

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

бакалавр

(ступінь вищої освіти)

на тему ПРОГРАМНА РЕАЛІЗАЦІЯ ЗАСТОСУНКУ ПОШТОВОГО
КЛІЄНТУ З СОРТУВАННЯМ ВХІДНИХ ЛИСТІВ
SOFTWARE IMPLEMENTATION OF AN EMAIL CLIENT
APPLICATION SORTING THE INCOMING EMAILS

Виконав(ла): студент(ка) 4 курсу, групи КНТ-141

Спеціальності 121 Інженерія програмного

(код і найменування спеціальності)

забезпечення

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

КУЛИКОВ А.С.

(ПРИЗВИЩЕ та ініціали)

Керівник КОЦУР М.І.

(ПРИЗВИЩЕ та ініціали)

Рецензент ІЛ'ЯШЕНКО М.Б.

(ПРИЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
 (повне найменування закладу вищої освіти)

Факультет КНТ
 Кафедра програмних засобів
 Ступінь вищої освіти бакалавр
 Спеціальність 121 Інженерія програмного забезпечення
(код і найменування)
 Освітня програма (спеціалізація) Інженерія програмного забезпечення
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
С.О. СУББОТІН
 « » 2024 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА

КУЛИКОВА Артема Степановича
(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Програмна реалізація застосунку поштового клієнту з сортуванням вхідних листів. Software Implementation of an Email Client Application Sorting the Incoming Emails
 керівник проєкту (роботи) канд. техн. наук, доцент КОЦУР Михайло Ігорович,
(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)
 затверджені наказом закладу вищої освіти від “28” квітня 2025 року № 209

2. Строк подання студентом проєкту (роботи) 02 червня 2025 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Матеріали і методи. 3. Основні рішення щодо реалізації компонентів системи. 4. Експлуатація, тестування та експериментальне дослідження програми.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, кількість слайдів, плакатів)
Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРІЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	КОЦУР М.І., доцент		
Нормоконтроль	КАЛІНІНА М.В., асистент		

7. Дата видачі завдання « 14 » квітня 2025 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	1 тиждень	Розділ 1
3	Вибір мови програмування та інших технологій розробки.	2 тиждень	Розділ 2
4	Розробка структури програми.	2 тиждень	Розділ 3
5	Розробка програми.	3-4 тижні	Розділи 3, 4
6	Тестування та експериментальне дослідження програмного забезпечення.	5 тиждень	Розділ 4
7	Оформлення пояснювальної записки та документів до неї.	6 тиждень	Додатки
8	Нормоконтроль та рецензування.	7 тиждень	
9	Захист роботи.	8 тиждень	

Студент(ка)

_____ Артем КУЛИКОВ
(підпис) (Ім'я ПРІЗВИЩЕ)

Керівник проєкту (роботи)

_____ Михайло КОЦУР
(підпис) (Ім'я ПРІЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра:
102 с., 3 табл., 26 рис., 4 дод., 27 джерел.

ПРОГРАМНА РЕАЛІЗАЦІЯ, ВЕБЗАСТОСУНОК, ПРОЄКТУВАННЯ
ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ, ПОШТОВИЙ КЛІЄНТ, ЕЛЕКТРОННІ
ЛИСТИ, СОРТУВАННЯ ПОШТИ, PYTHON.

Об'єкт дослідження – процес програмної реалізації застосунку поштового клієнту з сортуванням вхідних листів.

Предмет дослідження – програмні засоби для реалізації застосунку поштового клієнту з сортуванням вхідних листів.

Метою роботи є програмна реалізація застосунку поштового клієнту з сортуванням вхідних листів.

Матеріали, методи та технічні засоби: мова програмування Python, надбудови Django та середовища розробки PyCharm Free Educational Licenses.

Результати. Розроблено вебзастосунку поштового клієнту з сортуванням вхідних листів.

Практична цінність роботи полягає у програмній реалізації поштового клієнту з сортуванням вхідних листів у вигляді вебзастосунку.

Висновки. Реалізовано вебзастосунок поштового клієнту з сортуванням вхідних листів.

Галузь використання – користувачі мережі та робітники компаній.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 102 pages, 3 tables, 26 figures, 4 appendixes, 27 sources.

SOFTWARE IMPLEMENTATION, WEB APPLICATION, SOFTWARE DEVELOPMENT, EMAIL CLIENT, EMAILS, MAIL SORTING, PYTHON.

The object of research is the process of software implementation of an email client application with sorting incoming emails.

The subject of the research is software tools for implementing an email client application with sorting incoming emails.

The purpose of this work is to programmatically implement an email client application with sorting incoming emails.

Materials, methods, and technical tools: Python programming language, Django add-ons, and PyCharm Free Educational Licenses development environments.

Results. Developed a web application for an email client with sorting incoming emails.

The practical value of the work lies in the software implementation of the mail client with sorting incoming emails in the form of a web application.

Conclusions. Implemented a web application for the mail client with sorting incoming emails.

Scope of use – network users and employees of companies.

ЗМІСТ

	С.
Перелік скорочень та умовних познач	8
Вступ	9
1 Аналіз предметної області	16
1.1 Предметна область електронної пошти	16
1.2 Програмні застосунки для роботи із електронною поштою	21
1.3 Порівняння існуючих аналогів	26
1.4 Постановка завдання	33
1.4.1 Функціональність системи	34
1.4.2 Вимоги до інтерфейсу	36
1.4.3 Вимоги до безпеки	36
1.4.4 Не функціональні вимоги до продукту	36
1.5 Висновки за розділом 1	37
2 Матеріали і методи	38
2.1 Вибір мови програмування	38
2.2 Вибір середовища розробки	48
2.3 Структурна схема розробки	54
2.4 Висновки за розділом 2	56
3 Основні рішення щодо реалізації компонентів системи	57
3.1 Проєктування моделей системи	57
3.2 Проєктування інтерфейсу	63
3.3 Сорткування листів	65
4 Експлуатація, тестування та експериментальне дослідження програми	68
4.1 Призначення й умови застосування програми	68
4.2 Вимоги до інтерфейсу	68
4.3 Умови виконання програми	69
4.4 Методика та результати тестування	69
Висновки	73
Перелік джерел посилання	74

Додаток А Технічне завдання	77
Додаток Б Опис програми	82
Додаток В Текст програми	89
Додаток Д Слайди презентації	96

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

API – Application Programming Interface;

GUI – Graphical User Interface – графічного інтерфейсу користувача;

IDE – Integrated Development Environment – середовище розробки;

IMAP – Internet Message Access Protocol;

k-NN – k-Nearest Neighbors – метод k-найближчих сусідів;

MVC – Model-View-Controller;

ORM – Object-Relational Mapping;

POP3 – Post Office Protocol;

UI – User Interface – користувальницький інтерфейс;

UML – Unified Modeling Language;

ОС – операційна система;

ПЗ – програмне забезпечення;

СУБД – система управління базами даних.

ВСТУП

Електронна пошта залишається одним із найнадійніших та найнеобхідніших інструментів спілкування як особисто, так і професійно. Незважаючи на появу різних платформ обміну миттєвими повідомленнями (месенджери), інструментів співпраці та соціальних мереж, електронна пошта все ще відіграє центральну роль у цифровій комунікації [1]. Однією з найважливіших функцій, яка сприяла довговічності та корисності електронної пошти, є поштові клієнти – програмне забезпечення (ПЗ), яке дозволяє користувачам керувати, читати, надсилати та впорядковувати електронні листи на своїх пристроях. Серед найцінніших можливостей сучасних поштових клієнтів – можливість сортувати вхідні електронні листи різними способами [1]:

- за відправником;
- датою;
- темою;
- розміром;
- за допомогою розширених фільтрів, які дозволяють відокремлювати приватні повідомлення від рекламних або визначати пріоритет повідомлень від певних відправників [1].

Популярність поштових клієнтських застосунків, що пропонують функції сортування, можна пояснити зростаючим обсягом цифрових комунікацій, якими доводиться керувати приватним особам і організаціям. Щодня пересічна людина отримує десятки, а іноді й сотні електронних листів, особливо тих, хто працює в середовищі, яке вимагає спілкування з кількома командами, партнерами чи клієнтами. Без ефективного управління електронною поштою підтримання порядку стає серйозною проблемою. Механізми сортування служать першим рівнем захисту від перевантаження електронної пошти, допомагаючи користувачам розставляти пріоритети щодо важливих повідомлень і зменшувати кількість відволікаючих факторів [1].

Клієнти електронної пошти використовують різні методи сортування, кожен з яких відповідає різним уподобанням користувачів та потребам організації. Основні параметри сортування, такі як упорядкування повідомлень за часом отримання, допомагають користувачам швидко переглядати останні повідомлення. Сортування за відправником або темою дозволяє користувачам більш ефективно відстежувати листування, особливо в довгих повідомленнях електронної пошти. Більш просунуті функції сортування включають в себе позначки, маркування і використання папок. Мітки можуть вказувати на характер повідомлення-будь то термінове, інформаційне, що вимагає вжиття заходів або рекламне. Фільтри можуть автоматично переміщувати електронні листи до певних папок або застосовувати теги на основі заздалегідь визначених правил [2].

У багатьох професійних контекстах сортування – це не просто зручність, це необхідність. Розглянемо, наприклад, менеджера з персоналу, який щодня отримує резюме, підтвердження про проходження співбесід і внутрішню переписку. Завдяки ефективній системі сортування вхідні заявки можуть автоматично направлятися в папку з написом «кандидати», запрошення на зустрічі – в папку, прив'язану до календаря, а електронні листи внутрішньої команди можуть бути позначені кольором і мати пріоритет. Подібним чином, у системі обслуговування клієнтів сортування дозволяє миттєво Сортувати запити клієнтів, розрізняючи скарги, запити підтримки та відгуки. Це може значно скоротити час реагування та підвищити задоволеність клієнтів [2].

Крім організаційних переваг, сортування також підвищує продуктивність. Можливість класифікувати та фільтрувати повідомлення зменшує час, витрачений на пошук конкретних повідомлень. Замість нескінченного перегляду єдиної поштової скриньки користувач може швидко перейти до категорії або шукати папку з чіткими позначками. З часом така економія часу збільшується, що сприяє кращому управлінню часом і зниженню когнітивного навантаження на користувача. Більше того, коли

клієнти електронної пошти інтегрують сортування з функціями пошуку та автоматизації, такими як фільтри на основі правил або пріоритетні поштові скриньки, керовані машинним навчанням, ефективність управління великими обсягами електронної пошти значно покращується [2].

Навіть звичайні користувачі отримують вигоду від сортування електронної пошти. Наприклад, студент може відокремити електронні листи викладачів від тих, що стосуються онлайн-покупок або платформ соціальних медіа. Батьки можуть сортувати повідомлення від шкіл, медичних установ і працювати окремо з інформаційними бюлетенями без спаму. Така гнучкість дозволяє людям адаптувати своє цифрове комунікаційне середовище до свого способу життя, гарантуючи, що жодне важливе повідомлення не буде пропущено в спам [2].

Популярні клієнти електронної пошти, такі як Microsoft Outlook, Mozilla Thunderbird, Apple Mail і навіть вебплатформи, такі як Gmail і Yahoo Mail, пропонують різні варіанти сортування. Outlook особливо зручний для корпоративних користувачів, оскільки забезпечує правила, категорії та інтеграцію з календарями та завданнями. Gmail, хоча і працює в Інтернеті, популяризував використання вкладок і автоматичну фільтрацію за допомогою міток і інтелектуальну категоризацію – соціальні мережі, оновлення, рекламні акції і т. д. Thunderbird, з відкритим кодом, забезпечує високий ступінь налаштування, що підходить користувачам, які хочуть більше контролювати обробку та відображення своїх електронних листів [3].

Незважаючи на наявність багатьох відомих поштових клієнтів, все ще існують вагомі підстави для розробки нових поштових застосунків. Однією з основних причин цього є мінливий характер комунікацій та очікування користувачів. Сучасні користувачі більш мобільні, піклуються про конфіденційність і орієнтовані на багатозадачність, ніж будь-коли раніше. Застарілі поштові клієнти, незважаючи на свою потужність, часто базуються на застарілих платформах або філософії дизайну, які погано адаптуються до мобільних або децентралізованих середовищ. Більше того, нові покоління

користувачів звикли до більш чистих та інтуїтивно зрозумілих інтерфейсів і очікують високої продуктивності в режимі реального часу та розумної інтеграції з іншими інструментами [3].

Ще одним важливим стимулом для нових розробок є необхідність поліпшення користувацького досвіду і персоналізації. Хоча багато існуючих клієнтів дозволяють сортувати, мало хто пропонує справді розумну автоматизацію, яка враховує поведінку користувачів та адаптується відповідно. Штучний інтелект має все більше можливостей для покращення сортування електронної пошти-наприклад, він автоматично розпізнає повторювані зарплати та впорядковує їх у фінансовій папці, або виявляє повідомлення, пов'язані із зустрічами, та пропонує події календаря. Персоналізовані подання поштових скриньок, що відображають пріоритети користувача, його робочий графік і звички роботи з електронною поштою, можуть революціонізувати те, як користувачі звикли взаємодіяти з електронною поштою [3].

Безпека та конфіденційність також набувають все більшого значення. Багато популярних поштових клієнтів та послуг належать великим корпораціям, які використовують дані користувачів для цільової реклами чи інших комерційних цілей. Зростає попит на поштові клієнти з відкритим кодом або незалежно розроблені поштові клієнти, які поважають конфіденційність користувачів, пропонують наскрізне шифрування та забезпечують прозорість обробки даних. Нове покоління поштових клієнтів може надати пріоритет цим принципам, пропонуючи користувачам надійну платформу для конфіденційного спілкування [3].

Функціональна сумісність та інтеграція також є ключовими факторами, що стимулюють нові розробки. Сучасні робочі процеси часто включають в себе цілий ряд інструментів: платформи обміну повідомленнями, такі як Slack або Microsoft Teams, інструменти управління проектами, такі як Trello або Asana, хмарні сервіси зберігання даних, такі як Dropbox або Google Drive, а також календарі, замітки або CRM-системи. Поштовий клієнт, який легко

інтегрується з цими послугами, може стати потужним центром підвищення продуктивності. Наприклад, нова програма електронної пошти може включати можливість створювати завдання на основі електронних листів, ініціювати відеодзвінки або зберігати вкладення в хмарі одним клацанням миші, зберігаючи при цьому чистий та інтуїтивно зрозумілий інтерфейс [4].

Поширення віддаленої роботи та гібридних робочих середовищ з початку 2020-х років лише підкреслило важливість надійного та ефективного електронного листування. Оскільки члени команди розподіляються по часових поясах, а організації віддаляються від централізованих офісів, асинхронні засоби комунікації, такі як електронна пошта, стають ще більш актуальними. Поштовий клієнт, здатний адаптуватися до цих змін – можливо, забезпечуючи інтелектуальний підсумок потоків, планування з урахуванням часових поясів або інтеграцію з платформами колективної роботи – був би особливо корисним на поточному та майбутньому робочому місці [4].

Розробка нових застосунків поштових клієнтів також важлива для стимулювання інновацій. Конкуренція в сфері програмного забезпечення сприяє поліпшенню дизайну, продуктивності і функціональності. Без свіжих перспектив і нових учасників ринок застоюється, а користувачі залишаються із застарілими або неоптимальними інструментами. Нові розробники, стартапи і навіть любителі можуть запропонувати свіжі ідеї, кидаючи виклик домінуванню відомих гравців і пропонуючи користувачам кращий вибір [4].

Розробка нового поштового клієнту, зокрема, з розширеними можливостями сортування, – це можливість переосмислити те, як ми взаємодіємо з цифровими повідомленнями. Це дозволяє створювати більш інтелектуальні, адаптивні та відповідні потребам сучасних користувачів інструменти. Такий розвиток може включати такі інновації, як контекстне сортування, в якому електронні листи впорядковуються не лише за статичними критеріями, такими як відправник або дата, а й за динамічним контекстом, таким як терміновість, відносини або поточні проєкти. Система, яка розуміє робочий процес користувача та допомагає йому відповідно

розставляти пріоритети, може значно підвищити концентрацію уваги та зменшити стрес [4].

Більше того, оскільки машинне навчання та обробка природної мови продовжують вдосконалюватися, наступне покоління поштових клієнтів зможе краще зрозуміти зміст повідомлень. Замість того, щоб просто визначати ключові слова, ці системи можуть визначати ключ, виявляти неявні запитання чи завдання та навіть пропонувати відповіді чи наступні кроки. Це може підвищити роль поштового клієнта від пасивного сховища до активного помічника, допомагаючи користувачам активно керувати своїми комунікаціями [4].

Таким чином, поштові клієнтські програми з розширеними можливостями сортування продовжують користуватися популярністю, оскільки вони вирішують постійну і зростаючу проблему управління цифровими комунікаціями. Вони допомагають користувачам ефективно організовувати, розставляти пріоритети та отримувати доступ до інформації, що має вирішальне значення як в особистому, так і в професійному житті. Розробка нових поштових клієнтів залишається важливим завданням, оскільки потреби користувачів змінюються, технології розвиваються, а на ринку все ще існують прогалини в більш розумних, безпечних, зручних у використанні та інтегрованих рішеннях. Скромний поштовий клієнт, аж ніяк не застарілий, знаходиться на стику традицій та інновацій, пропонуючи розробникам захоплюючу можливість переосмислити один з найважливіших інструментів цифрової епохи [5].

Для розробки нового та актуального поштового клієнту з сортуванням вхідних листів необхідно розв'язати низку робочих задач:

- дослідити предметну область електронного листування та використання застосунків поштових клієнтів;
- дослідити існуючі рішення на ринку, виявити їхні переваги та недоліки;

- на основі виявлених переваг та недоліків – сформулювати перелік функціональних вимог до нового – якіснішого рішення;
- спроектувати та розробити новий застосунок – поштовий клієнт з сортуванням листів;
- виконати тестування нової розробки;
- надати ґрунтовні висновки за виконаною роботою.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

Перший розділ пояснювальної записки присвячено вирішенню трьох робочих задач, що необхідні для розробки нового та актуального поштового клієнту з сортуванням вхідних листів, а саме:

- дослідити предметну область електронного листування та використання застосунків поштових клієнтів;
- дослідити існуючі рішення на ринку, виявити їхні переваги та недоліки;
- на основі виявлених переваг та недоліків – сформувати перелік функціональних вимог до нового – якіснішого рішення.

Вирішення цих робочих задач необхідно для подальшої роботи над застосунком.

1.1 Предметна область електронної пошти

Електронна пошта залишається однією з найбільш широко використовуваних форм цифрового спілкування, незважаючи на розвиток сучасних платформ обміну повідомленнями, таких як WhatsApp, Telegram, Slack, Microsoft Teams, Discord та інших. Існує кілька глибоких причин такої незмінної популярності. Це не просто питання звички – електронна пошта виконує певну універсальну роль, яку нові платформи, при всіх своїх перевагах, все ще не можуть повністю замінити [6].

Основна причина, чому електронна пошта все ще актуальна, полягає в тому, що вона універсальна. Вам не потрібно використовувати одну платформу або встановлювати ту саму програму, щоб надсилати або отримувати електронні листи. З будь-якої адреси електронної пошти можна зв'язатися з будь-яким іншим, незалежно від постачальника. Користувач можете використовувати Gmail для надсилання повідомлення комусь, хто використовує Outlook, Yahoo або навіть власний домен. Це зручніше, в

порівнянні з платформами обміну повідомленнями, де обом сторонам потрібно встановити один і той же застосунок і часто створити обліковий запис, перш ніж вони зможуть спілкуватися [6].

Така універсальна сумісність робить електронну пошту засобом комунікації за замовчуванням, особливо в ситуаціях, коли відправник і одержувач можуть погано знати один одного, наприклад, при ділових запитах, обслуговуванні клієнтів, прийомі на роботу або академічному листуванні [6].

Електронній пошті притаманні формальність і професіоналізм. Це стандартний спосіб подання заявок на роботу, ділових контрактів, рахунків-фактур, пропозицій та взаємодії з клієнтами. Месенджери є більш простими та оперативними, але їм не вистачає структурованого форматування, вкладень та відстежуваності, які надає електронна пошта. Коли користувач надсилає електронний лист, у ньому є рядок теми, привітання, основна частина та підпис. Такий формат додає ясності і професіоналізму [6].

Організації використовують електронну пошту для обміну документами: запрошення на збори, внутрішні пам'ятки, листування з клієнтами та угоди зберігаються і на них можна посылатися протягом довгого часу. Для таких взаємодій потрібен чіткий звіт, який служби миттєвого обміну повідомленнями не завжди надійно або формально зберігають [7].

Електронна пошта за своєю суттю асинхронна – користувач надсилає повідомлення, не чекаючи негайної відповіді. Це сильна сторона, а не слабкість. Це враховує часові пояси, години роботи та пропускну здатність мозку. Навпаки, засоби обміну миттєвими повідомленнями часто вимагають негайних відповідей, що призводить до стресу та непорозумінь. Що стосується електронної пошти, загальноприйнятою нормою є те, що відповіді можуть зайняти години або навіть дні, що робить її ідеальною для детального, нетермінового спілкування [7].

