Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра програмного забезпечення

Бакалаврська дипломна робота

на тему: «Розробка веб-орієнтованого застосунку поштового клієнта для електронної розсилки»

Виконав: студент 4 курсу
групи 4ПІ-20б
спеціальності
121 – Інженерія програмного забезпечення
(шифр і назва напряму підготовки, спеціальності)
Мельничук М.Е.
(прізвище та ініціали)
Керівник: к.т.н., доц. каф. ПЗ Рейда О.М.
(прізвище та ініціали)
Рецензент: д.т.н., проф. каф. ОТ Мартинюк Т.Б.
(прізвище та ініціали)

Допущено до захисту
Зав. кафедри Романюк О.Н.
«10» 06 2024 р.

ВНТУ – 2024

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

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
Романюк О.Н._____
« 12 » березня 2024 р.

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

Мельничуку Михайлу Едуардовичу

1. Тема роботи – «Розробка веб-орієнтованого застосунку поштового клієнта для електронної розсилки»

Керівник роботи: Рейда О.М., к.т.н., доц. кафедри ПЗ, затверджені наказом вищого навчального закладу від 11 березня 2024 р. № 80.

2. Строк подання студентом роботи 3 червня 2024 р.

3. Вихідні дані до роботи: середовище розробки PyCharm, мова розробки Python, JavaScript, мова розмітки HTML5, CSS3, база даних DynamoDB, хмарні сервіси AWS.

4. Зміст розрахунково-пояснювальної записки: вступ; аналіз стану питання розробки; порівняльний аналіз аналогів; аналіз методів розв'язання задачі; постановка задачі; аналіз інфрмаційного забезпечення, розробка структури модуля клієнта, розробка моделі модуля клієнта, розробка алгоритмів роботи програми, варіантний аналіз та вибір мови програмування, варіантний аналіз та вибір середовища розробки, розробка інтерфейсу користувача; аналіз методів тестування, тестування розробленого програмного застосунку; висновки; список використаних джерел; додатки.

5. Перелік графічного матеріалу: титульний слайд; актуальність теми; мета, об'єкт та предмет дослідження; задачі дослідження, новизна, практична цінність одержаних результатів; порівняльний аналіз аналогів, використані технології, метод, модель та алгоритми роботи програми, тестування, висновки, апробація результатів роботи і публікації.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада Консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-4	Рейда О.М., к.т.н., доцент кафедри ПЗ	13.03.2024	03.06.2024

7. Дата видачі завдання 13 березня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської дипломної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз стану розвитку вебсистем для електронної розсилки	14.03.2024 – 24.03.2024	виконано
2	Порівняльний аналіз аналогів	25.03.2024 – 07.04.2024	виконано
3	Розробка поштового клієнта	08.04.2024 – 18.04.2024	виконано
4	Розробка методу масової відправки повідомлень	19.04.2024 – 19.05.2024	виконано
5	Тестування застосунку	20.05.2024 – 24.05.2024	виконано
6	Оформлення матеріалів до захисту БДР	25.05.2024 – 30.05.24	виконано

Студент _____ Мельничук М.Е.
(підпис) (прізвище та ініціали)

Керівник бакалаврської дипломної роботи _____ Рейда О.М.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Бакалаврська дипломна робота складається з 60 сторінок формату А4, на яких є 32 рисунки, 4 таблиці, список використаних джерел містить 20 найменувань.

У бакалаврській дипломній роботі проведено аналіз стану питання розробки системи масової розсилки електронних повідомлень. Було проведено аналіз аналогів, визначено їх переваги і недоліки та доведено доцільність розробки власної системи.

Виконане обґрунтування вибору мови та середовища програмування, а також технологій, які були використані під час розробки системи масової розсилки електронних повідомлень.

Розроблено веб-орієнтований застосунок поштового клієнта для електронної розсилки.

Для розробки використано мови програмування Python та JavaScript, середовища розробки PyCharm та Visual Studio Code.

Отримані в бакалаврській дипломній роботі результати можна використати для масової розсилки електронних листів.

Ключові слова: веб-система, кінотеатр, програмне забезпечення.

ANNOTATION

The bachelor's thesis consists of 60 A4 pages, including 32 figures, 4 tables, and a list of 20 references.

The bachelor's thesis analyzes the development of a mass email distribution system. It includes an analysis of existing systems, identifying their advantages and disadvantages, and demonstrates the feasibility of developing a custom system.

The thesis justifies the choice of programming language and environment, as well as the technologies used during the development of the mass email distribution system.

A web-based mail client application for electronic mailing has been developed.

The development used the programming languages Python and JavaScript, and the development environments PyCharm and Visual Studio Code.

The results obtained in the bachelor's thesis can be used for mass email distribution.

Keywords: web system, cinema, software.

ЗМІСТ

ВСТУП 4
1 АНАЛІЗ СТАНУ РОЗВИТКУ ВЕБСИСТЕМ ДЛЯ ЕЛЕКТРОННОЇ РОЗСИЛКИ 7
1.1 Аналіз стану питання розробки вебсервісу для електронної розсилки 7
1.2 Порівняльний аналіз аналогів 10
1.3 Аналіз методів розв’язання задачі..... 15
1.4 Постановка задач дослідження 17
1.5 Висновки 17
2 РОЗРОБКА СТРУКТУРИ ТА АЛГОРИТМІВ ПРОГРАМНОГО ЗАСТОСУНКУ 18
2.1 Аналіз інформаційного забезпечення..... 18
2.2 Розробка моделі системи..... 18
2.3 Розробка методу масової відправки повідомлень..... 20
2.4 Розробка алгоритмів роботи системи 23
2.5 Висновки..... 26
3 РОЗРОБКА ПОШТОВОГО КЛІЄНТА ДЛЯ ЕЛЕКТРОННОЇ РОЗСИЛКИ 27
3.1 Варіантний аналіз та вибір мови програмування розробки 27
3.2 Варіантний аналіз та вибір середовища розробки..... 33
3.3 Розробка інтерфейсу користувача..... 42
3.4 Висновки..... 46
4 ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 47
4.1 Аналіз методів тестування..... 47
4.2 Тестування поштового клієнта для електронної розсилки 49
4.3 Висновки 57
ВИСНОВКИ 58
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 59
ДОДАТКИ..... 61
Додаток А. Технічне завдання 62
Додаток Б. Протокол перевірки на наявність текстових запозичень..... 65
Додаток В – Лістинг програми 66

Додаток Г. Ілюстративна частина 85

ВСТУП

Обґрунтування вибору теми дослідження. Сучасні методи масової відправки електронних повідомлень для різних бізнес-цілей враховують різноманітність каналів зв'язку, що значно впливає на ефективність комунікації і обмежує можливості швидкого охоплення великої аудиторії.

Для забезпечення надійної доставки повідомлень використовують власну інфраструктуру та сервери, що дозволяє контролювати процес відправки. Як результат ускладнює масштабування системи і призводить до зниження продуктивності при збільшенні кількості отримувачів.

Таким чином, існуючі методи масової відправки електронних повідомлень характеризуються суттєвою обчислювальною складністю, що в значній мірі впливає на швидкість та ефективність комунікації. Тому актуальними є питання підвищення продуктивності методів і засобів масової відправки повідомлень.

Для існуючих методів масової розсилки електронних повідомлень, характерні проблеми з масштабуванням, надійністю доставки та відсутністю гнучкості у виборі каналів зв'язку.

Підвищення ефективності та швидкості доставки повідомлень великій кількості отримувачів, що дасть можливість покращити взаємодію з клієнтами та підвищити конверсію, тому важливими є питання розробки відповідних методів з використанням хмарних технологій.

При формуванні систем масової відправки повідомлень традиційні підходи використовуються, що не враховують особливостей сучасних хмарних сервісів, оскільки має місце обмеження масштабованості та гнучкості. Тому важливими є питання розробки комбінованих методів з використанням AWS SNS для підвищення ефективності.

Тому актуальними є питання підвищення продуктивності та ефективності масової відправки повідомлень, оскільки існуючі методи власної розсилки не задовольняють потреби сучасного бізнесу в комунікації, що передбачає розробку нових методів і засобів.

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

Мета та завдання дослідження. Метою роботи є підвищення продуктивності масової відправки повідомлень за рахунок використання сервісу відправлення повідомлень для встановлення зв'язку Amazon Web Services Simple Email Service (AWS SES).

Основними задачами дослідження є:

- провести аналіз існуючих методів і засобів масової відправки електронних повідомлень для визначення напрямків підвищення продуктивності;
- запропонувати новий метод масової відправки повідомлень з використанням Amazon Web Services Simple Email Service (AWS SES);
- розробити програмні компоненти та систему масової відправки повідомлень на основі запропонованих методів;
- провести дослідження розроблених засобів масової відправки повідомлень.

Об'єкт дослідження – процес розробки системи масової відправки електронних повідомлень у системах бізнес-комунікації.

Предмет дослідження – методи та засоби розробки веб-орієнтованого застосунку поштового клієнта для електронної розсилки з використанням Amazon Web Services Simple Email Service (AWS SES).

Методи дослідження. У процесі досліджень використовувались: теорія чисел та чисельних методів, теорія розподілених систем, методи для розробки моделей масової комунікації; комп'ютерне моделювання для аналізу та перевірки отриманих теоретичних положень.

Наукова новизна отриманих результатів. Подальшого розвитку отримав метод масової відправки повідомлень, який на відміну від існуючих, полягає у використанні шаблонів повідомлень зі змінними даними заповнення, та хмарного сервісу AWS SES для забезпечення автентичності відправлених повідомлень, що досягається через метод аутентифікації DKIM, і як наслідок, дає можливість підвищити користувацький досвід аудиторії та безпеку електронної комунікації.

Практична цінність отриманих результатів. Практична цінність одержаних результатів полягає в тому, що на основі отриманих в бакалаврській дипломній роботі теоретичних положень було проведено аналіз методів розсилки повідомлень, аналіз засобів розсилки електронних повідомлень. Результати аналізу зведені у таблиці.

Особистий внесок здобувача. Усі наукові результати, викладені у бакалаврській дипломній роботі, отримані автором особисто. У роботі [2], виконаній у співавторстві, автору належить аналіз поштових клієнтів розсилки електронних повідомлень.

Апробація результатів роботи. Описані у роботі положення доповідались на конференціях «LIII Науково-технічна конференція підрозділів Вінницького національного технічного університету (2024)».

Публікації. Результати роботи були опубліковані в матеріалах конференцій: LIII Науково-технічна конференція підрозділів Вінницького національного технічного університету (2024) [2].

1 АНАЛІЗ СТАНУ РОЗВИТКУ ВЕБСИСТЕМ ДЛЯ ЕЛЕКТРОННОЇ РОЗСИЛКИ

1.1 Аналіз стану питання розробки вебсервісу для електронної розсилки

Розробка вебсервісу для електронної розсилки є актуальним питанням для багатьох компаній та організацій, які прагнуть ефективно комунікувати зі своїми клієнтами, партнерами або підписниками. Такий сервіс дозволяє автоматизувати процес масової розсилки електронних листів, економлячи час та ресурси [2].

Електронна пошта залишається одним з найефективніших методів для побудови систем розсилок. Основний протокол, що використовується для відправки повідомлень, - це SMTP (Simple Mail Transfer Protocol). Він забезпечує передачу даних між різними поштовими серверами. На стороні клієнта для отримання повідомлень використовуються протоколи POP3 (Post Office Protocol) або IMAP (Internet Message Access Protocol). POP3 зазвичай завантажує повідомлення на пристрій користувача, а IMAP синхронізує повідомлення між серверами та пристроями. Для масових розсилок часто використовують ESP (Email Service Providers) такі як MailChimp, SendGrid, або Amazon SES. Ці сервіси надають інструменти для управління підписками, сегментації аудиторії, персоналізації повідомлень та аналітики.

SMS (Short Message Service) є ефективним способом швидкого досягнення користувачів на їхні мобільні телефони. Для реалізації SMS-розсилок використовують API операторів мобільного зв'язку або агрегаторів SMS. Агрегатори дозволяють відправляти повідомлення через різних операторів з єдиного інтерфейсу. Основний протокол, що використовується, - це SMPP (Short Message Peer-to-Peer), який дозволяє обмінюватися SMS між різними системами. Також широко використовуються HTTP API, які простіші в інтеграції. SMS-розсилки особливо ефективні для термінових повідомлень, оскільки вони зазвичай читаються протягом кількох хвилин після отримання.

Прикладом сервісу розсилки SMS-повідомлень є Textmagic, який наведено на

рисунку 1.1.

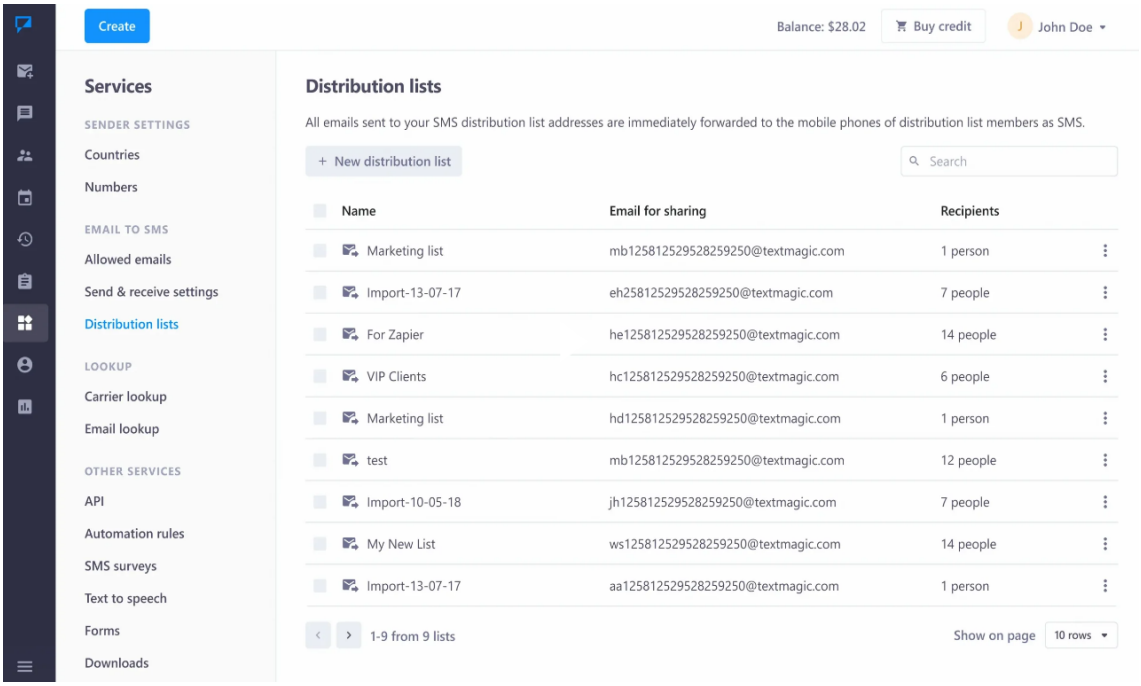

Рисунок 1.1 – Розсилка SMS-повідомлень у TextMagic

Push-повідомлення – це короткі сповіщення, що відправляються на мобільні пристрої або веб-браузери, навіть якщо додаток або сайт не активні.

Діаграма послідовності роботи Push-повідомлень наведено на рисунку 1.2.

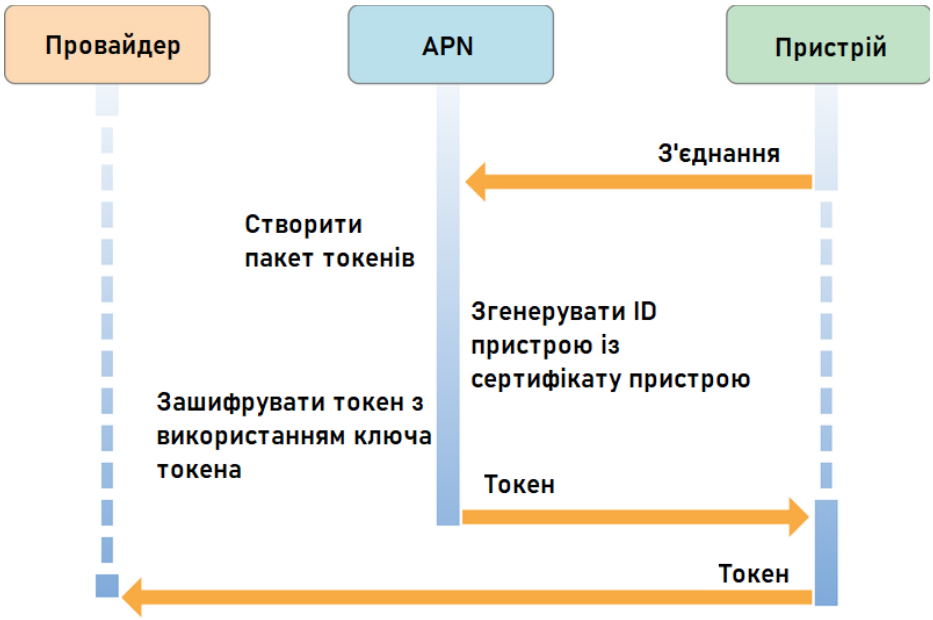

Рисунок 1.2 – Діаграма послідовності роботи Push-повідомлень

Для Android-пристроїв використовується FCM (Firebase Cloud Messaging) від Google, а для iOS - APNS (Apple Push Notification Service). Ці сервіси дозволяють відправляти тексти, зображення, та навіть дії, які можна виконати прямо з повідомлення. Push-повідомлення особливо ефективні для повторного залучення користувачів, нагадувань та миттєвих оновлень. Сервіси як OneSignal або Urban Airship надають додаткові можливості, такі як A/B тестування та аналітика.

Месенджери, такі як WhatsApp, Telegram, Facebook Messenger, стали популярною платформою для розсилок. Вони пропонують API для створення ботів, які можуть автоматично відповідати на повідомлення користувачів, відправляти новини або транзакційні сповіщення. Наприклад, Telegram Bot API дозволяє створювати ботів з багатим функціоналом, включаючи інтерактивні клавіатури. Facebook Messenger API підтримує чат-боти з можливістю прийому платежів.

RSS (Really Simple Syndication) - це формат на основі XML для публікації часто оновлюваного контенту, такого як блоги або новинні сайти. Контент генерується у вигляді XML-файлів у форматі RSS або його наступнику Atom. Користувачі підписуються на ці фіди через RSS-рідери, які автоматично перевіряють наявність нових записів. Перевага RSS в тому, що користувачі самі контролюють, що і коли вони читають, і не потрібно розкривати їхні контактні дані. RSS особливо цінний для блогерів, новинних сайтів та подкастів.

Взаємодія веб-браузерів, веб-серверів та RSS каналів наведена на рисунку 1.3.

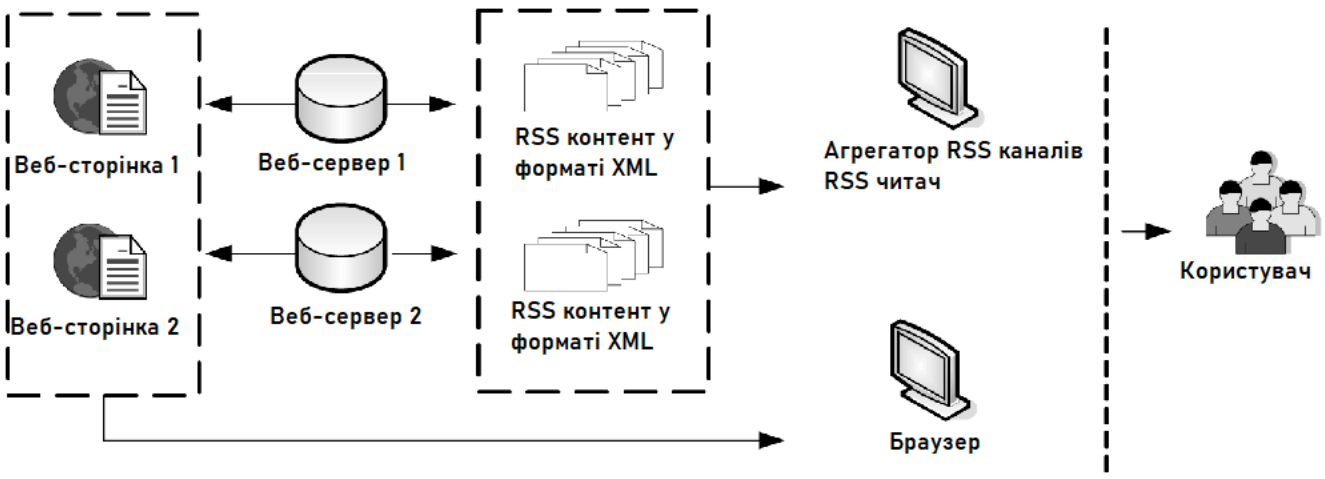

Рисунок 1.3 – Взаємодія RSS із веб-браузерами, серверами та користувачами

Ефективні розсилки ґрунтуються на розумінні аудиторії. CRM-системи (Customer Relationship Management) використовуються для зберігання даних про користувачів: їхню поведінку, інтереси, історію взаємодій.

Один із ключових аспектів створення вебсервісу для розсилки – це забезпечення зручного інтерфейсу для керування списками адрес, складання змісту листів та налаштування параметрів розсилки. Важливо також передбачити можливість сегментації аудиторії за різними критеріями для більш персоналізованого звернення.

Крім того, сучасний вебсервіс має надавати інструменти для аналізу ефективності розсилки, такі як відстеження відкриттів та прочитань листів, переходів за посиланнями тощо. Ці дані допоможуть оптимізувати майбутні кампанії та покращити залученість аудиторії.

Не слід забувати і про питання безпеки та конфіденційності даних. Вебсервіс повинен гарантувати захист персональної інформації підписників та забезпечувати відповідність діяльності з розсилки чинним нормативним актам, зокрема у сфері захисту персональних даних.

Розробка вебсервісу для електронної розсилки вимагає ретельного підходу до проектування архітектури, вибору технологій, реалізації функціоналу та інтеграції з іншими системами. Успішна імплементація такого сервісу може значно підвищити ефективність комунікацій компанії з її цільовою аудиторією.

1.2 Порівняльний аналіз аналогів

Розглянемо існуючі застосунки для електронної розсилки.

MailChimp [3] – це потужний вебсервіс для електронної розсилки, який користується великою популярністю серед компаній різних розмірів та сфер діяльності. Його функціонал дозволяє створювати, налаштовувати та відстежувати ефективність email-маркетингових кампаній.

Однією з ключових переваг MailChimp є його інтуїтивний та зручний інтерфейс, який спрощує процес створення та редагування листів. Сервіс пропонує

широкий вибір шаблонів дизайну, які можна легко персоналізувати відповідно до потреб користувача. Крім того, MailChimp дозволяє сегментувати аудиторію за різними критеріями, такими як демографічні дані, інтереси чи поведінка, що сприяє більш цілеспрямованому та ефективному таргетингу.

MailChimp також забезпечує детальні звіти та аналітику, які допомагають відстежувати ключові показники ефективності розсилки, такі як коефіцієнти відкриттів, кліків, відписок тощо. Ця інформація є корисною для оптимізації майбутніх кампаній та покращення залученості аудиторії. Крім того, сервіс пропонує потужні інструменти автоматизації, що дозволяють налаштовувати розсилки на основі певних тригерів або поведінки користувачів.

Варто відзначити, що MailChimp приділяє значну увагу питанням безпеки та конфіденційності даних. Сервіс відповідає вимогам законодавства щодо захисту персональних даних та забезпечує шифрування інформації під час передачі та зберігання. Крім того, MailChimp пропонує різні тарифні плани, що дозволяє обрати найбільш оптимальний варіант відповідно до розміру бізнесу та потреб в електронній розсилці.

Застосунок MailChimp продемонстровано на рисунку 1.4.

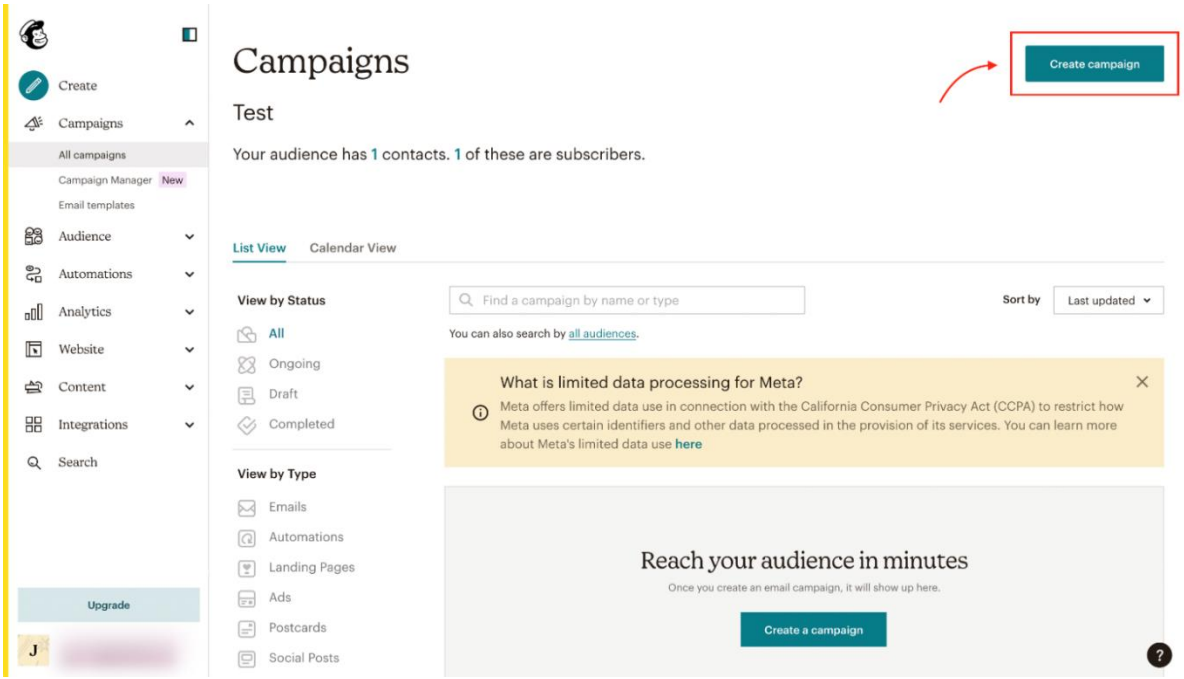

Рисунок 1.4 – Інтерфейс Mailchimp

SendPulse [4] – це багатофункціональний вебсервіс для електронних розсилок, SMS-розсилок та веб-push-сповіщень. Він надає комплексні рішення для ефективних маркетингових комунікацій з клієнтами.

Одна з ключових переваг SendPulse – це зручний інтерфейс для створення та налаштування email-розсилок. Сервіс пропонує широкий вибір шаблонів дизайну, які користувачі можуть легко персоналізувати відповідно до потреб своїх кампаній. Крім того, SendPulse дозволяє сегментувати списки підписників за різними критеріями, такими як демографічні дані, інтереси чи поведінка, що забезпечує більш цілеспрямовану комунікацію.

SendPulse також пропонує потужні інструменти для аналізу ефективності розсилок. Користувачі можуть відстежувати ключові показники, такі як коефіцієнти відкриттів, кліків, відписок, а також конверсії та дохід від кампаній. Ця аналітика допомагає оптимізувати майбутні розсилки та підвищити залученість аудиторії.

Застосунок SendPulse продемонстровано на рисунку 1.5.

