((By.XPATH, "//div[@data-
testid='story-subtitle']")))
    driver.find_elements_by_xpath("//div[@data-testid='story-
subtitle']")
```

```
    posts = driver.find_elements_by_class_name("timestampContent")
    posts[n].click()
    time.sleep(4)
    url_ret = driver.current_url
    time.sleep(2)
    return str(url_ret)

def get_mobile_post(driver,n):
    go_to_profile(driver)
    url = driver.current_url
    url2 = url[12:]
    mobile_url = 'https://.m.' + url2
    driver.get(mobile_url)

    WebDriverWait(driver,
5).until(EC.element_to_be_clickable((By.XPATH, "//div[@data-
sigil='m-feed-voice-subtitle']")))
    driver.find_elements_by_xpath("//div[@data-sigil='m-feed-
voice-subtitle']")
    posts = driver.find_elements_by_tag_name("abbr")
    posts[n].click()
    time.sleep(2)
    time.sleep(4)
    url_ret = driver.current_url
    time.sleep(2)
    return str(url_ret)

def get_mobile_post_url(driver,url):
    url2 = url[12:]
    mobile_url = 'https://m.' + url2
    driver.get(mobile_url)
    time.sleep(3)
    return str(mobile_url)

def get_all_posts(driver):
    post_links = []
    n = 0
    temp = True
    while temp:
        try:
            post_links.append(get_post(driver,n))
            time.sleep(2)
            driver.back()
            time.sleep(1)
            n = n + 1
        except:
            "End of list"
            temp = False
    return post_links
```

Продовження додатку А

```
def delete_post(driver,url):
    test = get_mobile_post_url(driver,url)
    driver.get(test)
    print(test)
    time.sleep(3.5)
    print("get mobile irl success")
    driver.find_element_by_xpath('//a[@aria-
haspopup="true"]').click()
    print("find popup")
    time.sleep(3.5)
    WebDriverWait(driver,
3).until(EC.element_to_be_clickable((By.XPATH, '//a[@data-
sigil="touchable touchable removeStoryButton
enabled_action"]'))).click()
    time.sleep(3)
    print("click popup")
    buttons = driver.find_elements_by_xpath('//a[@role="button"]')
    #print(buttons)
    print("find all buutons")
    time.sleep(4)
    for button in buttons:
        if button.text=='Видалити':
            button.click()
            print("click")
            break
        elif button.text=='Delete':
            button.click()
            print("click")
            break
        elif button.text=='Удалить':
            button.click()
            print("click")
            break
    print("all good")
    time.sleep(5)


def
facebook_create_post(facebook_login=None,facebook_password=None,st
atus=None,url_image=None):
    url = None
    try:
        driver = get_driver()

        try:

facebook_login_fun(driver,facebook_login,facebook_password)
            print('success login')
```

```
        except:
            print('login failed')
        try:
            url = publish_post(driver,status,url_image=url_image)
            print('success publish')
        except:
            print('publish failed')

        exit_driver(driver)
    except:
        print("something went wrong")
            #return False
    return url

def
facebook_create_post_public(facebook_login,facebook_password,statu
s,url_image=None,url=None):
    url_ret = None
    try:
        driver = get_driver()
        try:

facebook_login_fun(driver,facebook_login,facebook_password)
            print('success login')
        except:
            print('login failed')
        try:
            url_ret =
publish_post_public(driver,url,status,url_image=url_image)
            url_ret = url_ret[:84]
            print('success publish in public')
        except:
            print('publish public failed')
        exit_driver(driver)
    except:
        print("something went wrong")
            #return False
    return url_ret

def facebook_delete_post(facebook_login,facebook_password,url):
    try:
        driver = get_driver()
        try:

facebook_login_fun(driver,facebook_login,facebook_password)
            print('success login')
        except:
            print('login failed')
        try:
```

```
            delete_post(driver,url)
            print('success delete')
        except:
            print('delete failed')
        exit_driver(driver)
    except:
        print("something went wrong, dont delete")
```

__init__.py

```
from flask import Flask, request, jsonify, make_response
import uuid
from flask_sqlalchemy import SQLAlchemy
from flask_bcrypt import Bcrypt
from flask_login import LoginManager
from flask_migrate import Migrate, MigrateCommand
from flask_script import Manager
import os
from flask_admin import Admin, BaseView, expose
from flask_admin.contrib.sqla import ModelView
from flask_ckeditor import CKEditor
from selenium import webdriver
from selenium.webdriver.chrome.options import Options

config_file='settings.py'
app = Flask(__name__)
#basedir = os.path.abspath(os.path.dirname(__file__))'



chrome_options = webdriver.ChromeOptions()
chrome_options.binary_location =
os.environ.get("GOOGLE_CHROME_BIN")
chrome_options.add_argument("--headless")
chrome_options.add_argument("--disable-dev-shm-usage")
chrome_options.add_argument("--no-sandbox")
prefs = {"profile.default_content_setting_values.notifications" :
2}
chrome_options.add_experimental_option("prefs",prefs)
driver =
webdriver.Chrome(executable_path=os.environ.get("CHROMEDRIVER_PATH
"), chrome_options=chrome_options)
```