Для команд, клієнтів або партнерів, що працюють в різних часових поясах, електронна пошта залишається практичним способом спілкування, не перериваючи і не порушуючи синхронну доступність [7].

Електронна пошта відмінно справляється з вкладеннями різних типів: документами, електронними таблицями, PDF-файлами, зображеннями і т.д. хоча багато застосунків для чату підтримують загальний доступ до файлів, вони не завжди обробляють їх структурованим або доступним для пошуку способом. Електронна пошта також підтримує форматування розширеного тексту, підписи HTML, вбудовані посилання та заголовки – функції, які мають вирішальне значення для забезпечення ясності в офіційних повідомленнях [7].

Надсилення супровідного листа, рахунку-фактури або юридичного документа через додаток месенджер рідко є доречним. Електронна пошта все ще є найкращим інструментом для передачі та впорядкування структурованих документів [7].

Люди та організації покладаються на електронну пошту для довгострокового ведення записів. Більшість служб електронної пошти автоматично архівують надіслані та отримані повідомлення, а потужні інструменти пошуку дозволяють користувачам відновлювати старі листування через роки. Спробуйте знайти дворічне повідомлення в чаті програми обміну повідомленнями – це часто непрактично [7].

Ця історія з можливістю пошуку безцінна. Незалежно від того, чи шукає користувач рахунок-фактуру від постачальника, деталі з попередньої розмови чи документацію з юридичних причин, електронна пошта надає постійний архів для пошуку [8].

Для бізнесу електронна пошта є основним інструментом. Вона інтегрується з календарями, менеджерами завдань, CRM-системами і документообігом. Вона використовується для всього: від повідомлень відділу кадрів до автоматизації маркетингу і взаємодії з клієнтами. Жоден інший інструмент не має такої глибини інтеграції та універсальності [8].

Домени компаній часто розміщують власні сервери електронної пошти, що підкреслює індивідуальність бренду і професіоналізм. Ці користувацькі домени електронної пошти використовуються для надсилення офіційних повідомлень, рахунків-фактур, інформаційних бюлетенів та критичних

попереджень. Жоден застосунок для обміну повідомленнями не забезпечує такий рівень формальності або простежуваності [8].

Адреси електронної пошти все ще є ідентифікаційними даними за замовчуванням, які використовуються для реєстрації в більшості онлайн-сервісів. Соціальні медіа, хмарні сервіси, платформи електронної комерції, банківські програми використовують електронну адресу як прив'язку до цифрової ідентифікації. Коди підтвердження, скидання пароля, повідомлення про виставлення рахунків та оновлення умов надання послуг майже завжди надсилаються електронною поштою [8].

Завдяки цьому електронна пошта відіграє важливу роль у сучасному житті: це електронна поштова скринька, яка використовується для всього, від розсилки новин до отримання ПЗ. Такий рівень інтеграції в повсякденну діяльність означає, що люди не можуть просто відмовитися від електронної пошти, навіть якщо вони віддають перевагу додаткам для обміну повідомленнями для неформального спілкування [8].

Електронна пошта часто є юридичним засобом зв'язку. Суди та регулятори визнають електронні листи документальним підтвердженням угод, інструкцій та транзакцій. Підприємства та установи зобов'язані зберігати записи електронної пошти для проведення перевірок, дотримання вимог законодавства та вирішення спорів [8].

Застосунки для обміну повідомленнями, навпаки, не стандартизовані однаково, і їх використання в офіційних повідомленнях може спричинити проблеми з дотриманням законодавства. У таких секторах, як фінанси, охорона здоров'я, право та державне управління, електронна пошта не підлягає обговоренню [8].

Люди зазвичай використовують програми обміну повідомленнями для особистого спілкування, а електронну пошту – для роботи та виконання офіційних завдань. Цей поділ важливий з психологічної та функціональної точки зору. Це дозволяє людям перемикати контексти: коли користувач відкриває програму електронної пошти, він переходить в робочий режим або

режим адміністратора, тоді як WhatsApp або Telegram часто використовуються для спілкування з друзями та родиною [8].

Стирання кордону між ними може призвести до стресу та дезорганізації. Ведення професійного листування по електронній пошті допомагає створити кордон і зберегти концентрацію уваги [8].

Доступ до електронної пошти можна отримати з будь — якого пристрою, платформи чи інтерфейсу: веб застосунку, мобільного застосунку, desktop клієнтів або навіть інструментів командного рядка. Вона не прив'язана до якоїсь однієї екосистеми. Така відкритість рідко зустрічається серед сучасних інструментів. Якщо Gmail зникне, ви все одно зможете налаштувати інший клієнт за допомогою IMAP або перенести свою поштову скриньку до іншого постачальника. У програмах обміну повідомленнями користувач часто залежить від власних систем [9].

Більше того, поштові послуги часто надають гігабайти пам'яті, і користувачі можуть легко впорядковувати та створювати резервні копії своєї пошти. Це робить електронну пошту не тільки засобом спілкування, але і сховищем важливих життєвих даних: квитанцій, записів, підтверджень і багато чого іншого [9].

Технології електронної пошти більше 50 років. Протягом десятиліть вона довела свою стабільність, надійність і адаптивність. Платформи обміну повідомленнями приходять і йдуть (MSN Messenger, ICQ, Google Talk або BBM). Багато з них були закриті або замінені. Електронна пошта, навпаки, стала лише більш функціональною [9].

Така надійність зміцнює довіру. Надсилаючи важливе повідомлення електронною поштою, користувач знає, що воно потрапить у поштову скриньку одержувача і залишиться там, поки його не прочитають. Програми обміну повідомленнями, залежно від налаштувань push-розсилки або підключення, можуть не завжди доставляти або повідомляти про це [9].

Таким чином, електронна пошта залишається популярною, оскільки вона виконує функції, які не виконує жоден інший засіб комунікації. Це:

- універсальність для всіх платформ і сервісів;
- формальність – підходить для професійного використання;
- асинхронність – що скорочує часові витрати;
- надійність і хороша документованість;
- інтегрованість в ділові та юридичні системи;
- використання для ідентифікації та реєстрації;
- налаштованість і незалежність від власних платформ.

Незважаючи на те, що застосунки для обміну повідомленнями стають швидшими та повсякденнішими, електронна пошта все ще залишається цифровою основою для серйозного, відстежуваного та структурованого спілкування. Вона глибоко вкорінена в архітектурі Інтернету, комерції, управління та професійного життя – і саме тому все ще користується успіхом через десятиліття після появи [9].

1.2 Програмні застосунки для роботи із електронною поштою

З початку існування Інтернету поштові клієнти відігравали основоположну роль у формуванні способів спілкування людей у цифровому форматі. До появи вебслужб електронної пошти, таких як Gmail або Outlook.com, більша частина електронного листування здійснювалася за допомогою автономних застосунків, встановлених безпосередньо на персональних комп'ютерах (ПК). Ці програми підключалися до поштових серверів, завантажували вхідну пошту та дозволяли користувачам читати та складати повідомлення в автономному режимі. Такий підхід давав значні переваги в той час, коли інтернет-з'єднання були більш повільними, менш надійними і не завжди доступними [9].

Одним з перших і найбільш широко використовуваних поштових клієнтів був Eudora, запущений наприкінці 1980-х. Він завоював популярність завдяки своїй простоті і зручності використання. Незабаром за ним пішли інші, включаючи Pegasus Mail, Lotus Notes та Microsoft Outlook. Ці програми

підключалися до поштових серверів за такими протоколами, як Post Office Protocol (POP3) і більш пізній Internet Message Access Protocol (IMAP). POP3 завантажував електронні листи та часто видаляв їх із сервера, що було ідеальним рішенням для користувачів з обмеженим обсягом пам'яті на сервері або повільним доступом до Інтернету. З іншого боку, протокол IMAP дозволяв повідомленням залишатися на сервері та синхронізуватися між пристроями, відкриваючи шлях для доступу з декількох пристроїв у майбутньому [10].

На той час найпривабливішими функціями поштового клієнта були автономний доступ та інтеграція з настільними комп'ютерами. Користувачі могли читати, сортувати та писати електронні листи, навіть якщо вони були відключені від інтернету. При повторному підключенні клієнт відправляв повідомлення з черги і завантажував нові. Це було вкрай важливо в епоху, коли плата за користування Інтернетом стягувалася щохвилини або по комутованих з'єднаннях. Автономна функціональність заощадила гроші та зробила електронну пошту більш доступною для широкої аудиторії [10].

Організація та сортування вхідних електронних листів на ранньому етапі стала основною функцією. У міру зростання поштових скриньок необхідність ефективного управління ними ставала все більш актуальною. Поштові клієнти впровадили системи папок, фільтри та правила для автоматизації сортування повідомлень. Наприклад, користувач може створити правило для автоматичного переміщення всіх електронних листів від певного відправника до папки. Це дозволило покращити організацію та зменшити когнітивне навантаження під час навігації по все більш переповненій поштовій скриньці. Також були впроваджені такі функції, як позначення тегів, мітки тегів та повнотекстовий пошук, які полегшують керування великими обсягами електронної пошти [10].

Інтеграція календарів, довідників контактів і списків завдань також сприяла зростанню популярності поштових клієнтів, особливо в бізнес-середовищі. Microsoft Outlook став домінуючим у корпоративному середовищі завдяки своїй безперебійній інтеграції з Microsoft Exchange, що дозволяє не

тільки надсилати електронні листи, але й планувати зустрічі, керувати контактами та налаштовувати нагадування – все це з однієї програми. Цей набір інструментів для підвищення продуктивності дозволив створити потужний і зручний центр документообігу, який було важко замінити [10].

Безпека була ще однією причиною того, що клієнти електронної пошти залишалися популярними. У корпоративних середовищах організації могли контролювати та налаштовувати поштові клієнти відповідно до конкретних вимог безпеки, таких як зберігання зашифрованих повідомлень, захищені протоколи зв'язку та перевірка цифрового підпису. Навпаки, служби вебпошти спочатку були більш вразливими і пропонували менше можливостей для налаштування [10].

Незважаючи на поліпшення інтернет-з'єднань і підвищення надійності поштових сервісів, багато користувачів продовжували покладатися на desktop поштові клієнти. Причиною такої довговічності є переваги, які клієнти продовжували пропонувати, особливо досвідченим користувачам, підприємствам та приватним особам з особливими потребами в продуктивності. Однією з таких переваг є продуктивність. Desktop клієнти, як правило, забезпечують швидшу навігацію та пошук у великих поштових скриньках, оскільки дані зберігаються локально. Користувачам не потрібно чекати завантаження сторінок або отримання повідомлень із сервера [10].

Можливість керувати кількома обліковими записами в межах єдиного інтерфейсу – це ще одна сфера, в якій клієнти електронної пошти традиційно досягають успіху. Для професіоналів, які керують декількома адресами електронної пошти: особистими, діловими, для конкретних проєктів і т.д., – наявність єдиної поштової скриньки і можливість налаштовувати параметри для кожного облікового запису окремо є важливою перевагою. Хоча вебпошта поступово вдосконалюється в цьому відношенні, більшість клієнтів все ще пропонують простіший і надійніший спосіб управління кількома обліковими записами [10].

Поштові клієти також надають вдосконалені інструменти для форматування та редагування повідомлень. Вони підтримують текстові редактори, вбудовані зображення та просте управління вкладеннями. Такі функції, як перевірка орфографії, граматичні рекомендації та шаблони повідомлень, часто є більш складними або налаштованими в desktop застосунках. Це особливо важливо для професіоналів і підприємств, де грамотно організована комунікація впливає на бренд і довіру до нього [10].

Інтеграція з іншими застосунками – ще одна перевага поштових клієнтів. Наприклад, настільні клієнти можна налаштувати для роботи з програмами для редагування документів, CRM-системами або інструментами управління завданнями. Користувачі можуть конвертувати електронний лист у завдання, подію календаря або контакт кількома клацаннями миші. Ці інтеграції допомагають оптимізувати робочі процеси та зменшити перемикання контексту, що часто є проблемою для вебплатформ без складних плагінів або сторонніх служб [11].

Ще однією перевагою є можливість настройки. Багато клієнтів електронної пошти пропонують широкий спектр налаштувань для теми інтерфейсу, макетів, налаштувань ярликів та сповіщень. Користувачі, які проводять години на день, керуючи електронною поштою, часто цінують можливість адаптувати додаток до свого робочого процесу. Сервіси вебпошти, як правило, мають більш обмежені можливості налаштування, часто пов'язані з вибором дизайну, розрахованого на широку аудиторію [10].

Незважаючи на зростання популярності мобільних пристроїв і повсюдне поширення вебплатформ, поштові клієнти еволюціонували, щоб залишатися актуальними. Зараз багато сучасних клієнтів синхронізуються з хмарними службами, підтримують автентифікацію OAuth та пропонують наскрізне шифрування. Деякі з них, такі як Mozilla Thunderbird, використовують модель з відкритим кодом, що дозволяє спільноті додавати нові функції та підтримувати платформу. Інші, такі як Apple Mail або Outlook,

продовжують тісно інтегруватися у свої екосистеми, забезпечуючи безперебійну роботу на різних пристроях та сервісах [11].

Ще одним фактором, що сприяє збереженню актуальності поштових клієнтів, є зростаюче занепокоєння з приводу конфіденційності та контролю даних. Користувачі стають все більш обережними щодо того, як технологічні компанії обробляють свої дані, особливо при використанні служб електронної пошти, які монетизують поведінку користувачів за допомогою реклами. За допомогою настільних поштових клієнтів, особливо налаштованих на використання користувацьких або приватних поштових серверів, користувачі можуть краще контролювати свої повідомлення. Вони можуть вибирати стандарти шифрування, зберігати локальні копії та уникати небажаного відстеження або сканування свого вмісту для націлювання на рекламу [11].

Окрім ділового та професійного використання, існує сегмент технічно підкованих користувачів, які цінують клієнтів електронної пошти з відкритим кодом, які поважають конфіденційність, для особистого користування. Такі інструменти, як Thunderbird або Mailspring, привабливі для користувачів, яким потрібне середовище без реклами, без відволікань та з можливістю прихованого налаштування параметрів. Ці користувачі часто віддають перевагу настільним клієнтам саме тому, що вони не прив'язані до браузера і пропонують більш детальний контроль над поведінкою та зовнішнім виглядом електронної пошти [11].

Незважаючи на все це, клієнти електронної пошти стикаються зі зростаючою конкуренцією з боку інструментів співпраці, таких як Slack, Microsoft Teams та Google Chat. Ці платформи пропонують обмін повідомленнями в режимі реального часу, обмін файлами та інтеграцію з іншими інструментами, що зменшує залежність електронної пошти від внутрішнього спілкування. Однак електронна пошта залишається незамінною для зовнішнього спілкування, офіційного листування та асинхронного обміну повідомленнями. Поштові клієнти все ще служать інтерфейсом, за допомогою

якого користувачі отримують доступ до таких комунікацій та керують ними, часто ефективніше, ніж дозволяють платформи вебпошти [11].

Навіть молоді користувачі, які, можливо, звикли до інструментів браузера та мобільних застосункв, починають усвідомлювати цінність наявності потужного клієнта електронної пошти для настільних комп'ютерів, особливо коли вони починають працювати і від них очікують більш структурованого та професійного спілкування. Інструменти, що підтримують ярлики, фільтри, запрошення до календаря та управління листуванням, стають все більш необхідними у міру зростання обов'язків. Сучасні поштові клієнти адаптуються до цих потреб, пропонуючи більш чіткі інтерфейси, покращену інтеграцію та кросплатформну синхронізацію [11].

В решті решт, популярність поштових клієнтів, як в минулому, так і сьогодні, пов'язана з поєднанням продуктивності, комплексності, кастомізації та контролю. З часом їх роль змінилася – від простих засобів читання повідомлень до повноцінних комунікаційних центрів, але їх основні переваги залишаються актуальними. Незалежно від того, чи використовуються вони професіоналами, які керують кількома обліковими записами, компаніями, які вимагають суворого дотримання законодавства, або приватними особами, які піклуються про конфіденційність, поштові клієнти все ще пропонують унікальні переваги. До тих пір, поки електронна пошта залишається центральною формою цифрової комунікації, поштові клієнти будуть займати своє місце в цифровому інструментарії користувачів по всьому світу [11].

1.3 Порівняння існуючих аналогів

Найбільш популярні поштові клієнти, це desktop застосунки Microsoft Outlook [12]-[13] та Mozilla Thunderbird [14]-[15], й вебзастосунок Gmail [16]-[17]. Саме тому, на їх аналізі та порівнянні зосередимо увагу.

Microsoft Outlook – це персональний інформаційний менеджер, розроблений корпорацією Майкрософт, який широко використовується в

якості поштового клієнта, але також включає функції для управління календарем, контактами, завданнями, нотатками і записами в журналі. Він доступний як desktop застосунок (як частина Microsoft Office або Microsoft 365), вебзастосунок (Outlook в Інтернеті) та мобільний застосунок (Outlook для iOS та Android).

Microsoft Outlook – це поштовий клієнт і персональний органайзер від Microsoft. Він дозволяє надсилати та отримувати електронні листи, керувати календарем, контактами, завданнями та нотатками. Ви можете використовувати його на своєму комп'ютері, у веббраузері або на мобільному пристрої [12].

Outlook працює, підключаючись до поштових серверів, таких як Microsoft Exchange, Gmail та інших. Для надсилання та отримання повідомлень він використовує Інтернет-протоколи (наприклад, IMAP, POP3 та SMTP). Під час підключення програма синхронізує ваші електронні листи, події календаря та контакти, щоб ви могли отримати доступ до них з будь-якого місця. Це також дозволяє організувати свою поштову скриньку, планувати зустрічі та встановлювати нагадування – все це в одному місці (рис.1.1) [13].

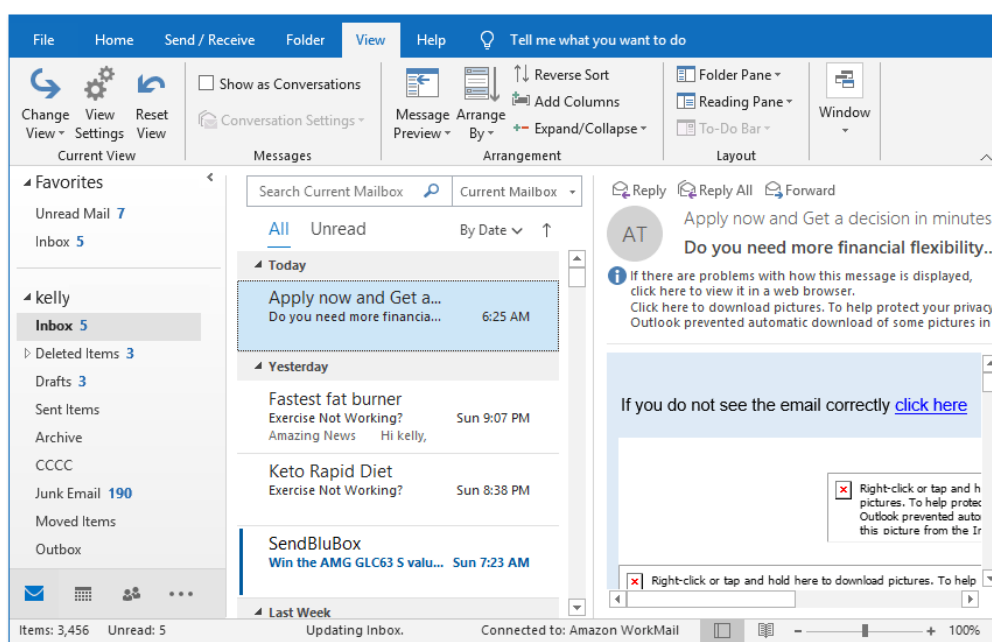


Рисунок 1.1 – Приклад інтерфейсу застосунку Microsoft Outlook [13]

Mozilla Thunderbird – це безкоштовний поштовий клієнт з відкритим кодом, розроблений Mozilla. Він дозволяє надсилати, отримувати та впорядковувати електронні листи, а також керувати контактами, календарями та стрічками новин [14].

Thunderbird підключається до сервера постачальника послуг електронної пошти, використовуючи стандартні протоколи, такі як IMAP (для синхронізації повідомлень), POP3 (для завантаження повідомлень) та SMTP (для надсилання повідомлень). Програма зберігає електронну пошту локально на вашому комп'ютері і може працювати з кількома обліковими записами. Thunderbird також підтримує застосунки, які дозволяють налаштувати його функції та зовнішній вигляд (рис.1.2) [15].