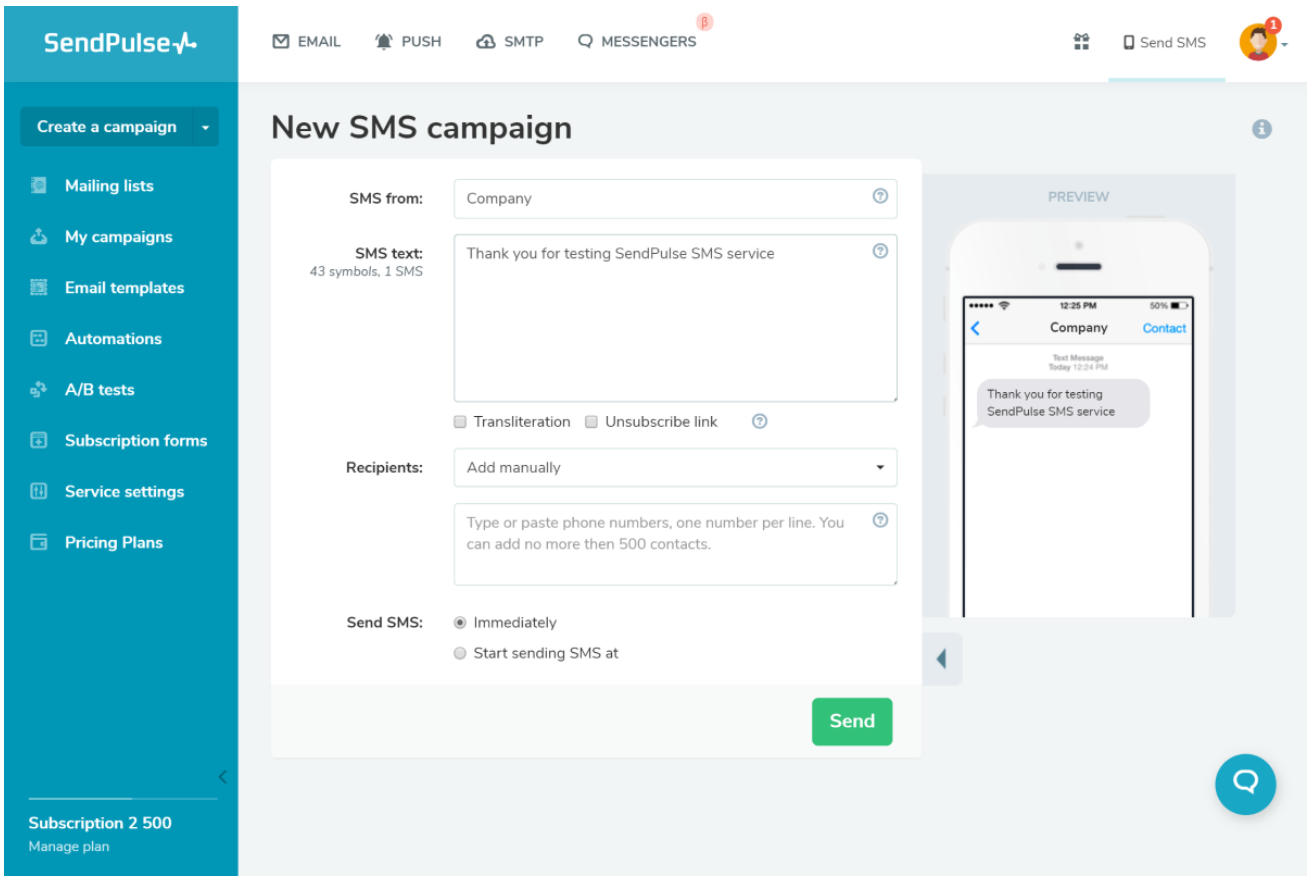

Рисунок 1.5 – Інтерфейс SendPulse

Окрім електронних розсилок, SendPulse також пропонує можливість здійснювати SMS-розсилки та веб-push-сповіщення. Ці канали комунікації можуть бути корисними для різних сценаріїв, таких як нагадування, оповіщення про акції чи розсилка транзакційних повідомлень. SendPulse забезпечує захист персональних даних та конфіденційність інформації відповідно до вимог законодавства, пропонуючи різні тарифні плани, що дозволяють обрати найбільш оптимальний варіант для потреб бізнесу.

GetResponse [5] – це потужна маркетингова платформа для електронних розсилок, автоматизації маркетингу та створення лендингів. Вона пропонує всеохоплюючі рішення для малого та середнього бізнесу для покращення взаємодії з клієнтами та збільшення конверсій.

Основною функцією GetResponse є створення та управління email-маркетинговими кампаніями. Платформа має зручний drag-and-drop редактор для розробки професійних емейл-шаблонів та автоматизованих послідовностей розсилок. Користувачі можуть сегментувати своїх підписників за різними критеріями для більш персоналізованих повідомлень. GetResponse також пропонує аналітику з деталізованими звітами про показники ефективності кампаній, такі як відкриття, клікі, конверсії тощо.

Окрім розсилок, GetResponse надає інструменти для створення лендингів, веб-форм та маркетингових воронок. Вбудований конструктор дозволяє швидко розробляти привабливі лендинги без навичок програмування. Платформа також підтримує автоматизацію маркетингових процесів за допомогою налаштовуваних робочих процесів та тригерів на основі дій користувачів.

GetResponse приділяє велику увагу безпеці та конфіденційності даних. Платформа відповідає стандартам захисту даних та пропонує різні тарифні плани, адаптовані до потреб бізнесу будь-якого розміру. Крім того, GetResponse має широкі можливості інтеграції з популярними інструментами, такими як CRM, CMS, соціальні мережі тощо, що робить її гнучким рішенням для комплексних маркетингових стратегій.

Застосунок GetResponse продемонстровано на рисунку 1.6.

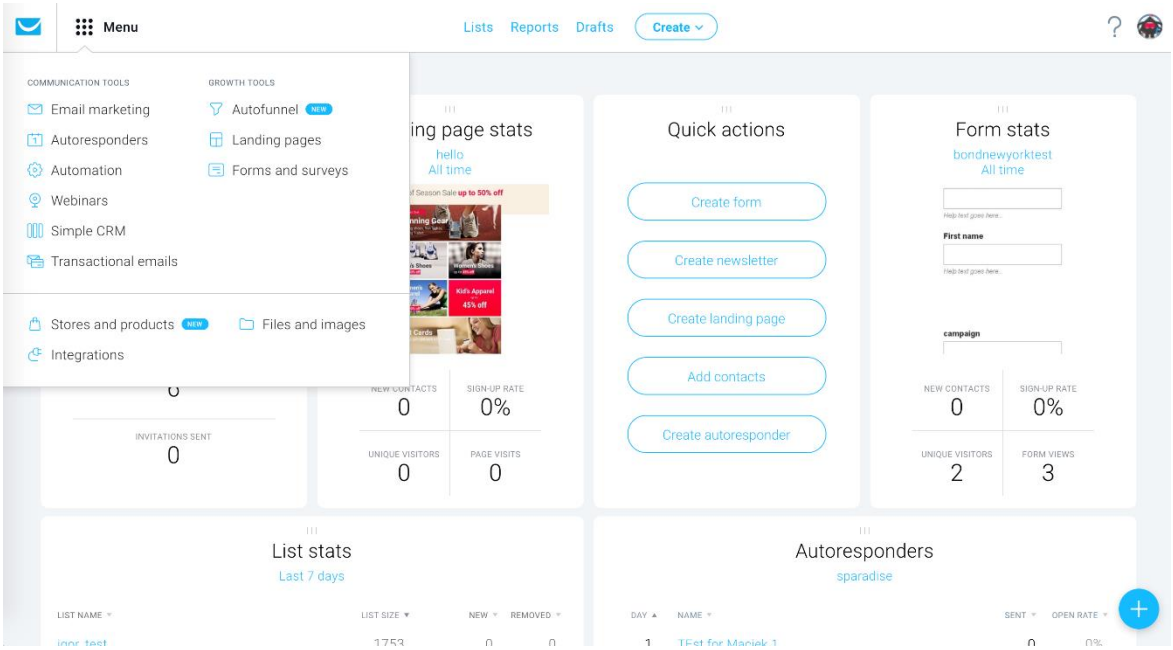

Рисунок 1.6 – Інтерфейс GetResponse

На основі аналізу аналогів, було сформовано таблицю 1.1 порівняння характеристик.

Таблиця 1.1 – Порівняння характеристик аналогів

	GetResponse	MailChimp	SendPulse	Власна розробка
Формування груп адресатів	1	0	1	1
Фільтрація адресатів відповідно до критеріїв	1	1	1	1
Планування розсилок	0	1	1	1
Управління контентом	1	1	0	1
Підтримка багатомовних розсилок	1	1	1	1
Аналітика та звітність	1	0	1	1
Тестування розсилок	1	1	1	1
Сумарний коефіцієнт	6	5	6	7

Таким чином, власна розробка є актуальною та випереджає аналоги за характеристиками.

1.3 Аналіз методів розв'язання задачі

Розглянемо аналіз методів розробки вебсервісу поштового клієнта для електронної розсилки. Спочатку визначимо основні вимоги та функціональність такого веб-сервісу:

1. Зберігання та управління списками розсилки (додавання, видалення, редагування).
2. Створення та редагування шаблонів електронних листів.
3. Планування та відправлення електронних розсилок.
4. Відстеження та аналітика (відкриття листів, переходи за посиланнями).
5. Інтеграція з зовнішніми сервісами (CRM, аналітика тощо).

Тепер проаналізуємо підходи розробки з точки зору монолітної архітектури.

Переваги монолітної архітектури:

1. Простота розробки та розгортання – монолітний додаток є єдиним цілим, що спрощує розробку, тестування та розгортання. Немає необхідності керувати декількома службами та їх взаємодією.
2. Продуктивність – оскільки всі компоненти знаходяться в одному процесі, монолітний додаток може бути більш продуктивним, ніж розподілена система через відсутність міжсервісних викликів.
3. Контроль над залежностями – у монолітній системі легше контролювати та керувати залежностями між різними компонентами, оскільки вони знаходяться в одному місці.
4. Масштабованість – монолітний додаток можна масштабувати горизонтально, додаючи більше екземплярів для обробки більшого навантаження.

Недоліки монолітної архітектури:

1. Складність внесення змін. Оскільки весь додаток є одним цілим, будь-які зміни вимагають повного розгортання всього додатка. Це може бути проблемою для

великих систем з частими оновленнями.

2. Обмежена масштабованість. Хоча монолітний додаток можна масштабувати горизонтально, він все одно буде мати обмеження щодо масштабованості через обмеження ресурсів на окремому сервері.

3. Нездатність використовувати різні технології. Монолітна архітектура обмежує використання різних технологій для різних компонентів, оскільки вся система повинна бути написана однією мовою програмування.

4. Залежність від середовища. Монолітний додаток часто залежить від певного середовища, що ускладнює його перенесення та розгортання на різних платформах.

Незважаючи на ці недоліки, монолітна архітектура може бути найкращим вибором для вебсервісу поштового клієнта для електронної розсилки з наступних причин:

1. Проста функціональність. Вебсервіс поштового клієнта має відносно просту функціональність, яка може бути реалізована в межах монолітної системи без значних проблем.

2. Компактний масштаб. Для більшості клієнтів електронної розсилки масштаб буде відносно невеликим, що робить монолітну архітектуру цілком прийнятною.

3. Швидкий запуск та ітерації. Монолітна архітектура дозволяє швидко розробляти, тестувати та випускати нові версії продукту, що є важливим для проектів, які потребують частих ітерацій.

4. Простота розгортання та обслуговування. Розгортання та обслуговування монолітного додатка є простішим, ніж керування розподіленою системою з декількома сервісами.

Отже, хоча монолітна архітектура має свої недоліки, для вебсервісу поштового клієнта для електронної розсилки вона може бути найкращим вибором завдяки відносно простій функціональності, обмеженому масштабу та вимогам до швидкого циклу розробки та ітерацій.

1.4 Постановка задач дослідження

В результаті аналізу стану питання розробки вебсервісу поштового клієнта для електронної розсилки, порівняльного аналізу аналогів та аналізу методів розв’язання задачі, було поставлено такі задачі дослідження:

– провести аналіз сучасного стану питання розробки та обґрунтування задач;

– розробити модель системи розсилки електронних повідомлень;

– розробити інтерфейс користувача для поштового клієнта для електронної розсилки;

– розробити метод створення шаблонів електронних листів;

– розробити алгоритми роботи вебсистеми;

– розробити функціонал для формування шаблонів електронних листів, списків розсилки та розсилки електронних листів;

– розробити інтерфейс користувача застосунку;

– провести тестування веб-застосунку.

1.5 Висновки

У першому розділі було проведено аналіз стану питання розробки вебсервісу поштового клієнта для електронної розсилки, порівняльний аналіз аналогів, аналіз методів розв’язання задачі. На їх основі було доведено доцільність та актуальність власної розробки, поставлено задачі дослідження

2 РОЗРОБКА СТРУКТУРИ ТА АЛГОРИТМІВ ПРОГРАМНОГО ЗАСТОСУНКУ

2.1 Аналіз інформаційного забезпечення

Застосунки для електронної розсилки збирають різні дані про користувачів, включаючи контактну інформацію, таку як імена, електронні адреси, номери телефонів та адреси доставки. Також можуть збиратися дані про поведінку користувачів, такі як відкриття листів, кліки по посиланнях та інші дії, які виконуються користувачами в рамках розсилки.

Дані про користувачів можуть бути зібрані різними способами, включаючи реєстрацію на сайті, підписку на розсилку, заповнення форм зворотного зв'язку, купівлю товарів або послуг та інші дії, які виконуються користувачами. Також можуть збиратися дані з соціальних мереж, зовнішніх баз даних та інших джерел.

Застосунки для електронної розсилки також генерують дані, такі як статистика розсилки, звіти про результати розсилки, дані про відкриття листів, кліки по посиланнях та інші дії користувачів. Ці дані можуть бути використані для аналізу ефективності розсилки, визначення цільової аудиторії, оптимізації розсилки та інших цілей.

Дані, зібрані та згенеровані застосунками для електронної розсилки, можуть бути оброблені різними способами. Наприклад, дані про користувачів можуть бути оброблені для сегментації бази даних, персоналізації розсилки, цільового маркетингу та інших цілей. Дані про результати розсилки можуть бути оброблені для аналізу ефективності розсилки, визначення цільової аудиторії, оптимізації розсилки та інших цілей.

2.2 Розробка моделі системи

Розробка моделі програмного застосунку є важливим етапом у процесі створення програмного забезпечення. Модель допомагає візуалізувати та

спроектувати структуру, поведінку та взаємодії компонентів застосунку.

Було розроблено модель системи, яка наведена на рисунку 2.1.

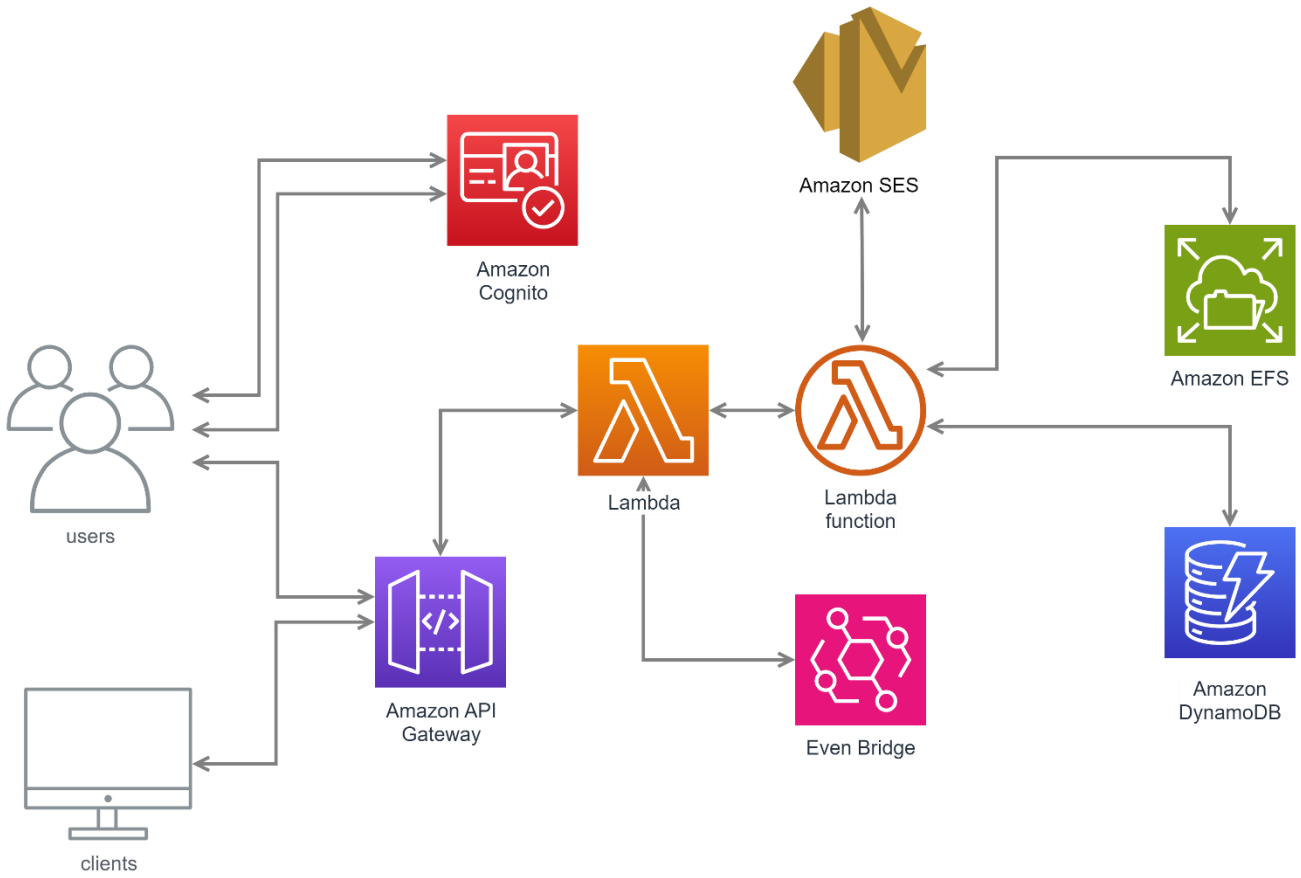

Рисунок 2.1 – Модель системи

Складові системи:

1. Users. Кінцеві користувачі, які взаємодіють з системою через клієнтські програми.

2. Amazon Cognito [6]. служба для керування користувачами та контролем доступу. Забезпечує автентифікацію, авторизацію, федерацію ідентифікаторів.

3. Amazon API Gateway [7]. керований AWS API-шлюз, який працює як "вхідні двері" для додатків, перенаправляє запити до інших сервісів.

4. Lambda функції. Serverless функції для виконання коду без управління інфраструктурою. Один Lambda взаємодіє з Cognito, інший - з подіями від SES, S3 та Event Bridge.

5. Amazon SES [8]. простий поштовий сервіс для відправки та приймання електронних листів.

6. Amazon EFS [9]. файлове сховище даних для зберігання файлів, резервних копій, логів тощо.

7. Amazon DynamoDB [10]. повністю керована NoSQL база даних для зберігання та запитів структурованих даних з високою продуктивністю.

8. AWS Event Bridge [11]. сервіс для керування подіями з усієї AWS інфраструктури. Lambda функція реагує на події з SES, S3 та Event Bridge.

Модель демонструє розподілену монолітну архітектуру з різними керованими сервісами AWS для автентифікації, API, обробки даних, сховищ та обробки подій без управління фізичними серверами.

2.3 Розробка методу масової відправки повідомлень

Створення шаблонів електронних листів - це процес створення стандартного дизайну та змісту для електронних листів, які можуть бути використані для масової розсилки або для швидкого створення нових листів. Шаблони електронних листів дозволяють заощадити час та зусилля, оскільки не потрібно створювати кожен лист з нуля.

Для налаштування параметрів розсилки листа використовується мова програмування YAML.

YAML – це зручний для читання формат серіалізації даних, який використовується для конфігураційних файлів та обміну даними між різними мовами програмування. Основною перевагою YAML є його простота та зрозумілість: він використовує відступи для позначення вкладеності та не потребує складних синтаксичних конструкцій, таких як лапки та фігурні дужки, що робить його легким для читання і редагування людиною.

Для створення же наповнення листа для розсилки використовується Liquid.

Liquid – це потужна і гнучка шаблонна мова програмування, розроблена компанією Shopify для використання в їх платформі електронної комерції. Liquid

дозволяє розробникам створювати динамічний вміст, який може бути зручно інтегрований у HTML-код веб-сторінок. Основною перевагою Liquid є його здатність дозволяти користувачам використовувати змінні, цикли та умовні вирази, що значно спрощує процес створення персоналізованих шаблонів для різних сторінок і продуктів.

Liquid дозволяє додавати до шаблонів листів змінні, які будуть заповнюватися даними користувачів із списку розсилки при відправці листів. Таким чином можна легко персоналізувати листи при розсилці.

Блок-схема алгоритму створення шаблонів електронних листів наведена на рисунку 2.2.

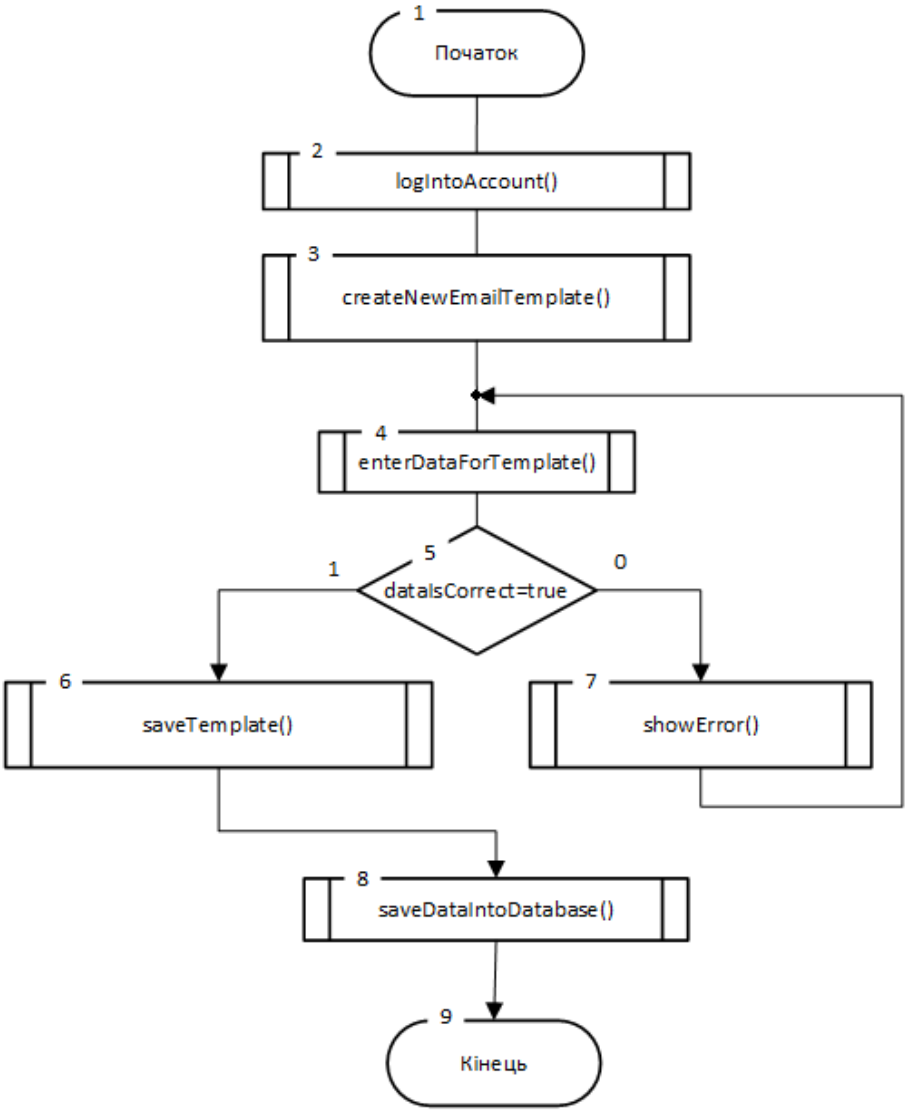

Рисунок 2.2 – Блок-схема алгоритму створення шаблонів електронних листів

Алгоритм масової відправки повідомлень:

1. Користувач входить до свого акаунту в вебсервісі поштового клієнта.
2. Користувач створює новий шаблон електронного листа, вказуючи його назву, тему листа, текст повідомлення та динамічні поля (наприклад, ім'я одержувача, дату тощо).
3. Система перевіряє коректність введених даних (перевірка тексту повідомлення, динамічних полів тощо).
4. Якщо дані коректні, система зберігає шаблон електронного листа в базі даних.
5. При створенні нової розсилки користувач може вибрати збережений шаблон та вказати список одержувачів.
6. Користувач ініціює розсилку листів.
7. Система проводить розсилку листів.

Для підтвердження автентичності надісланих листів, а також підвищення якості класифікації та ідентифікації легітимної електронної пошти використовується метод DKIM.

DKIM (DomainKeys Identified Mail) — це метод аутентифікації електронної пошти, який дозволяє одержувачам перевіряти, чи справді електронний лист було надіслано з домену, з якого він стверджує своє походження, і чи його вміст не був змінений під час передачі. Це досягається шляхом додавання цифрового підпису до заголовків повідомлення, яка створюється з використанням приватного ключа відправника. Відповідний публічний ключ розміщується у DNS-записах домену відправника, що дозволяє одержувачам перевіряти підпис за допомогою цього ключа.

Основною перевагою DKIM є підвищення рівня безпеки електронної пошти, оскільки він допомагає запобігати спуфінгу (підробці) адрес відправників, що є поширеним методом шахрайства. Підтвердження автентичності повідомлень також сприяє підвищенню довіри до відправника, що особливо важливо для бізнесу і комунікацій з клієнтами. Крім того, DKIM допомагає покращити доставку електронної пошти, оскільки багато поштових сервісів враховують наявність

аутентифікації при визначенні, чи вважати листи спамом.

Алгоритм роботи DKIM наведено на рисунку 2.3.

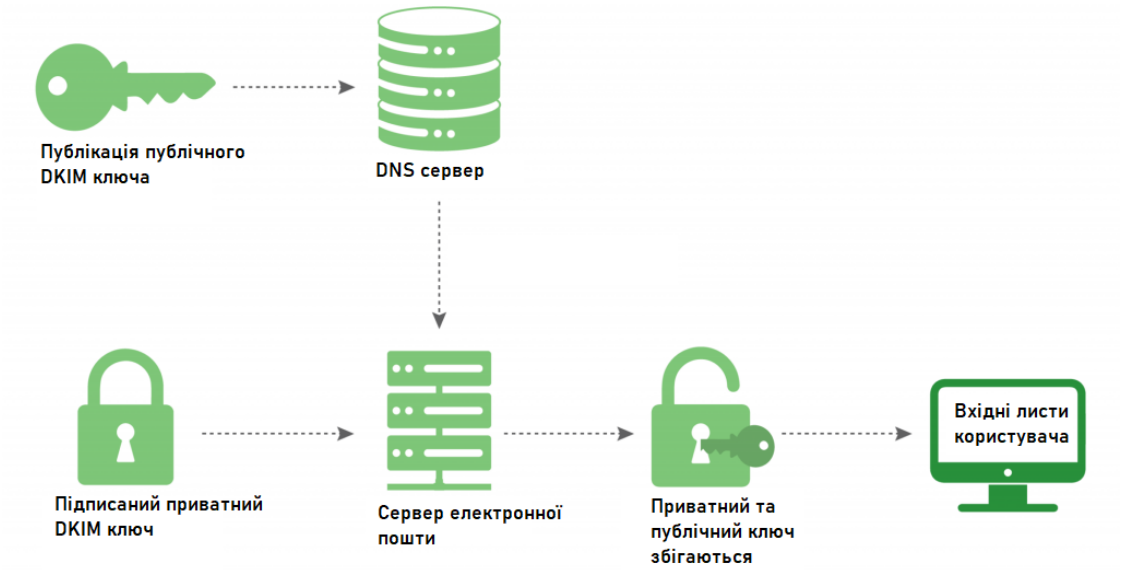

Рисунок 2.3 – Алгоритм роботи DKIM

2.4 Розробка алгоритмів роботи системи

Головним алгоритмом є алгоритм створення та відправки електронних листів. Він реалізує головний базовий функціонал веб системи та дозволяє створювати розсилки листів, формувати списки для розсилки.