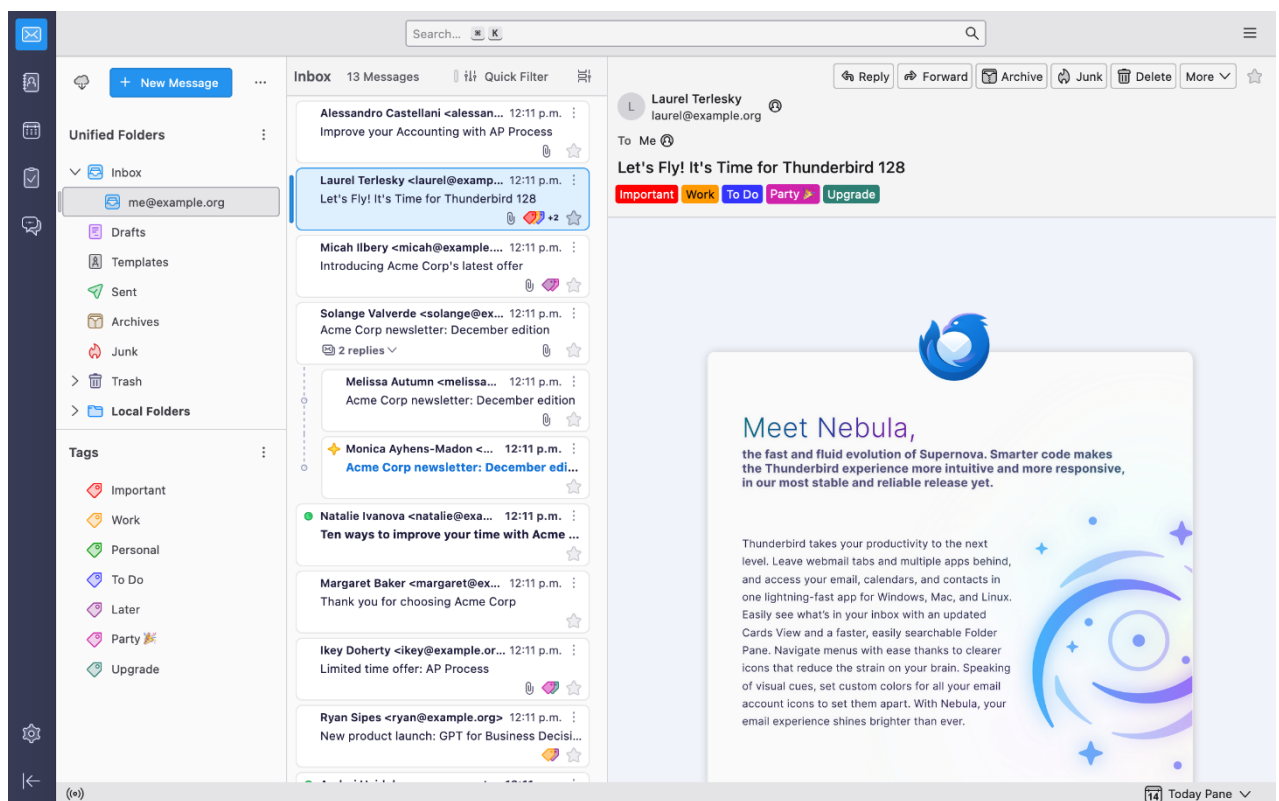


Рисунок 1.2 – Приклад інтерфейсу застосунку Mozilla Thunderbird [15]

Gmail – аорштовий клієнт в форматі вебзастосунку, розроблений Google. Він дозволяє користувачам надсилати, отримувати та впорядковувати електронні листи через браузер або мобільний застосунок. Gmail

підключається до серверів Google і зберігає всі повідомлення в хмарі, тому користувачі можуть отримувати доступ до своїх поштових скриньок з будь-якого пристрою, підключеного до Інтернету [16].

Gmail популярний завдяки своїй безкоштовності, простоті у використанні та потужним функціям, таким як фільтрація спаму, швидкий пошук, великий обсяг пам'яті, розумні відповіді та інтеграція з іншими службами Google, такими як Диск, Календар та Meet. Він також підтримує IMAP та POP3, тому користувачі можуть використовувати Gmail з іншими програмами електронної пошти, якщо вони цього захочуть (рис.1.3) [17].

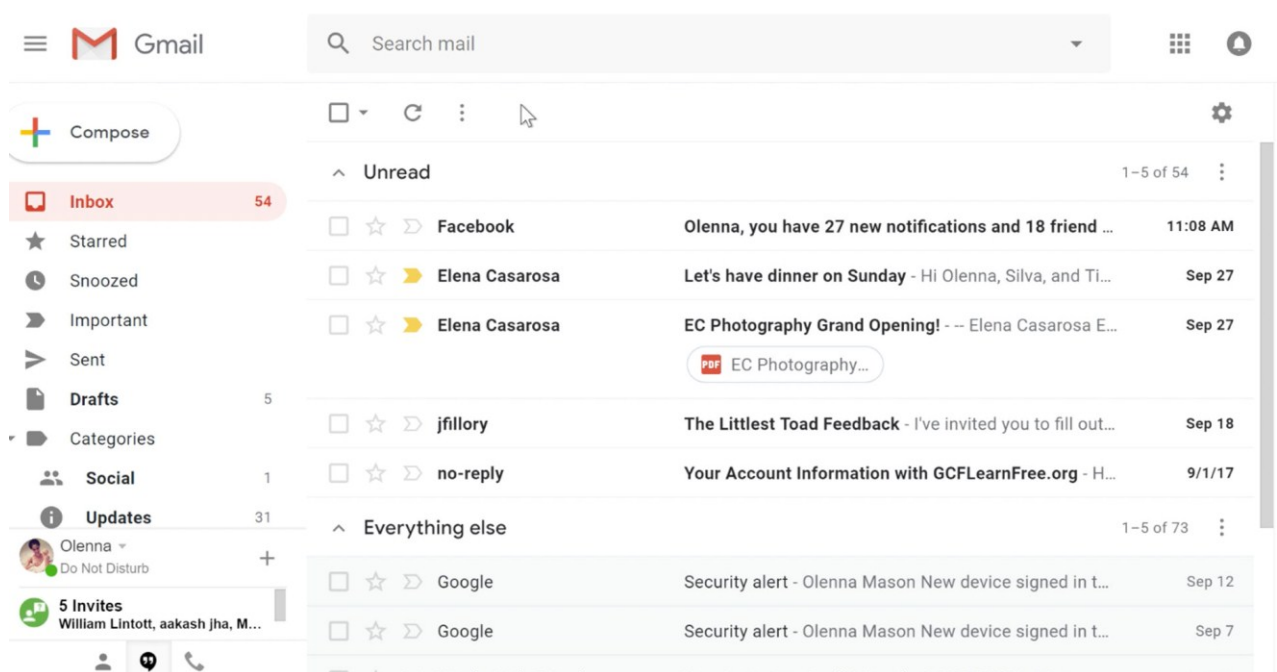


Рисунок 1.3 – Приклад інтерфейсу вебзастосунку Gmail [17]

Для аналізу переваг та недоліків існуючих застосунків слід виконати їх порівняння. Порівняння в табличному вигляді представлено в табл. 1.1.

Таблиця 1.1 – Порівняння існуючих аналогів

Критерії порівняння	Microsoft Outlook	Mozilla Thunderbird	Gmail
1	2	3	4
UI	Складний користувальницький інтерфейс, особливо для нових користувачів; перевантажений	Застарілий, менш відшліфований; не дуже інтуїтивно зрозумілий	Проста, але обмежена настройка
Продуктивність	Може працювати повільно через великі поштові скриньки "Вхідні" або спільні поштові скриньки	Лаги з величезними поштовими архівами; затримки при запуску	В цілому швидка, але з обмеженнями, пов'язаними з браузером
Підтримка платформи	Чудово працює в Windows; на Mac та мобільних пристроях можливості обмежені	В основному це десктопна версія; офіційної мобільної версії немає	В Інтернеті; непослідовна робота на мобільних пристроях
Можливість налаштування	Деякі опції, але обмежена гнучкість	Легко настраюється інтерфейс (розширення, теми)	Дуже обмежена настройка
Сортування та фільтрація	Потужний механізм правил, але втомлива настройка	Хороші фільтри, потрібна ручна настройка	Автоматичні категорії часто класифікуються неправильно
Використання в автономному режимі	Хороша Автономна підтримка	Повноцінне Автономне використання	Обмежений автономний режим у Chrome
Обробка небажаної пошти	Іноді трапляються помилкові спрацьовування	В значній мірі залежить від ручного налаштування	Загалом ефективний, але не дозволяє вимкнути фільтри Google
Функція пошуку	Потужний, але іноді неінтуїтивний інструментарій	Слабкий пошук у порівнянні з хмарними сервісами	Швидкий і точний

Продовження таблиці 1.1

1	2	3	4
Підтримка та оновлення	Часті оновлення; хороша підтримка підприємств	Оновлення, створені спільнотою; неузгоджений цикл випуску	Регулярні оновлення, але проблеми користувачів часто ігноруються
Доступність	Розумна підтримка; можливості для поліпшення	Мінімальні можливості доступності	Базові можливості, а не розширені

Критеріями для порівняння будуть слугувати:

- користувацький інтерфейс (UI) – це перше, з чим взаємодіють користувачі. Захаращений або неінтуїтивний інтерфейс може зробити недоступними навіть потужні функції. Багато користувачів відмовляються від поштових клієнтів просто тому, що вони здаються занадто складними або візуально пригнічують. Сучасні користувачі, особливо молоді та користувачі мобільних пристроїв, очікують чистого, мінімалістичного та адаптивного дизайну. Хороший UI сприяє більш швидкому впровадженню, скорочує час на вивчення і прискорює виконання завдань;

- продуктивність – навіть якщо застосунок досить функціональний, низька продуктивність (повільне завантаження, затримки, збої) погіршує роботу користувачів, особливо при роботі з великими поштовими скриньками або корпоративною електронною поштою. Продуктивність безпосередньо впливає на продуктивність. Клієнт, який ефективно обробляє тисячі електронних листів, забезпечує безперебійний робочий процес, особливо для досвідчених користувачів у бізнесі чи наукових колах;

- підтримка платформи – користувачі все частіше перемикаються між пристроями: настільними комп'ютерами на роботі, планшетами в дорозі, телефонами вдома. Електронна пошта повинна бути доступною та узгодженою для всіх них. Невідповідність взаємодії на різних платформах

зменшує довіру та зручність використання. Єдина підтримка забезпечує користувачам доступ до функцій, макетів і налаштувань в будь-якому місці;

- можливість налаштування – різні користувачі мають різні робочі процеси. Одні хочуть автоматизації, інші - кольорових написів, а треті — мінімалістичного оформлення. Кастомізація дозволяє користувачам формувати додаток відповідно до своїх звичок. Воно також підтримує різні можливості доступу і настройки робочого процесу, підвищуючи задоволеність і знижуючи когнітивне навантаження;

- сортування та фільтрація – перевантаження поштових скриньок – одна з найбільш поширених проблем користувачів. Сортування і фільтрація необхідні для підтримки порядку, особливо при великому обсязі електронної пошти. Автоматична інтелектуальна фільтрація допомагає користувачам зосередитися на важливих повідомленнях і скорочує час, витрачений на обробку не відносяться до справи. Це підвищує продуктивність і ясність розуму;

- використання в автономному режимі – користувачам необхідно отримувати доступ до електронних листів, читати їх і навіть складати чернетки без підключення до Інтернету — під час поїздок, надзвичайних ситуацій або в регіонах з обмеженим зв'язком. Автономний доступ забезпечує безперервність роботи і пошуку інформації. Надійний автономний режим особливо важливий для професіоналів, журналістів та студентів;

- обробка небажаної пошти – спам неминучий. Неякісні спам-фільтри можуть пропускати небажану пошту в поштову скриньку або помилково приховувати важливі електронні листи. Ефективна обробка небажаної пошти захищає користувачів від фішингу та відволікання, а також зберігає довіру до системи електронної пошти. Воно також повинно бути учнем і прозорим для користувачів, щоб уникнути помилкових спрацьовувань;

- функція пошуку – швидкий пошук конкретних електронних листів важливий, особливо для постійних користувачів або для пошуку юридичних осіб. Пошук - одна з найбільш часто використовуваних функцій. Розширений

пошук з контекстним ранжуванням (наприклад, за найбільш актуальним або останнім) економить час і зменшує кількість розчарувань. Слабкий пошук знижує продуктивність при роботі з великими поштовими скриньками;

- підтримка та оновлення – клієнти електронної пошти потребують регулярних оновлень для забезпечення безпеки, сумісності та покращення функцій. Погана підтримка призводить до усунення помилок та загроз безпеці. Активне обслуговування свідчить про надійність;

- доступність – мільйони користувачів мають проблеми із зором, мобільністю або когнітивними порушеннями. Багато поштових клієнтів погано оптимізовані для цих цілей. Інклюзивний дизайн гарантує, що кожен може спілкуватися цифровим способом — це не тільки етично, але й юридично необхідно в багатьох контекстах (наприклад, стандарти WCAG, відповідність ada). Доступність розширює коло користувачів і підвищує зручність використання для всіх.

Ці функції не є довільними; вони відображають основні принципи розробки сучасного програмного забезпечення: зручність використання, безпеку, продуктивність, адаптивність і довіру. Новий поштовий клієнт, який продумано враховує ці аспекти, не тільки заповнить прогалини, залишені Outlook, Thunderbird та Gmail, але й має зробити цифрове спілкування більш ефективним, зручним та приємним для всіх.

1.4 Постановка завдання

Важливо розробити новий вебзастосунок поштовий клієнт з сортуванням листів, оскільки люди отримують занадто багато електронної пошти, а важливі повідомлення часто втрачаються. Сучасний застосунок має автоматично впорядковувати електронні листи за темами, відправниками чи пріоритетами, заощаджуючи час користувачів та зменшуючи стрес. Багато сучасних поштових клієнтів перевантажені, повільні або недостатньо гнучкі. Нове рішення може запропонувати більш зрозумілий дизайн, більш високу

продуктивність та інтелектуальні функції, такі як теги, фільтри та сортування, які допоможуть користувачам зосередитися на найважливішому.

Метою роботи є програмна реалізація застосунку поштового клієнту з сортуванням вхідних листів.

1.4.1 Функціональність системи

Розподілимо функціональні вимоги [18]-[19] до застосунку за групами функціональних можливостей використання:

а) управління електронною поштою:

- 1) надсилання та отримання електронних листів через протокол IMAP / SMTP;
- 2) підтримка декількох облікових записів електронної пошти (наприклад, Gmail, Outlook, Yahoo, користувацьких доменів);
- 3) перегляд і читання повідомлень електронної пошти в режимі потокової передачі;
- 4) складання, відповідь, пересилання та видалення електронних листів;
- 5) позначення листів як прочитані / непрочитані, позначені зірочкою або важливі;
- 6) додавання файлів та перегляд вкладення;

б) сортування та систематизація:

- 1) автоматичне сортування вхідних електронних листів за категоріями (наприклад, особисті, робочі, новинні розсилки, рекламні акції);
- 2) спеціальні правила сортування за відправником, темою, ключовими словами або часом;
- 3) система тегів для маркування та фільтрації електронних листів;
- 4) папки та підпапки для організації вручну;

5) функція пошуку з фільтрами (за датою, відправнику, тегами, непрочитаним і т.д.);

6) механізм визначення пріоритетів для виділення термінових або особливо важливих повідомлень;

в) додаткові функції:

1) інтеграція з календарем для управління подіями та зустрічами;

2) нагадування про завдання або прапорці для відстеження виконання, прив'язані до певних електронних листів;

3) інтелектуальні підказки (наприклад, швидкі відповіді, нагадування про подальші дії);

4) виявлення спаму та фішингу з автоматичною фільтрацією;

5) функція відкладеного перегляду електронної пошти для тимчасового приховування повідомлень і їх подальшого відображення;

г) UI та спеціальні можливості:

- адаптивний користувальницький інтерфейс для настільних комп'ютерів, планшетів і мобільних пристроїв;
- темний режим і налаштування теми оформлення;
- комбінації клавіш для досвідчених користувачів;
- підтримка перетягування електронних листів між папками;
- доступний дизайн, що відповідає стандартам WCAG;

д) адміністрування та налаштування:

- управління профілем користувача та уподобаннями;
- налаштування повідомлень про нові електронні листи, нагадування і т.д.;
- імпорт / експорт контактів і правил електронної пошти;
- підтримка плагінів або розширень для додаткових функцій;
- журнал активності для останніх дій і відстеження безпеки.

1.4.2 Вимоги до інтерфейсу

Графічний інтерфейс для нового поштового клієнту повинен бути зрозумілим, простим та адаптивним. Він повинен включати перегляд папки з вхідними, з відсортованими папками або категоріями (наприклад, для роботи, особистих цілей, рекламних акцій), бічну панель для навігації та окрему область для читання та перегляду повідомлень. Користувачі повинні мати можливість перетягувати електронні листи, встановлювати прапорці для масових дій та легко застосовувати фільтри. Інтерфейс повинен підтримувати світлий / темний режим, добре працювати на мобільних пристроях та настільних комп'ютерах та забезпечувати швидкий доступ до пошуку, тегів та налаштувань. Кнопки й написи повинні бути чіткими, з однаковими значками і плавними переходами.

1.4.3 Вимоги до безпеки

Застосунок має забезпечувати конфіденційність та безпеку даних користувачів:

- гешування паролей користувачів;
- застосування захищених механізмів автентифікації (двофакторна авторизація) користувачів та безпечної передачі даних.

1.4.4 Не функціональні вимоги до продукту

Обов'язкові вимоги.

Для сервера: серверна операційна система (ОС); підтримка фреймворку Django, підтримка системи управління базами даних (СУБД) MySQL.

Мінімальні системні вимоги та ПЗ, що необхідні для функціонування серверної частини застосунку:

- продуктивний хостинг (висока пропускна можливість);

- підтримка MySQL.

Для експлуатації клієнтської частини необхідні такі програмно-технічні засоби:

- ПК або смартфон з можливістю виходу в Інтернет;
- актуальна версія браузера.

1.5 Висновки за розділом 1

Перший розділ пояснювальної записки було присвячено вирішенню трьох робочих задач, що необхідні для розробки нового та актуального поштового клієнту з сортуванням вхідних листів, а саме:

- було досліджено предметну область електронного листування та використання застосунків поштових клієнтів;
- було досліджено існуючі рішення на ринку, виявити їхні переваги та недоліки;
- на основі виявлених переваг та недоліків – було сформовано перелік функціональних вимог до нового – якіснішого рішення.

Вирішення цих робочих задач необхідно для подальшої роботи над застосунком.

2 МАТЕРІАЛИ І МЕТОДИ

Другий розділ буде присвячено висвітленню вибору технічних засобів розробки. Зокрема буде обґрунтовано вибір мови програмування, фреймворку та середовища розробки. Окрему увагу буде приділено структурному плануванню розробки.

2.1 Вибір мови програмування

За останнє десятиліття Python і Django стали популярним поєднанням для веброзробки. Їх популярність продовжує зростати завдяки їх простоті, гнучкості та здатності обробляти складні програми за допомогою чистого коду та високої швидкості розробки. Python відомий своїм читабельним синтаксисом та простотою навчання, що робить його привабливим як для початківців, так і для професіоналів. З іншого боку, Django – це високорівневий вебфреймворк, побудований поверх Python, що надає безліч готових інструментів і бібліотек [20].

Однією з ключових причин популярності Python у веброзробці є його простота та читабельність. Python використовує чіткий і стислий синтаксис, який допомагає розробникам писати зрозумілий код з меншою кількістю рядків. Це має вирішальне значення в середовищі спільної роботи, де команди працюють спільно і читаність коду відіграє життєво важливу роль. Крім того, філософія дизайну Python наголошує на читабельності та простоті коду, що допомагає зменшити ймовірність помилок та полегшує підтримку коду з часом [20].

Ще однією перевагою є велика стандартна бібліотека Python та пакети сторонніх розробників. Python поставляється з безліччю вбудованих модулів, які охоплюють все – від обробки файлів до інтернет-протоколів. Крім цього, Python Package Index пропонує широку екосистему сторонніх бібліотек, які полегшують все – від доступу до баз даних та інтеграції API до машинного

навчання та обробки природної мови. Ця екосистема необхідна веброзробникам, яким часто потрібно швидко розширювати свої програми, не винаходячи велосипед [20].

Django спирається на сильні сторони Python і пропонує потужний набір інструментів, спеціально розроблений для створення вебдодатків. Однією з його основних функцій є система Object-Relational Mapping (ORM), яка дозволяє розробникам взаємодіяти з базою даних, використовуючи код Python замість запитів SQL. Це підвищує продуктивність, зменшує кількість помилок і забезпечує узгодженість і чистоту кодової бази. ORM обробляє міграції баз даних, запити та взаємозв'язки логічним та Python-способом, усуваючи багато повторюваних завдань, пов'язаних з управлінням базами даних [20].

Django також поставляється з вбудованою панеллю адміністратора, яка автоматично генерується на основі моделей, визначених у застосунку. Цей інтерфейс адміністратора дозволяє розробникам та адміністраторам сайтів легко керувати вмістом, користувачами та даними без написання додаткового коду. Це значно економить час при розробці і особливо корисно для створення внутрішніх інструментів та інформаційних панелей [21].

Безпека – ще одна важлива причина, чому Django воліють використовувати у веброзробці. Django включає надійні функції безпеки, такі як захист від ін'єкцій SQL, міжсайтових сценаріїв (тип XSS), підробка міжсайтових запитів та перехоплення кліків. Він заохочує безпечні методи розробки та включає компоненти проміжного програмного забезпечення, які допомагають застосовувати ці стандарти за замовчуванням. Для розробників, які створюють вебзастосунки, які обробляють конфіденційні дані або вимагають автентифікації користувача, Django забезпечує надійну основу, не вимагаючи великих налаштувань [21].

Швидка розробка є відмінною рисою Django. Фреймворк заснований на принципі DRY і підкреслює можливість повторного використання компонентів. За допомогою Django розробники можуть швидко створювати новий проєкт, використовуючи вбудовані команди та інструменти. У

платформу інтегровані такі функції, як маршрутизація URL-адрес, створення шаблонів та обробка форм, що дозволяє розробникам швидше переходити від ідеї до реалізації. Така швидкість особливо корисна для стартапів, створення прототипів або будь-якої іншої ситуації, коли час виходу на ринок має вирішальне значення [21].

Іншим важливим фактором є потужна підтримка спільноти та документація, доступна як для Python, так і для Django. Обидва мають великі активні спільноти, які сприяють екосистемі за допомогою посібників, бібліотек, розширень та фреймворків. Офіційна документація Django добре написана та всебічно описує більшість аспектів фреймворку. Форуми спільноти, Stack Overflow, GitHub та інші платформи пропонують рішення поширених проблем та рекомендації щодо найкращої практики [21].

Масштабованість також важлива для вебзастосунків, і Python з Django здатні працювати із застосунками з високим трафіком. Деякі з найбільших вебсайтів у світі, включаючи Instagram та Pinterest, використовують Django для створення елементів своєї архітектури. Django підтримує модульну розробку, дозволяючи програмам масштабуватися горизонтально, додаючи більше серверів та балансуючи навантаження. Він також підтримує асинхронні можливості і може інтегруватися з системами кешування, такими як Redis або Memcached, для підвищення продуктивності [21].

Сумісність Python з іншими технологіями також робить його привабливим. Python може легко взаємодіяти з іншими мовами та інструментами, що корисно при інтеграції із зовнішніми системами або застарілим програмним забезпеченням. Це особливо корисно в корпоративних середовищах, де вебдодаткам може знадобитися взаємодія з існуючою інфраструктурою [21].

З точки зору навчання, Python і Django зручні для початківців. Синтаксис Python близький до природної мови, що скорочує час навчання, а структурований підхід Django допомагає початківцям розробникам зрозуміти типовий процес роботи вебпрограми. Інструменти побудови каркасів Django

та чітка структура проєкту забезпечують міцну основу для вивчення концепцій веброзробки, таких як Model-View-Controller (MVC), маршрутизація, моделювання даних та управління користувачами [21].

Тестування – це ще одна сфера, в якій Django досягає успіху. Django включає тестову платформу, вбудовану в ядро, що заохочує розробників писати тести та дотримуватися підходу до розробки, заснованого на тестуванні. Автоматизоване тестування важливо для підтримки якості додатків, особливо в міру їх зростання в розмірах і складності. Django підтримує модульні тести, інтеграційні тести та спеціальні тестові випадки, що дозволяє легко переконатися, що нові функції або виправлення помилок не порушують існуючу функціональність [22].

Наявність опцій хостингу є ще однією практичною перевагою. Багато популярних хостингових сервісів пропонують вбудовану підтримку додатків Python та Django. Такі платформи, як Heroku, DigitalOcean та AWS, забезпечують прості робочі процеси розгортання, а конфігурацію Django можна налаштувати відповідно до різних налаштувань середовища розробки, проміжного етапу та виробництва [22].

Інтеграція з інтерфейсними технологіями також проста. У той час як Django обробляє серверну частину, розробники можуть легко інтегрувати її з інтерфейсними платформами, такими як React, Vue.js або Angular, для створення сучасних, чуйних користувацьких інтерфейсів. Платформа Django Rest Framework розширює можливості Django, спрощуючи створення RESTful API, дозволяючи створювати повнофункціональні або ізольовані додатки, в яких інтерфейс і серверна частина взаємодіють по протоколу http [22].

Для корпоративного використання Django забезпечує зручність обслуговування, стабільність і довгострокову підтримку. Випуски Django добре керуються та дотримуються чіткого плану, включаючи версії з довгостроковою підтримкою, які підтримуються протягом декількох років. Це

забезпечує підприємствам передбачуване та стабільне середовище розробки, необхідне для критично важливих застосунків [22].

Python та Django також підтримують інтернаціоналізацію та локалізацію, які важливі для програм, що обслуговують глобальну аудиторію. Вбудовані інструменти полегшують переклад вмісту, підтримують різні часові пояси та регіональне форматування, дозволяючи розробникам створювати багатомовні програми без значних витрат часу [22].

Що стосується інструментів розробки, Python та Django добре підтримуються сучасними IDE та редакторами. Такі інструменти, як PyCharm, забезпечують чудову підтримку Django, включаючи такі функції, як завершення коду, Налаштування та перегляд баз даних. Навіть легші редактори, такі як Visual Studio Code, мають потужну підтримку плагінів, що робить процес розробки плавним та продуктивним [22].

Таким чином, популярність Python та Django для веброботки зумовлена їх простотою, потужними функціями, можливостями швидкої розробки та потужною екосистемою. Вони пропонують безпечну, масштабовану та добре задокументовану основу для створення всього, від невеликих вебсайтів до масштабних корпоративних додатків. Їх постійне зростання і впровадження в різних галузях промисловості відображає їх ефективність в задоволенні різноманітних потреб сучасних веброботників [22].

Вибір Python та фреймворку Django на основі засвідчених переваг та недоліків наведено у вигляді табл. 2.1. У якості критеріїв порівняння використано:

- мовна узгодженість – деякі стеки дозволяють писати як інтерфейс, так і серверну частину однією мовою (наприклад, JavaScript), тоді як інші, такі як Python та Django, розділяють їх. У поштових застосунках внутрішня логіка часто складна (синтаксичний аналіз, фільтрація, виявлення спаму, IMAP/SMTP і т.д.). Python добре підходить для цього. Незважаючи на зручність використання JS з повним стеком, розділення завдань часто призводить до

більш чистого та простішого в обслуговуванні коду в довгостроковій перспективі;

- надійність фреймворку – хороший фреймворк визначає, наскільки швидко і безпечно ви зможете створити багатофункціональний застосунок. Django має включені батареї, що означає, що він надає ORM, інтерфейс адміністратора, управління користувачами та багато іншого прямо з коробки. Це критично важливо для програми електронної пошти, яка з самого початку потребує надійних систем облікових записів, зберігання повідомлень та обробки завдань;

- продуктивність (необроблена швидкість) – продуктивність завжди викликає занепокоєння, особливо у адаптивних застосунках. Незважаючи на те, що Node.js може перевершувати Django у швидкодії вводу-виводу, більшість програм електронної пошти не є інструментами для спілкування в режимі реального часу. Вони справляються з більш повільними комунікаційними та фоновими завданнями, такими як синхронізація повідомлень, синтаксичний аналіз та фільтрація спаму. Швидкість роботи Django достатня для цих випадків використання, особливо в поєднанні з підтримкою асинхронності і кешування;

- швидкість розробки – швидший розвиток означає Швидше виконання ітерацій, менше помилок та менші витрати. Django надає вбудовані компоненти (ORM, адміністрування, форми, аутентифікація), що означає, що розробникам не потрібно винаходити велосипед. Для стартапів або дослідницьких проєктів, що створюють поштовий клієнт, Django значно прискорює виведення на ринок;

- крива навчання – наскільки легко розробникам зрозуміти, підтримувати його чи сприяти проєкту. Синтаксис Python простіший і читабельніший, ніж часто складний інтерфейс JavaScript + стек бекенда. Спрощені кодові бази особливо важливі для довгострокової підтримки та залучення нових розробників;

– безпека – електронна пошта є надзвичайно конфіденційною. Безпека не підлягає обговоренню. Django автоматично захищає від багатьох поширених вебвразливостей (XSS, CSRF, SQL-ін'єкція). Неправильні налаштування безпеки є однією з найпоширеніших причин злому в реальному світі. Django допомагає запобігти їх за замовчуванням;

– функції в режимі реального часу – електронна пошта може включати сповіщення або синхронізацію в режимі реального часу, хоча це не так важливо, як програми обміну повідомленнями. Оновлення в режимі реального часу корисні (наприклад, сповіщення про нові електронні листи), але не відіграють вирішальної ролі. Django може підтримувати це через канали Django або WebSockets, і такої додаткової гнучкості часто буває достатньо;

– масштабованість – у міру зростання бази користувачів застосунок повинен плавно масштабуватися. Платформи електронної пошти розвиваються повільно і передбачувано. Django добре масштабується по вертикалі і тепер підтримує асинхронне масштабування по горизонталі. Node.js має великий потенціал мікросервісу, але Django дуже ефективно обробляє монолітне ядро поштової системи;

– інтеграція з базами даних – системи електронної пошти в значній мірі орієнтовані на дані. Оскільки під час роботи доведеться мати справу з обліковими записами користувачів, папками, ярликами, фільтрами, журналами – все це вимагає надійної структури реляційної бази даних. ORM в Django тісно інтегрований і відрізняється високою надійністю, спрощуючи доступ до даних і зменшуючи кількість помилок;

– обробка зовнішнього інтерфейсу – інтерфейси електронної пошти, як правило, складні – наприклад, багатопанельні макети, перетягування, пошук та фільтрація. React є потужним інструментом для створення таких інтерфейсів користувача, але Django може обслуговувати динамічні інтерфейси користувача, використовуючи React /Vue, або простіші інтерфейси користувача, використовуючи рендеринг на стороні сервера Така гнучкість дозволяє проводити оптимізацію залежно від розміру команди та потреб UX;

– спільнота та підтримка – сильна спільнота забезпечує довгострокову життєздатність і допомагає при виникненні проблем. Django та Python підтримуються сильними спільнотами, де доступно багато прикладів, пов'язаних з електронною поштою та пакетами з відкритим кодом. Node.js / React також має величезну екосистему, але Django часто краще підходить для таких складних програм, як електронна пошта;

– складність розгортання – наскільки складно запуснути додаток і підтримувати його працездатність. Програми Django можна легко розгорнути за допомогою серверів WSGI, таких як Gunicorn, та служб, таких як Heroku або Docker. Стеки JavaScript часто вимагають об'єднання пакетів, зворотного проксі-сервера та оркестрування контейнерів. Більше гнучкості, але і складності;

– відповідність корпоративним вимогам – клієнти електронної пошти часто орієнтовані на команди, підприємства чи установи, які піклуються про конфіденційність. Django широко використовується в уряді, охороні здоров'я та фінансах — сферах, де пріоритетними є безпека та зрозумілість. Зрозумілість і перевіряемость Python є перевагою в регульованому середовищі;

– інтерфейс адміністратора – програми електронної пошти потребують інструментів для підтримки користувачів, управління повідомленнями, діагностики та ведення журналів. Django поставляється з вбудованою спеціальною панеллю адміністратора, що скорочує час розробки серверної панелі. Node / React вимагають створення з нуля або інтеграції сторонніх інформаційних панелей;

– пакети з відкритим кодом – використання готових пакетів дозволяє не винаходити велосипед заново. У Django і Python є готові пакети для відправки / отримання електронних листів, синтаксичного аналізу MIME, IMAP, SMTP, Celery для черг і т. д. Вони є основоположними для поштового додатку. Параметри JavaScript у цьому просторі слабші або новіші;

– робота з протоколами електронної пошти – це головне в розробці поштових додатків-такі протоколи, як IMAP, SMTP, POP3, синтаксичний аналіз MIME. Тут Python чудовий з бібліотеками, такими як imaplib, smtplib, email та mail-parser. У JavaScript відсутня надійна, продумана підтримка низькорівневих поштових протоколів, що ускладнює розробку серверної частини.

У таблиці 2.1 наведено порівняння мов програмування.

Таблиця 2.1 – Порівняння мов програмування

Критерій	Python + Django	JavaScript + React + Node.js
Мовна узгодженість	Лише для бекенда(Python); для інтерфейсу потрібен HTML / JS	JS з повним стеком (інтерфейс React, серверна частина Node)
Надійність фреймворку	Django - це фреймворк, що працює від батарейок, з ORM, панеллю адміністратора, авторизацією і т. д.	Node + Express мінімальний; потрібні сторонні модулі
Продуктивність (необроблена швидкість)	Повільніше, ніж Node, у необроблених завданнях вводу-виводу	Швидка, керована подіями, неблокуюча Архітектура
Швидкість розробки	Високий рівень завдяки вбудованим інструментам, ORM і admin	Помірний рівень; потрібно більше налаштувань та бібліотек
Крива навчання	Простий для початківців завдяки високорівневим абстракціям	Крутіше з декількома інструментами / бібліотеками (React, Webpack і т. д.)
Безпека	Надійні налаштування за замовчуванням (CSRF, XSS, захист від введення SQL)	Безпека повинна бути налаштована вручну
Функції в режимі реального часу	Для WebSockets потрібні канали або сторонні інструменти.	Вбудована в Node.js (socket.io і т. д.)
Масштабованість	Добре для вертикального масштабування; асинхронний за допомогою Django + ASGI	Відмінно підходить для горизонтального масштабування та мікросервісів

Продовження таблиці 2.1

Критерій	Python + Django	JavaScript + React + Node.js
Інтеграція з базами даних	Жорсткий ORM (Django ORM) з PostgreSQL, MySQL і т. д.	Використовує ORMS, такі як Sequelize або Prisma
Обробка зовнішнього інтерфейсу	Потрібен окремий інтерфейс або використовуються шаблони Django	React ефективний для динамічних інтерфейсів користувача
Спільнота та підтримка	Великий, продуманий, особливо для серверної розробки	Він широко використовується, особливо в інтерфейсній екосистемі JS
Складність розгортання	Простіше з такими інструментами, як Gunicorn, Docker, Heroku	Більше рухомих компонентів (Node, react build, Зворотні проксі)
Відповідність корпоративним вимогам	Широко використовується в уряді, охороні здоров'я, наукових дослідженнях	Популярний у стартапах та користувацьких вебдодатках
Інтерфейс адміністратора	Поставляється з потужним, автоматично генерується призначенням для користувача інтерфейсом адміністратора	Вимагає створення панелей адміністратора вручну
Пакети з відкритим кодом	Багата екосистема (електронна пошта, авторизація, фонові завдання)	Величезна бібліотека NPM, але більш фрагментована
Робота з протоколами електронної пошти	Чудово (за допомогою таких бібліотек, як 'imaplib', 'smtplib', 'email', 'celery')	Слабка вбудована підтримка; покладається на сторонні модулі

Вбудовані інструменти для виконання складних серверних завдань, таких як аутентифікація, адміністрування, обробка сеансів і моделювання баз даних, роблять Django ідеальною платформою для роботи з повідомленнями, користувачами і правилами. Відмінна підтримка роботи з протоколами електронної пошти, вкладеннями, синтаксичним аналізом і плануванням – Python володіє кращими бібліотеками для роботи з електронною поштою.

Більш висока швидкість розробки, особливо для команд, які віддають перевагу серверній частині і бізнес-логіці, а не ультрадинамічним користувацьким інтерфейсам.

Підвищена безпека за замовчуванням, що особливо важливо при роботі з особистою або корпоративною електронною поштою. Спрощене довгострокове обслуговування і зручність читання, що особливо важливо в застосунках, що працюють зі складними, регульованими або чутливими до конфіденційності робочими процесами. Для створення поштового клієнту, де ключовими є надійність, безпечне управління користувачами, аналіз повідомлень і ефективність серверної частини, Python + Django пропонує більш стабільну і безпечну основу, при цьому при бажанні дозволяючи інтегрувати його з сучасними інтерфейсами, такими як React.

2.2 Вибір середовища розробки

Розробники часто використовують PyCharm для проєктів на Django, оскільки це середовище розробки (IDE) надає безліч вбудованих інструментів, які прискорюють і спрощують роботу з Django. PyCharm розуміє структуру Django і пропонує такі функції, як інтелектуальне завершення коду, швидка навігація та пропозиції щодо автоматичного імпорту. Воно також підтримує шаблони Django, маршрутизацію URL-адрес, взаємозв'язки моделей та міграції за допомогою спеціальних інструментів виділення та навігації [23]-[24].

Вбудований в PyCharm потужний налагоджувач допомагає швидко знаходити проблеми. Він включає візуальний інтерфейс для перевірки змінних, покрокового виконання коду та перевірки трасування стека. PyCharm також має вбудований термінал, браузер баз даних, інтеграцію Git та підтримку віртуальних середовищ, тому розробники можуть керувати всіма частинами свого проєкту на Django в одному місці [24].

Інша причина полягає в тому, що PyCharm полегшує тестування. Він підтримує платформу тестування Django нестандартно і дозволяє розробникам запускати тести та переглядати результати безпосередньо в IDE. Він також підтримує інструменти редагування, форматування та рефакторингу коду, які допомагають підтримувати чистоту та впорядкованість коду [23].

В цілому, PyCharm економить час і підвищує продуктивність, пропонуючи багатий набір інструментів, спеціально розроблених для розробки на Django, тому розробники часто вибирають його [24].

Обґрунтування вибору PyCharm в порівнянні з іншими IDE представлено у табл. 2.2. Серед критеріїв порівняння будемо розглядати наступні:

- підтримка платформи Django – Django – це повнофункціональна вебплатформа з моделями, переглядами, шаблонами, формами, менеджерами URL-адрес та адміністраторами, які тісно інтегровані. Тісний зв'язок між компонентами Django означає, що розробники IDE повинні глибоко розуміти ці зв'язки. PyCharm Pro забезпечує автоматичне прив'язування пов'язаних файлів, таких як моделі до форм або подання до шаблонів – те, що VS Code не може зробити без складної настройки. Це підвищує точність і продуктивність;

- час налаштування – важливо розпочати роботу, особливо для нових проєктів або залучення нових розробників. Підтримка Django в PyCharm доступна з самого початку. Вона без проблем виявляє settings.py, налаштовує manage.py завдання і налаштовує інтерпретатор. У VS Code вам потрібно вручну встановлювати плагіни, налаштовувати завдання та змінні середовища. Це економить години роботи в командному середовищі;

- інтелектуальний аналіз коду – можливість автоматичного заповнення, навігації та визначення контексту має вирішальне значення у великих кодових базах. Програми Django використовують Конвенції (наприклад, імена полів, контекстні змінні в шаблонах), які не завжди підбираються загальними інструментами IntelliSense. PyCharm надає

рекомендації, специфічні для Django, такі як пошук по полях (`author__name`, `created__lt` і т.д.) і шляхи перевірки форми, що допомагає запобігти помилкам;

- налагоджувач – налагодження займає центральне місце в розробці, особливо в складних серверних додатках, таких як поштові клієнти. Налагоджувач PyCharm може працювати з поданнями, шаблонами та фоновими завданнями, забезпечуючи доступ до даних запитів, контекстних змінних та запитів ORM. На відміну від цього, VS Code має обмежену підтримку повного стека Django і вимагає додаткової конфігурації для налагодження шаблонів або асинхронних подань;

- інтеграція з базами даних – Django – це платформа на основі ORM, що означає, що ваша програма працює у своїй базі даних. PyCharm надає вам вбудований переглядач SQL, тестер запитів та відображення моделі до таблиці. Це допомагає візуально досліджувати вашу базу даних та розуміти взаємозв'язки даних. Навпаки, VS Code використовує зовнішні інструменти або плагіни з менш плавною інтеграцією;

- налагодження шаблонів – саме в них виникає більшість помилок інтерфейсу в Django. PyCharm може перевіряти та налагоджувати змінні шаблону в режимі реального часу. Це дозволяє відстежити, чому значення `None`, де відбувається збій циклу або чому умова не виконується. У VS Code це не підтримується, тому доводиться покладатися на еквіваленти `print()` або `console.log()`;

- інструменти тестування – тестування має вирішальне значення для підтримки надійності в міру зростання вашого проєкту на Django. PyCharm має вбудовані панелі запуску тестів на Django, інструменти покриття та виявлення тестів. Ви можете запускати окремі тести, візуально перевіряти помилки та впевнено виконувати рефакторинг. У VS Code тестування виконується швидше вручну і залежить від CLI або окремих конфігурацій;

- навігація по панелі адміністратора – адміністратор Django є одним з найбільш часто використовуваних компонентів фреймворку. PyCharm розпізнає моделі адміністратора і дозволяє швидко переходити від

zareєстрованих класів адміністратора до визначень моделей. Це полегшує налагодження або розширення функцій адміністратора, особливо у великих програмах, де ви налаштовуєте відображення списків, фільтри та дозволи;

- управління версіями – кожен сучасний проєкт використовує Git, а Django часто включає відгалуження для функцій, міграції або виправлення помилок. Обидва інструменти підтримують Git, але інтеграція PyCharm є більш глибокою. Він включає графічні подання, інструменти для вирішення конфліктів злиття, фільтрацію історії комітів та перевірку виправлень – і все це без виходу з IDE. Це важливо для спільної розробки на Django;

- якість коду та рефакторинг – проєкти на Django можуть швидко розростатися, що ускладнює підтримання чистоти коду. PyCharm може розумно рефакторувати конструкції, специфічні для Django. Наприклад, перейменування поля `model` оновлює його використання у шаблонах, формах, серіалізаторах та фільтрах. Це зменшує помилки та ручну працю порівняно з VS Code, де вам потрібно виконати пошук-заміну або сподіватися, що плагін знайде це;

- UI та навігація по проєкту – Django використовує структурований макет проєкту (програми, моделі, шаблони, статичні файли). PyCharm покращує навігацію за допомогою переглядів та ярликів, специфічних для Django. Ви можете, наприклад, перейти від шаблону URL безпосередньо до подання або шаблону. У VS Code для цього потрібні зовнішні розширення, і їм все ще не вистачає глибини;

- розширюваність – налаштування корисно, особливо в середовищі з великою кількістю мов. VS Code тут просто чудовий-Ви можете розширити його за допомогою тисяч плагінів та мовної підтримки. Але для Django PyCharm вже включає все, що потрібно більшості розробників, не складаючи все разом;

- ліцензування та вартість – ціноутворення по-різному впливає на команди та окремих працівників. VS Code доступний безкоштовно, що чудово підходить для студентів та любителів. PyCharm Pro платний, але вартість

часто виправдовується підвищенням продуктивності і підтримкою, яку він забезпечує при професійній розробці на Django;

– Docker & Remote Dev – Django часто використовується в контейнерах або в середовищах віддаленої розробки. PyCharm має потужні інструменти інтерфейсу Docker та вбудовану підтримку віддаленої розробки. VS Code може відповідати цьому за допомогою плагінів, але вбудований набір інструментів PyCharm є більш зручним і відшліфованим для складних проєктів.