Опис алгоритму:

1. Користувач входить до свого акаунту в вебсервісі поштового клієнта.

2. Користувач створює нову електронну розсилку, вказуючи тему листа, текст повідомлення та список одержувачів.

3. Система перевіряє коректність введених даних (перевірка електронних адрес, тексту повідомлення тощо).

4. Якщо дані коректні, система формує електронні листи для кожного одержувача.

5. Система відправляє створені листи через SMTP-сервер.

6. Після успішної відправки, система зберігає інформацію про розсилку та статус відправки в базі даних.

Блок-схема алгоритму роботи створення та відправки електронних листів наведено на рисунку 2.4.

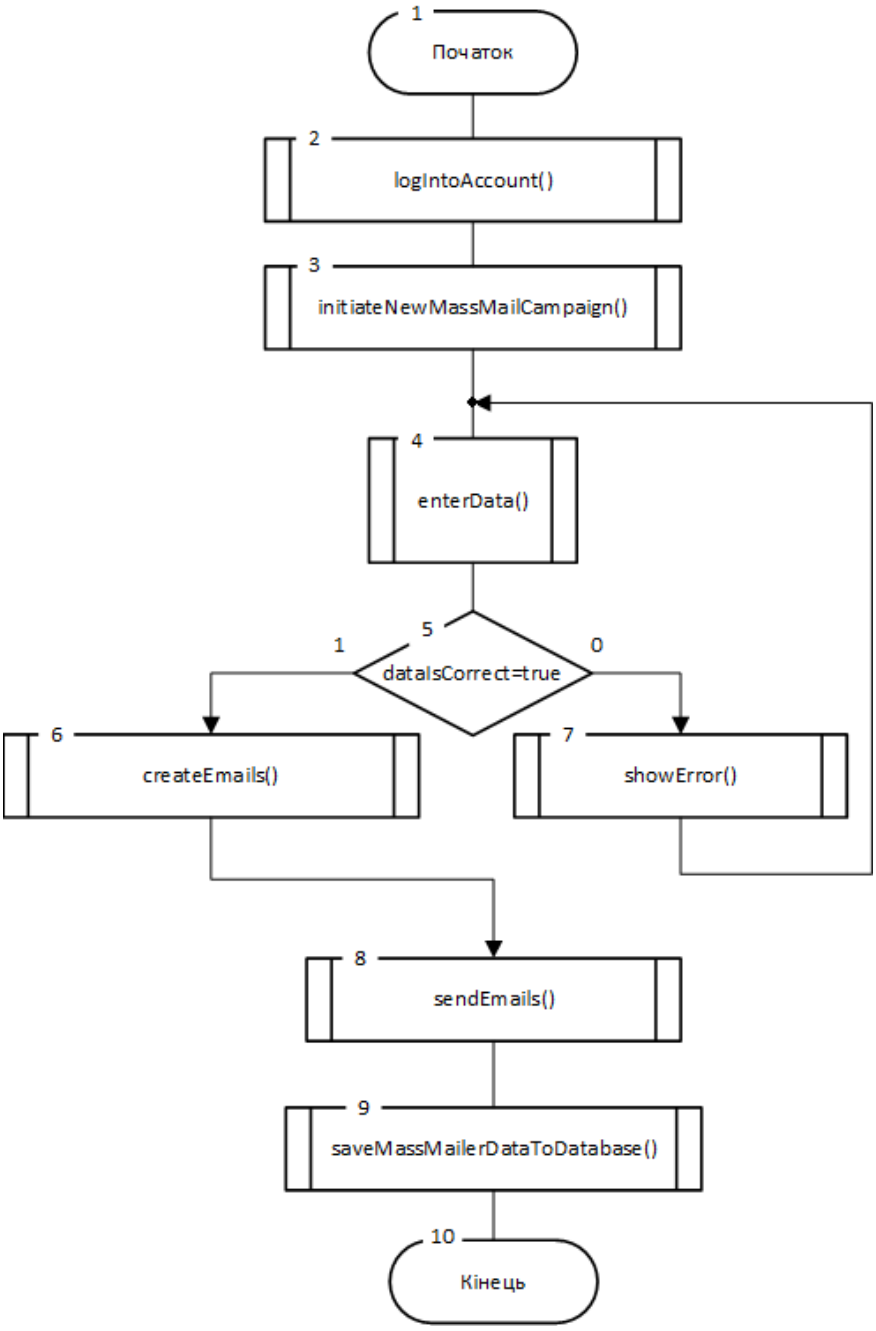

Рисунок 2.4 – Блок-схема алгоритму створення та відправки електронних листів

Розроблено також алгоритм автоматичної відправки електронних листів за розкладом. Він дозволяє створювати листи та надсилати їх користувачам саме в той

час, коли це зручно чи більш доцільно.

Блок-схема алгоритму роботи автоматичної відпраки електронних листів наведено на рисунку 2.5.

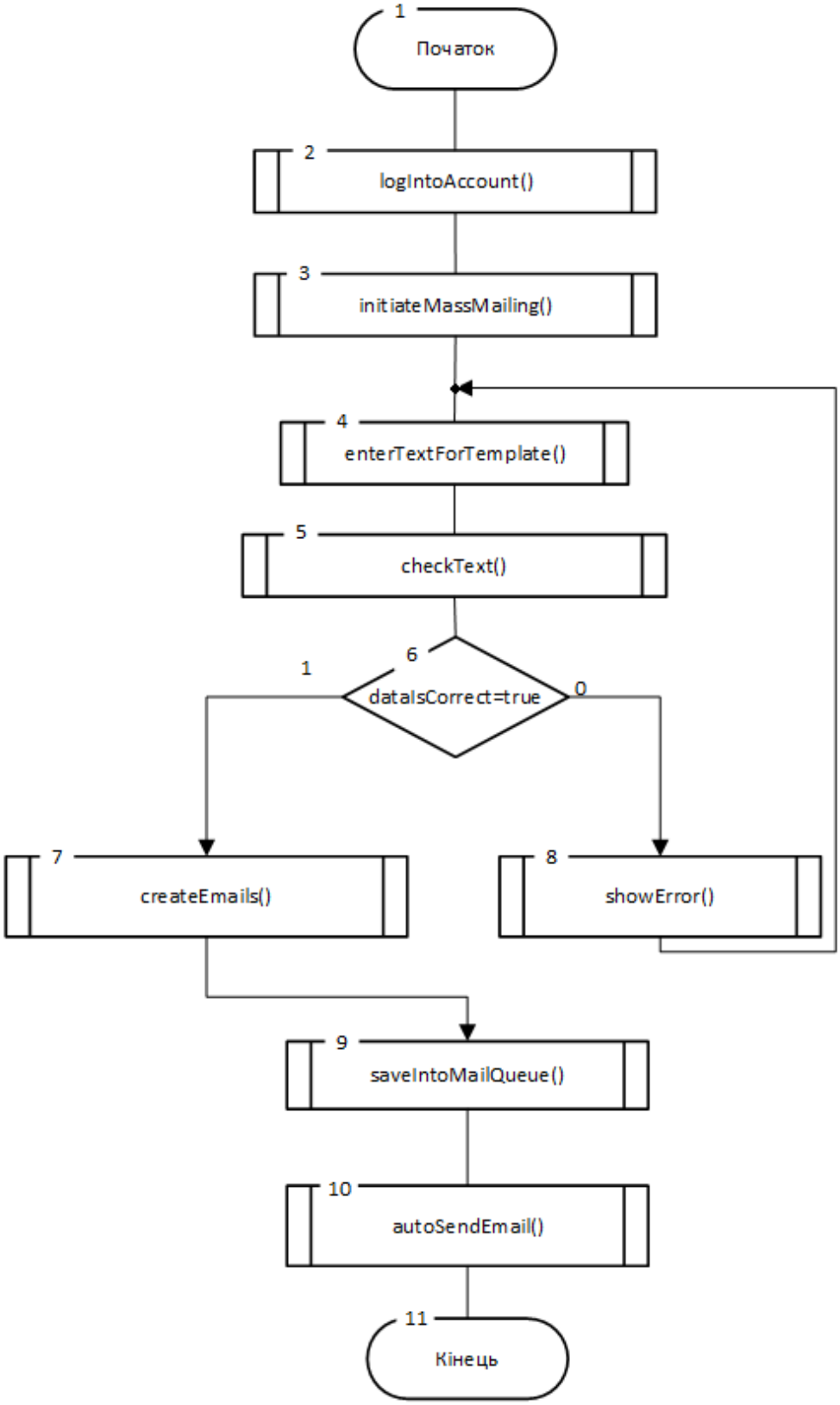

Рисунок 2.5 – Блок-схема алгоритму роботи автоматичної відпраки електронних

листів

Опис алгоритму:

1. Користувач входить до свого акаунту в вебсервісі поштового клієнта.
2. Користувач створює нову електронну розсилку, вказуючи тему листа, текст повідомлення, список одержувачів та час відправки.
3. Система перевіряє коректність введених даних (перевірка електронних адрес, тексту повідомлення, часу відправки тощо).
4. Якщо дані коректні, система формує електронні листи для кожного одержувача та зберігає їх у черзі відправки.
5. Система автоматично відправляє листи через SMTP-сервер у вказаний час.
6. Після успішної відправки, система зберігає інформацію про розсилку та статус відправки в базі даних.

2.5 Висновки

У другому розділі було проведено аналіз інформаційного забезпечення системи, модель системи, метод створення шаблонів електронних листів, алгоритми роботи вебсистеми для розсилки електронних листів.

3 РОЗРОБКА ПОШТОВОГО КЛІЄНТА ДЛЯ ЕЛЕКТРОННОЇ РОЗСИЛКИ

3.1 Варіантний аналіз та вибір мови програмування розробки

Вибір мови програмування для розробки вебсервісу поштового клієнта для електронної розсилки залежить від різних факторів, таких як досвід розробників, вимоги до продуктивності, масштабованості, надійності та зручності в роботі з відповідними бібліотеками та фреймворками.

Розглянемо деякі популярні мови програмування.

Java [12] - це об'єктно-орієнтована мова програмування, яка була розроблена компанією Sun Microsystems (нині належить Oracle) у 1995 році. Java є однією з найпопулярніших мов програмування у світі, яка широко використовується для створення різних типів програмного забезпечення, включаючи вебзастосунки, мобільні додатки, настільні програми та вбудовані системи.

Java є потужною та гнучкою мовою програмування, яка підходить для широкого спектра завдань, включаючи розробку вебсервісу поштового клієнта для електронної розсилки.

Архітектура Java продемонстрована на рисунку 3.1.

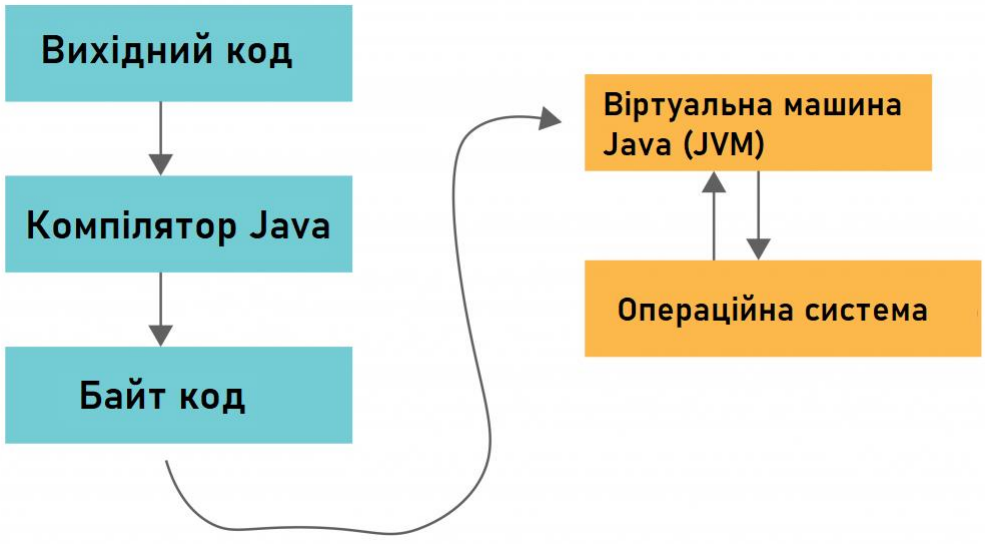

Рисунок 3.1 – Архітектура Java

Python — високорівнева мова програмування загального призначення, яка вирізняється своєю простотою та легкістю у навчанні.

Python надає підтримку для кількох програмних парадигм, включаючи об'єктно-орієнтоване, процедурне та функціональне програмування. Основна ідея мови — підвищення продуктивності розробників за рахунок читабельного і компактного коду. Завдяки цьому, Python став популярним серед новачків і досвідчених програмістів, а також використовується у різних сферах, таких як веб-розробка, наукові дослідження, аналіз даних, штучний інтелект і автоматизація.

Архітектура Python наведена на рисунку 3.2.

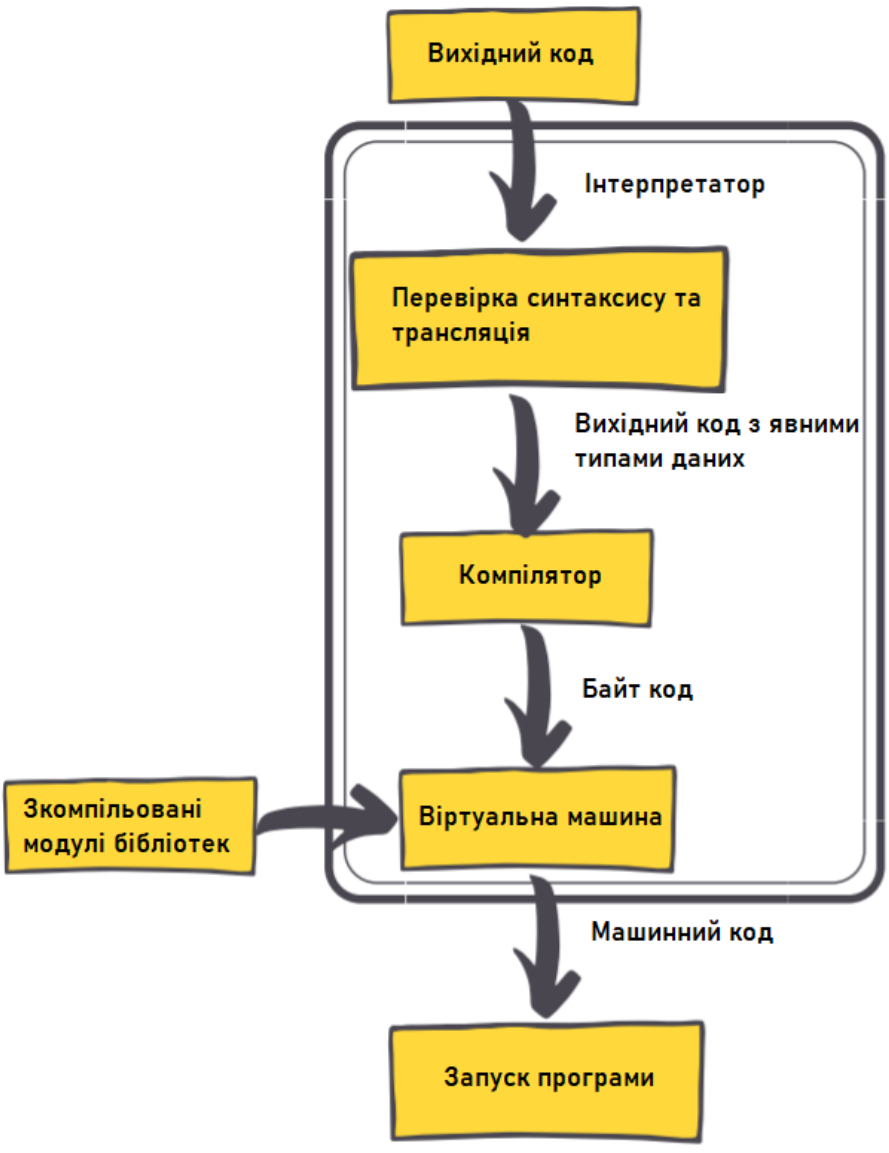

Рисунок 3.2 – Архітектура Python

Однією з головних переваг Python є його велика стандартна бібліотека, яка забезпечує широкий набір інструментів для вирішення найрізноманітніших завдань. Крім того, Python має багату екосистему сторонніх бібліотек і фреймворків, таких як Django та Flask для веб-розробки, NumPy і pandas для обробки даних, TensorFlow і PyTorch для машинного навчання. Це дозволяє швидко створювати і розгортати проекти, не витрачаючи багато часу на написання базового функціоналу. Python також відомий своєю кросплатформеністю, що дозволяє використовувати його на різних операційних системах без значних змін у коді.

Важливою особливістю Python є його інтерактивний режим, що дозволяє виконувати код рядок за рядком, що особливо корисно для тестування, налагодження та експериментів. Python також підтримує різноманітні інструменти для розробки, такі як інтегровані середовища розробки (IDE) PyCharm, VS Code і Jupyter Notebook. Ці інструменти забезпечують зручність у написанні, тестуванні та налагодженні коду, що сприяє підвищенню ефективності роботи програмістів. Загалом, Python — це потужна і гнучка мова програмування, яка продовжує розвиватися і залишатися однією з найпопулярніших у світі.

PHP [13] (Hypertext Preprocessor) - це популярна серверна мова програмування, яка використовується для створення динамічних вебсторінок та вебзастосунків. PHP була створена Расмусом Лердорфом у 1994 році та спочатку називалася Personal Home Page Tools. Згодом PHP була перейменована в PHP: Hypertext Preprocessor, що відображає її здатність обробляти HTML-код та генерувати динамічний контент на стороні сервера.

PHP має простий та зрозумілий синтаксис, який легко вивчити, особливо для тих, хто вже знайомий з мовами програмування, такими як C або Perl. PHP також має велику кількість готових скриптів та бібліотек, які можна використовувати для швидкого створення вебзастосунків.

PHP має вбудовану підтримку HTML, що дозволяє легко генерувати динамічний контент на вебсторінках. PHP-код може бути вбудований безпосередньо в HTML-код, що полегшує створення динамічних вебсторінок.

PHP є безкоштовною мовою програмування з відкритим вихідним кодом, що

означає, що кожен може вільно використовувати, модифікувати та поширювати PHP. PHP також має велику та активну спільноту розробників, які сприяють розвитку мови та створенню нових бібліотек та фреймворків.

PHP підтримується практично всіма популярними вебсерверами, включаючи Apache, Nginx та Microsoft IIS. PHP також підтримує широкий спектр систем керування базами даних, включаючи MySQL, PostgreSQL, Oracle та SQLite.

JavaScript [14]- це популярна мова програмування, яка використовується для створення інтерактивних вебсторінок та вебзастосунків. JavaScript була створена Бренданом Айхом у 1995 році під час його роботи в Netscape Communications Corporation. Спочатку JavaScript називалася LiveScript, але пізніше була перейменована в JavaScript, щоб відобразити її зв'язок з мовою програмування Java.

JavaScript дозволяє створювати інтерактивні вебсторінки, які реагують на дії користувача. Завдяки JavaScript можна створювати вебформи, слайдери, модальні вікна, анімації та інші інтерактивні елементи.

JavaScript дозволяє обробляти дані на клієнтському боці, що дозволяє зменшити навантаження на сервер та збільшити швидкість роботи вебзастосунку.

JavaScript має простий та зрозумілий синтаксис, який легко вивчити, особливо для тих, хто вже знайомий з мовами програмування, такими як C або Java.

JavaScript має вбудовану підтримку HTML та CSS, що дозволяє легко маніпулювати елементами вебсторінки та змінювати їх стилі.

JavaScript має велику кількість фреймворків та бібліотек, які полегшують розробку вебзастосунків. Деякі з популярних фреймворків JavaScript включають React, Angular, Vue.js та Node.js.

Основні складові JavaScript:

1. JavaScript-скрипти - це файли з розширенням .js, які містять JavaScript-код. JavaScript-скрипти можуть бути вбудовані безпосередньо в HTML-код або підключені як зовнішні файли.

2. JavaScript має велику кількість вбудованих функцій, які виконують різні завдання, такі як робота з рядками, масивами, датами та часом. Розробники також можуть створювати власні функції.

3. JavaScript є об'єктно-орієнтованою мовою програмування, що дозволяє створювати об'єкти для організації коду та повторного використання.

4. JavaScript дозволяє створювати події, які реагують на дії користувача, такі як натискання кнопок миші, натискання клавіш та прокрутка сторінки.

5. JavaScript має велику кількість бібліотек та фреймворків, які полегшують розробку вебзастосунків. Деякі з популярних бібліотек JavaScript включають jQuery, Lodash, Axios та Moment.js.

В цілому, JavaScript є популярною та потужною мовою програмування, яка підходить для створення інтерактивних вебсторінок та вебзастосунків. JavaScript має простий та зрозумілий синтаксис, вбудовану підтримку HTML та CSS, а також багату екосистему бібліотек та фреймворків. JavaScript є відмінним вибором для розробки клієнтської частини вебсервісу поштового клієнта для електронної розсилки.

TypeScript [15] - це статично типізована мова програмування, яка є надбудовою над JavaScript. TypeScript був створений компанією Microsoft у 2012 році з метою покращення розробки великих та складних вебзастосунків. TypeScript дозволяє використовувати статичну типізацію, інтерфейси, класи, модулі та інші сучасні концепції об'єктно-орієнтованого програмування, які недоступні в JavaScript.

TypeScript дозволяє використовувати статичну типізацію, що дозволяє виявляти помилки в коді на ранніх стадіях розробки. Статична типізація також покращує читабельність коду та полегшує його підтримку.

TypeScript дозволяє використовувати інтерфейси, що дозволяє визначати контракти для класів та об'єктів, а також забезпечує сумісність між різними частинами коду.

TypeScript підтримує об'єктно-орієнтоване програмування, включаючи класи, модулі, успадкування та поліморфізм.

TypeScript є надбудовою над JavaScript, що означає, що будь-який код JavaScript є дійсним кодом TypeScript. Це дозволяє легко інтегрувати TypeScript в існуючі проекти JavaScript.

TypeScript має велику кількість фреймворків та бібліотек, які полегшують розробку вебзастосунків. Деякі з популярних фреймворків TypeScript включають Angular, React, Vue.js та Node.js.

TypeScript є потужною та гнучкою мовою програмування, яка підходить для розробки великих та складних вебзастосунків. TypeScript дозволяє використовувати статичну типізацію, інтерфейси, класи, модулі та інші сучасні концепції об'єктно-орієнтованого програмування, що покращує якість коду та зменшує кількість помилок. TypeScript є відмінним вибором для розробки клієнтської частини вебсервісу поштового клієнта для електронної розсилки.

Для розробки програмних засобів вебсервісу поштового клієнта для електронної розсилки, необхідно обрати мову програмування окремо для серверної та клієнтської частини.

Для того, аби порівняти можливості описаних мов програмування для серверної частини системи, складемо таблицю 3.1 порівняння характеристик.

Таблиця 3.1 – Порівняння характеристик мов програмування для серверної частини системи

	Java	Python	PHP
Наявність шаблонів для розробки застосунків електронної розсилки	0	1	0
Наявність фреймворків для веб-розробки	1	1	1
Підтримка багатопоточності	1	1	1
Підтримка у AWS	1	1	0
Підтримка кешування для підвищення продуктивності застосунку	1	1	1
Підтримка API для інтеграції з іншими сервісами	1	1	1
Інтеграція з сервісми для аналітики взаємодії користувачів із застосунком	1	1	1
Сумарний коефіцієнт	6	7	5

Для серверної частини найкраще підходить Python. Python має високу швидкість при початковому запуску серверлес (serverless) системи розсилки, що є важливим, адже коли система не використовується, вона автоматично зупиняється, а при її виклику потрібен перезапуск. Використання мови Python дозволяє досить швидко і майже непомітно для користувача провести цей запуск на відміну від мови Java, яка потребує значно більше часу для запуску через особливості її віртуальної машини JVM.

Крім цього, Python має великий набір бібліотек та фреймворків, призначених саме для розробки систем розсилки електронних повідомлень. До таких належать «smtplib», «email» та «Flask-mail». Використання цих бібліотек може значно спростити процес розробки застосунків.

Щодо мови програмування для розробки клієнтської частини системи, то найкращим вибором є JavaScript.

JavaScript дозволяє створювати інтерактивні вебсторінки, що реагують на дії користувача. Це дозволяє створювати динамічний та привабливий інтерфейс користувача, який забезпечує кращий досвід взаємодії з користувачем.

JavaScript має велику кількість фреймворків та бібліотек для розробки клієнтської частини, таких як React, Angular, Vue.js, jQuery, Bootstrap та багато інших. Ці фреймворки та бібліотеки полегшують розробку, тестування та розгортання клієнтської частини, а також забезпечують високу продуктивність та надійність.

JavaScript підтримує асинхронне програмування, що дозволяє виконувати кілька завдань одночасно та уникати блокування інтерфейсу користувача. Це дозволяє створювати швидкі та відзивчі клієнтські застосунки, які забезпечують кращий досвід взаємодії з користувачем.

3.2 Варіантний аналіз та вибір середовища розробки

Окрім вибору мов програмування для розробки, важливо також обрати і середовище (чи середовища) розробки. Розглянемо деякі відомі середовища

розробки.

Visual Studio Code [17] (VS Code) – це безкоштовний та відкритий текстовий редактор, розроблений компанією Microsoft. Він підтримує багато мов програмування, включаючи JavaScript, TypeScript, Python, Java, C++, PHP, Go, Ruby, Rust, Swift та багато інших. VS Code має багато функцій, що полегшують розробку веб-застосунків.

VS Code автоматично виділяє синтаксис коду, що допомагає легше читати та розуміти код. VS Code пропонує автодоповнення коду, що допомагає швидше та точніше вводити код. VS Code має вбудований налагоджувач, що дозволяє ставити точки зупинки, переглядати значення змінних та виконувати код крок за кроком.

VS Code має вбудовану підтримку систем керування версіями, таких як Git, SVN та Mercurial. Це дозволяє легко керувати версіями коду, робити коміти, створювати гілки та розв'язувати конфлікти.

VS Code має велику кількість розширень, що дозволяють розширити функціональність редактора. Розширення можуть додавати нові мови програмування, теми оформлення, функції рефакторингу, інструменти для тестування, налагодження та розгортання застосунків та багато іншого.