Обґрунтування вибору PyCharm в порівнянні з іншими IDE представлено у таблиці 2.2.

Таблиця 2.2 – Порівняння IDE

Критерій	PyCharm	Visual Studio Code
Підтримка платформи	Вбудована глибока інтеграція (моделі, подання, шаблони, маршрутизація URL-адрес)	Потрібні розширення; автолінковка обмежена
Час налаштування	Попередньо налаштована для проєктів на Django	Потрібно ручна настройка і модуль, що підключається
Інтелектуальний аналіз коду	Додатково (підказки по типу поля, дозвіл шаблону, самоаналіз моделі)	Хороший, але обмежений без великої кількості плагінів
Налагоджувач	Повна інтеграція з сервером Django, контекст шаблону, асинхронні подання	Основна налагодження на Python; додаткові налаштування для Django
Інтеграція з базами даних	Вбудований браузер баз даних, редактор SQL, інтеграція з ORM	Розширення доступні, але менш інтегровані
Налагодження шаблонів	Так-налагодження шаблонів Django з повним контекстом	Немає вбудованої підтримки

Продовження таблиці 2.2

Критерій	PyCharm	Visual Studio Code
Інструменти тестування	Інтегрований Django test runner, користувацький інтерфейс для тестового покриття	Потрібні розширення або інтеграція з CLI
Навігація по панелі адміністратора	Розпізнає та пов'язує маршрути та моделі адміністратора Django	Немає вбудованого аналізу моделі адміністратора
Управління версіями	Інтегровані інструменти VCS, інтерактивне перебазування, вирішення конфліктів	Хороша підтримка Git за допомогою розширень
Якість коду та рефакторинг	Складний рефакторинг (наприклад, перейменування полів моделі в рамках проєкту)	Від базового до помірного рефакторингу
UI та навігація по проєкту	Представлення проєкту, специфічне для Django, автоматичне зв'язування між поданнями, моделями, шаблонами	Базове дерево проєктів
Розширюваність	Помірний рівень; plugin marketplace менше, ніж VS Code	Дуже висока; велика екосистема розширень
Docker & Remote Dev	Професійна версія Платна; спільноті не вистачає інструментів Django	Безкоштовне і з відкритим вихідним кодом
Підтримка платформи	Потужна підтримка (інтерфейс користувача для Docker, SSH, WSL)	Добре працює з розширеннями

З урахуванням низки мобільних можливостей для розробки з використанням Django вибір було зроблено на користь PyCharm.

2.3 Структурна схема розробки

На рис. 2.1 наведено структурну схему вебзастосунку поштового клієнта, що включає функції сортування вхідних електронних листів.

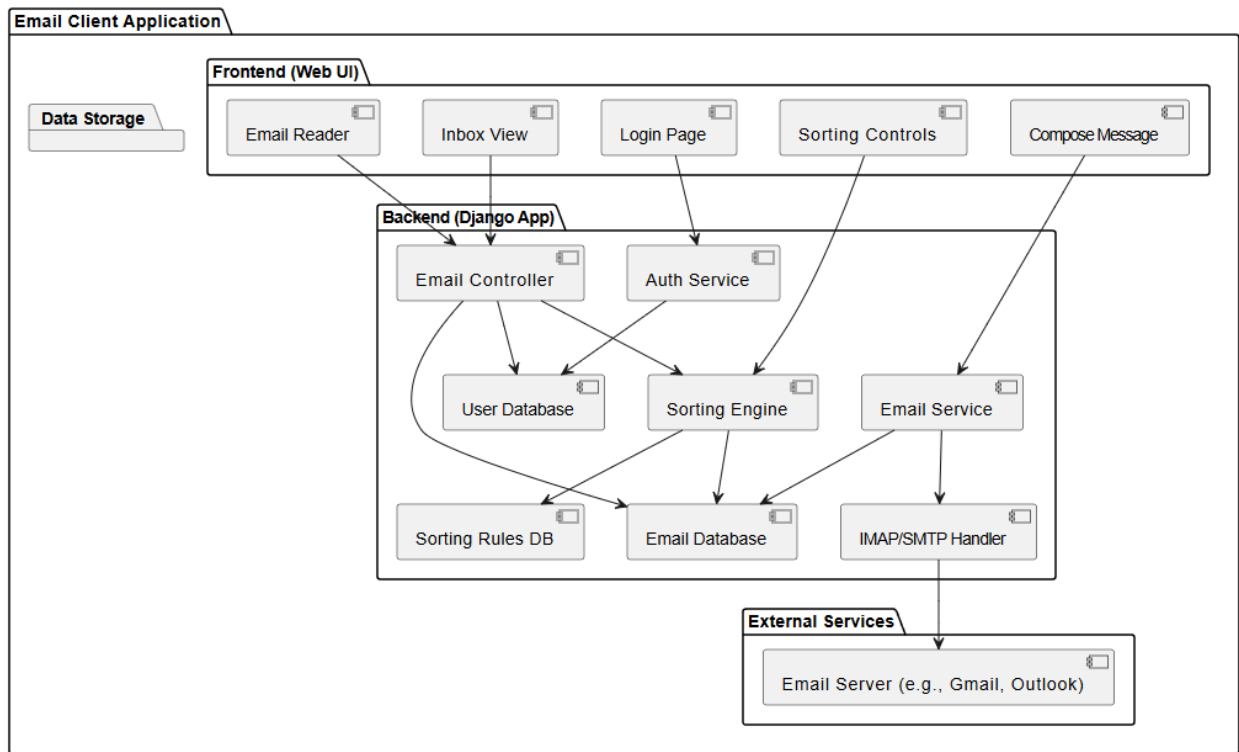


Рисунок 2.1 – Структурна схема застосунку

Ось що показує кожна частина структури:

а) інтерфейс (Web UI) – це інтерфейс користувача, який працює у браузері. Він включає в себе:

- 1) сторінка входу – для аутентифікації користувача;
- 2) перегляд вхідних повідомлень – відображає електронні листи, часто згруповані за категоріями (наприклад, робочі, особисті);
- 3) програма для читання електронної пошти – дозволяє користувачам відкривати та читати окремі електронні листи;
- 4) створити повідомлення – дозволяє створювати і відправляти нові електронні листи;

5) елементи керування сортуванням – тут користувачі визначають або налаштовують порядок сортування електронних листів (наприклад, за відправником, темою, важливістю);

б) серверна частина (бізнес-логіка на базі Django) – серверна логіка, написана на Django (Python), відповідає за обробку та зберігання даних:

1) служба авторизації – управляє входом користувача в систему і аутентифікацією;

2) служба електронної пошти – надсилає електронні листи через протокол SMTP та отримує електронні листи через протокол IMAP;

3) контролер електронної пошти – центральна логіка, яка витягує, обробляє та відображає електронні листи;

4) механізм сортування – застосовує правила сортування вхідних електронних листів за категоріями або папками;

5) обробник IMAP / SMTP – взаємодіє із зовнішніми поштовими серверами (такими як Gmail або Outlook);

в) зовнішні служби – поштові сервери, до яких програма підключається для надсилання та отримання повідомлень:

1) сервер електронної пошти (наприклад, Gmail, Outlook) – це фактичні служби електронної пошти, які зберігають ваші електронні листи;

г) сховище даних – бази даних, що використовуються для зберігання даних користувача та електронної пошти та управління ними:

1) база даних користувачів – зберігає дані облікового запису користувача (наприклад, облікові дані для входу, налаштування);

2) база даних електронної пошти – зберігає вибраний та класифікований вміст електронної пошти;

3) база даних правил сортування – зберігає власні або автоматичні правила сортування;

д) взаємозв'язки і робочий процес – інтерфейс надсилає серверні запити (наприклад, для читання або сортування електронних листів):

- 1) серверна частина виконує аутентифікацію, підключається до поштових серверів і обробляє електронні листи;
- 2) механізм сортування застосовує правила для сортування електронних листів у міру їх надходження;
- 3) усі дані зберігаються в базах даних для швидкого доступу та повторного використання.

Ця структура є масштабованою та модульною, що означає, що компоненти можуть бути вдосконалені або замінені незалежно один від одного: наприклад, оновлення механізму сортування без зміни всієї програми.

2.4 Висновки за розділом 2

Другий розділ було присвячено висвітленню вибору технічних засобів розробки. Зокрема було обґрунтовано вибір мови програмування, фреймворку та середовища розробки. Окрему увагу було приділено структурному плануванню розробки.

3 ОСНОВНІ РІШЕННЯ ЩОДО РЕАЛІЗАЦІЇ КОМПОНЕНТІВ СИСТЕМИ

3.1 Проєктування моделей системи

Розробники використовують Unified Modeling Language (UML) для планування, візуалізації та передачі структури та поведінки вебзастосунків протягом усього процесу розробки. Хоча формально UML використовується не в кожному проєкті, багато розробників використовують його діаграми для підвищення наочності, управління архітектурою та зменшення непорозумінь [25].

Збір вимог та раннє планування. Діаграми прецедентів показують взаємодію між користувачами (учасниками) та системою. Розробники використовують їх, щоб зрозуміти та задокументувати, для чого користувачам потрібен вебзастосунок [25].

Проєктування системи – використання діаграм класів. Представляють модель даних і взаємозв'язки між класами, такими як користувачі, електронні листи, папки і т.д. у вебзастосунках вони часто відображають схеми баз даних або серверні об'єкти (наприклад, моделі Django).

Схеми компонентів використовуються для планування структури і взаємозв'язку даних. Демонструє, як застосунок розбито на логічні частини, такі як інтерфейс, серверна частина, база даних і сторонні сервіси [25].

Моделювання поведінки за допомогою діаграми послідовностей. Описує, як компоненти або класи взаємодіють з часом для виконання завдання (наприклад, отримання та сортування електронної пошти).

Використовується для визначення взаємодії між інтерфейсним і серверним рівнями або API. Діаграми активностей показують процес управління або прийняття рішень (наприклад, етапи входу користувача або класифікації електронної пошти) [25].

Діаграми станів допомагають моделювати логіку перед кодуванням. Представляють стани, в яких може перебувати об'єкт (наприклад, електронна пошта: непрочитана – прочитана – архівована) [25].

Діаграми класів (часто з використанням нотації в стилі UML) допомагають проєктувати бази даних – допомагають наочно уявити, як пов'язані таблиці і які поля включені в них. Цей крок допомагає при плануванні моделі Django або налаштуванні ORM [25].

Посилання на впровадження:

- розробники звертаються до діаграм UML під час написання коду, щоб відповідати запланованій архітектурі та поведінці;
- допомагає швидко впровадити нових членів команди, показуючи, як влаштована система.

Тестування та документація:

- UML-діаграми часто включаються в технічну документацію і можуть використовуватися для створення тестових кейсів;
- діаграми дій та послідовності корисні для визначення сценаріїв тестування та розуміння того, чого очікувати від кожної дії користувача [25].

Переваги використання UML у веброботі:

- заохочує структуроване мислення – перед написанням коду;
- спрощує командну взаємодію – особливо у великих командах;
- допомагає забезпечити чітке розуміння вимог;
- допомагає в налагодженні та оптимізації, показуючи передбачуваний системний процес [25].

Хоча не кожен проєкт вебзастосунок використовує повне моделювання на мові UML, вибіркоче використання ключових діаграм, таких як діаграми класів та діаграми послідовностей, може значно покращити процес проєктування та розробки. Дайте мені знати, якщо вам потрібні шаблони або приклади для поштової програми на основі Django [25].

Діаграма класів на рис. 3.1 представляє основну структуру застосунку поштового клієнта з сортуванням вхідних електронних листів. Вона показує, як взаємодіють і взаємопов'язані основні компоненти системи.

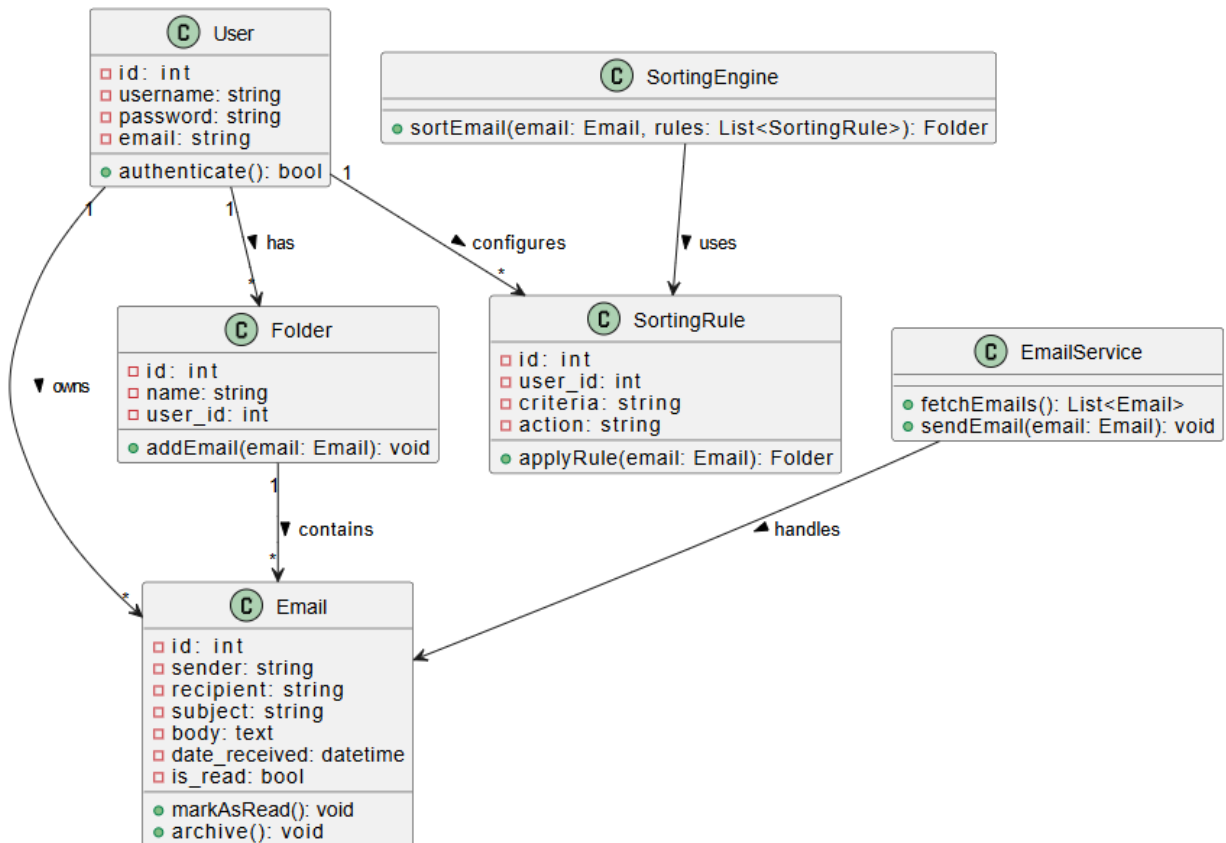


Рисунок 3.1 – Діаграма класів

Користувач: представляє людину, яка використовує додаток. Кожен користувач може мати кілька електронних листів, папок та правил сортування.

Електронна пошта: це окреме електронне повідомлення з такими властивостями, як відправник, тема та вміст. Воно включає в себе способи позначити як прочитане або архівоване.

Папка: представляє категорію або папку, в якій можуть бути впорядковані електронні листи (наприклад, вхідні, робочі або спам). Папка може містити багато електронних листів і належати одному користувачеві.

Правило сортування: визначає, як слід класифікувати вхідні електронні листи (наприклад, залежно від відправника або теми). Кожне правило належить користувачеві і вказує, куди слід надіслати електронний лист.

EmailService: виконує отримання та надсилання електронних листів за допомогою протоколів електронної пошти, таких як IMAP та SMTP.

SortingEngine: застосовує правила сортування до електронних листів і вирішує, в яку папку їх слід перемістити.

Схема показує, як електронні листи пов'язані з користувачами та папками, як застосовуються правила та як серверні служби працюють з цими об'єктами. Ця структура підтримує такі функції, як автоматична Класифікація Електронної Пошти, спеціальні фільтри та організоване управління поштовими скриньками.

Діаграма прецедентів наведена на рис. 3.2. Ілюструє ключові взаємодії між Користувачем і Застосунком з сортуванням вхідних електронних листів.

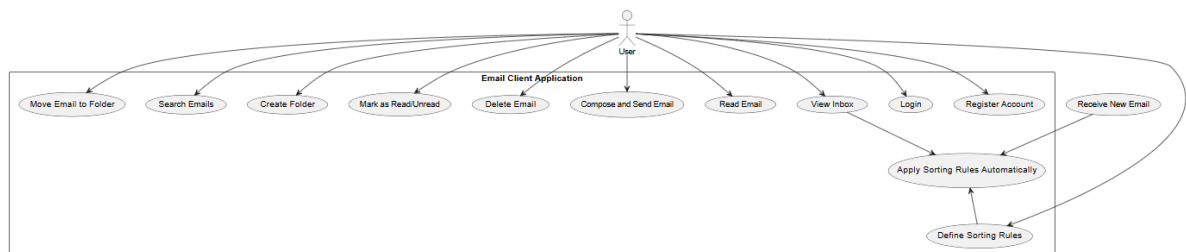


Рисунок 3.2 – Діаграма прецедентів

Користувач є головним актором, який виконує такі дії, як:

- реєстрація;
- вхід;
- перегляд папки вхідних листів;
- читання;
- надсилання;
- видалення електронних листів.

Користувач також може створювати папки та визначати правила сортування для автоматичної організації своїх електронних листів.

При надходженні електронних листів або при зверненні користувача до папки Вхідні система автоматично застосовує правила сортування для переміщення електронних листів у відповідні папки.

Додаткові випадки використання включають позначення електронних листів як прочитаних чи непрочитаних, пошук певних повідомлень та ручне переміщення електронних листів у папки.

Ця діаграма дає загальне уявлення про те, які функціональні можливості система пропонує користувачам і як сортування інтегрована в основний робочий процес електронної пошти.

Діаграма станів (рис. 3.3) показує, як окремий електронний лист рухається по системі:

- а) отриманий електронний лист: електронний лист завантажується з сервера;
- б) застосунок намагається застосувати правила сортування:
 - 1) якщо будь-яке правило відповідає, електронний лист сортується і поміщається в папку;
 - 2) якщо жодне правило не відповідає, лист залишається в папці Вхідні.

Після сортування або розміщення в папці Вхідні електронний лист переходить у стан непрочитаний.

Коли користувач читає його, він переходить у стан Прочитано.

У режимі Прочитано користувач може архівувати або видаляти електронний лист.

Електронні листи також можна видаляти безпосередньо зі стану Непрочитане.

Ця діаграма корисна для моделювання того, як система обробляє переходи стану кожного електронного листа, а також для розробки як внутрішньої логіки, так і поведінки інтерфейсу користувача.