VS Code має вбудований термінал, що дозволяє виконувати команди в операційній системі, не виходячи з редактора. VS Code має функцію Live Share, що дозволяє спільно редагувати код з іншими розробниками в реальному часі.

VS Code використовує Language Server Protocol (LSP), що дозволяє інтегрувати інструменти для роботи з мовами програмування, такі як лінтери, форматирувальники, статичні аналізатори та інші.

Отже, VS Code є потужним та гнучким інструментом для розробки веб-застосунків. Він має багато функцій, що полегшують розробку, тестування та налагодження застосунків, а також велику кількість розширень, що дозволяють розширити функціональність редактора. VS Code є відмінним вибором для розробки клієнтської частини вебсервісу поштового клієнта для електронної розсилки.

Visual Studio Code продемонстровано на рисунку 3.3.

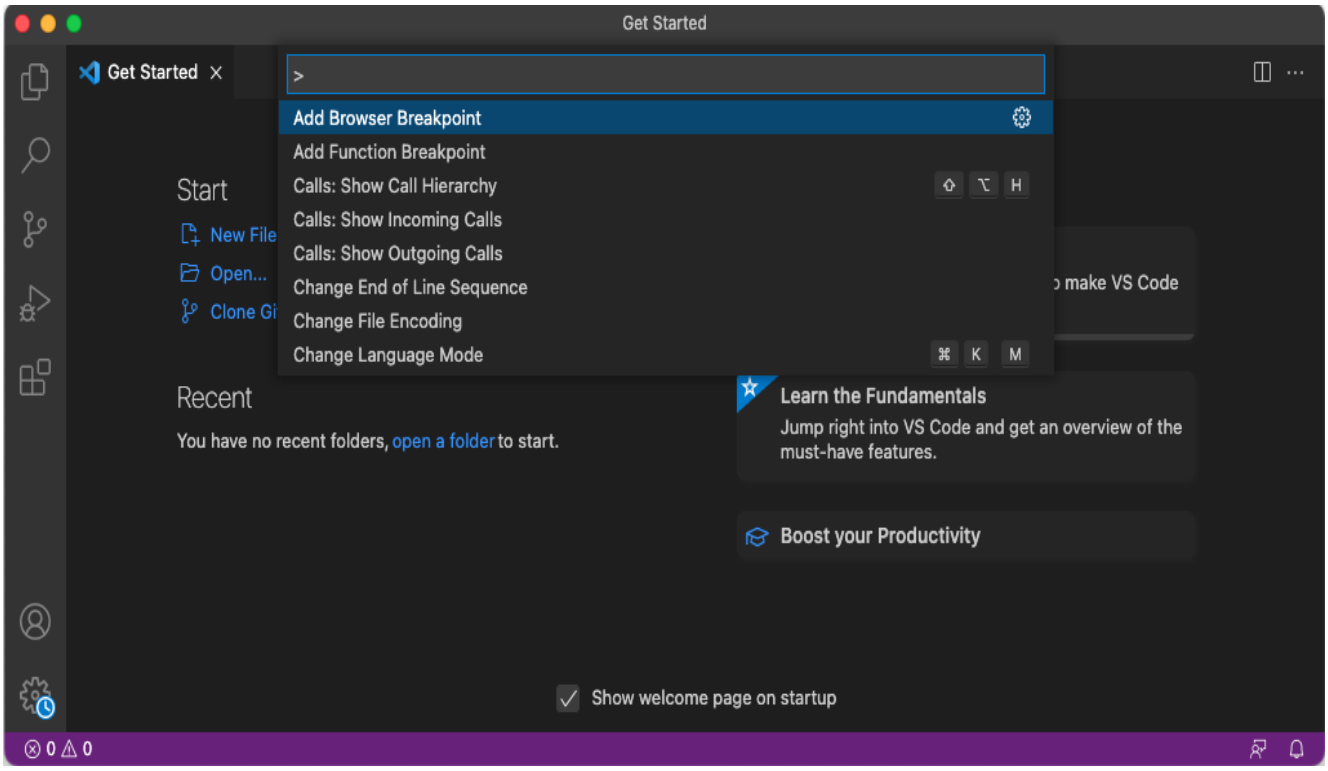

Рисунок 3.3 – Інтерфейс Visual Studio Code

WebStorm [18] – це комерційне середовище розробки, розроблене компанією JetBrains. Воно призначене для розробки веб-застосунків на JavaScript, TypeScript, HTML, CSS та інших веб-технологіях. WebStorm має багато функцій, що полегшують розробку, тестування та налагодження веб-застосунків, серед яких можна виділити такі:

WebStorm пропонує інтелектуальне автодоповнення коду, що допомагає швидше та точніше вводити код. Автодоповнення працює на основі аналізу коду, бібліотек та фреймворків, що використовуються в проекті.

WebStorm має вбудований налагоджувач, що дозволяє ставити точки зупинки, переглядати значення змінних, виконувати код крок за кроком та відстежувати помилки. Налагоджувач працює з JavaScript, TypeScript, Node.js та іншими технологіями.

WebStorm має вбудовану підтримку систем керування версіями, таких як Git, SVN, Mercurial та Perforce. Це дозволяє легко керувати версіями коду, робити коміти, створювати гілки та розв'язувати конфлікти.

WebStorm має багато інструментів для рефакторингу коду, таких як перейменування змінних, виділення змінних, вилучення функцій, злиття файлів та багато іншого. Ці інструменти допомагають покращити структуру коду та зробити його більш зрозумілим.

WebStorm має інтегровані інструменти для тестування коду, такі як Jest, Mocha, Karma, Jasmine та інші. Ці інструменти дозволяють створювати та виконувати тести для перевірки роботи коду.

Інтерфейс WebStorm наведено на рисунку 3.4.

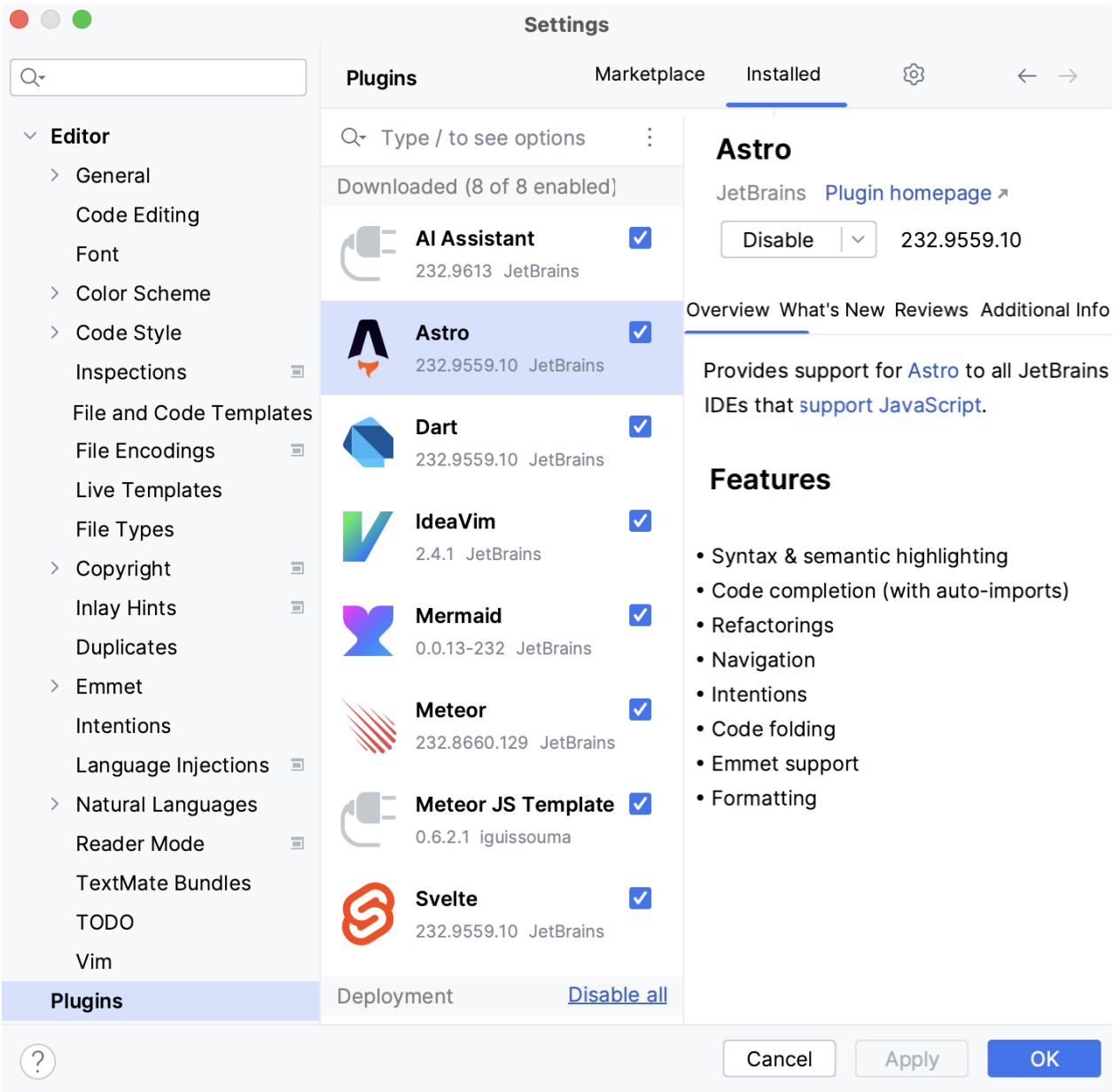

Рисунок 3.4 – Інтерфейс WebStorm

WebStorm має вбудовану підтримку багатьох популярних фреймворків, таких як React, Angular, Vue.js, Express, Meteor, Webpack та багато інших. Це дозволяє легко створювати, налагоджувати та тестувати застосунки, що використовують ці фреймворки.

WebStorm має вбудований термінал, що дозволяє виконувати команди в операційній системі, не виходячи з середовища розробки. WebStorm має функцію Live Edit, що дозволяє відстежувати зміни в коді в реальному часі, не перезавантажуючи сторінку браузера.

WebStorm є потужним та гнучким середовищем розробки для веб-застосунків. Він має багато функцій, що полегшують розробку, тестування та налагодження застосунків, а також велику кількість інструментів для роботи з JavaScript, TypeScript, HTML, CSS та іншими веб-технологіями. WebStorm є відмінним вибором для розробки клієнтської частини вебсервісу поштового клієнта для електронної розсилки.

Sublime Text – це комерційний текстовий редактор, розроблений компанією Sublime HQ. Він підтримує багато мов програмування, включаючи JavaScript, Python, HTML, CSS, PHP, Ruby, C++, Java та багато інших. Sublime Text має багато функцій, що полегшують роботу з кодом. Sublime Text автоматично виділяє синтаксис коду, що допомагає легше читати та розуміти код.

Sublime Text пропонує автодоповнення коду, що допомагає швидше та точніше вводити код. Воно дозволяє виконувати множинне редагування коду, що дозволяє одночасно редагувати кілька рядків коду.

Sublime Text дозволяє переходити до визначення функції або змінної, натиснувши клавішу F12. Це допомагає швидше навігавувати по коду та розуміти його структуру. Sublime Text має велику кількість плагінів, що дозволяють розширити функціональність редактора. Плагіни можуть додавати нові мови програмування, теми оформлення, функції рефакторингу, інструменти для тестування, налагодження та розгортання застосунків та багато іншого.

Sublime Text має палітру команд, що дозволяє швидко виконувати різні команди, не використовуючи меню або клавіатурні скорочення. Sublime Text має

велику кількість тем оформлення, що дозволяють змінити зовнішній вигляд редактора. Sublime Text доступний для Windows, MacOS та Linux, що дозволяє використовувати його на різних платформах.

Sublime Text є потужним та гнучким інструментом для роботи з кодом. Він має багато функцій, що полегшують роботу з кодом, а також велику кількість плагінів, що дозволяють розширити функціональність редактора. Sublime Text є відмінним вибором для розробки клієнтської частини вебсервісу поштового клієнта для електронної розсилки.

Побудовано таблицю 3.2 порівняння функціональних характеристик описаних середовищ розробки для веб-застосунків.

Таблиця 3.2 – Порівняння функціональних характеристик середовищ розробки для веб-застосунків

	Visual Studio Code	WebStorm	Sublime Text
Безкоштовний доступ	1	0	1
Інтеграція з системами контролю версій	1	1	1
Автодоповнення коду	1	1	0
Синтаксичний аналіз коду	1	1	0
Низькі системні вимоги	1	0	1
Підтримка плагінів та розширень	1	1	1
Вбудована підтримка відладки	1	1	0
Підтримка рефакторингу коду	1	1	1
Сумарний коефіцієнт	8	6	5

Таким чином, для розробки клієнтської частини системи обрано Visual Studio Code. Це середовище розробки необхідне для створення користувацького інтерфейсу системи розсилки електронних повідомлень та інтеграцію з серверною частиною.

VS Code має велику кількість розширень, що дозволяють розширити функціональність редактора. Розширення можуть додавати нові мови програмування, теми оформлення, функції рефакторингу, інструменти для тестування, налагодження та розгортання застосунків та багато іншого. Це дозволяє розробникам підлаштувати VS Code під свої потреби та робити розробку більш ефективною.

Розглянемо середовища розробки для Python.

PyCharm — інтегроване середовище розробки (IDE) для мови програмування Python, розроблене компанією JetBrains. Серед ключових особливостей PyCharm варто відзначити розширене автозаповнення коду, зручну навігацію, інструменти для налагодження та тестування, інтеграцію з системами контролю версій, такими як Git, а також підтримку веб-розробки через Django та Flask. Це робить PyCharm вибором багатьох професійних розробників, які цінують продуктивність і надійність у роботі.

PyCharm пропонує два основних видання - Community Edition, яке є безкоштовним та з відкритим вихідним кодом, і Professional Edition, що є платним і надає розширені можливості. Community Edition підходить для базових потреб Python-розробників, тоді як Professional Edition включає додаткові функції для роботи з веб-фреймворками, базами даних, науковими інструментами та підтримку інструментів для розробки на інших мовах. PyCharm також відзначається інтеграцією з популярними технологіями та інструментами, такими як Docker, Kubernetes та різні сервіси хмарних обчислень, що забезпечує повний цикл розробки та деплойменту програмних продуктів.

Однією з найбільш корисних функцій PyCharm є його потужний налагоджувач, який дозволяє легко відслідковувати і виправляти помилки в коді. Крім того, PyCharm підтримує аналіз коду в реальному часі, що допомагає виявляти

потенційні проблеми та покращувати якість коду під час написання. Вбудований інструмент тестування забезпечує швидкий запуск та аналіз результатів тестів, що сприяє розробці надійного та безпомилкового програмного забезпечення. Завдяки цим та багатьом іншим можливостям, PyCharm є незамінним інструментом для Python-розробників, які прагнуть досягти високої продуктивності та якості в своїй роботі.

Інтерфейс PyCharm наведено на рисунку 3.5.

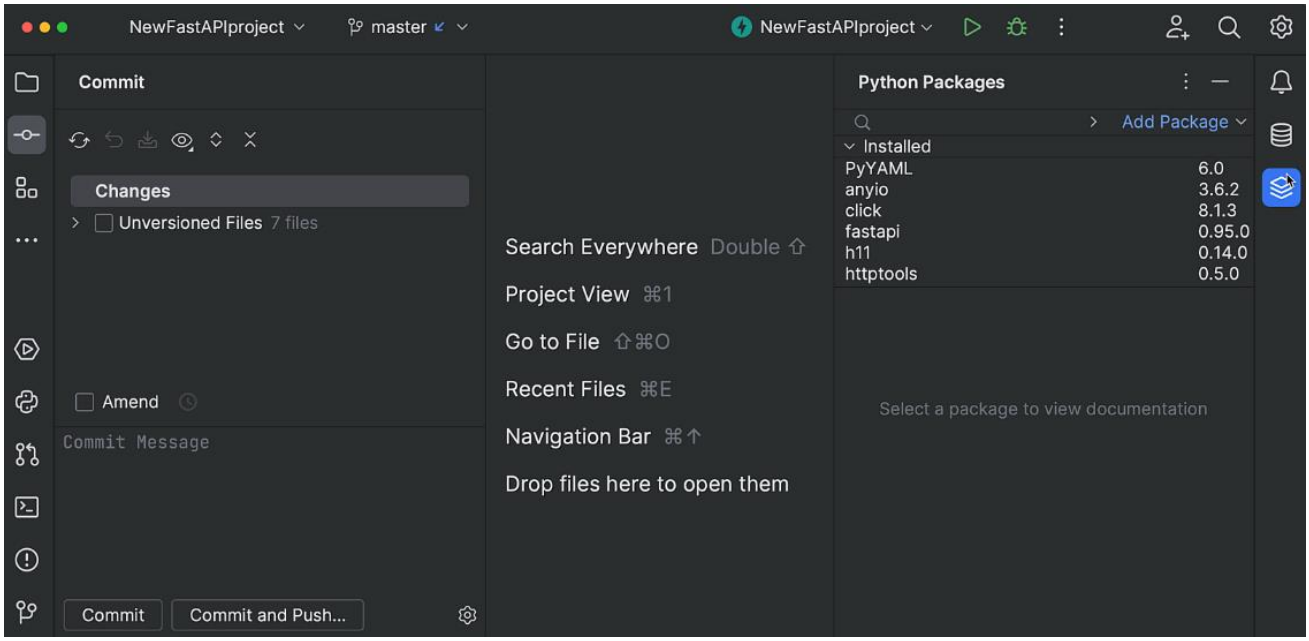

Рисунок 3.5 – Інтерфейс PyCharm

Spyder — це інтегроване середовище розробки (IDE) для мов програмування Python, спрямоване на наукові обчислення та аналіз даних. Відоме своєю простотою використання та інтуїтивно зрозумілим інтерфейсом, Spyder забезпечує потужні інструменти для розробки, включаючи підтримку автодоповнення коду, відладку, візуалізацію даних та інтерактивну консоль IPython прямо всередині IDE. Він також надає доступ до різних пакетів наукових обчислень, що робить його популярним в середовищах, де велике значення має аналітична обробка даних і статистика.

Інтерфейс Spyder наведено на рисунку 3.6.

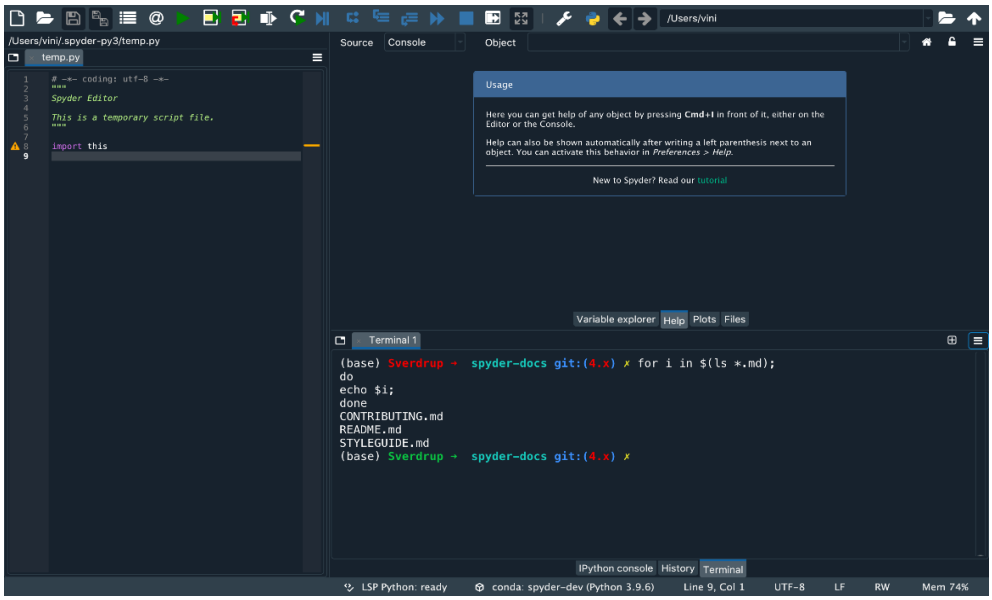

Рисунок 3.6 – Інтерфейс Spyder

Jupyter Notebook — це інтерактивне середовище для створення та виконання документів, що містять живий код, рівень оформлення тексту, математичні рівняння, графіки та інші медіа. В основі Jupyter Notebook лежить концепція клітинок, які можуть містити код Python, текстові описи, форматовані за допомогою Markdown, а також результати виконання коду, що відображаються прямо в нотатці. Це середовище особливо корисне для дослідження даних, викладання та спільної роботи, оскільки дозволяє створювати і документувати аналізи та обчислення у зручній і доступній формі.

Побудовано таблицю 3.3 порівняння функціональних характеристик описаних середовищ розробки застосунків на Python.

Таблиця 3.3 – Порівняння функціональних характеристик середовищ розробки застосунків на Python

	PyCharm	Spyder	Jupyter Notebook
1	2	3	4
Безкоштовний доступ	1	1	1
Інтеграція з системами контролю версій	1	1	1

Продовження таблиці 3.3

1	2	3	4
Автодоповнення коду	1	1	1
Синтаксичний аналіз коду	1	1	0
Підтримка плагінів і розширень	1	0	0
Підтримка рефакторингу коду	1	0	0
Вбудована підтримка відладки	1	1	0
Сумарний коефіцієнт	7	5	3

Отже, PyCharm є потужним та гнучким середовищем розробки для застосунків на різних мовах програмування. Він має багато функцій, що полегшують розробку, тестування та налагодження застосунків. PyCharm є відмінним вибором для розробки серверної частини вебсервісу поштового клієнта для електронної розсилки.

Таким чином, було обрано мову програмування Python та середовище розробки PyCharm для розробки серверної частини системи, JavaScript та Visual Studio Code для розробки клієнтської частини.

3.3 Розробка інтерфейсу користувача

Розробка інтерфейсу допомагає зрозуміти, як саме користувачі будуть взаємодіяти з продуктом. Це допомагає розробникам зрозуміти потреби користувачів і створити інтерфейс, який буде інтуїтивно зрозумілим.

Добре спроектований інтерфейс допомагає користувачам виконувати завдання ефективніше. Під час проектування можна визначити найкращий спосіб розміщення елементів інтерфейсу, щоб користувачі могли легко знайти те, що їм потрібно.

Добре продуманий інтерфейс може значно покращити задоволеність користувачів. Користувачі цінують продукти, які легко використовувати і які мають

приємний зовнішній вигляд.

Проектування інтерфейсу допомагає виявити потенційні проблеми на ранніх стадіях розробки. Це може заощадити час і ресурси, оскільки виправлення проблем на ранніх стадіях є набагато легшим і дешевшим.

Структурна схема вікна вхідних та вихідних повідомлень наведена на рисунку 3.7.

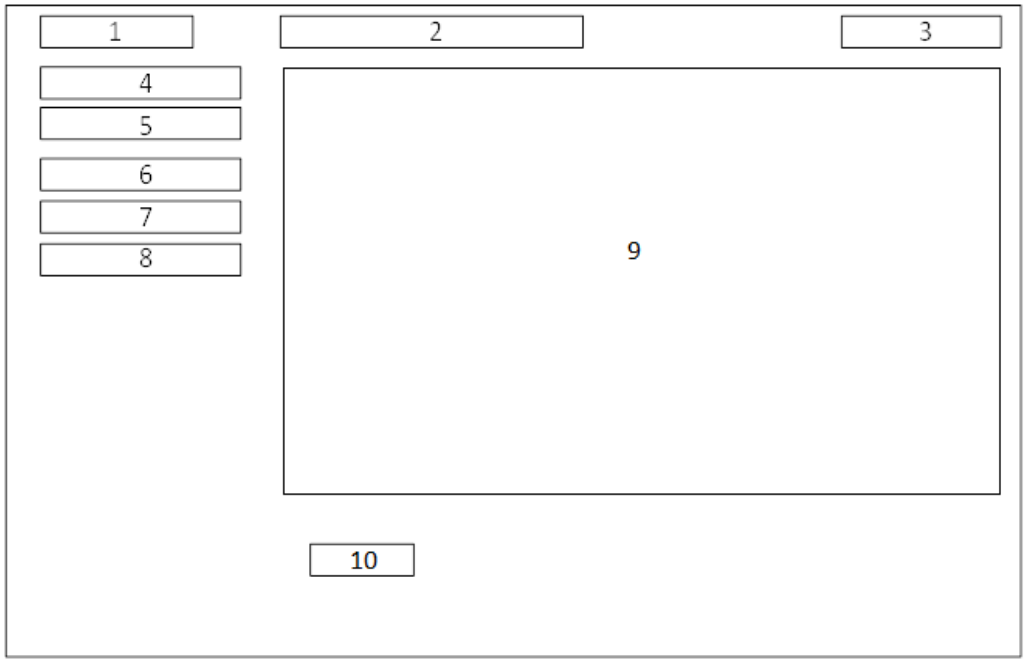

Рисунок 3.7 – Структурна схема вікна вхідних та вихідних повідомлень

Елементи вікна вхідних та вихідних повідомлень:

1. Логотип системи.
2. Пошукове поле.
3. Електронна пошта профілю.
4. Кнопка для вхідних повідомлень.
5. Кнопка для надісланих повідомлень.
6. Кнопка для контактів.
7. Кнопка для запланованих повідомлень.
8. Кнопка для вкладки спаму.
9. Список отриманих або надісланих листів.

10. Меню перемикання між сторінками з листами.

Структурна схема електронного листа у вікні вхідних та вихідних повідомлень наведена на рисунку 3.8.

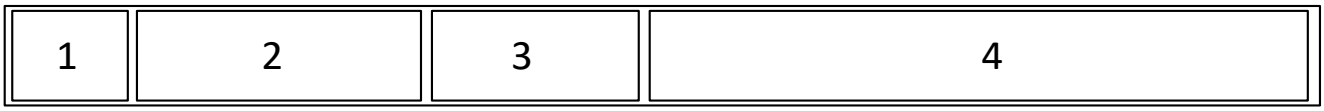

Рисунок 3.8 – Структурна схема електронного листа у вікні з листами

Елементи листа:

1. Кнопка виділення листа
2. Відправник або отримувач листа.
3. Тема листа.
4. Фрагмент тексту листа.

Вікно створення листів необхідне для формування розсилки, надсилання необхідної інформації іншим користувачам. При цьому, важливою є наявність інструментів для редагування та стилізації тексту та формування списків розсилки.

Елементи вікна:

1. Логотип системи.
2. Пошукове поле.
3. Електронна пошта профілю.
4. Кнопка для написання листа окремому користувачу.
5. Кнопка для написання листа групі користувачів.
6. Кнопка для вхідних повідомлень.
7. Кнопка для надісланих повідомлень.
8. Кнопка для контактів.
9. Кнопка для запланованих повідомлень.
10. Кнопка для вкладки спаму.
11. Поле отримувача листа.
12. Поле для теми листа.
13. Поле введення та редагування тексту листа.

14. Кнопка для надсилання листа.

Структурна схема вікна створення електронних листів наведена на рисунку 3.9.