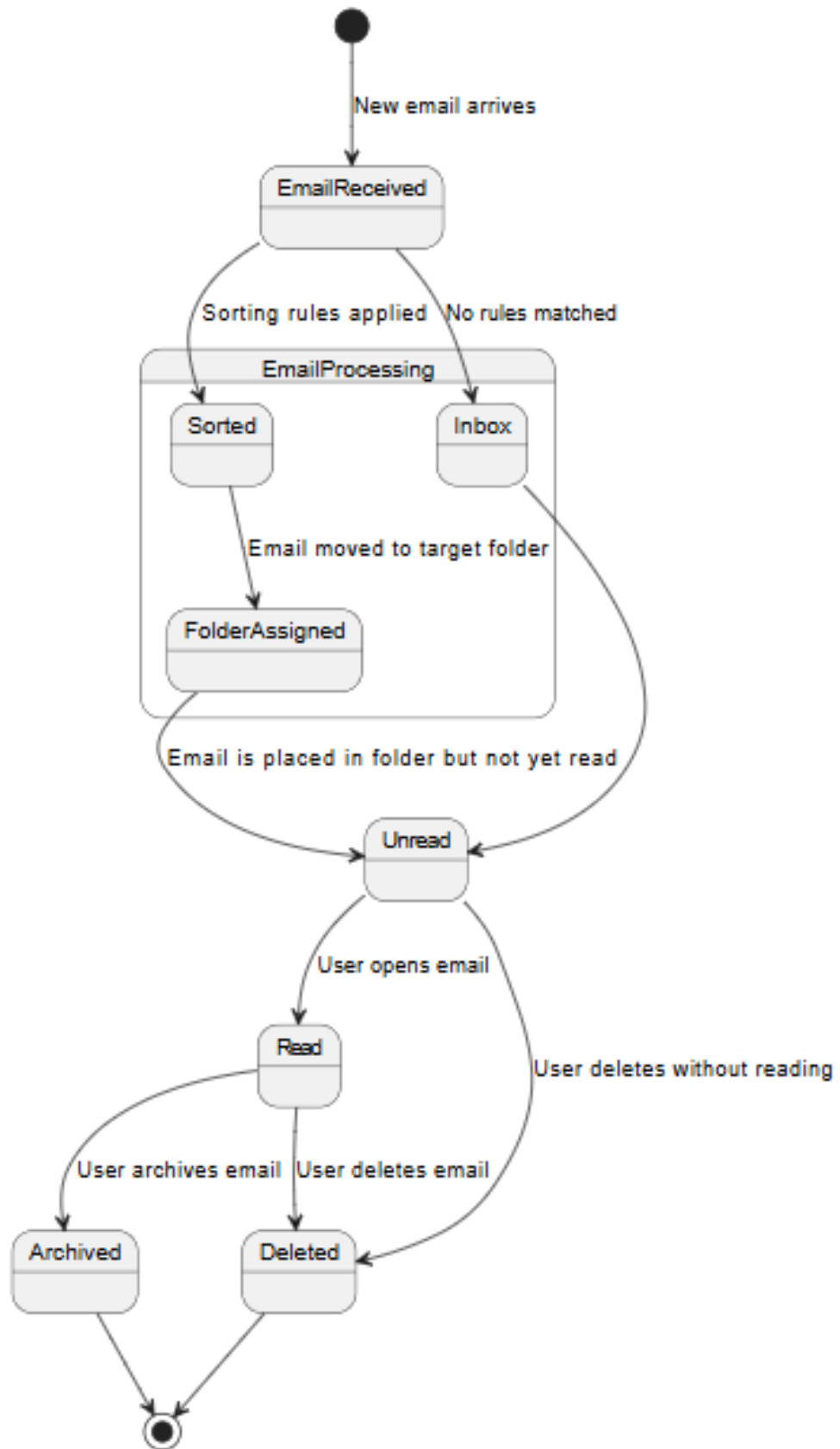


Рисунок 3.3 – Діаграма станів системи

3.2 Проектування інтерфейсу

Розробка графічного інтерфейсу користувача (Graphical User Interface – GUI) поштового клієнту з сортуванням вхідних листів вимагає пильної уваги до зручності використання, ясності та ефективності [26].

Дизайн має бути орієнтований на користувача. Інтерфейс повинен бути інтуїтивно зрозумілим для користувачів усіх рівнів. Слід віддавати перевагу простоті і звичності — користувачі очікують, що макети будуть схожі на популярні поштові клієнти (наприклад, Gmail або Outlook). Слід мінімізувати час навчання, зберігаючи дизайн послідовним і передбачуваним [26].

Очистити та мінімізувати структуру поштових скриньок і папок, зокрема вхідних листів. Чітко відобразити папку Вхідні та всі створені користувачем папки. Дозволити переміщення електронних листів між папками за допомогою перетягування або установки прапорців. Підтримка вкладених папок або міток, якщо це необхідно для просунутих користувачів [26].

Інтерфейс сортування електронної пошти. Застосунок має надавати просту у використанні панель для створення правил сортування та управління ними. Слід використовувати прості випадаючі списки або поля форми. Показати візуальний перегляд або імітацію, як будуть працювати правила сортування. Дозволити змінювати порядок сортування або вмикати / вимикати його.

Відображення поштових скриньок і попередній перегляд електронної пошти. Варто створити чистий макет з двома або трьома панелями:

- список папок (ліворуч), список адрес електронної пошти (у центрі), попередній перегляд або повний вміст (праворуч);
- чітке відображення статусу прочитано / непрочитано, важливих позначок, вкладень і тегів;
- увімкнення інструментів пошуку та фільтрації для швидкого пошуку конкретних повідомлень.

Оперативність і доступність. Слід переконатися, що інтерфейс добре працює на настільних комп'ютерах, планшетах та смартфонах. Використовувати адаптивний дизайн, щоб макет адаптувався до розміру екрана. Забезпечити доступність для користувачів з обмеженими можливостями:

- навігація з клавіатури;
- сумісність з програмою читання з екрану;
- режим високої контрастності та настроюваний розмір шрифту.

Необхідно чітко вказувати помилки при введенні правил сортування, збої при відправці електронної пошти або проблеми з сервером. Запобігати випадковому видаленню за допомогою підтверджень або опцій скасування [26].

Дозволити користувачам налаштовувати:

- правила сортування;
- перегляд папок за замовчуванням;
- параметри відображення (щільність, тема, розмір шрифту);
- увімкнення опції для імпорту / експорту правил сортування або синхронізації налаштувань на різних пристроях.

Індикатори безпеки та конфіденційності. Чітко відображати, зашифрований електронний лист чи потенційно підозрілий. Забезпечити видимі елементи керування для виходу з системи, керування налаштуваннями облікового запису та зміни паролів [26].

Хороший графічний інтерфейс для поштового клієнта з функціями сортування повинен бути:

- зрозумілим і звичним;
- ефективним для управління великими обсягами повідомлень;
- прозорим в тому, як правила сортування впливають на поведінку поштових скриньок;
- доступним, швидким і надійним.

Проектування з урахуванням реальної поведінки та очікувань користувачів допомагає створити продукт, який здається корисним і природним, особливо в такій складній і чутливій сфері, як електронна пошта. Дайте мені знати, якщо вам потрібен макет каркасу або пропозиції щодо дизайну для цих функцій [26].

3.3 Сортування листів

Метод k-найближчих сусідів (k-nearest neighbors – k-NN) може бути використаний в застосунку поштового клієнту для автоматичного сортування вхідних електронних листів за відповідними папками або категоріями на основі їх подібності з раніше класифікованими електронними листами (рис. 3.4) [27].

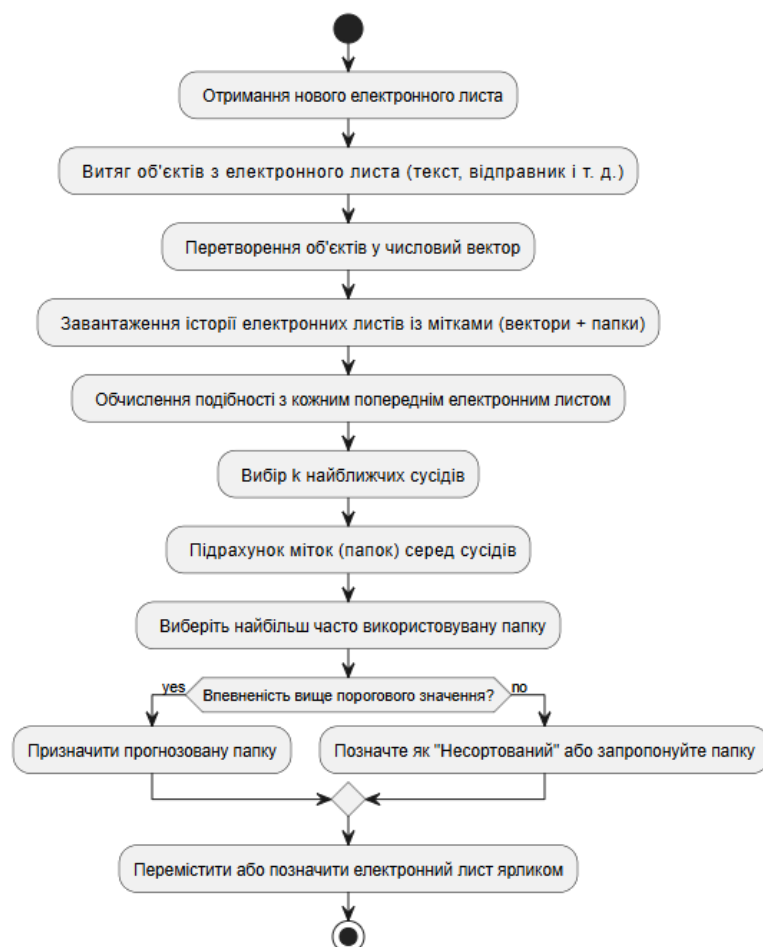


Рисунок 3.4 – Використання k-NN для автоматичного сортування листів у застосунку

k-NN – це керований метод машинного навчання, який класифікує нові точки даних (у цьому випадку вхідні електронні листи) на основі міток найближчих сусідів у навчальних даних [27].

Кроки для використання k-NN для сортування електронних листів:

а) підготовка даних – збір попередніх електронних листів, які вже були відсортовані користувачем за папками;

б) кожен електронний лист стає точкою даних, а папка, в якій він знаходиться, є його міткою (класом);

в) вилучення ознак – перетворення кожного електронний лист у вектор числових ознак. Це може включати:

- 1) набір слів або оцінки TF-IDF за темою / текстом;
- 2) адреса електронної пошти відправника / домен;
- 3) наявність певних ключових слів;
- 4) час отримання;
- 5) довжина електронного листа;
- 6) чи є в ньому вкладення;
- 7) попередня обробка (наприклад, виділення ключових слів, видалення стоп-слів) допомагає поліпшити якість;
- 8) застосування моделі (логічний висновок);

г) коли надходить новий електронний лист – витягуються його характеристики таким же чином. k-NN буде використано, щоб знайти k найбільш схожих електронних листів в історичних даних (наприклад, використовуючи косинусоїдальну подібність або евклідову відстань) [27];

д) визначення (на основі відстані до найближчого сусіда), до яких папок належать цей лист;

е) призначення новому електронному листу найбільш часто зустрічається папку серед сусідніх.

Переваги використання k-NN:

– не потрібно попередня побудова моделі – метод використовує весь набір даних безпосередньо;

- інтерпретується: легко зрозуміти, як приймаються рішення, оглядаючи сусідів;
- адаптивність: дозволяє швидко відображати зміни в поведінці користувачів при переміщенні електронних листів.

4 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

4.1 Призначення й умови застосування програми

Важливо розробити новий вебзастосунок поштовий клієнт з сортуванням листів, оскільки люди отримують занадто багато електронної пошти, а важливі повідомлення часто втрачаються. Сучасний застосунок має автоматично впорядковувати електронні листи за темами, відправниками чи пріоритетами, заощаджуючи час користувачів та зменшуючи стрес. Багато сучасних поштових клієнтів перевантажені, повільні або недостатньо гнучкі. Нове рішення може запропонувати більш зрозумілий дизайн, більш високу продуктивність та інтелектуальні функції, такі як теги, фільтри та сортування, які допоможуть користувачам зосередитися на найважливішому.

Усі функціональні вимоги до нового застосунку можна уявити таким чином:

- управління електронною поштою;
- сортування та систематизація;
- додаткові функції (інтеграція з календарем, нагадування про завдання, тощо);
- UI та спеціальні можливості;
- адміністрування та налаштування.

4.2 Вимоги до інтерфейсу

Графічний інтерфейс для нового поштового клієнту повинен бути зрозумілим, простим та адаптивним. Він повинен включати перегляд папки з вхідними, з відсортованими папками або категоріями (наприклад, для роботи, особистих цілей, рекламних акцій), бічну панель для навігації та окрему область для читання та перегляду повідомлень. Користувачі повинні мати можливість перетягувати електронні листи, встановлювати прапорці для

масових дій та легко застосовувати фільтри. Інтерфейс повинен підтримувати світлий / темний режим, добре працювати на мобільних пристроях та настільних комп'ютерах та забезпечувати швидкий доступ до пошуку, тегів та налаштувань. Кнопки й написи повинні бути чіткими, з однаковими значками і плавними переходами.

4.3 Умови виконання програми

Обов'язкові вимоги.

Для сервера: серверна ОС; підтримка фреймворку Django, підтримка СУБД MySQL.

Мінімальні системні вимоги та ПЗ, що необхідні для функціонування серверної частини застосунку:

- продуктивний хостинг (висока пропускна можливість);
- підтримка MySQL.

Для експлуатації клієнтської частини необхідні такі програмно-технічні засоби:

- ПК або смартфон з можливістю виходу в Інтернет;
- актуальна версія браузера.

4.4 Методика та результати тестування

Під час тестування було опрацьовані головні сторінки застосунку, з якими взаємодіє користувач.

Тож на рис. 4.1 наведено головну сторінку з вхідними листами користувача.

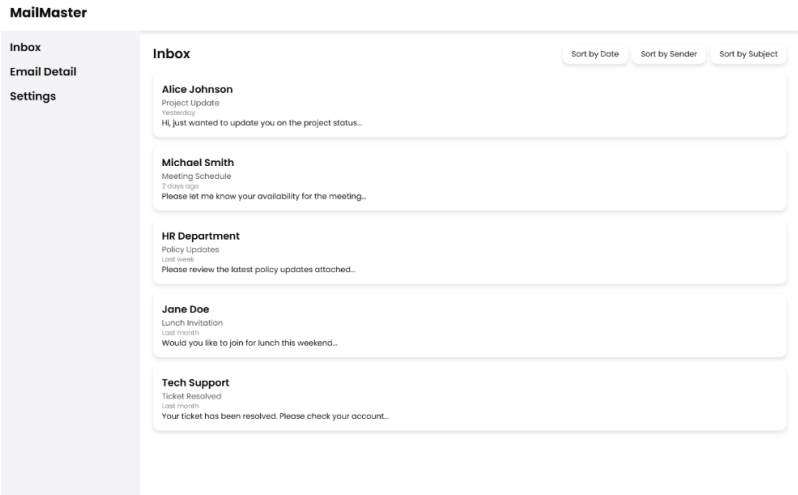


Рисунок 4.1 – Сторінка вхідних листів

На рис. 4.2 наведено фільтрацію та варіанти сортувань листів зв ключовими словами, з налаштуванням критеріїв відображення результатів пошуку.

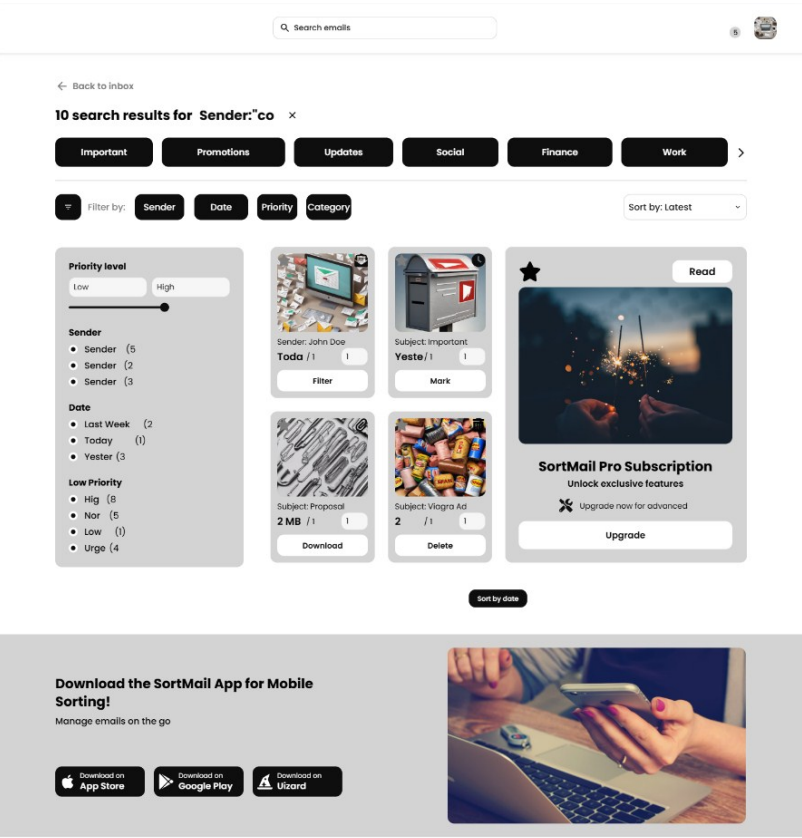


Рисунок 4.2 – Сторінка пошуку та сортування листів

На рис. 4.3 представлено сторінку налаштувань користувача його скриньок.

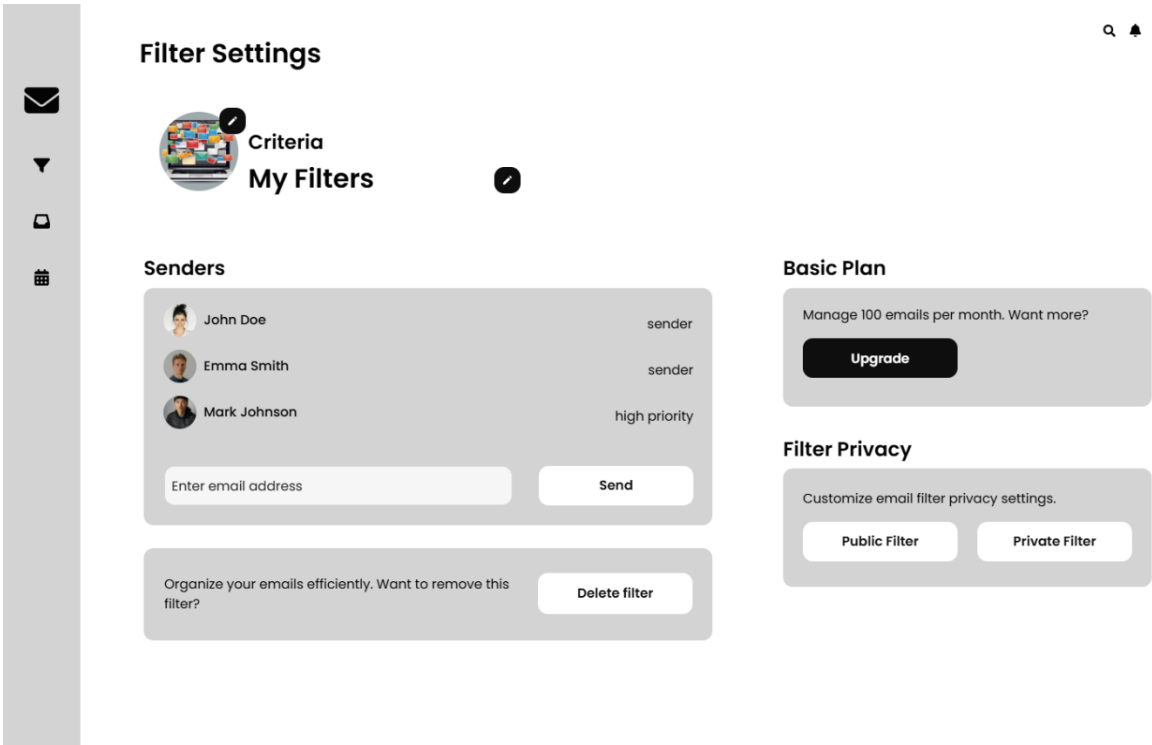


Рисунок 4.3 – Сторінка налаштування поштових скриньок

На рис. 4.4 продемонстровано сторінку з прикладом відкритого листа.