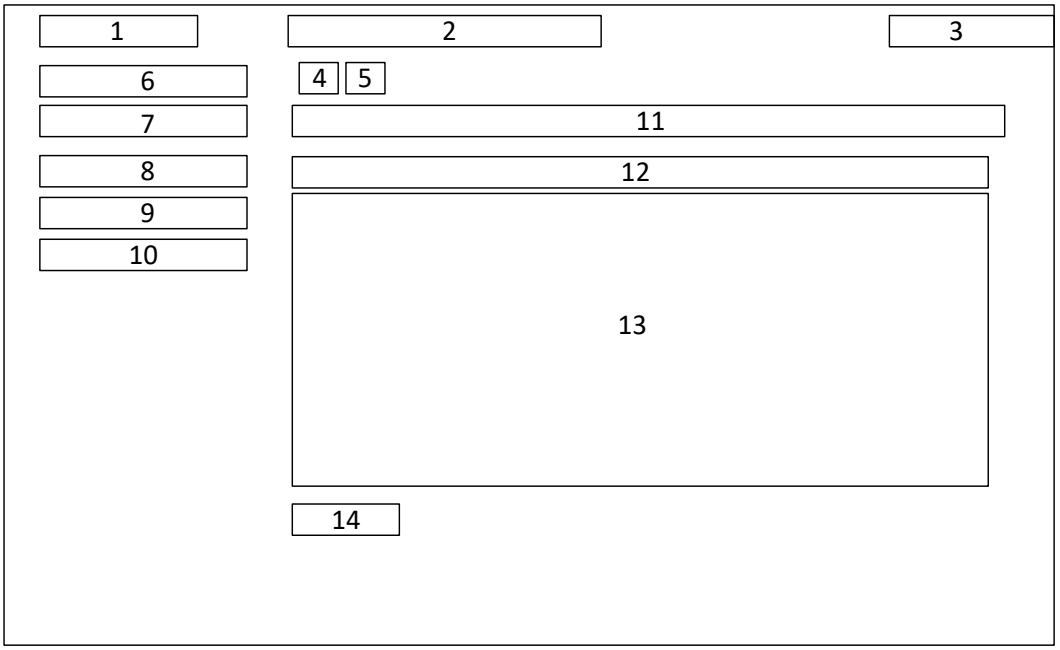

Рисунок 3.9 – Вікно створення електронних листів

Структурна схема вікна створення шаблонів листів наведена на рисунку 3.10.

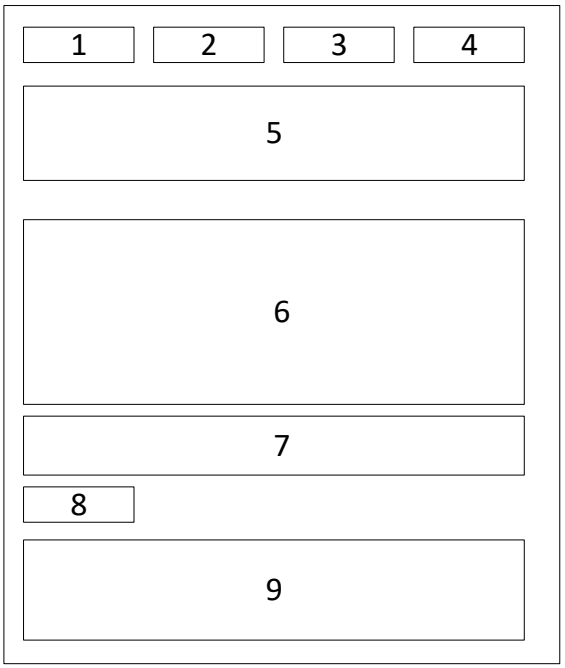

Рисунок 3.10 – Структурна схема вікна створення шаблонів листів

Елементи вікна:

1. Кнопка для редагування шаблону.
2. Кнопка тестування створеного шаблону листа.
3. Кнопка створення копії шаблону.
4. Кнопка для прикріплення файлу до листа.
5. Поле для налаштування параметрів листа.
6. Поле для налаштування тексту листа.
7. Поле для прикріплення файлів до листа.
8. Кнопка збереження шаблону.
9. Статистика створеного шаблону листа.

3.4 Висновки

У третьому розділі було проведено варіантний аналіз середовищ розробки застосунку та мов програмування. Було обрано мови програмування Java та JavaScript для розробки серверної та клієнтської частини вебсистеми відповідно та середовища розробки IntelliJ IDEA та Visual Studio Code. Було розроблено структурні схеми інтерфейсу користувача.

4 ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Аналіз методів тестування

Тестування веб-застосунків є невід'ємною частиною розробки програмного забезпечення. Це процес, який дозволяє виявити та виправити помилки, недоліки та вразливості в системі.

Методи тестування веб-застосунків можуть бути різними. Вони залежать від типу застосунку, його функціональності, цільової аудиторії та багатьох інших факторів.

Функціональне тестування є фундаментом процесу. Воно охоплює перевірку основних можливостей: створення та редагування шаблонів листів, управління списками контактів (включаючи їх сегментацію), планування та відправку розсилок. Важливим аспектом є відстеження статистики, такої як кількість відкриттів, кліків та відписок.

Складові функціонального тестування наведені на рисунку 4.1.

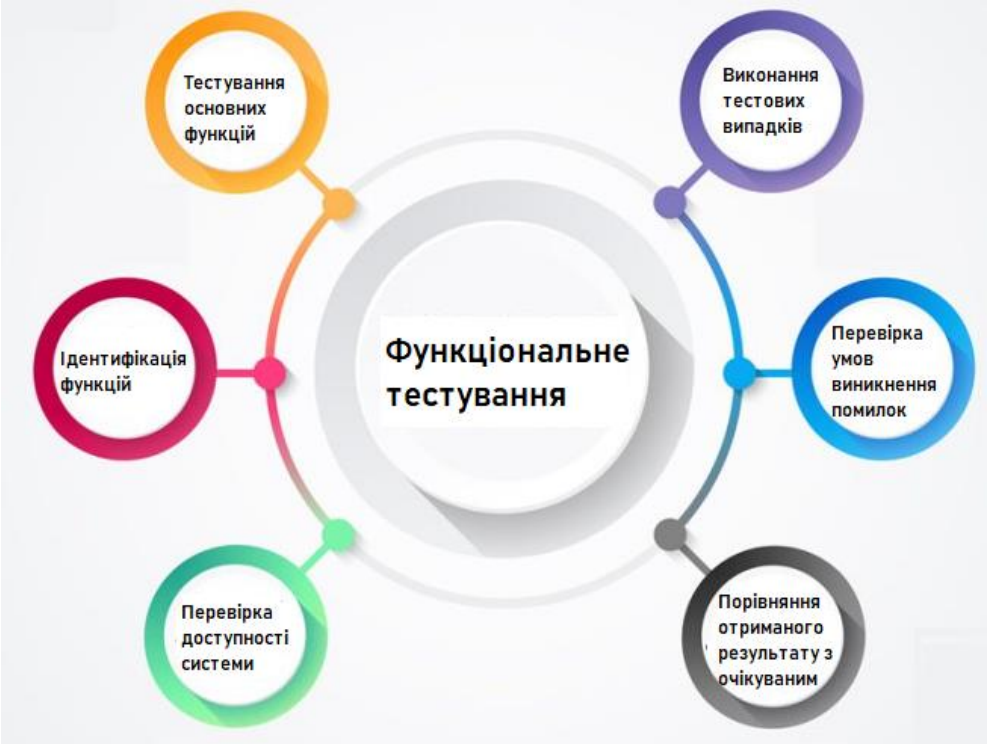

Рисунок 4.1 – Складові функціонального тестування

Тестування інтеграцій забезпечує безперебійну взаємодію з іншими сервісами. Це включає імпорт контактів з CRM-систем чи Google Contacts, інтеграцію з аналітичними інструментами як Google Analytics або Mixpanel, а також перевірку коректності роботи API для взаємодії з іншими сервісами.

Продуктивність є ключовою для таких систем. Навантажувальне тестування перевіряє здатність системи працювати з великою кількістю одночасних розсилок. Стрес-тестування симулює пікові навантаження, наприклад, під час великих розпродажів. Тестування масштабованості гарантує, що система витримає зростання бази користувачів.

Навантажувальне тестування є важливою складовою стратегії тестування системи, спрямованою на оцінку її продуктивності під високим навантаженням. Цей тип тестування дозволяє визначити максимальні межі, до яких система може працювати ефективно, включаючи обробку багатьох одночасних запитів або транзакцій.

Складові навантажувального тестування наведені на рисунку 4.2.

Рисунок 4.2 – Складові навантажувального тестування

Автоматизація тестування економить час та підвищує надійність. Unit-тести для backend-логіки, UI-тести з використанням інструментів як Selenium або Cypress, та API-тести з Postman чи RestAssured допомагають швидко виявляти регресії.

Піраміда автоматизації тестів продемонстрована на рисунку 4.3.

Рисунок 4.3 – Піраміда автоматизації тестів

Отже, тестування застосунків для електронної розсилки є дуже важливим елементом розробки та впровадження таких застосунків. Воно дозволяє забезпечити високу якість роботи застосунку, підвищити ефективність маркетингових кампаній та запобігти потенційним проблемам.

4.2 Тестування поштового клієнта для електронної розсилки

Для перевірки функціоналу роботи клієнта для електронної розсилки, проведемо його тестування. Було обрано метод мануального (ручного) тестування.

Мануальне (ручне) тестування - це процес, коли тестувальник вручну виконує тестові сценарії, не використовуючи жодних автоматизованих інструментів або

скриптів.

Основні характеристики мануального тестування:

1. Виконується людиною (тестувальником).

2. Не потребує знання мов програмування.

3. Ефективне для перевірки зручності використання та візуального дизайну.

4. Гнучке і дозволяє тестувальнику відхилятися від сценаріїв, базуючись на інтуїції.

5. Може бути повільнішим та менш надійним у порівнянні з автоматизованим тестуванням для повторюваних тестів.

На рисунку 4.4 зображено вкладку із вхідними листами.

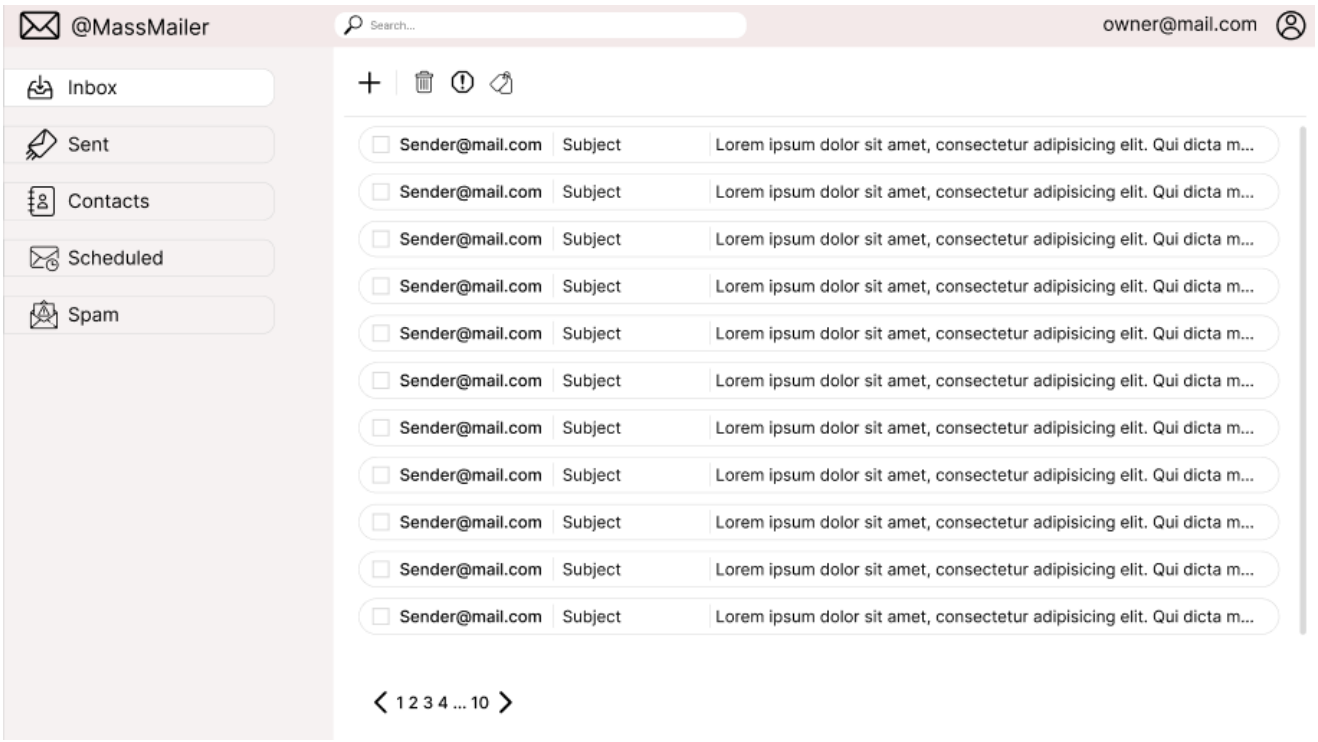

Рисунок 4.4 – Вкладка із вхідними листами

На рисунку 4.5 зображено вкладку із надісланими повідомленнями. Тут відображено результат розсилки листів на окремі електронні адреси.

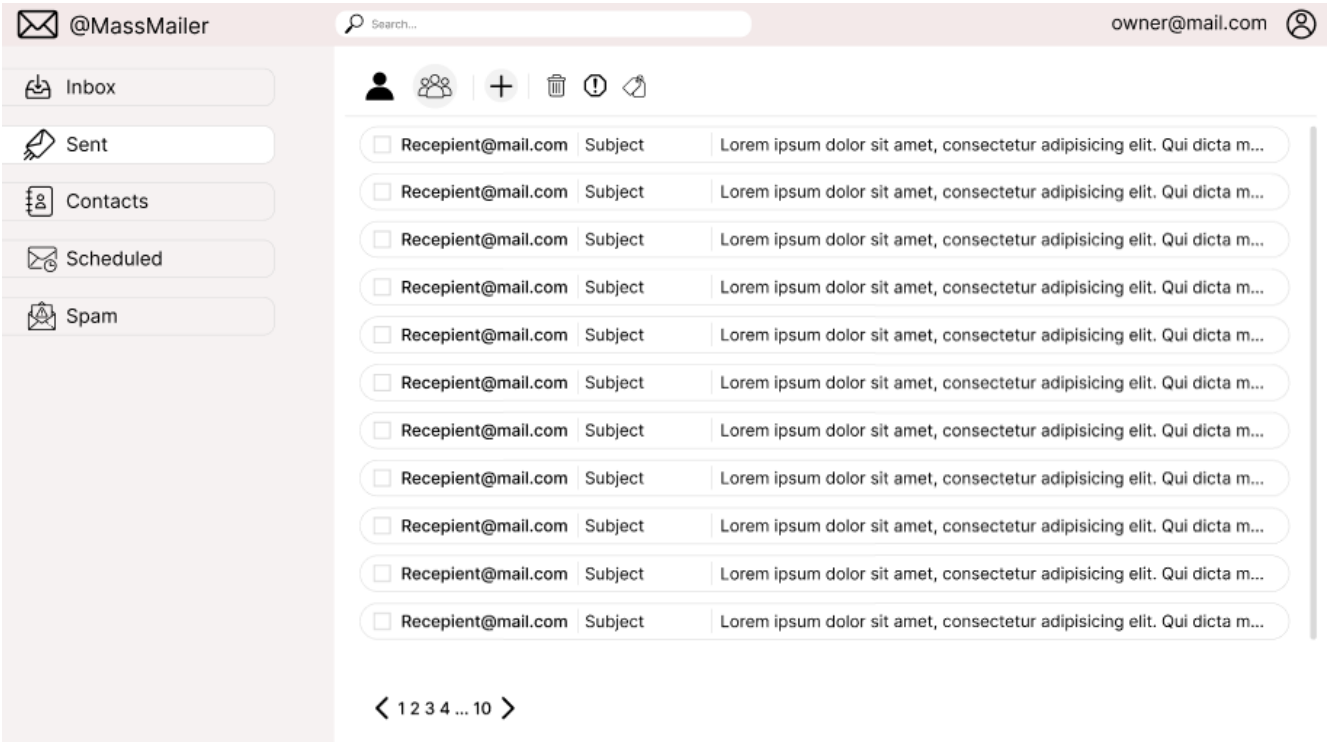

Рисунок 4.5 – Розсилка на окремі електронні адреси

На рисунку 4.6 зображено розсилку групам користувачів. Він відрізняється від попередньої тим, що відправлення листів здійснено певним групам користувачів, які згруповані за різними ознаками.

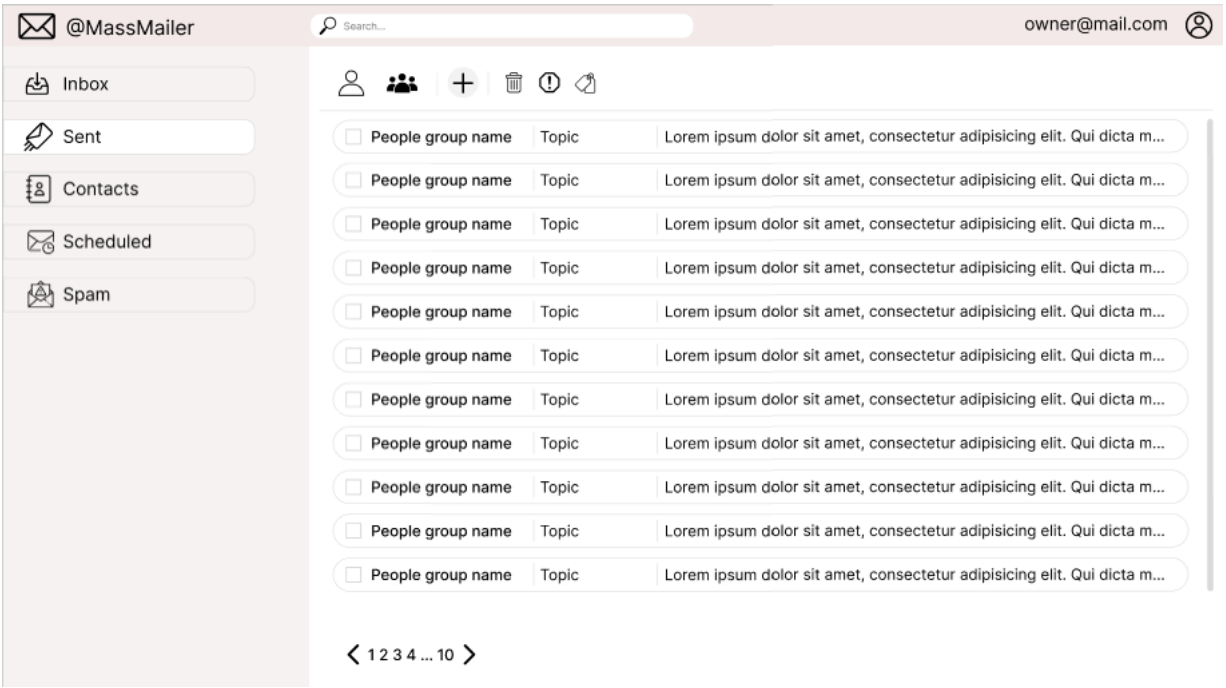

Рисунок 4.6 – Розсилка групам користувачів

На рисунку 4.7 зображено вікно написання електронного листа окремому користувачу.

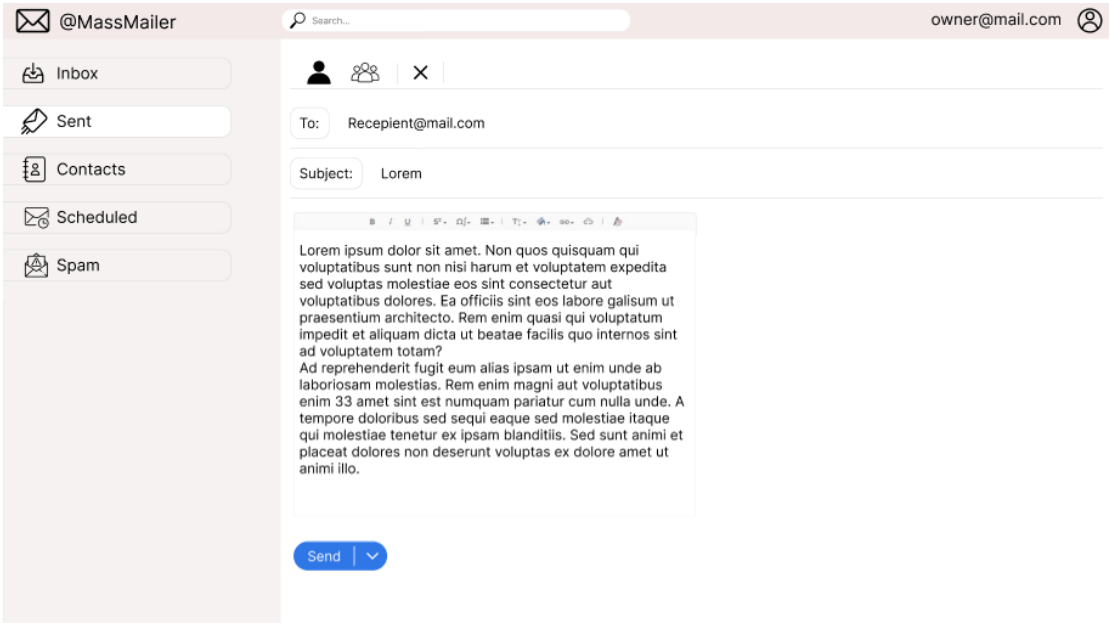

Рисунок 4.7 – Вікно написання електронного листа окремому користувачу

На рисунку 4.8 зображено вікно написання електронного листа групі користувачів. Тут, в якості адресата вказується не електронна адреса, а назва певної групи адрес.

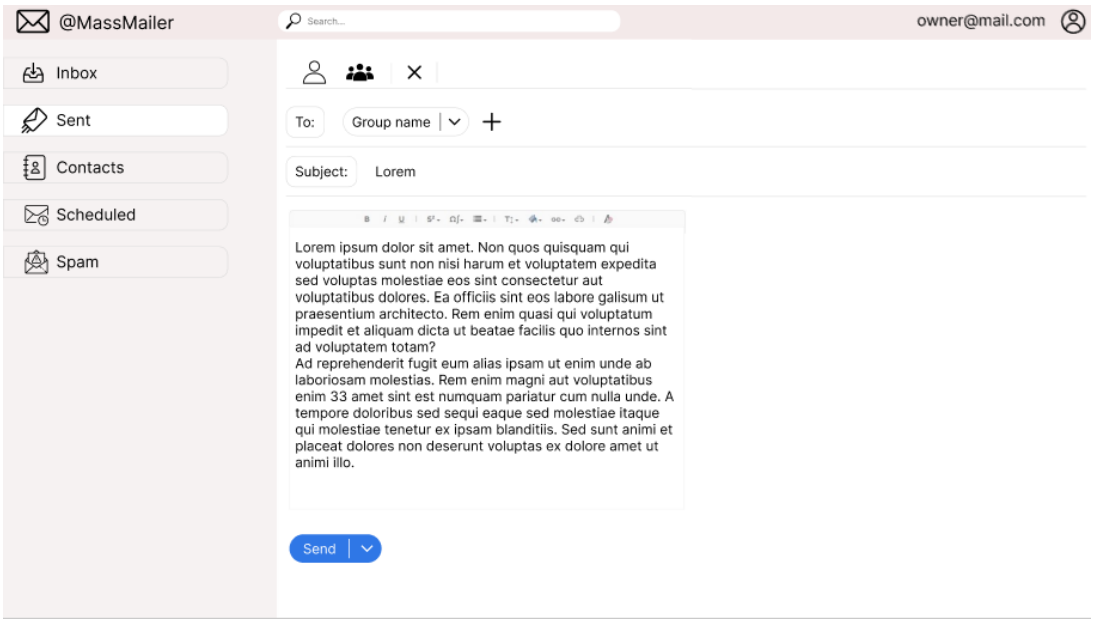

Рисунок 4.8 – Вікно написання електронного листа групі користувачів

Для того, аби користувач був включений в розсилку електронних повідомлень, він повинен спершу підтвердити своє бажання отримувати листи. Для цього надсилається електронний лист для підтвердження долучення до розсилки (рисунок 4.9).

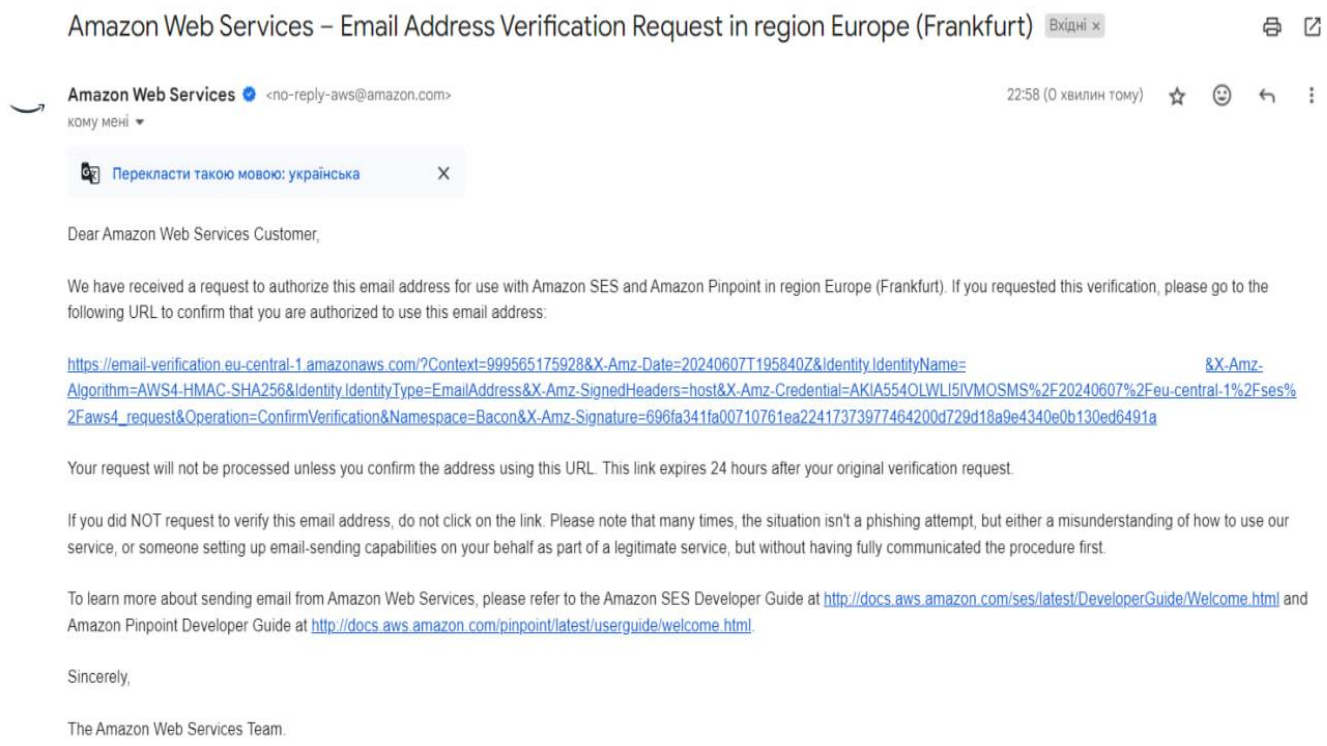

Рисунок 4.9 – Електронний лист для підтвердження участі в розсилці

Список електронних пошт та статус їх підтвердження для отримання розсилки продемонстровано на рисунку 4.10.

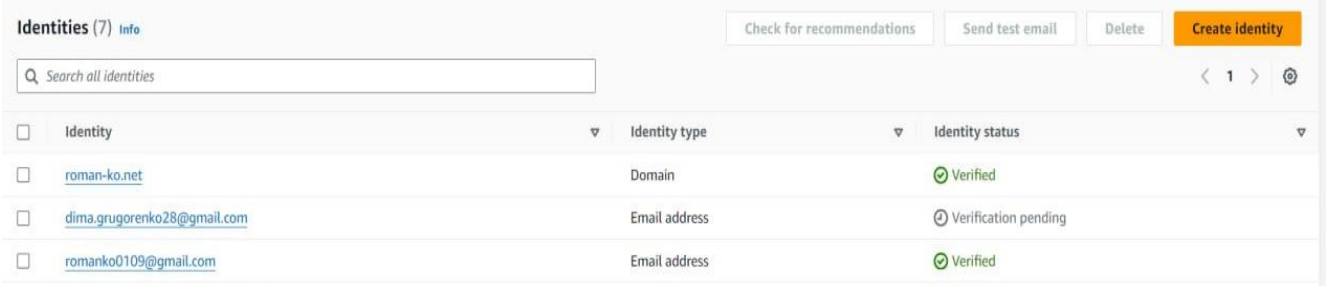

Рисунок 4.10 – Список електронних пошт з їх статусом підтвердження

Поки користувач не підтвердив своє бажання отримувати електронну розсилку, листи йому не надсилатимуться.

Формування шаблону листа наведено на рисунку 4.11.

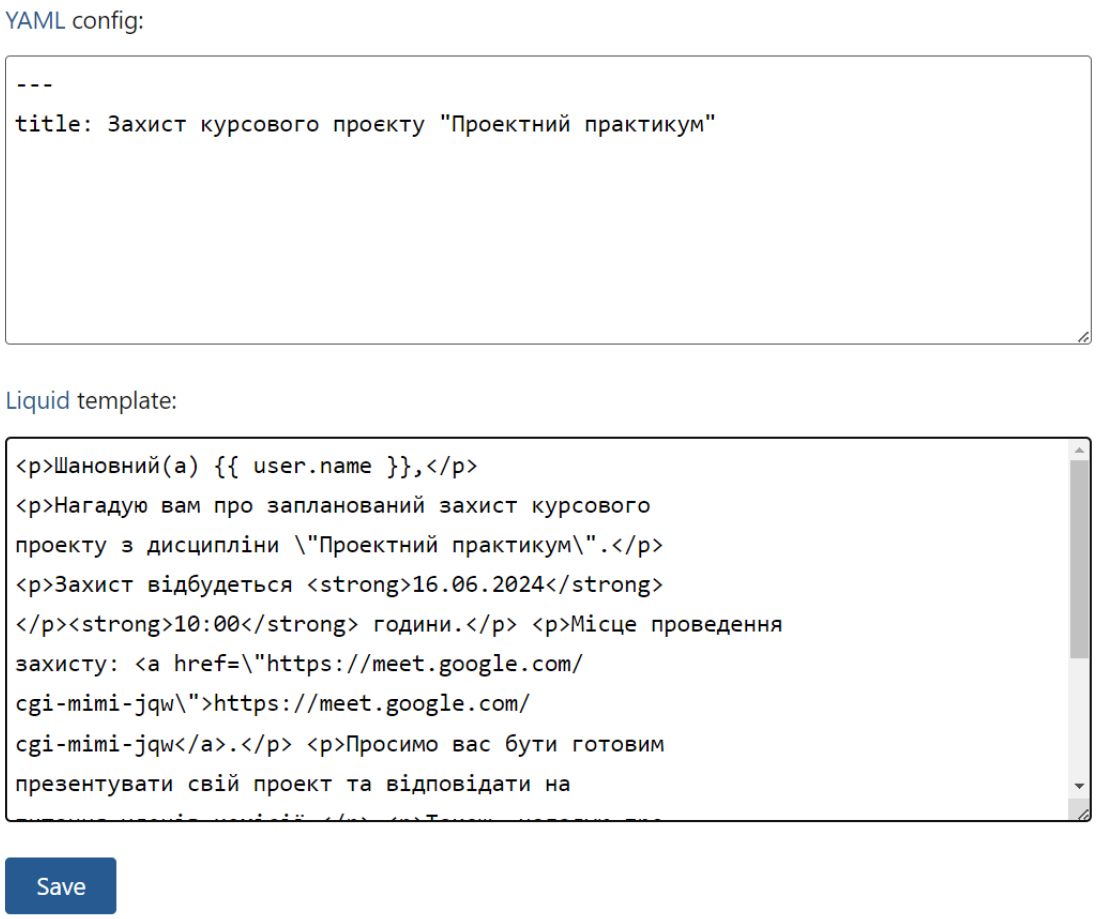

Рисунок 4.11 – Формування шаблону листа

Статистика надісланого листа продемонстрована на рисунку 4.12.

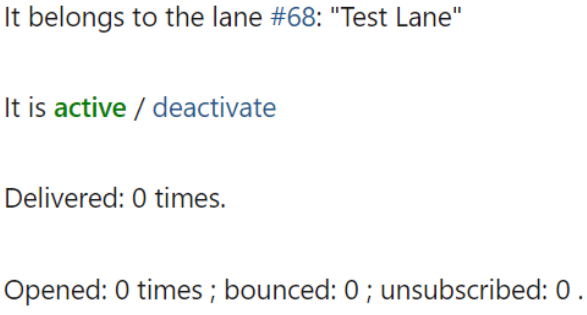

Рисунок 4.12 – Статистика надісланого листа

Відправлений електронний лист продемонстровано на рисунку 4.13.

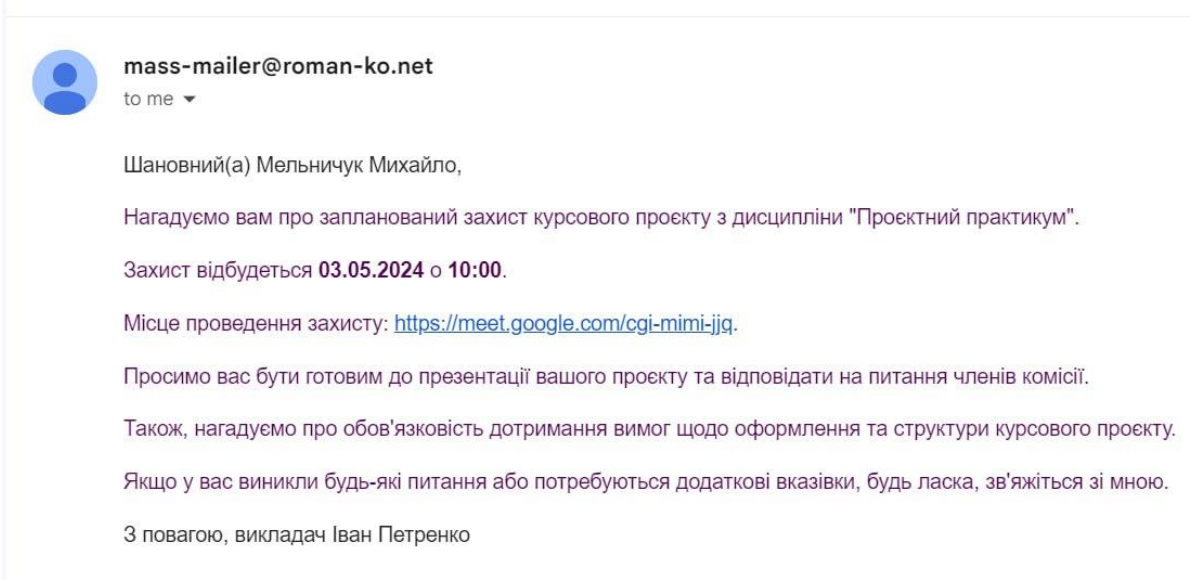

Рисунок 4.13 – Надісланий електронний лист

На рисунку 4.14 продемонстровано інформацію про лист від Gmail, в якій вказано, що електронний лист підписаний, використовує протокол шифрування.

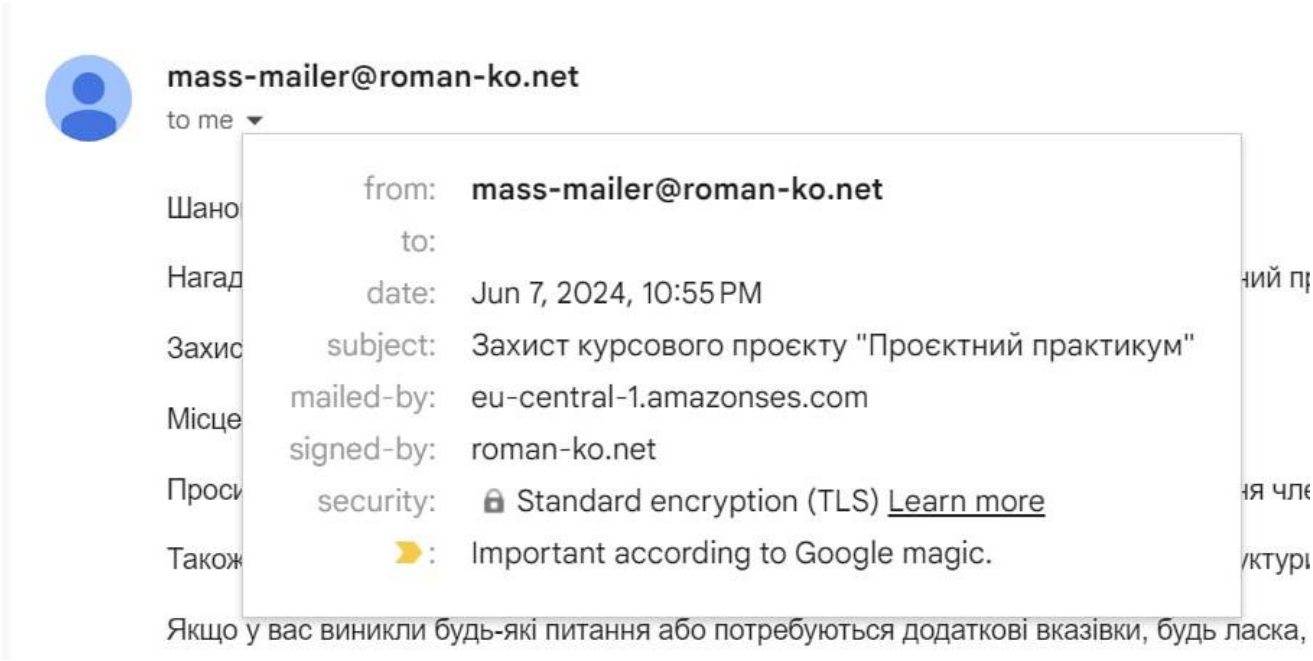

Рисунок 4.14 – Інформація про надісланий лист

На рисунку 4.15 зображено тестування відправки цього електронного листа.

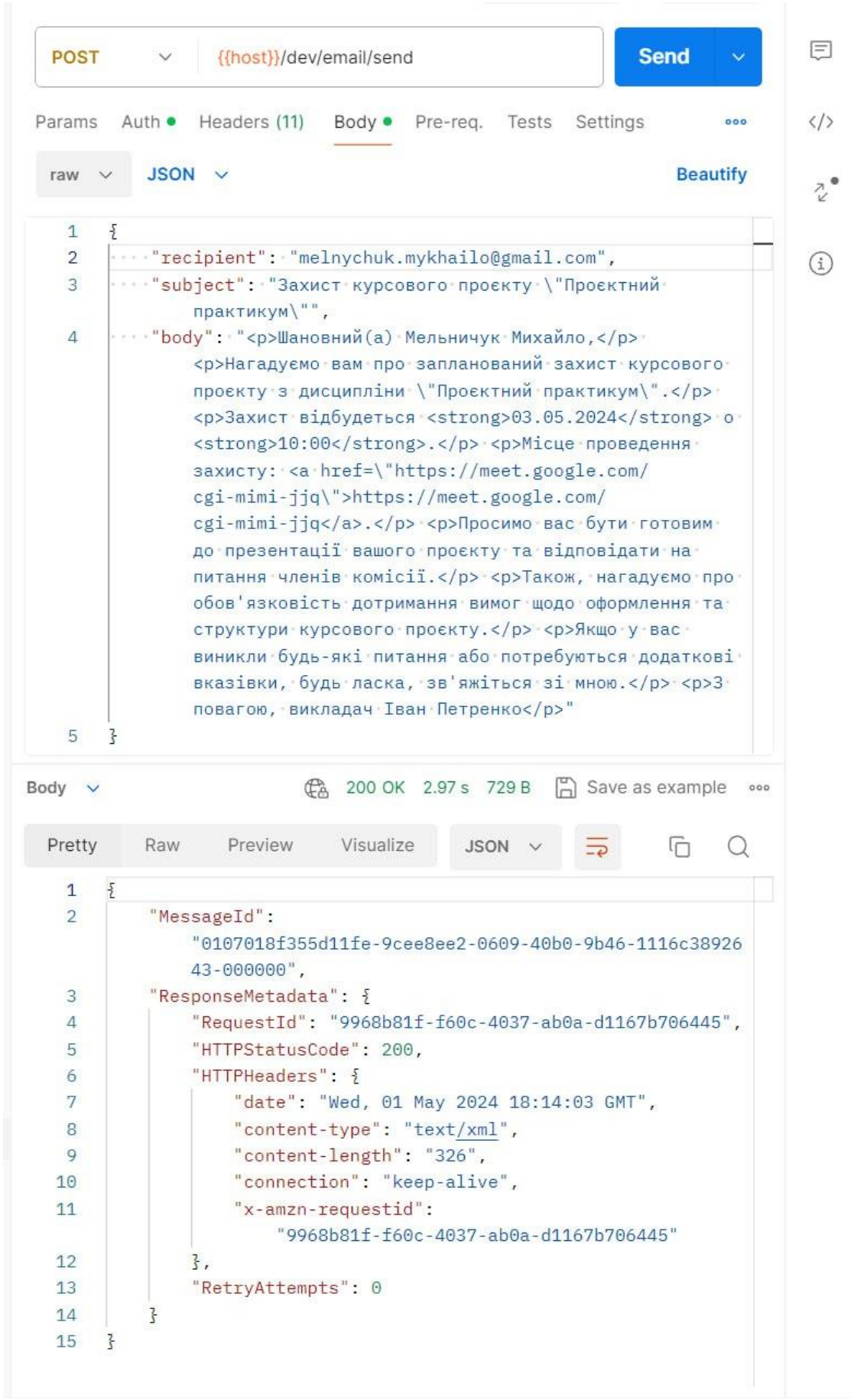

Рисунок 4.15 – Тестування відправки електронного листа через Postman

Таким чином, було проведено тестування функціоналу поштового клієнта для електронної розсилки, продемонстровано, що він виконує поставлені на початку розробки задачі.

4.3 Висновки

У четвертому розділі було проведено аналіз методів тестування веб-застосунків загалом та клієнтів та розсилки електронних листів зокрема. Було проведено тестування розробленого клієнта для розсилки електронних листів, продемонстровано його функціонал.

ВИСНОВКИ

В результаті виконання бакалаврської дипломної роботи, було реалізовано програмні засоби вебсервісу поштового клієнта для електронної розсилки. Вони забезпечують процес створення електронних листів для розсилок, формування груп для розсилок, надсилання електронних листів та отримання електронних листів.

У роботі розвязано такі задачі: проведено аналіз існуючих методів і засобів масової відправки електронних повідомлень для визначення напрямків підвищення продуктивності; розроблено метод масової відправки повідомлень з використанням Amazon Web Services Simple Email Service (AWS SES); розроблено програмні компоненти та систему масової відправки повідомлень на основі запропонованих методів; проведено дослідження розроблених засобів масової відправки повідомлень.

Для розробки було використано мови програмування Python та JavaScript, середовища розробки PyCharm та Visual Studio Code.

У першому розділі, було проведено аналіз стану питання розробки вебсервісу поштового клієнта для електронної розсилки, порівняльний аналіз аналогів, аналіз методів розв’язання задачі. На їх основі було доведено доцільність та актуальність власної розробки, поставлено задачі дослідження.

У другому розділі, було розроблено модель застосунку, проведено аналіз інформаційного забезпечення, розроблено метод створення шаблонів електронних листів, алгоритми роботи вебсервісу.

У третьому розділі, було проведено варіантний аналіз засобів розробки, розроблено алгоритми роботи застосунку, розроблено структурні схеми інтерфейсу застосунку, розроблено вебсервіс.

У четвертому розділі, було проведено аналіз методів тестування, проведено тестування розробленого вебсервісу. Тестування довело його працездатність та коректність роботи. Розроблений застосунок виконує поставлені перед ним на початку розробки задачі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. A Guide to Email Distribution System [Електронний ресурс] – Режим доступу до ресурсу: https://velocitymedia.agency/latest-news/a-guide-to-email-distribution-systems

2. Мельничук М.Е. Аналіз поштових клієнтів для електронної розсилки / М.Е.Мельничук, О.М. Рейда // Матеріали конференції «LIII Науково-технічна конференція підрозділів Вінницького національного технічного університету (2024)», Вінниця, 2024. 2774 с.

3. What is a distribution list in email? [Електронний ресурс] – Режим доступу до ресурсу: https://www.techtarget.com/whatis/definition/distribution-list

4. MailChimp [Електронний ресурс] – Режим доступу до ресурсу: https://templates.mailchimp.com/getting-started/template-language/

5. SendPulse [Електронний ресурс] – Режим доступу до ресурсу: https://sendpulse.ua/

6. GetResponse [Електронний ресурс] – Режим доступу до ресурсу: https://www.getresponse.com/

7. Amazon Cognito [Електронний ресурс] – Режим доступу до ресурсу: https://aws.amazon.com/cognito/?nc1=h_ls

8. Amazon API Gateway [Електронний ресурс] – Режим доступу до ресурсу: https://aws.amazon.com/api-gateway/

9. Amazon SES [Електронний ресурс] – Режим доступу до ресурсу: https://aws.amazon.com/ses/

10. Amazon EFS [Електронний ресурс] – Режим доступу до ресурсу: https://aws.amazon.com/efs/?nc1=h_ls

11. Amazon DynamoDB [Електронний ресурс] – Режим доступу до ресурсу: https://aws.amazon.com/dynamodb/

12. AWS Event Bridge [Електронний ресурс] – Режим доступу до ресурсу: https://aws.amazon.com/eventbridge/

13. What is Java? [Електронний ресурс] – Режим доступу до ресурсу: https://aws.amazon.com/what-is/java/

14. Що таке PHP? [Електронний ресурс] – Режим доступу до ресурсу: https://freehost.com.ua/ukr/faq/wiki/chto-takoe-php/

15. JavaScript [Електронний ресурс] – Режим доступу до ресурсу: https://w3schoolsua.github.io/js/index.html#gsc.tab=0

16. TypeScript [Електронний ресурс] – Режим доступу до ресурсу: https://www.typescriptlang.org/

17. Webstorm [Електронний ресурс] – Режим доступу до ресурсу: https://www.jetbrains.com/webstorm/

18. Sublime Text [Електронний ресурс] – Режим доступу до ресурсу: https://www.sublimetext.com/

19. Spring [Електронний ресурс] – Режим доступу до ресурсу: https://spring.io/

20. Методичні вказівки до виконання бакалаврських кваліфікаційних робіт для студентів спеціальності 121 "Інженерія програмного забезпечення" / Уклад. О. Н. Романюк, Г. О. Черноволик – Вінниця: ВНТУ, 2022. – 52с.

ДОДАТКИ

Додаток А. Технічне завдання

62

Додаток А – Технічне завдання

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

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
Романюк О.Н.
« 12 » березня 2024 р.

Технічне завдання
на бакалаврську дипломну роботу «Розробка веб-орієнтованого застосунку поштового клієнта для електронної розсилки»
за спеціальністю
121 – Інженерія програмного забезпечення

Керівник бакалаврської дипломної роботи:
к.т.н., доц. О.М. Рейда
« 12 » березня 2024 року.
Виконав:
студент гр. 4ПІ-20б М.Е. Мельничук
« 12 » березня 2024 року.

м. Вінниця – 2024 рік

1. Найменування та галузь застосування

Бакалаврська дипломна робота: «Розробка веб-орієнтованого застосунку поштового клієнта для електронної розсилки».

Галузь застосування – веб-технології.

2. Підстава для розробки.

Завдання на роботу, яке затверджене на засіданні кафедри програмного забезпечення – протокол №17 від 20 березня 2024 р.

3. Мета та призначення розробки.

Метою роботи є підвищення продуктивності масової відправки повідомлень за рахунок використання AWS SNS.

4. Вихідні дані для проведення НДР

1. A Guide to Email Distribution System [Електронний ресурс] – Режим доступу до ресурсу: https://velocitymedia.agency/latest-news/a-guide-to-email-distribution-systems

2. What is a distribution list in email? [Електронний ресурс] – Режим доступу до ресурсу: https://www.techtarget.com/whatis/definition/distribution-list

3. Amazon SES [Електронний ресурс] – Режим доступу до ресурсу: https://aws.amazon.com/ses/

4. Spring [Електронний ресурс] – Режим доступу до ресурсу: https://spring.io/

5. Технічні вимоги

Вхідні дані — інформація про користувача, текст електронних листів.

Вихідні дані — надіслані електронні листи, сформовані групи розсилок.

6. Конструктивні вимоги.

Інтерфейс програми поовинен відповідати естетичним та ергономічним вимогам, повинна бути зручною в обслуговуванні та керуванні.

Графічна та текстова документація повинна відповідати діючим стандартам України.

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

–пояснювальна записка до бакалаврської дипломної роботи;

–технічне завдання;

–лістинги програми.

8. Вимоги до рівня уніфікації та стандартизації

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ.

9. Стадії та етапи розробки

№ з/п	Назва етапів бакалаврської дипломної роботи	Строк виконання етапів роботи
1	Аналіз стану розвитку вебсистем для електронної розсилки	14.03.2024 – 24.03.2024
2	Порівняльний аналіз аналогів	25.03.2024 – 07.04.2024
3	Розробка поштового клієнта	08.04.2024 – 18.04.2024
4	Розробка методу масової відправки повідомлень	19.04.2024 – 19.05.2024
5	Тестування застосунку	20.05.2024 – 24.05.2024
6	Оформлення матеріалів до захисту БДР	25.05.2024 – 30.05.24

10. Порядок контролю та прийняття.

Усі етапи бакалаврської роботи контролюються науковим керівником згідно плану по виконанню роботи. Прийняття бакалаврської кваліфікаційної роботи здійснюється ДЕК, затвердженою зав. кафедрою згідно з графіком захисту.

Додаток Б. Протокол перевірки на наявність текстових запозичень

65

Додаток Б – Протокол перевірки на наявність текстових запозичень

ПРОТОКОЛ
ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Розробка веб-орієнтованого застосунку поштового клієнта для електронної розсилки

Тип роботи: БДР

Підрозділ : кафедра програмного забезпечення, ФІТКІ

Науковий керівник: к.т.н., доц. каф. ПЗ Рейда О.М.

Оригінальність	95,8%
Схожість	4,2%

Аналіз звіту подібності

▪ **Запозичення, виявлені у роботі, оформленні коректно і не містять ознак плагіату.**

☐ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цілісності роботи і відсутності самостійності її автора. Роботу направити на доопрацювання.

☐ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховання недобросовісних запозичень.

Особа, відповідальна за перевірку _____ Черноволик Г. О.

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck

Автор роботи _____ Мельничук М.Е.

Керівник роботи _____ Рейда О.М.

Додаток В – Лістинг програми

mass-mailer_verify-email-identity.py

```
import json
import boto3


def lambda_handler(event, context):
    body = json.loads(event['body']);
    try:
        validateRequestBody(body)
    except RequestBodyValidationError as e:
        print('Handled exception: ', e);
        return create4xxErrorResponse(e);

    identity = body['identity'];

    response = verify_email_identity(identity);

    print(f'verifying identity: {body.get('identity')}');
    return response;


def validateRequestBody(body):
    errorCauses = []
    if ('identity' not in body):
        errorCauses.append('Missing identity')

    if (len(errorCauses) > 0):
        raise RequestBodyValidationError("Invalid request body", errorCauses)


def create2xxResponse(sent_response):
    print('sent_response: ', sent_response)
```

```
    return {
     'statusCode': sent_response['ResponseMetadata']['HTTPStatusCode'],
     'body': json.dumps(sent_response)
    };


def create5xxErrorResponse(e):
    return {
        'statusCode': 500,
        'body': json.dumps({
            'message': str(e)
        })
    };


def verify_email_identity(identity):
    AWS_REGION = "eu-central-1";
    client = boto3.client('ses',region_name=AWS_REGION);
    try:
        response = client.verify_email_identity(
            EmailAddress=identity
        );
    except Exception as e:
        print('Caught Exception: ', e);
        return create5xxErrorResponse(e);
    else:
        return create2xxResponse(response);



class RequestBodyValidationError(Exception):
    def __init__(self, message, errorCauses=None):
        super().__init__(message);
        self.errorCauses = json.dumps(errorCauses);
```

mass-mailer_send-email.py

```
import json
import boto3
from botocore.exceptions import ClientError


def lambda_handler(event, context):
    print('Send email function invoked...');
    body = json.loads(event['body']);
    try:
        validateRequestBody(body)
    except RequestBodyValidationError as e:
        print('Handled exception: ', e);
        return create4xxErrorResponse(e);

    response = send_email(body);

    print(f'recipient: {body.get('recipient')}, subject: {body['subject']}, body: {body['body']}');
    return create2xxResponse(response);



def validateRequestBody(body):
    errorCauses = []
    if ('recipient' not in body):
        errorCauses.append('Missing recipient')
    if ('subject' not in body):
        errorCauses.append('Missing subject')
    if ('body' not in body):
        errorCauses.append('Missing body')

    if (len(errorCauses) > 0):
        raise RequestBodyValidationError("Invalid request body", errorCauses)
```

```
def create4xxErrorResponse(e):
    return {
        'statusCode': 400,
        'body': json.dumps({
            'message': str(e),
            'errors': e.errorCauses
        })
    };

def create5xxErrorResponse(e):
    return {
        'statusCode': 500,
        'body': json.dumps({
            'message': str(e)
        })
    };

def create2xxResponse(sent_response):
    print('sent_response: ', sent_response)
    return {
     'statusCode': sent_response['ResponseMetadata']['HTTPStatusCode'],
     'body': json.dumps(sent_response)
    };

def send_email(request_body):
    AWS_REGION = "eu-central-1";
    SENDER = "mass-mailer@roman-ko.net";
    CHARSET = "UTF-8";
    client = boto3.client('ses',region_name=AWS_REGION);

    try:
        recipient = request_body["recipient"];
        print(f'attempting to send an email to: {recipient}');
        response = client.send_email(
            Destination={
```

```
            'ToAddresses': [
                recipient
            ]
        },
        Message={
            'Body': {
                'Html': {
                    'Charset': CHARSET,
                    'Data': request_body["body"],
                }
            },
            'Subject': {
                'Charset': CHARSET,
                'Data': request_body["subject"],
            },
        },
        Source=SENDER
    )
except ClientError as e:
    print('Caught ClientError: ', e);
    print(e.response['Error']['Message']);
    return create5xxErrorResponse(e);
except Exception as e:
    print('Caught Exception: ', e);
    return create5xxErrorResponse(e);
else:
    print(f'Email sent! Message ID: {response['MessageId']}');
    return response;
return None

class RequestBodyValidationError(Exception):
    def __init__(self, message, errorCauses=None):
        super().__init__(message);
        self.errorCauses = json.dumps(errorCauses);
```