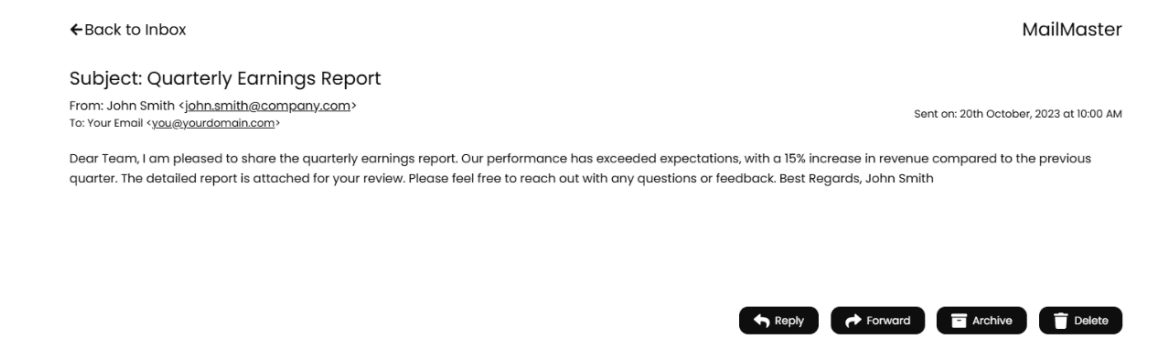


Рисунок 4.4 – Сторінка з прикладом відкритого листа

На рис. 4.5 наведено приклад налаштування автоматичних сортувань листів.

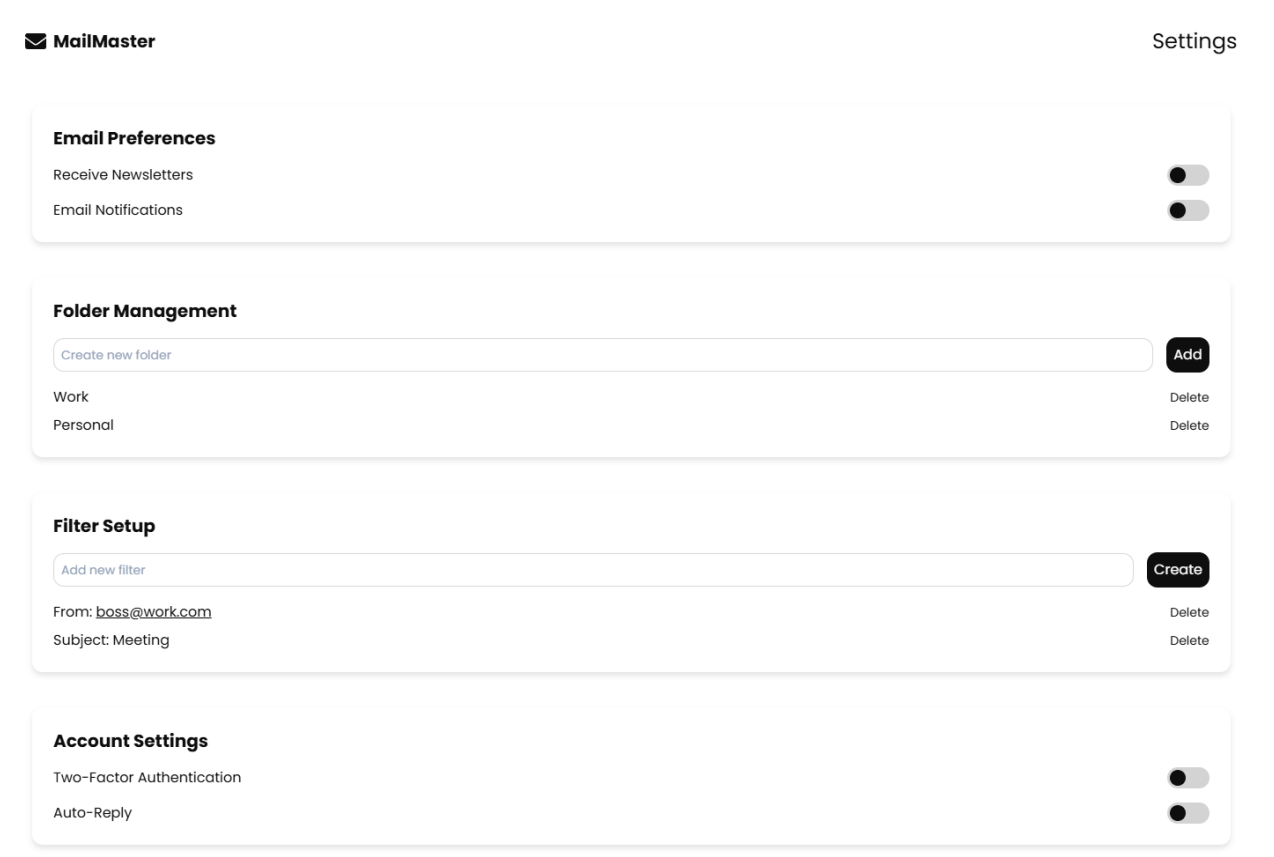


Рисунок 4.5 – Приклад налаштування автоматичних сортувань листів

ВИСНОВКИ

В ході виконання дипломної кваліфікаційної роботи бакалавра було досягнуто головну мету – виконано програмну реалізації застосунку поштового клієнту з сортуванням вхідних листів. Для цього було виконано низку задач, серед яких:

- досліджено предметну область електронного листування та використання застосунків поштових клієнтів;
- досліджено існуючі рішення на ринку, виявлено їхні переваги та недоліки;
- на основі виявлених переваг та недоліків – сформовано перелік функціональних вимог до нового – якіснішого рішення;
- спроектовано та розроблено новий застосунок – поштовий клієнт з сортуванням листів;
- виконано тестування нової розробки;
- надано ґрунтовні висновки за виконаною роботою.

Розроблений вебзастосунок був реалізований із використанням мови програмування Python, надбудови Django та середовища розробки PyCharm Free Educational License.

Виконано функціональне тестування застосунку за сценаріями використання застосунку.

На основі розв’язаних задач було отримано програмний продукт, що в повній мірі реалізує поставлені задачі та в повній мірі відповідає всім функціональним вимогам визначеним на початку розробки.

В наступних релізах застосунку можна покращити механізми автоматичного сортування листів, використовуючи складніші методи кластеризації.

Усі завдання дипломної кваліфікаційної роботи бакалавра повністю виконано.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. The 7 best email clients for Windows / [Electronic resource] – Access mode: <https://zapier.com/blog/best-email-clients-windows/>.
2. 12 Best Email Clients 2025 / [Electronic resource] – Access mode: <https://efficient.app/best/email>.
3. The Best Email Clients for 2025 / [Electronic resource] – Access mode: https://www.pcmag.com/picks/best-email-clients?test_uuid=02LlF0iWKsilxYTJVF8uH5y&test_variant=A.
4. Component «Mail» from OneBox / [Electronic resource] – Access mode: <https://1b.app/en/app/email/>.
5. The 9 Best Email Client Apps for Gmail for Every Use Case / [Electronic resource] – Access mode: <https://missiveapp.com/blog/best-email-client-for-gmail>.
6. Top 10 Email Clients & Software for 2025 / [Electronic resource] – Access mode: <https://www.helpdesk.com/blog/email-clients/>.
7. Email Client Essentials: Choosing the Best for Small Business Success / [Electronic resource] – Access mode: <https://www.engagebay.com/blog/email-client/>.
8. Email Marketing Management, Made Easy / [Electronic resource] – Access mode: <https://www.daysmart.com/features-email-marketing-management/>.
9. 3 Effective Emails to Book More Event Planning Clients / [Electronic resource] – Access mode: <https://www.eventcertificate.com/emails-event-planning-clients/>.
10. Transform Your Legal Communication with the Best Email Tool for Lawyers / [Electronic resource] – Access mode: <https://mailsuite.com/en/solutions/lawyers-and-attorneys>.
11. 4 Project Management Tips For Faster Customer Email Service / [Electronic resource] – Access mode: <https://www.mailmunch.com/blog/project-management-tips-for-faster-service>.

12. Microsoft Outlook / [Електрон. ресурс]. – Режим доступу: <https://www.microsoft.com/uk-ua/microsoft-365/outlook/email-and-calendar-software-microsoft-outlook>.
13. Про Microsoft Outlook / [Електрон. ресурс]. – Режим доступу: <https://uk.extendoffice.com/outlook-tutorials-get-started/about-microsoft-outlook.html>.
14. Thunderbird / [Электрон. ресурс]. – Режим доступа: <https://www.thunderbird.net/ru/>.
15. Початок роботи з Thunderbird / [Электрон. ресурс]. – Режим доступа: <https://support.mozilla.org/ru/kb/nachalo-raboty-v-thunderbird>.
16. Gmail пошта від Google / [Электрон. ресурс]. – Режим доступа: <https://www.gmail.com/mail/help/intl/ru/about.html?de>.
17. History of Gmail / [Electronic resource] – Access mode: https://en.wikipedia.org/wiki/History_of_Gmail.
18. 10 Essential Tips for Requirements Analysis in Custom Software Development / [Electronic resource] – Access mode: <https://www.comidor.com/blog/low-code/requirements-analysis-tips/>.
19. Requirement Analysis Techniques / [Electronic resource] – Access mode: <https://www.visual-paradigm.com/guide/requirements-gathering/requirement-analysis-techniques/>.
20. Meet Django / [Electronic resource] – Access mode: <https://www.djangoproject.com/>.
21. Django for Web Development / [Electronic resource] – Access mode: <https://realpython.com/learning-paths/django-web-development/>.
22. What Is the Django Framework? The Complete Beginner's Guide / [Electronic resource] – Access mode: <https://careerfoundry.com/en/blog/web-development/django-framework-guide/>.
23. Основи Pycharm / [Електрон. ресурс]. – Режим доступу: <https://w3schoolsua.github.io/hyperskill/pycharm-basics.html#gsc.tab=0>.

24. PyCharm The only Python IDE you need / [Electronic resource] – Access mode: <https://www.jetbrains.com/pycharm/>.
25. UML modeling and visual editing / [Electronic resource] – Access mode: <https://www.ibm.com/docs/en/dma?topic=users-uml-modeling-visual-editing>.
26. A New Graphic User Interface Design for 3D Modeling Software for Children / [Electronic resource] – Access mode: https://link.springer.com/chapter/10.1007/978-3-030-77040-2_20.
27. What is the k-nearest neighbors (KNN) algorithm? / [Electronic resource] – Access mode: <https://www.ibm.com/think/topics/knn>.

ДОДАТОК А
Технічне завдання

Вступ

Даний застосунок поштового клієнту присвячений веденню поштових акаунтів користувачів, з автоматичною функцією сортування листів.

A.1 Підстава для розробки

Підставою для розробки є завдання на дипломну кваліфікаційну роботу на тему «Програмна реалізація застосунку поштового клієнту з сортуванням вхідних листів», затверджене наказом Національного університету «Запорізька політехніка» № 209 від 28 квітня 2025 р.

A.2 Основні вимоги до програми, що розробляється

Важливо розробити новий вебзастосунок поштовий клієнт з сортуванням листів, оскільки люди отримують занадто багато електронної пошти, а важливі повідомлення часто втрачаються. Сучасний застосунок має автоматично впорядковувати електронні листи за темами, відправниками чи пріоритетами, заощаджуючи час користувачів та зменшуючи стрес. Багато сучасних поштових клієнтів перевантажені, повільні або недостатньо гнучкі. Нове рішення може запропонувати більш зрозумілий дизайн, більш високу продуктивність та інтелектуальні функції, такі як теги, фільтри та сортування, які допоможуть користувачам зосередитися на найважливішому.

Метою роботи є програмна реалізація застосунку поштового клієнту з сортуванням вхідних листів.

A.2.1 Вимоги до функціональних характеристик

Розподілимо функціональні вимоги до застосунку за групами функціональних можливостей використання:

а) управління електронною поштою:

- 1) надсилання та отримання електронних листів через протокол IMAP / SMTP;
- 2) підтримка декількох облікових записів електронної пошти (наприклад, Gmail, Outlook, Yahoo, користувацьких доменів);
- 3) перегляд і читання повідомлень електронної пошти в режимі потокової передачі;
- 4) складання, відповідь, пересилання та видалення електронних листів;
- 5) позначення листів як прочитані / непрочитані, позначені зірочкою або важливі;
- 6) додавання файлів та перегляд вкладення;

б) сортування та систематизація:

- 1) автоматичне сортування вхідних електронних листів за категоріями (наприклад, особисті, робочі, новинні розсилки, рекламні акції);
- 2) спеціальні правила сортування за відправником, темою, ключовими словами або часом;
- 3) система тегів для маркування та фільтрації електронних листів;
- 4) папки та підпапки для організації вручну;
- 5) функція пошуку з фільтрами (за датою, відправнику, тегами, непрочитаним і т.д.);
- 6) механізм визначення пріоритетів для виділення термінових або особливо важливих повідомлень;

в) додаткові функції:

- 1) інтеграція з календарем для управління подіями та зустрічами;
- 2) нагадування про завдання або прапорці для відстеження виконання, прив'язані до певних електронних листів;

- 3) інтелектуальні підказки (наприклад, швидкі відповіді, нагадування про подальші дії);
- 4) виявлення спаму та фішингу з автоматичною фільтрацією;
- 5) функція відкладеного перегляду електронної пошти для тимчасового приховування повідомлень і їх подальшого відображення;
- г) UI та спеціальні можливості:
 - 1) адаптивний користувальницький інтерфейс для настільних комп'ютерів, планшетів і мобільних пристроїв;
 - 2) темний режим і налаштування теми оформлення;
 - 3) комбінації клавіш для досвідчених користувачів;
 - 4) підтримка перетягування електронних листів між папками;
 - 5) доступний дизайн, що відповідає стандартам WCAG;
- д) адміністрування та налаштування:
 - 1) управління профілем користувача та уподобаннями;
 - 2) налаштування повідомлень про нові електронні листи, нагадування і т.д.;
 - 3) імпорт / експорт контактів і правил електронної пошти;
 - 4) підтримка плагінів або розширень для додаткових функцій;
 - 5) журнал активності для останніх дій і відстеження безпеки.

A.2.2 Вимоги до інтерфейсу програми

Графічний інтерфейс для нового поштового клієнту повинен бути зрозумілим, простим та адаптивним. Він повинен включати перегляд папки з вхідними, з відсортованими папками або категоріями (наприклад, для роботи, особистих цілей, рекламних акцій), бічну панель для навігації та окрему область для читання та перегляду повідомлень. Користувачі повинні мати можливість перетягувати електронні листи, встановлювати прапорці для масових дій та легко застосовувати фільтри. Інтерфейс повинен підтримувати світлий / темний режим, добре працювати на мобільних пристроях та

настільних комп'ютерах та забезпечувати швидкий доступ до пошуку, тегів та налаштувань. Кнопки й написи повинні бути чіткими, з однаковими значками і плавними переходами.

А.2.4 Вимоги до складу та параметрів технічний засобів

Обов'язкові вимоги.

Для сервера: серверна ОС; підтримка фреймворку Django, підтримка СУБД MySQL.

Мінімальні системні вимоги та ПЗ, що необхідні для функціонування серверної частини застосунку:

- продуктивний хостинг (висока пропускна можливість);
- підтримка MySQL.

Для експлуатації клієнтської частини необхідні такі програмно-технічні засоби:

- ПК або смартфон з можливістю виходу в Інтернет;
- актуальна версія браузера.

ДОДАТОК Б
Опис програми

Б.1 Загальні відомості

Важливо розробити новий вебзастосунок поштовий клієнт з сортуванням листів, оскільки люди отримують занадто багато електронної пошти, а важливі повідомлення часто втрачаються. Сучасний застосунок має автоматично впорядковувати електронні листи за темами, відправниками чи пріоритетами, заощаджуючи час користувачів та зменшуючи стрес. Багато сучасних поштових клієнтів перевантажені, повільні або недостатньо гнучкі. Нове рішення може запропонувати більш зрозумілий дизайн, більш високу продуктивність та інтелектуальні функції, такі як теги, фільтри та сортування, які допоможуть користувачам зосередитися на найважливішому.

Метою роботи є програмна реалізація застосунку поштового клієнту з сортуванням вхідних листів.

Б.2 Функціональне призначення

Розподілимо функціональні вимоги до застосунку за групами функціональних можливостей використання:

а) управління електронною поштою:

- 1) надсилання та отримання електронних листів через протокол IMAP / SMTP;
- 2) підтримка декількох облікових записів електронної пошти (наприклад, Gmail, Outlook, Yahoo, користувацьких доменів);
- 3) перегляд і читання повідомлень електронної пошти в режимі потокової передачі;
- 4) складання, відповідь, пересилання та видалення електронних листів;
- 5) позначення листів як прочитані / непрочитані, позначені зірочкою або важливі;
- 6) додавання файлів та перегляд вкладення;

б) сортування та систематизація:

1) автоматичне сортування вхідних електронних листів за категоріями (наприклад, особисті, робочі, новинні розсилки, рекламні акції);

2) спеціальні правила сортування за відправником, темою, ключовими словами або часом;

3) система тегів для маркування та фільтрації електронних листів;

4) папки та підпапки для організації вручну;

5) функція пошуку з фільтрами (за датою, відправнику, тегами, непрочитаним і т.д.);

б) механізм визначення пріоритетів для виділення термінових або особливо важливих повідомлень;

в) додаткові функції:

1) інтеграція з календарем для управління подіями та зустрічами;

2) нагадування про завдання або прапорці для відстеження виконання, прив'язані до певних електронних листів;

3) інтелектуальні підказки (наприклад, швидкі відповіді, нагадування про подальші дії);

4) виявлення спаму та фішингу з автоматичною фільтрацією;

5) функція відкладеного перегляду електронної пошти для тимчасового приховування повідомлень і їх подальшого відображення;

г) UI та спеціальні можливості:

1) адаптивний користувальницький інтерфейс для настільних комп'ютерів, планшетів і мобільних пристроїв;

2) темний режим і налаштування теми оформлення;

3) комбінації клавіш для досвідчених користувачів;

4) підтримка перетягування електронних листів між папками;

5) доступний дизайн, що відповідає стандартам WCAG;

д) адміністрування та налаштування:

- 1) управління профілем користувача та уподобаннями;
- 2) налаштування повідомлень про нові електронні листи, нагадування і т.д.;
- 3) імпорт / експорт контактів і правил електронної пошти;
- 4) підтримка плагінів або розширень для додаткових функцій;
- 5) журнал активності для останніх дій і відстеження безпеки.

Б.3 Опис логічної структури

На рис. Б.1 наведено структурну схему вебзастосунку поштового клієнта, що включає функції сортування вхідних електронних листів.

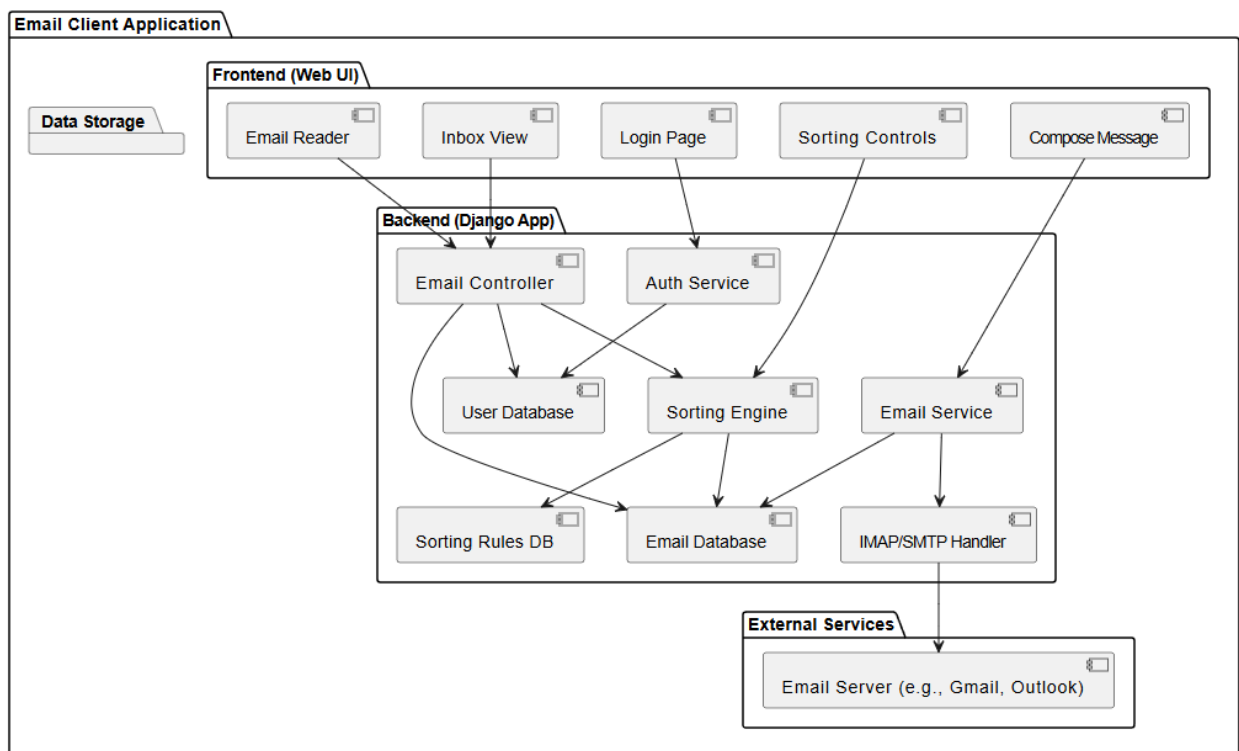


Рисунок Б.1 – Структурна схема застосунку

Ось що показує кожна частина структури:

а) інтерфейс (Web UI) – це інтерфейс користувача, який працює у браузері. Він включає в себе:

- 1) сторінка входу – для аутентифікації користувача;
- 2) перегляд вхідних повідомлень – відображає електронні листи, часто згруповані за категоріями (наприклад, робочі, особисті);
- 3) програма для читання електронної пошти – дозволяє користувачам відкривати та читати окремі електронні листи;
- 4) створити повідомлення – дозволяє створювати і відправляти нові електронні листи;
- 5) елементи керування сортуванням – тут користувачі визначають або налаштовують порядок сортування електронних листів (наприклад, за відправником, темою, важливістю);

б) серверна частина (бізнес-логіка на базі Django) – серверна логіка, написана на Django (Python), відповідає за обробку та зберігання даних:

- служба авторизації – управляє входом користувача в систему і аутентифікацією;
- служба електронної пошти – надсилає електронні листи через протокол SMTP та отримує електронні листи через протокол IMAP;
- контролер електронної пошти – центральна логіка, яка витягує, обробляє та відображає електронні листи;
- механізм сортування – застосовує правила сортування вхідних електронних листів за категоріями або папками;
- обробник IMAP / SMTP – взаємодіє із зовнішніми поштовими серверами (такими як Gmail або Outlook);

в) зовнішні служби – поштові сервери, до яких програма підключається для надсилання та отримання повідомлень:

- сервер електронної пошти (наприклад, Gmail, Outlook) – це фактичні служби електронної пошти, які зберігають ваші електронні листи;
- г) сховище даних – бази даних, що використовуються для зберігання даних користувача та електронної пошти та управління ними:
 - база даних користувачів – зберігає дані облікового запису користувача (наприклад, облікові дані для входу, налаштування);
 - база даних електронної пошти – зберігає вибраний та класифікований вміст електронної пошти;
 - база даних правил сортування – зберігає власні або автоматичні правила сортування;
- д) взаємозв'язки і робочий процес – інтерфейс надсилає серверні запити (наприклад, для читання або сортування електронних листів):
 - серверна частина виконує аутентифікацію, підключається до поштових серверів і обробляє електронні листи;
 - механізм сортування застосовує правила для сортування електронних листів у міру їх надходження;
 - усі дані зберігаються в базах даних для швидкого доступу та повторного використання.