mass-mailer_verify-email-identity-sam-template.yml

```
# This AWS SAM template has been generated from your function's configuration. If
# your function has one or more triggers, note that the AWS resources associated
# with these triggers aren't fully specified in this template and include
# placeholder values. Open this template in AWS Application Composer or your
# favorite IDE and modify it to specify a serverless application with other AWS
# resources.
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
Description: An AWS Serverless Application Model template describing your function.
Resources:
  massmailerverifyemailidentity:
    Type: AWS::Serverless::Function
    Properties:
      CodeUri: .
      Description: ''
      MemorySize: 128
      Timeout: 3
      Handler: lambda_function.lambda_handler
      Runtime: python3.12
      Architectures:
        - x86_64
      EphemeralStorage:
        Size: 512
      EventInvokeConfig:
        MaximumEventAgeInSeconds: 21600
        MaximumRetryAttempts: 2
      PackageType: Zip
      Policies:
        - Statement:
            - Effect: Allow
              Action:
                - logs:CreateLogGroup
                - logs:CreateLogStream
                - logs:PutLogEvents
              Resource: '*'
            - Sid: VisualEditor0
              Effect: Allow
              Action:
                - ses:SendEmail
                - ses:SendRawEmail
              Resource:
                - arn:aws:ses:eu-central-1:999565175928:identity/roman-ko.net
                - arn:aws:ses:eu-central-1:999565175928:identity/*@gmail.com
                - arn:aws:ses:*:999565175928:configuration-set/*
      SnapStart:
        ApplyOn: None
      Events:
        Api1:
```

```
    Type: Api
    Properties:
      Path: /email/verify
      Method: POST
RuntimeManagementConfig:
  UpdateRuntimeOn: Auto
```

mass-mailer_send-email_sam-template.yml

```
# This AWS SAM template has been generated from your function's configuration. If
# your function has one or more triggers, note that the AWS resources associated
# with these triggers aren't fully specified in this template and include
# placeholder values. Open this template in AWS Application Composer or your
# favorite IDE and modify it to specify a serverless application with other AWS
# resources.
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
Description: An AWS Serverless Application Model template describing your function.
Resources:
  massmailersendemail:
    Type: AWS::Serverless::Function
    Properties:
      CodeUri: .
      Description: ''
      MemorySize: 128
      Timeout: 3
      Handler: lambda_function.lambda_handler
      Runtime: python3.12
      Architectures:
        - x86_64
      EphemeralStorage:
        Size: 512
      EventInvokeConfig:
        MaximumEventAgeInSeconds: 21600
        MaximumRetryAttempts: 2
      PackageType: Zip
      Policies:
```

```
    - Statement:
        - Effect: Allow
          Action:
            - logs:CreateLogGroup
            - logs:CreateLogStream
            - logs:PutLogEvents
          Resource: '*'
        - Sid: VisualEditor0
          Effect: Allow
          Action:
            - ses:SendEmail
            - ses:SendRawEmail
          Resource:
            - arn:aws:ses:eu-central-1:999565175928:identity/roman-ko.net
            - arn:aws:ses:eu-central-1:999565175928:identity/*@gmail.com
            - arn:aws:ses:*:999565175928:configuration-set/*
  SnapStart:
    ApplyOn: None
  Events:
    Api1:
      Type: Api
      Properties:
        Path: /email/send
        Method: POST
  RuntimeManagementConfig:
    UpdateRuntimeOn: Auto
```

mass-mailer_inbox-page.html

<html class="wf-inter-n1-active wf-inter-n2-active wf-inter-n3-active wf-inter-n4-active wf-inter-n5-active wf-inter-n6-active wf-inter-n7-active wf-inter-n8-active wf-active"><head> <meta charset="utf-8"> <title>Melnychuk.misha03518042's site</title> <meta name="description" content=""> <meta property="og:type" content="article"> <meta property="og:title" content="Melnychuk.misha03518042's site"> <meta property="og:description" content=""> <meta property="og:image" content="https://s3.us-east-1.amazonaws.com/cdn-siter/ac295bf3-1a4b-4723-8294-efd2ad823498-url_og_image.png"> <meta property="twitter:card"

content="summary_large_image"> <meta property="twitter:title" content="Melnychuk.misha03518042's site"> <meta property="twitter:description" content=""> <meta property="twitter:image" content="https://s3.us-east-1.amazonaws.com/cdn-siter/ac295bf3-1a4b-4723-8294-efd2ad823498-url_og_image.png"> <script type="text/javascript" id="www-widgetapi-script" src="https://www.youtube.com/s/player/8fc6998a/www-widgetapi.vflset/www-widgetapi.js" async=""></script><script type="text/javascript" id="variables">

window.site_id = 'melnychukmisha0-1714591726516'

window.page_id = 'pg_TcCQ5AYnoUPm29yq3m7uJuYj5'

window.pages = [{"path":"melnychukmisha03518042s-site","id":"pg_TcCQ5AYnoUPm29yq3m7uJuYj5","home":true}];

window.preview = true;

window.devices = [{"name":"res375","title":"Mobile","width":375},{"name":"res640","title":"Tablet","width":640},{"name":"desktop","title":"Desktop","width":1200}]

</script> <link rel="stylesheet" media="screen" href="https://api.siter.io/assets/application-47d9c80bc006194ad960d6dc2d9708abf9e984360309d1d501e7acce8017c331.css"><link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Inter:100,200,300,400,500,600,700,800" media="all"> <script type="text/javascript" charset="UTF-8" src="https://maps.googleapis.com/maps-api-v3/api/js/56/10/intl/uk_ALL/common.js"></script><script type="text/javascript" charset="UTF-8" src="https://maps.googleapis.com/maps-api-v3/api/js/56/10/intl/uk_ALL/util.js"></script></head><body class="siter-ready">

<div data-unit="px" class="page-content__component rectangle_fe2a334b-3e64-49e8-b248-5b59107147b7 rectangle_fe2a334b-3e64-49e8-b248-5b59107147b7_hover" data-container="grid" id="fe2a334b-3e64-49e8-b248-5b59107147b7" data-type="rectangle">

<div data-unit="px" class="page-content__component rectangle_5cf2501e-d3eb-4f60-a3e6-c4452211177f rectangle_5cf2501e-d3eb-4f60-a3e6-c4452211177f_hover" data-container="grid" id="5cf2501e-d3eb-4f60-a3e6-c4452211177f" data-type="rectangle">

<div data-unit="px" class="page-content__component rectangle_37e08837-5525-487b-9d69-2a5ae32cc3ab" data-container="grid" id="37e08837-5525-487b-9d69-2a5ae32cc3ab" data-type="rectangle" style="background-image:url('https://cdn.siter.io/assets/ast_ukLBEMyeggGHq4aijbR1dp71d/76688b89-76b7-4403-9e38-4d1d1fdd1240.webp');">

Inbox <script>const siterEffects = {}</script>   <svg

onclick="EditorPageWidgets.notifyHide()" width="16" height="16" viewBox="0 0 16 16" fill="none" xmlns="http://www.w3.org/2000/svg" style="position: absolute; top: 20px; right: 22px;"> <g opacity="0.5"> <path d="M15 1L1 15" stroke="#333333"></path> <path d="M1 1L15 15" stroke="#333333"></path> </g> </svg> <h4 class="title"></h4>   <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script> <script src="https://www.youtube.com/iframe_api"></script> <script src="https://player.vimeo.com/api/player.js"></script> <script src="https://maps.googleapis.com/maps/api/js?key=AIzaSyDrutUSPfjuwC8b4W9pitclowi3btRQkMo&v=3.exp&libraries=geometry,drawing,places"></script> <script src="https://api.siter.io/assets/anime.min-8439bccbde3050f7ead5acddedd897fd20bc7203a3ba209514f38680f12ebab0.js"></script> <link rel="stylesheet" href="https://api.siter.io/assets/magnific-popup-2f7f85183333c84a42262b5f8a4f8251958809e29fa31c65bdee53c4603502cd.css"> <script src="https://api.siter.io/assets/magnific-popup.min-df6ae89414437784941d5dc0ada892e5c27aac18e58676eba0fa1b5e6b288b74.js"></script> <script src="https://api.siter.io/assets/swiper.min-dceaef8c10f13b33b0328938ecbe2b8c3f0d03f28ae56a8864c767b8a2d21caf.js"></script> <link rel="stylesheet" href="https://api.siter.io/assets/swiper-18be8aa3f032dded246a45a9da3dafdb3934e39e1f1b3b623c1722f3152b2788.css"> <script src="https://api.siter.io/assets/application-b7c06fec0e822cccd2ffcea972d7ee8ce166e0b0162f68adb89219130bf4f5ad.js"></script> </body></html>

mass-mailer_send-email-page.html

<html class="wf-inter-n1-active wf-inter-n2-active wf-inter-n3-active wf-inter-n4-active wf-inter-n5-active wf-inter-n6-active wf-inter-n7-active wf-inter-n8-active wf-active"><head> <meta charset="utf-8"> <title>Melnychuk.misha03518042's site</title> <meta name="description" content=""> <meta property="og:type" content="article"> <meta property="og:title" content="Melnychuk.misha03518042's site"> <meta property="og:description" content=""> <meta property="og:image" content="https://s3.us-east-1.amazonaws.com/cdn-siter/63bcd99e-2bd2-465f-a01d-ca80ff12b391-url_og_image.png"> <meta property="twitter:card" content="summary_large_image"> <meta property="twitter:title" content="Melnychuk.misha03518042's site"> <meta property="twitter:description" content=""> <meta property="twitter:image" content="https://s3.us-east-1.amazonaws.com/cdn-siter/63bcd99e-2bd2-

465f-a01d-ca80ff12b391-url_og_image.png"> <script type="text/javascript" id="www-widgetapi-script" src="https://www.youtube.com/s/player/8fc6998a/www-widgetapi.vflset/www-widgetapi.js" async=""></script><script type="text/javascript" id="variables">
window.site_id = 'melnychuk-misha0-1714591726516'
window.page_id = 'pg_TcCQ5AYnoUPm29yq3m7uJuYj5'
window.pages = [{"path":"melnychuk-misha03518042s-site","id":"pg_TcCQ5AYnoUPm29yq3m7uJuYj5","home":true}];
window.preview = true;
window.devices = [{"name":"res375","title":"Mobile","width":375},{"name":"res640","title":"Tablet","width":640},{"name":"desktop","title":"Desktop","width":1200}]
</script> <link rel="stylesheet" media="screen" href="https://api.siter.io/assets/application-47d9c80bc006194ad960d6dc2d9708abf9e984360309d1d501e7acce8017c331.css"><link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Inter:100,200,300,400,500,600,700,800" media="all"> <script type="text/javascript" charset="UTF-8" src="https://maps.googleapis.com/maps-api-v3/api/js/56/10/intl/uk_ALL/common.js"></script><script type="text/javascript" charset="UTF-8" src="https://maps.googleapis.com/maps-api-v3/api/js/56/10/intl/uk_ALL/util.js"></script>
<style>.rectangle_169629c3-459b-406f-8509-f59745fe2f95 .strw-shape {
content: '';
position: absolute;
top:0;
left: 0;
right:0;
bottom:0;
background-image: -webkit-linear-gradient(to right, rgba(255, 255, 255, 1), rgba(255, 255, 255, 1));
background-image: -moz-linear-gradient(to right, rgba(255, 255, 255, 1), rgba(255, 255, 255, 1));
background-image: -ms-linear-gradient(to right, rgba(255, 255, 255, 1), rgba(255, 255, 255, 1));
background-image: -o-linear-gradient(to right, rgba(255, 255, 255, 1), rgba(255, 255, 255, 1));
background-image: linear-gradient(to right, rgba(255, 255, 255, 1), rgba(255, 255, 255, 1));

```
}

.rectangle_169629c3-459b-406f-8509-f59745fe2f95:hover .strw-shape, .rectangle_169629c3-459b-406f-8509-f59745fe2f95_hover .strw-shape {

  opacity: 1;
  background-image: -webkit-;
  background-image:  -moz-;
  background-image:  -ms-;
  background-image:  -o-;
  background-image:  ;

}.rectangle_169629c3-459b-406f-8509-f59745fe2f95 { top: 179px;left: -360px;width: 396px;border: 1px solid rgba(217, 217, 217, 1);height: 55px;opacity: 1;overflow: hidden;position: absolute;box-shadow: none;visibility: inherit;background-size: cover;background-repeat: no-repeat;background-position: center;border-radius: 0px 16px 16px 0px; }.rectangle_169629c3-459b-406f-8509-f59745fe2f95 {
        left: -moz-calc(50% - 600px + -360px);
        left: -webkit-calc(50% - 600px + -360px);
        left: -o-calc(50% - 600px + -360px);
        left: calc(50% - 600px + -360px)
      }.text_3bde17a4-1cca-4415-9396-a83b2dfb02c7 [data-wysiwyg="true"] {
  display: table-cell;
  vertical-align: middle
}
.text_3bde17a4-1cca-4415-9396-a83b2dfb02c7 {
  color: rgba(0, 0, 0, 1);

}
.text_3bde17a4-1cca-4415-9396-a83b2dfb02c7 { top: 189px;left: -266px;width: 110px;height: 34px;display: table;opacity: 1;font-size: 28px;position: absolute;text-align: left;font-family: Inter;font-weight: 400;line-height: 20px;text-shadow: none;visibility: inherit;letter-spacing: 0em;box-shadow: none;text-transform: normal;text-decoration: none; }.text_3bde17a4-1cca-4415-9396-a83b2dfb02c7 {
        left: -moz-calc(50% - 600px + -266px);
        left: -webkit-calc(50% - 600px + -266px);
```

```
                left: -o-calc(50% - 600px + -266px);
                left: calc(50% - 600px + -266px)
              }</style>
        <style>.rectangle_67289745-0675-4e6d-88d4-1e6df67ad51a .strw-shape {
          content: '';
          position: absolute;
          top:0;
          left: 0;
          right:0;
          bottom:0;
          background-image: -webkit-linear-gradient(to right, rgba(246, 242, 242, 1), rgba(246, 242, 242,
1));
          background-image:    -moz-linear-gradient(to right, rgba(246, 242, 242, 1), rgba(246, 242, 242,
1));
          background-image:     -ms-linear-gradient(to right, rgba(246, 242, 242, 1), rgba(246, 242, 242,
1));
          background-image:      -o-linear-gradient(to right, rgba(246, 242, 242, 1), rgba(246, 242, 242,
1));
          background-image:         linear-gradient(to right, rgba(246, 242, 242, 1), rgba(246, 242, 242,
1));
        }

        .rectangle_67289745-0675-4e6d-88d4-1e6df67ad51a:hover .strw-shape, .rectangle_67289745-
0675-4e6d-88d4-1e6df67ad51a_hover .strw-shape {

          opacity: 1;
          background-image: -webkit-;
          background-image:    -moz-;
          background-image:     -ms-;
          background-image:      -o-;
          background-image:         ;

        }.rectangle_67289745-0675-4e6d-88d4-1e6df67ad51a { top: 262px;left: -360px;width:
396px;border: 1px solid rgba(217, 217, 217, 1);height: 55px;opacity: 1;overflow: hidden;position:
absolute;box-shadow: none;visibility: inherit;background-size: cover;background-repeat: no-
```

repeat;background-position: center;border-radius: 0px 16px 16px 0px; }.rectangle_67289745-0675-4e6d-88d4-1e6df67ad51a {
left: -moz-calc(50% - 600px + -360px);
left: -webkit-calc(50% - 600px + -360px);
left: -o-calc(50% - 600px + -360px);
left: calc(50% - 600px + -360px)
}.text_b66eae90-aa12-4569-8834-a82469c3427a [data-wysiwyg="true"] {
display: table-cell;
vertical-align: middle
}

.text_b66eae90-aa12-4569-8834-a82469c3427a {

color: rgba(0, 0, 0, 1);

}

.text_b66eae90-aa12-4569-8834-a82469c3427a { top: 272px;left: -266px;width: 284px;height: 34px;display: table;opacity: 1;font-size: 28px;position: absolute;text-align: left;font-family: Inter;font-weight: 400;line-height: 20px;text-shadow: none;visibility: inherit;letter-spacing: 0em;box-shadow: none;text-transform: normal;text-decoration: none; }.text_b66eae90-aa12-4569-8834-a82469c3427a {
left: -moz-calc(50% - 600px + -266px);
left: -webkit-calc(50% - 600px + -266px);
left: -o-calc(50% - 600px + -266px);
left: calc(50% - 600px + -266px)
}</style>
</head>
<body class="siter-ready">

<div data-unit="px" class="page-content__component rectangle_37e08837-5525-487b-9d69-2a5ae32cc3ab" data-container="grid" id="37e08837-5525-487b-9d69-2a5ae32cc3ab" data-type="rectangle" style="background-

image:url('https://cdn.siter.io/assets/ast_ukLBEMyeggGHq4aijbR1dp71d/76688b89-76b7-4403-9e38-4d1d1fdd1240.webp');">

Inbox <script>const siterEffects = {}</script>   <svg onclick="EditorPageWidgets.notifyHide()" width="16" height="16" viewBox="0 0 16 16" fill="none" xmlns="http://www.w3.org/2000/svg" style="position: absolute; top: 20px; right: 22px;"> <g opacity="0.5"> <path d="M15 1L1 15" stroke="#333333"></path> <path d="M1 1L15 15" stroke="#333333"></path> </g> </svg> <h4 class="title"></h4>   <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script> <script src="https://www.youtube.com/iframe_api"></script> <script src="https://player.vimeo.com/api/player.js"></script>

<style>.group_0d729529-77a1-4b18-ac68-12486139da12 { top: 0px;left: 0px;width: 100%;height: 0px;opacity: 1;position: absolute;visibility: inherit;background-color: transparent; }.group_803c38a6-cc77-4d8b-a27f-f2d3da84d536 { top: 0px;left: 0px;width: 100%;height: 0px;opacity: 1;position: absolute;visibility: inherit;background-color: transparent; }</style><style>.page_pg_TcCQ5AYnoUPm29yq3m7uJuYj5 { height: 1080px;border-radius: 0px 0px 0px 0px; }.nl-desktop {
display: inline;
content: normal;
}.nl-res640 {
content: " ";
display: none;
}.nl-res375 {
content: " ";
display: none;
}@media (max-width: 1199px) {.nl-desktop {
content: " ";
display: none;
}.nl-res640 {

```
        display: inline;
        content: normal;
      }.nl-res375 {
        content: " ";
        display: none;
      }}@media (max-width: 639px) {.nl-desktop {
        content: " ";
        display: none;
      }.nl-res640 {
        content: " ";
        display: none;
      }.nl-res375 {
        display: inline;
        content: normal;
      }}body {
    background-image: -webkit-linear-gradient(to right, rgba(255, 255, 255, 1), rgba(255, 255, 255,
1));
    background-image:   -moz-linear-gradient(to right, rgba(255, 255, 255, 1), rgba(255, 255, 255,
1));
    background-image:    -ms-linear-gradient(to right, rgba(255, 255, 255, 1), rgba(255, 255, 255,
1));
    background-image:     -o-linear-gradient(to right, rgba(255, 255, 255, 1), rgba(255, 255, 255,
1));
    background-image:        linear-gradient(to right, rgba(255, 255, 255, 1), rgba(255, 255, 255,
1));
    background-size: cover;
    background-repeat: no-repeat;
    background-position: top center;
  }

  .page_pg_TcCQ5AYnoUPm29yq3m7uJuYj5 {
    background-repeat: no-repeat;
    background-position: top center;
    background-size: cover;
  }</style>
```

<script src="https://maps.googleapis.com/maps/api/js?key=AIzaSyDrutUSPfjuwC8b4W9pitclowi3btRQkMo&v=3.exp&libraries=geometry,drawing,places"></script> <script src="https://api.siter.io/assets/anime.min-8439bccbde3050f7ead5acddedd897fd20bc7203a3ba209514f38680f12ebab0.js"></script> <link rel="stylesheet" href="https://api.siter.io/assets/magnific-popup-2f7f85183333c84a42262b5f8a4f8251958809e29fa31c65bdee53c4603502cd.css"> <script src="https://api.siter.io/assets/magnific-popup.min-df6ae89414437784941d5dc0ada892e5c27aac18e58676eba0fa1b5e6b288b74.js"></script> <script src="https://api.siter.io/assets/swiper.min-dceaef8c10f13b33b0328938ecbe2b8c3f0d03f28ae56a8864c767b8a2d21caf.js"></script> <link rel="stylesheet" href="https://api.siter.io/assets/swiper-18be8aa3f032dded246a45a9da3dafdb3934e39e1f1b3b623c1722f3152b2788.css"> <script src="https://api.siter.io/assets/application-b7c06fec0e822cccd2ffcea972d7ee8ce166e0b0162f68adb89219130bf4f5ad.js"></script>
</body></html>

Додаток Г. Ілюстративна частина

ІЛЮСТРАТИВНА ЧАСТИНА

РОЗРОБКА ВЕБ-ОРІЄНТОВАНОГО ЗАСТОСУНКУ ПОШТОВОГО КЛІЄНТА ДЛЯ ЕЛЕКТРОННОЇ РОЗСИЛКИ

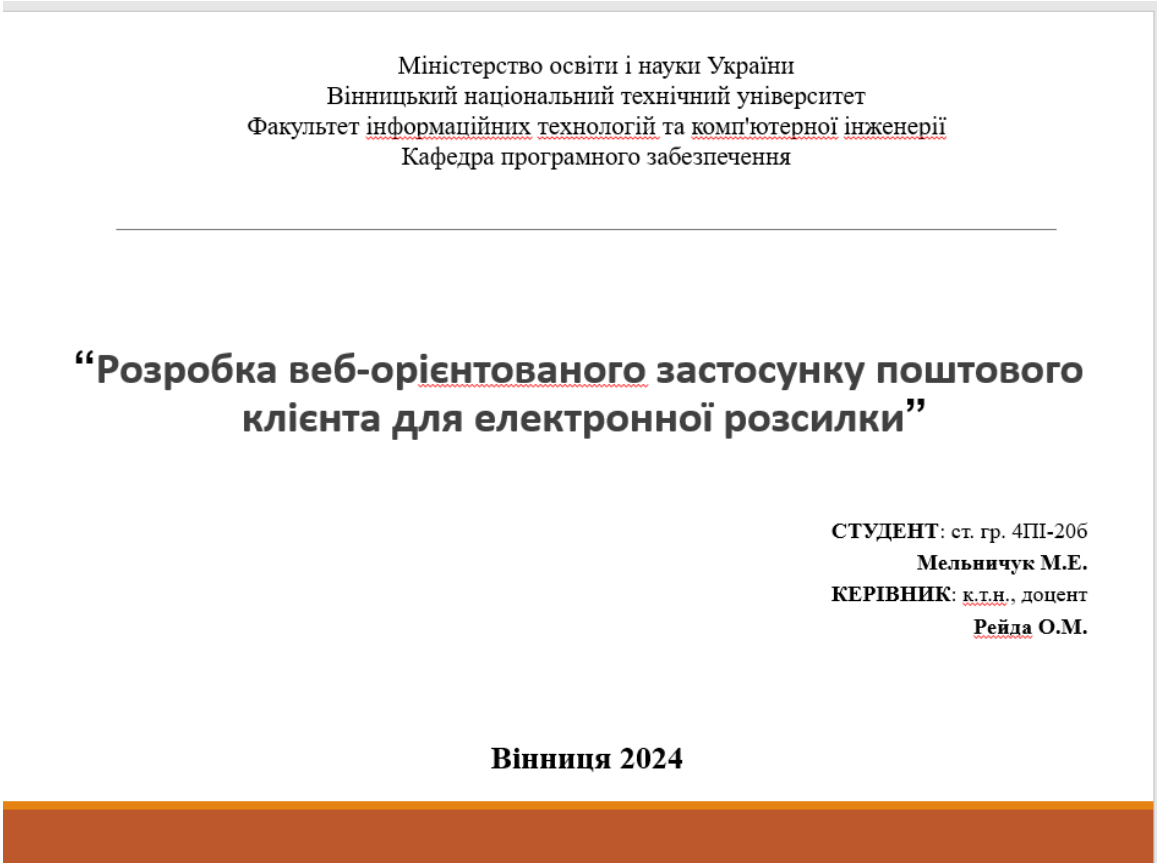

Рисунок Г.1 – Титульний слайд

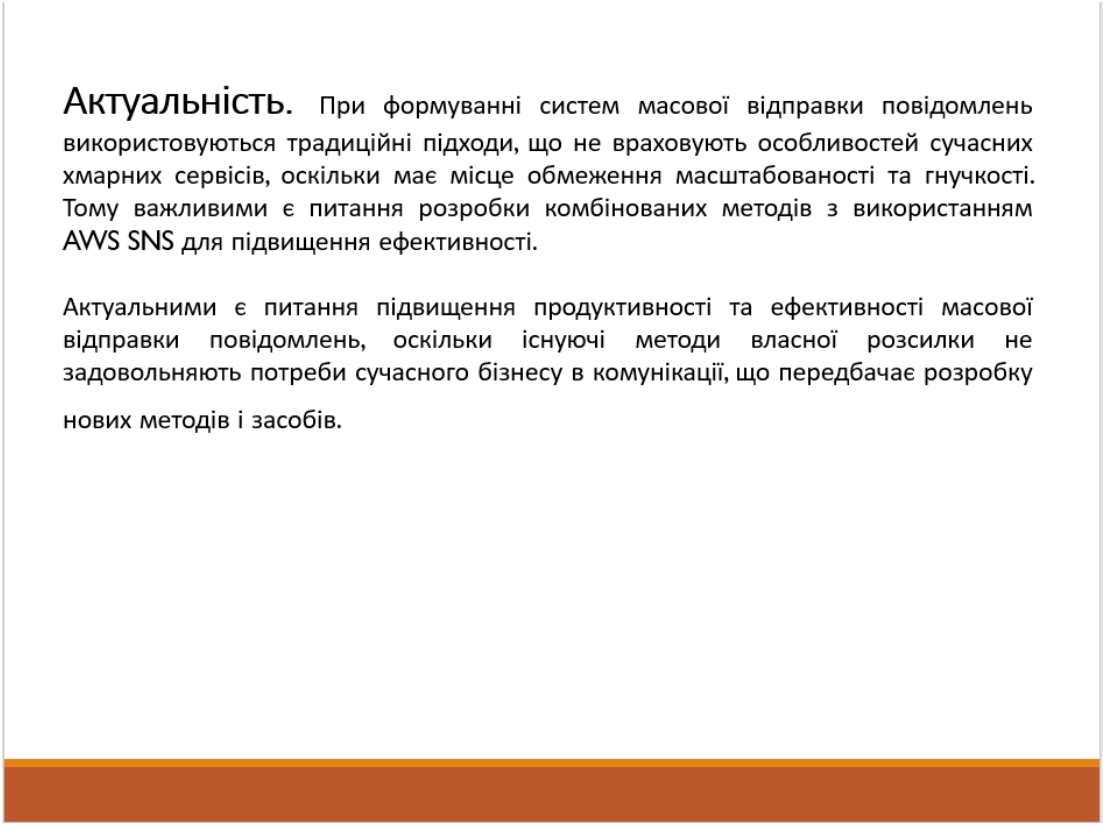

Рисунок Г.2 – Актуальність

Мета і завдання дослідження. Метою роботи є підвищення продуктивності масової відправки повідомлень за рахунок використання сервісе відправлення повідомлень для встановлення зв'язку Amazon Web Services Simple Notification Service (AWS SNS).

Предмет дослідження. Методи та засоби розробки веб-орієнтованого застосунку поштового клієнта для електронної розсилки з використанням Amazon Web Services Simple Notification Service (AWS SNS).

Об'єктом дослідження. Процес розробки системи масової відправки електронних повідомлень у системах бізнес-комунікації.

3

Рисунок Г.3 – Мета, завдання, предмет, і об'єкт достілження

Задачі

1. провести аналіз існуючих методів і засобів масової відправки електронних повідомлень для визначення напрямків підвищення продуктивності;
2. запропонувати новий метод масової відправки повідомлень з використанням Amazon Web Services Simple Notification Service (AWS SNS);
3. розробити програмні компоненти та систему масової відправки повідомлень на основі запропонованих методів;
4. провести дослідження розроблених засобів масової відправки повідомлень.

Рисунок Г.4 – Задачі

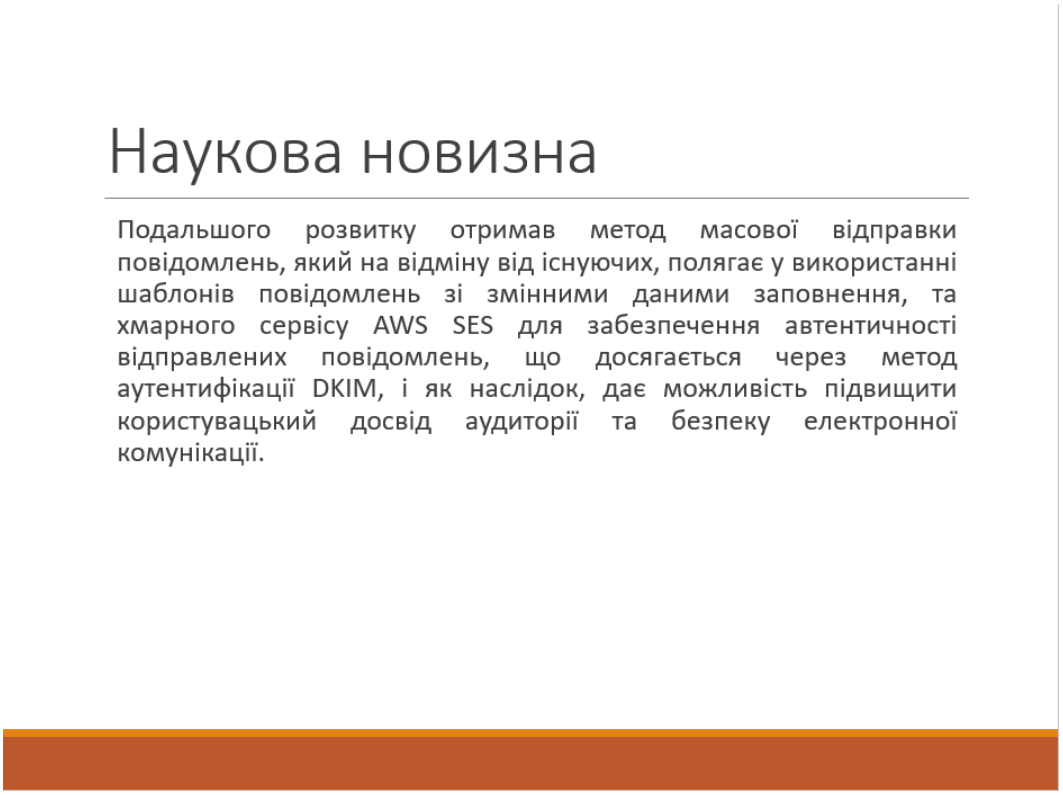

Рисунок Г.5 – Наукова новизна

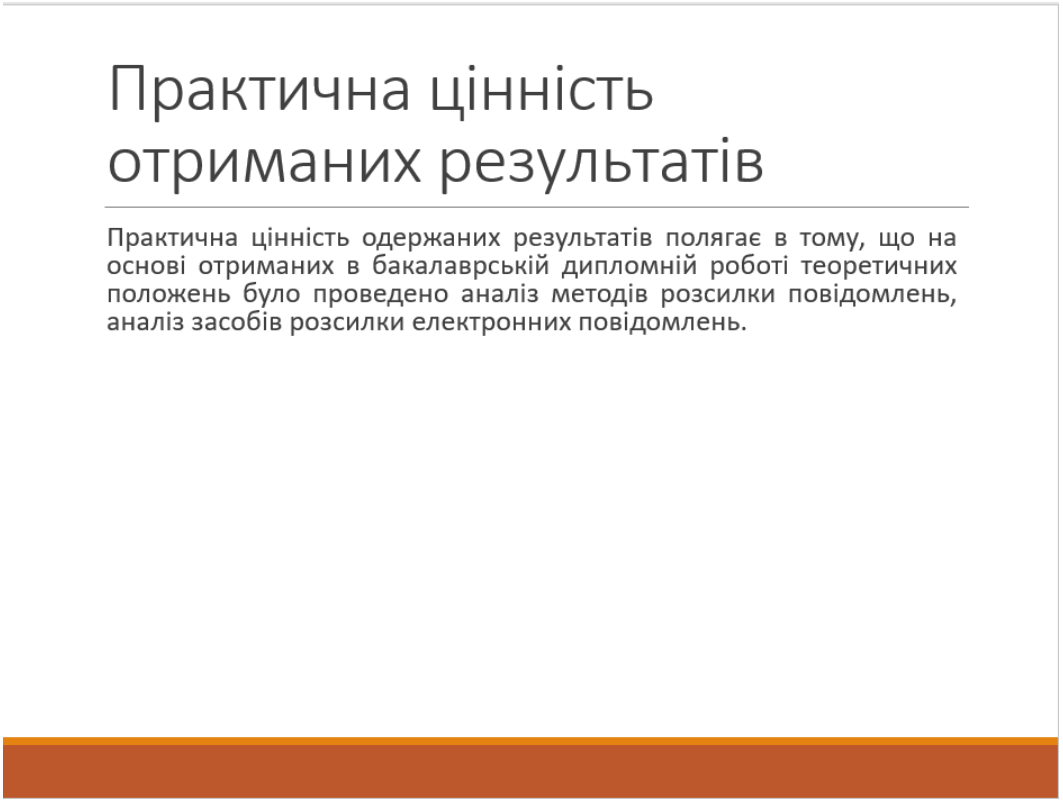

Рисунок Г.6 – Практична цінність отриманих результатів

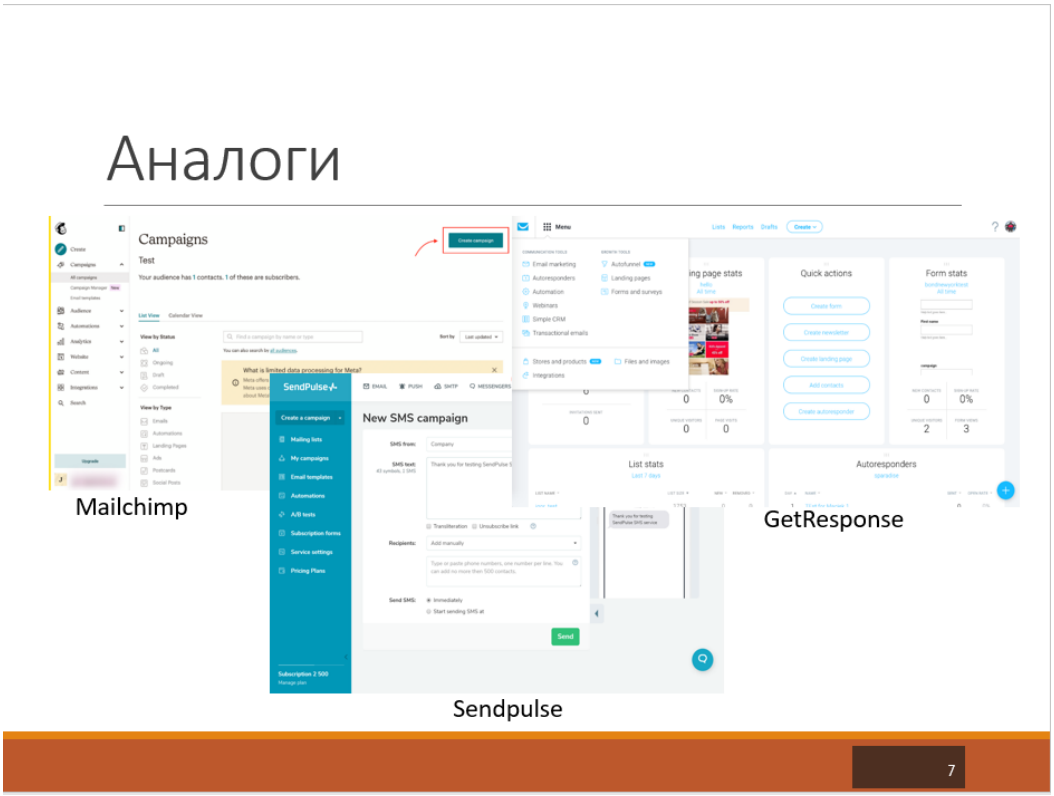

Рисунок Г.7 – Аналоги

	GetResponse	MailChimp	SendPulse	Власна розробка
Формування груп адресатів	1	0	1	1
Фільтрація адресатів відповідно до критеріїв	1	1	1	1
Планування розсилок	0	1	1	1
Управління контентом	1	1	0	1
Підтримка багатомовних розсилок	1	1	1	1
Аналітика та звітність	1	0	1	1
Тестування розсилок	1	1	1	1
Сумарний коефіцієнт	6	5	6	7

Рисунок Г.8 – Порівняльний аналіз аналогів

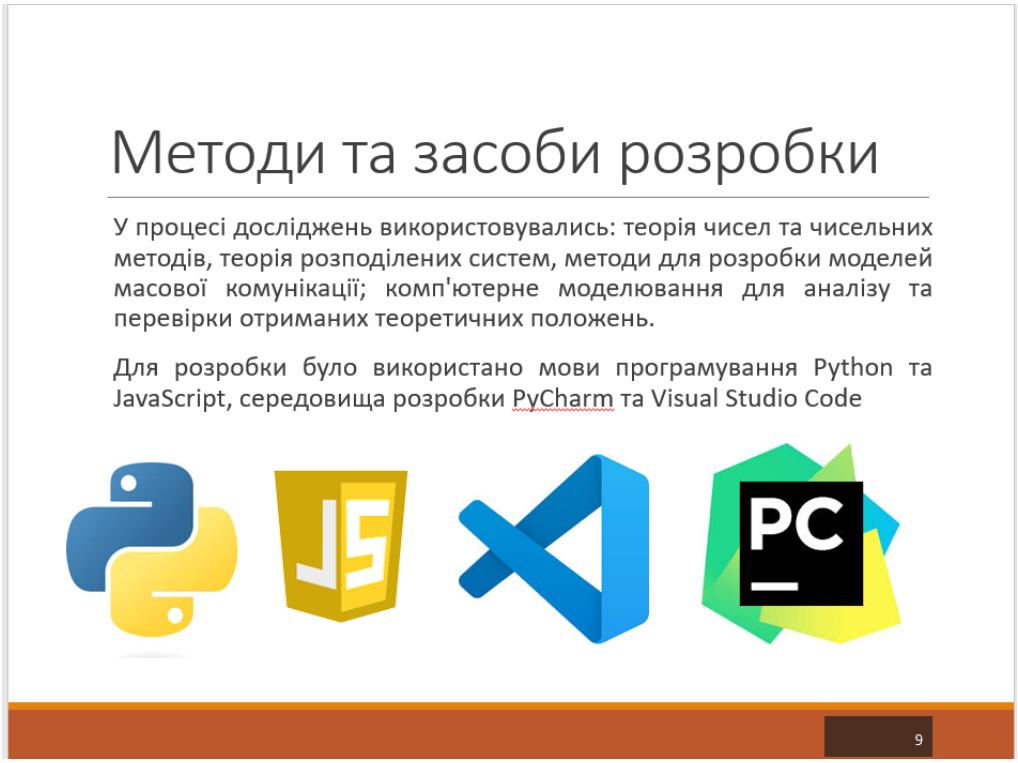

Рисунок Г.9 – Методи та засоби розробки

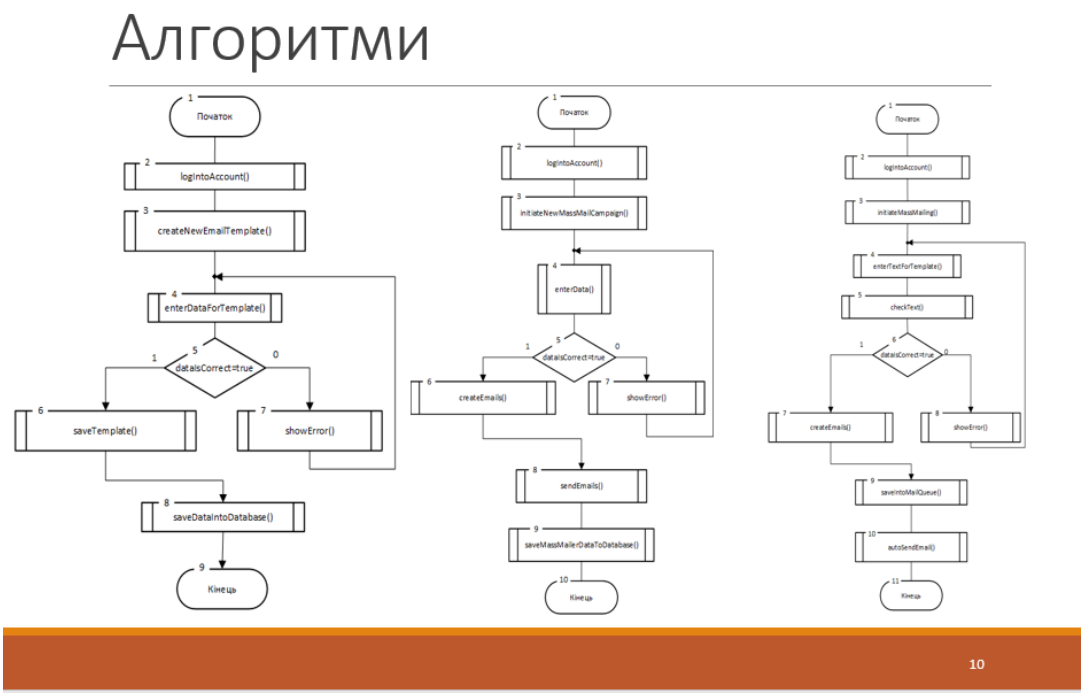

Рисунок Г.10 – Алгоритми

Рисунок Г.11 – Модель застосунку

Рисунок Г.12 – Алгоритм створення шаблонів електронних листів

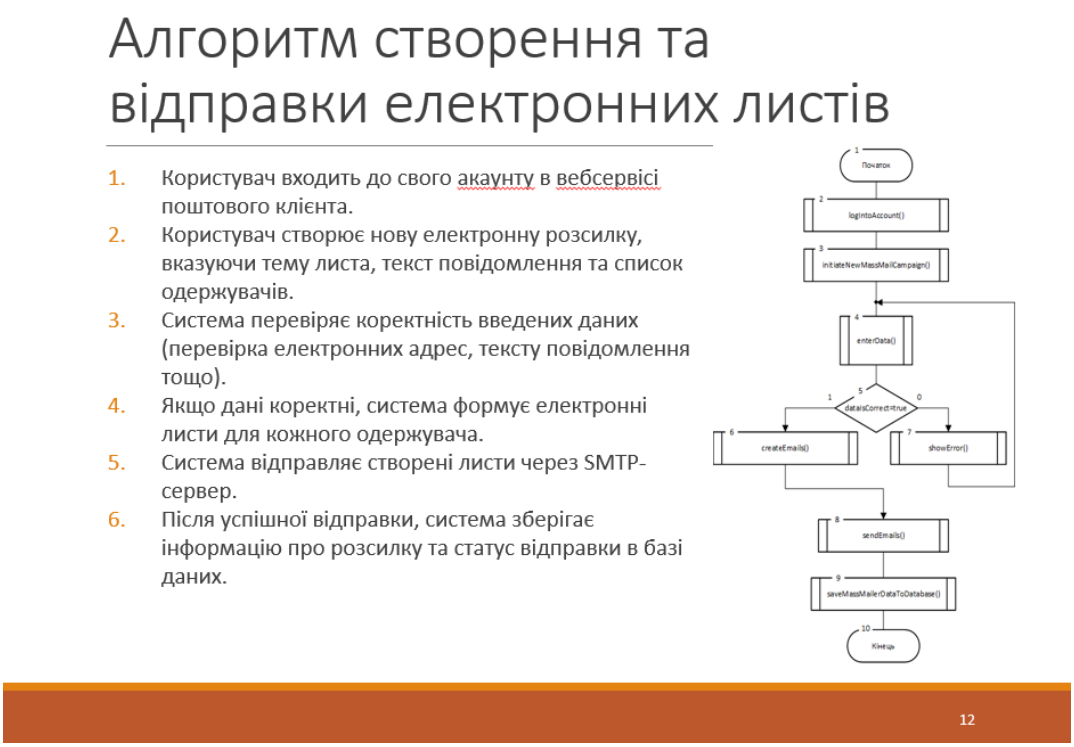

Рисунок Г.13 – Алгоритм створення та відправки електронних листів

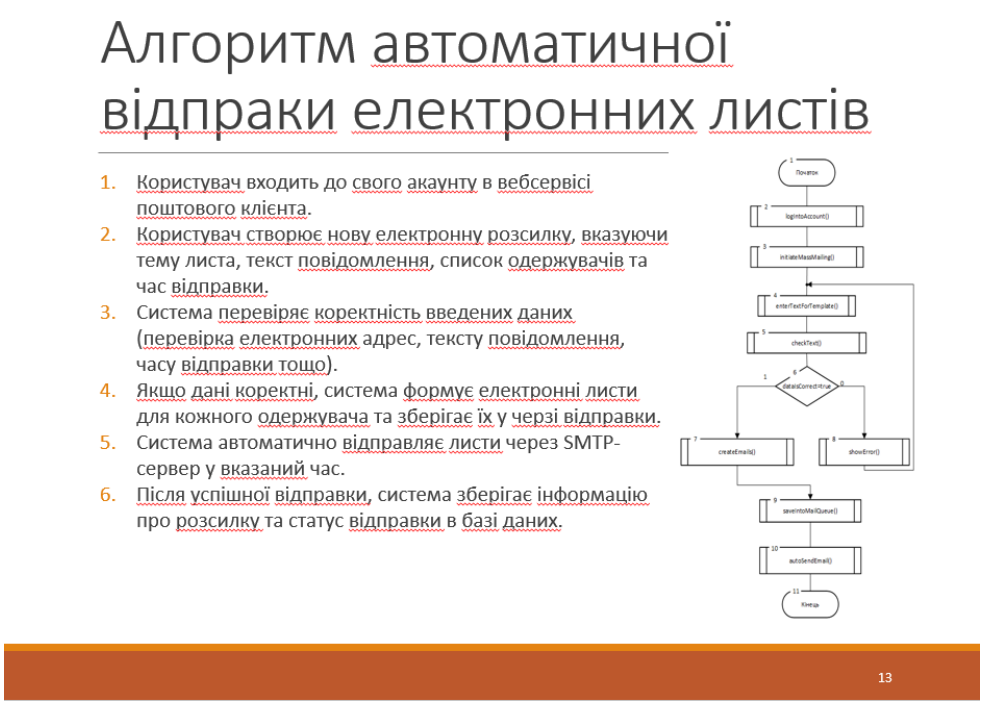

Рисунок Г.14 – Алгоритм автоматичної відправки електронних листів

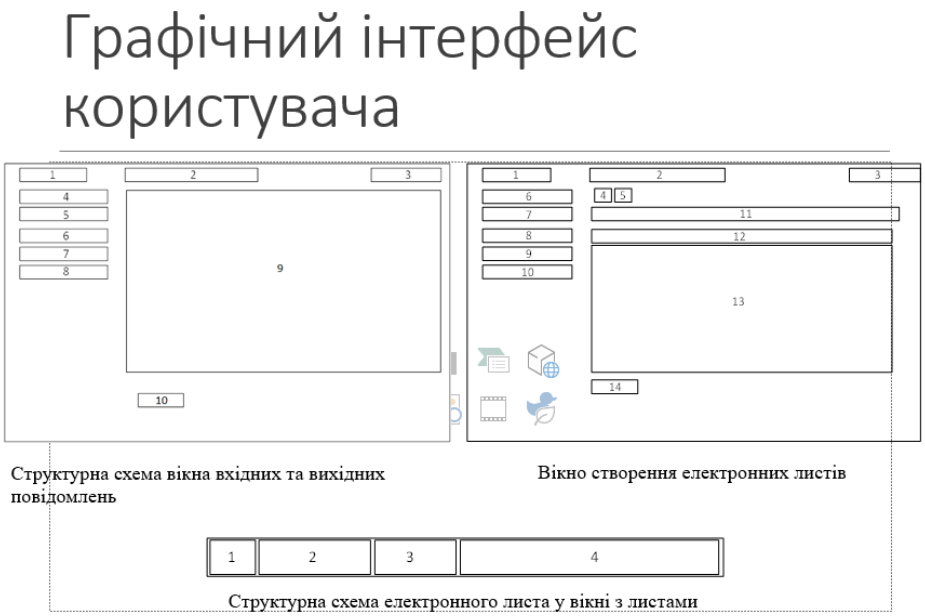

Рисунок Г.15 – Графічний інтерфейс користувача

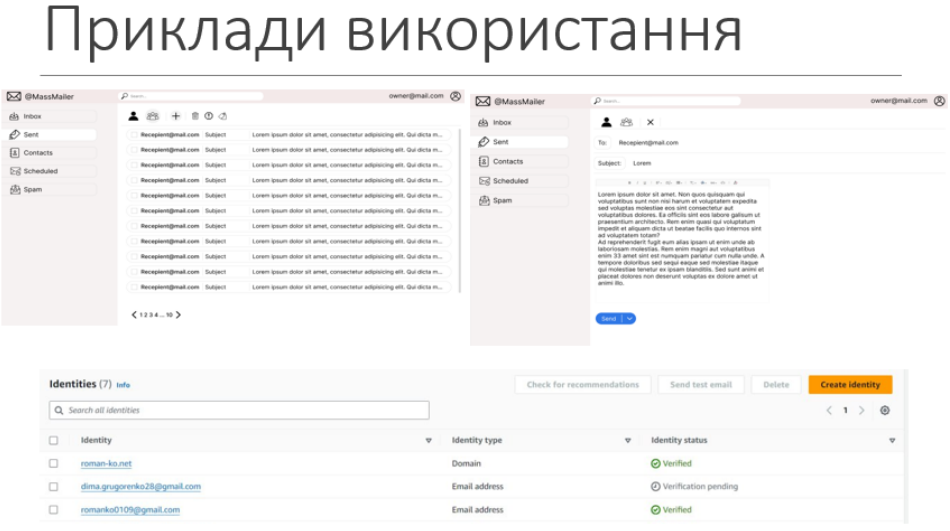

Рисунок Г.16 – Приклади використання

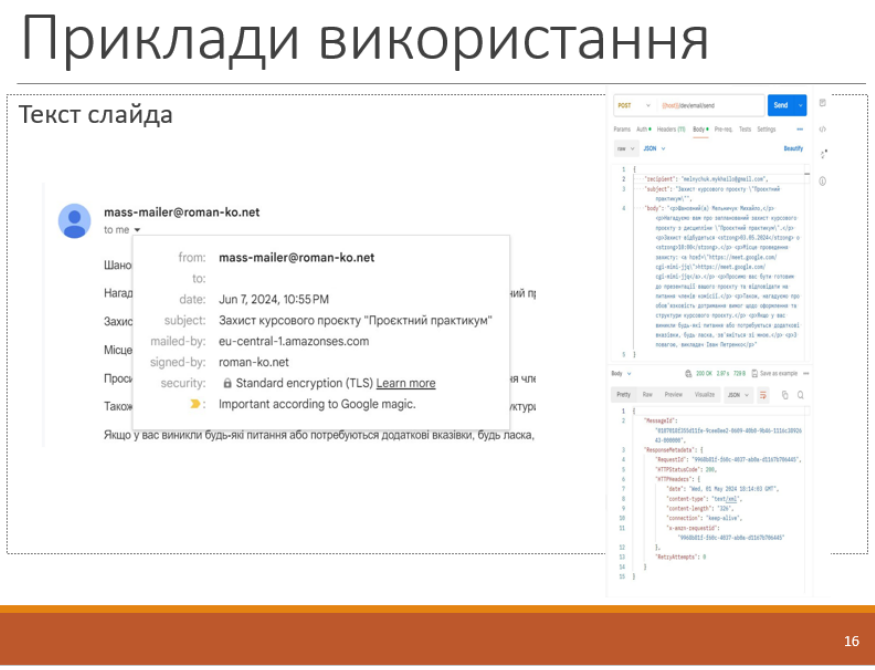

Рисунок Г.17 – Приклади використання

Рисунок Г.18 – Висновки

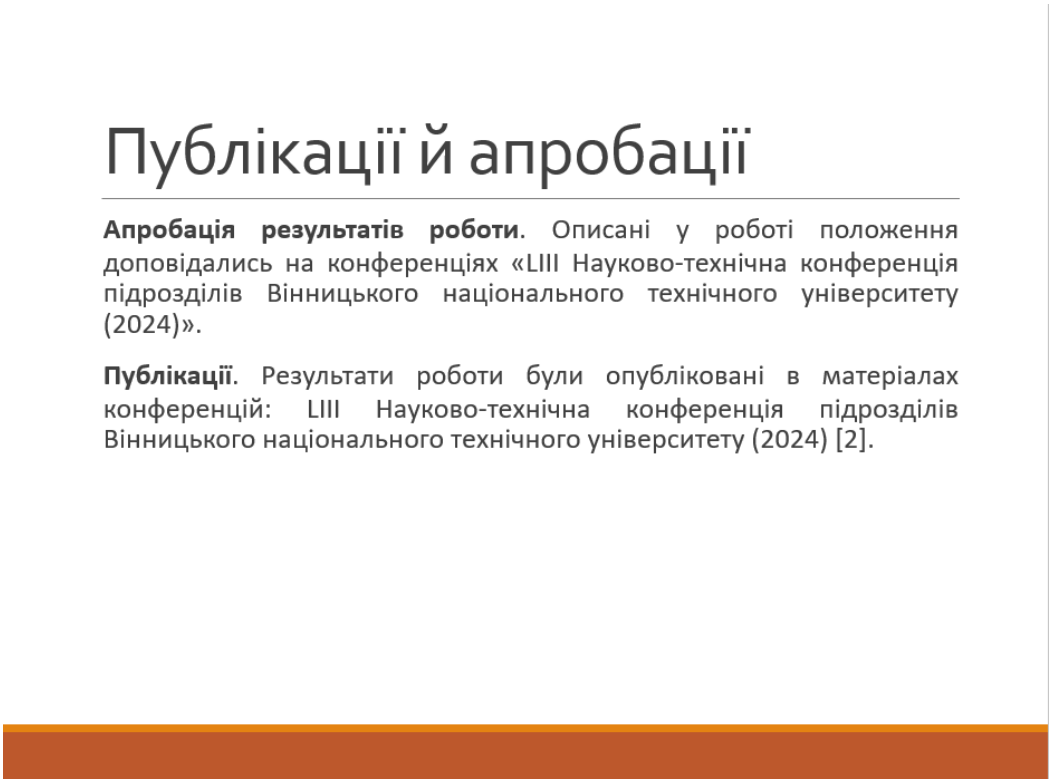

Рисунок Г.19 – Публікації й апробації