Ця структура є масштабованою та модульною, що означає, що компоненти можуть бути вдосконалені або замінені незалежно один від одного: наприклад, оновлення механізму сортування без зміни всієї програми.

Б.4 Використані технічні засоби

Обов'язкові вимоги.

Для сервера: серверна ОС; підтримка фреймворку Django, підтримка СУБД MySQL.

Мінімальні системні вимоги та ПЗ, що необхідні для функціонування серверної частини застосунку:

- продуктивний хостинг (висока пропускна можливість);
- підтримка MySQL.

Для експлуатації клієнтської частини необхідні такі програмно-технічні засоби:

- ПК або смартфон з можливістю виходу в Інтернет;
- актуальна версія браузера.

ДОДАТОК В
Текст програми

```

models.py (Email & Folder models)

python

from django.db import models
from django.contrib.auth.models import User


class Folder(models.Model):
    user = models.ForeignKey(User,
on_delete=models.CASCADE)
    name = models.CharField(max_length=64)

    def __str__(self):
        return self.name


class Email(models.Model):
    user = models.ForeignKey(User,
on_delete=models.CASCADE)
    sender = models.CharField(max_length=255)
    subject = models.CharField(max_length=255)
    body = models.TextField()
    date_received = models.DateTimeField(auto_now_add=True)
    folder = models.ForeignKey(Folder, null=True,
blank=True, on_delete=models.SET_NULL)
    is_read = models.BooleanField(default=False)

    def __str__(self):
        return f"{self.subject} from {self.sender}"

forms.py (Optional Form)

python

from django import forms

```

```

from .models import Email

class EmailForm(forms.ModelForm):
    class Meta:
        model = Email
        fields = ['sender', 'subject', 'body']
knn_sorter.py (Email Sorting Logic)
python
from sklearn.feature_extraction.text import
TfidfVectorizer
from sklearn.neighbors import KNeighborsClassifier
from .models import Email, Folder

def sort_email(user, new_email):
    # Get user's labeled email history
    emails = Email.objects.filter(user=user,
folder__isnull=False)
    if not emails.exists():
        return None

    texts = [f"{e.subject} {e.body}" for e in emails]
    labels = [e.folder.name for e in emails]

    vectorizer = TfidfVectorizer(stop_words="english")
    X = vectorizer.fit_transform(texts)
    knn = KNeighborsClassifier(n_neighbors=3)
    knn.fit(X, labels)

    new_text = f"{new_email.subject} {new_email.body}"
    X_new = vectorizer.transform([new_text])

```

```

predicted = knn.predict(X_new)[0]

try:
    return Folder.objects.get(user=user,
name=predicted)
except Folder.DoesNotExist:
    return None
views.py (Basic views for inbox & adding email)
python
from django.shortcuts import render, redirect
from django.contrib.auth.decorators import
login_required
from .models import Email, Folder
from .forms import EmailForm
from .knn_sorter import sort_email

@login_required
def inbox(request):
    emails = Email.objects.filter(user=request.user).order_by('-
date_received')
    folders = Folder.objects.filter(user=request.user)
    return render(request, "inbox.html", {"emails":
emails, "folders": folders})

@login_required
def create_email(request):
    if request.method == "POST":
        form = EmailForm(request.POST)
        if form.is_valid():

```

```

        email = form.save(commit=False)
        email.user = request.user
        folder = sort_email(request.user, email)
        email.folder = folder
        email.save()
        return redirect("inbox")
    else:
        form = EmailForm()
        return render(request, "create_email.html",
{"form": form})

```

urls.py (App URLs)

```

python
from django.urls import path
from . import views

urlpatterns = [
    path("", views.inbox, name="inbox"),
    path("new/", views.create_email,
name="create_email"),
]

```

emailclient/urls.py (Project URLs)

```

python
Копировать
Редактировать
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include("emailapp.urls")),

```

```

]
templates/inbox.html (Basic Inbox View)
<!DOCTYPE html>
<html>
<head>
<title>Inbox</title>
</head>
<body>
<h1>Inbox</h1>
<a href="/new/">Send New Email</a>
<ul>
{% for email in emails %}
<li>
    <strong>{{ email.subject }}</strong>    from    {{
email.sender }}
    -    {{ email.folder.name    if    email.folder    }}    ({{
email.date_received }})
</li>
{% empty %}
<li>No emails found.</li>
{% endfor %}
</ul>
</body>
</html>

In admin.py:
from django.contrib import admin
from .models import Email, Folder

admin.site.register(Email)
admin.site.register(Folder)

```

Setup Commands

```
bash
```

```
# Create project and app
```

```
django-admin startproject emailclient
```

```
cd emailclient
```

```
python manage.py startapp emailapp
```

```
# Add 'emailapp' to settings.py INSTALLED_APPS
```

```
# Migrate DB
```

```
python manage.py makemigrations
```

```
python manage.py migrate
```

```
# Create superuser for admin
```

```
python manage.py createsuperuser
```

```
# Run server
```

```
python manage.py runserver;
```

ДОДАТОК Д
Слайди презентації

Міністерство освіти і науки України
кафедра програмних засобів

Програмна реалізація застосунку поштового клієнту з сортуванням вхідних листів

Software Implementation of an Email Client Application
Sorting the Incoming Emails

Виконав	студ. КНТ-141 Артем КУЛИКОВ
Керівник	канд. техн. наук, доцент Михайло КОЦУР
Нормоконтроль	асист. Марина КАЛІНІНА
Рецензент	канд. техн. наук, доцент Матвій ІЛЬЯШЕНКО

Рисунок Д.1 – Слайд 1

Об'єкт, предмет та мета роботи

Об'єкт дослідження – процес програмної реалізації застосунку поштового клієнту з сортуванням вхідних листів.

Предмет дослідження – програмні засоби для реалізації застосунку поштового клієнту з сортуванням вхідних листів.

Метою роботи є програмна реалізація застосунку поштового клієнту з сортуванням вхідних листів.

Рисунок Д.2 – Слайд 2

Аналіз предметної області

Критерії порівняння	Microsoft Outlook	Mozilla Thunderbird	Gmail
UI	Складний користувацький інтерфейс, особливо для нових користувачів; перевантажений	Застарілий, менш відшліфований; не дуже інтуїтивно зрозумілий	Проста, але обмежена настройка
Продуктивність	Може працювати повільно через великі поштові скриньки "Вхідні" або спільні поштові скриньки	Лаги з величезними поштовими архівами; затримки при запуску	В цілому швидка, але з обмеженнями, пов'язаними з браузером
Підтримка платформи	Чудово працює в Windows; на Mac та мобільних пристроях можливості обмежені	В основному це десктопна версія; офіційної мобільної версії немає	В Інтернеті; непослідовна робота на мобільних пристроях
Можливість налаштування	Деякі опції, але обмежена гнучкість	Легко настраюється інтерфейс (розширення, теми)	Дуже обмежена настройка
Сортування та фільтрація	Потужний механізм правил, але втомлива настройка	Хороші фільтри, потрібна ручна настройка	Автоматичні категорії часто класифікуються неправильно
Використання в автономному режимі	Хороша Автономна підтримка	Повноцінне Автономне використання	Обмежений автономний режим у Chrome
Обробка небажаної пошти	Іноді трапляються помилкові спрацювання	В значній мірі залежить від ручного налаштування	Загалом ефективний, але не дозволяє вимкнути фільтри Google

3

Рисунок Д.3 – Слайд 3

Постановка завдання

Важливо розробити новий вебзастосунок поштової клієнт з сортуванням листів, оскільки люди отримують занадто багато електронної пошти, а важливі повідомлення часто втрачаються. Сучасний застосунок має автоматично впорядковувати електронні листи за темами, відправниками чи пріоритетами, заощаджуючи час користувачів та зменшуючи стрес. Багато сучасних поштових клієнтів перевантажені, повільні або недостатньо гнучкі. Нове рішення може запропонувати більш зрозумілий дизайн, більш високу продуктивність та інтелектуальні функції, такі як теги, фільтри та сортування, які допоможуть користувачам зосередитися на найважливішому.

4

Рисунок Д.4 – Слайд 4

Вибір технічних засобів розробки (мова програмування та фреймворк)

Критерій	Python + Django	JavaScript + React + Node.js
Мовна узгодженість	Лише для бекенда(Python); для інтерфейсу потрібен HTML / JS	JS з повним стеком (Інтерфейс React, серверна частина Node)
Надійність фреймворку	Django - це фреймворк, що працює від батарейок, з ORM, панеллю адміністратора, авторизацією і т. д.	Node + Express мінімальний; потрібні сторонні модулі
Продуктивність (необроблена швидкість)	Повільніше, ніж Node, у необроблених завданнях вводу-виводу	Швидка, керована подіями, неблокуюча Архітектура
Швидкість розробки	Високий рівень завдяки вбудованим інструментам, ORM і admin	Помірний рівень; потрібно більше налаштувань та бібліотек
Крива навчання	Простий для початківців завдяки високорівневим абстракціям	Крутіше з декількома інструментами / бібліотеками (React, Webpack і т. д.)
Безпека	Надійні налаштування за замовчуванням (CSRF, XSS, захист від введення SQL)	Безпека повинна бути налаштована вручну
Функції в режимі реального часу	Для WebSockets потрібні канали або сторонні інструменти.	Вбудована в Node.js (socket.io і т. д.)
Масштабованість	Добре для вертикального масштабування; асинхронний за допомогою Django + ASGI	Відмінно підходить для горизонтального масштабування та мікросервісів
Інтеграція з базами даних	Жорсткий ORM (Django ORM) з PostgreSQL, MySQL і т. д.	Використовує ORMs, такі як Sequelize або Prisma
Обробка зовнішнього інтерфейсу	Потрібен окремий інтерфейс або використовуються шаблони Django	React ефективний для динамічних інтерфейсів користувача

5

Рисунок Д.5 – Слайд 5

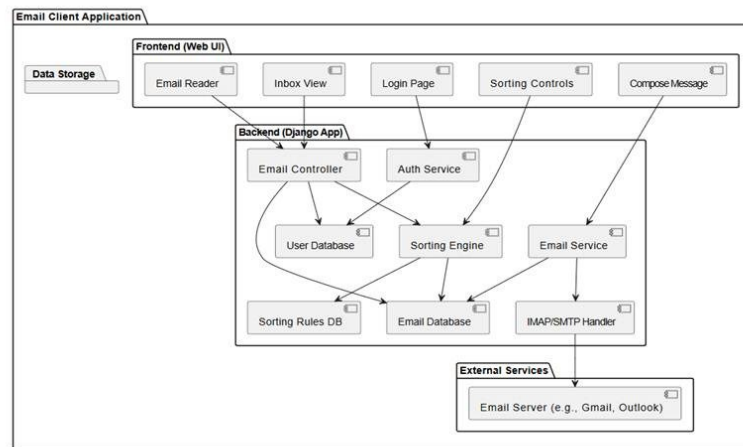
Вибір технічних засобів розробки (середовище розробки)

Критерій	PyCharm	Visual Studio Code
Підтримка платформи	Вбудована глибока інтеграція (моделі, подання, шаблони, маршрутизація URL-адрес)	Потрібні розширення
Час налаштування	Попередньо налаштована для проєктів на Django	Потрібно ручна настройка і модуль, що підключається
Інтелектуальний аналіз коду	Додатково (підказки по типу поля, дозвіл шаблону, самоаналіз моделі)	Хороший, але обмежений без великої кількості плагінів
Налагоджувач	Повна інтеграція з сервером Django, контекст шаблону, асинхронні подання	Основна налагодження на Python; додаткові налаштування для Django
Інтеграція з базами даних	Вбудований браузер баз даних, редактор SQL, інтеграція з ORM	Розширення доступні, але менш інтегровані
Налагодження шаблонів	Так-налагодження шаблонів Django з повним контекстом	Немає вбудованої підтримки
Інструменти тестування	Інтегрований Django test runner, користувацький інтерфейс для тестового покриття	Потрібні розширення або інтеграція з CLI
Навігація по панелі адміністратора	Розпізнає та пов'язує маршрути та моделі адміністратора Django	Немає вбудованого аналізу моделі адміністратора
Управління версіями	Інтегровані інструменти VCS, інтерактивне перебазування, вирішення конфліктів	Хороша підтримка Git за допомогою розширень
Якість коду та рефакторинг	Складний рефакторинг (наприклад, перейменування полів моделі в рамках проєкту)	Від базового до помірного рефакторингу

6

Рисунок Д.6 – Слайд 6

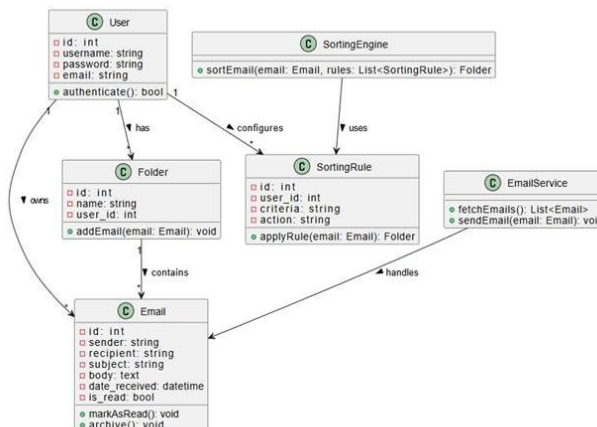
Структурна схема застосунку



7

Рисунок Д.7 – Слайд 7

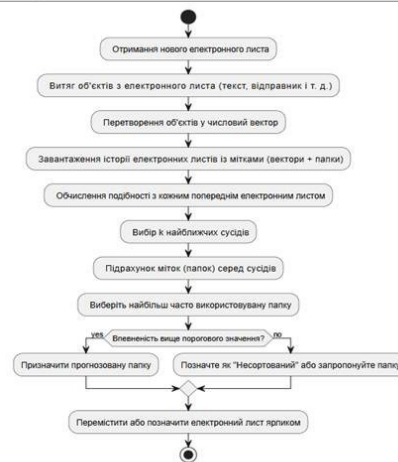
Проектування архітектури застосунку



8

Рисунок Д.8 – Слайд 8

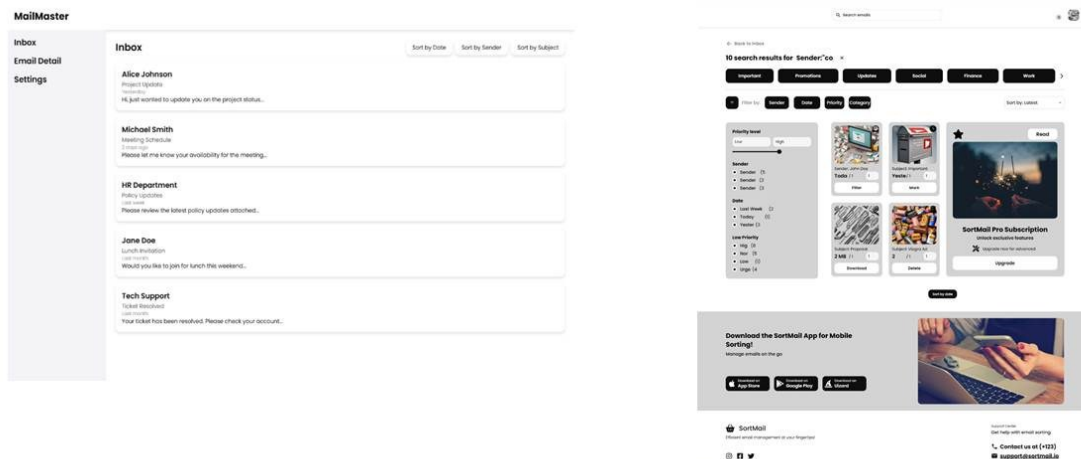
Метод сортування листів



9

Рисунок Д.9 – Слайд 9

Робота та тестування застосунку



10

Рисунок Д.10 – Слайд 10

Робота та тестування застосунку

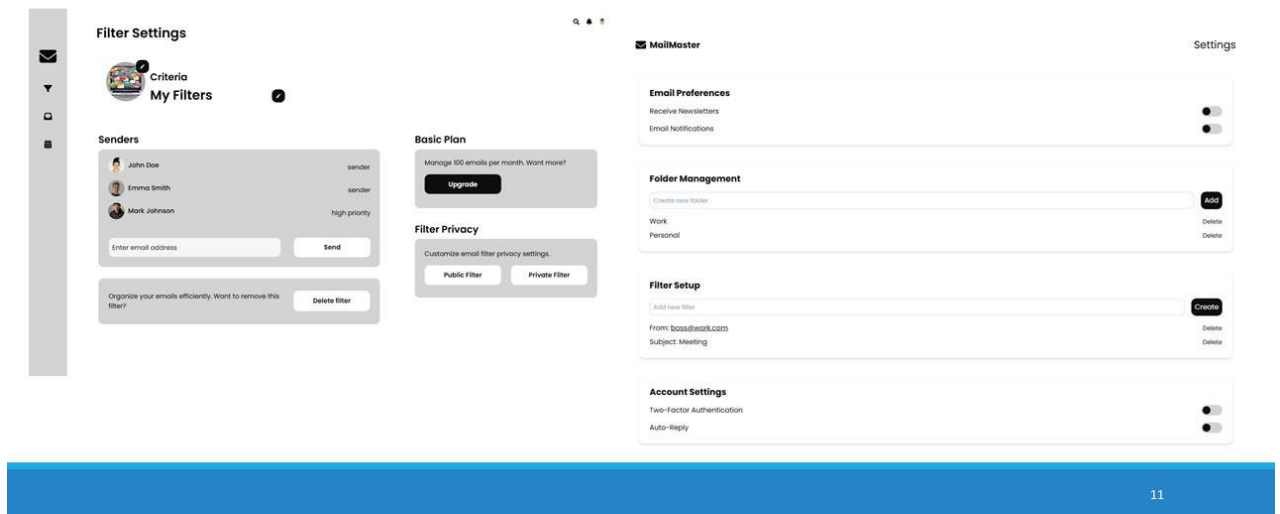


Рисунок Д.11 – Слайд 11

Висновки

В ході виконання дипломної кваліфікаційної роботи бакалавра було досягнуто головну мету – виконано програмну реалізації застосунку поштового клієнту з сортуванням вхідних листів. Для цього було виконано низку задач, серед яких:

- ❖ досліджено предметну область електронного листування та використання застосунків поштових клієнтів;
- ❖ досліджено існуючі рішення на ринку, виявлено їхні переваги та недоліки;
- ❖ на основі виявлених переваг та недоліків – сформовано перелік функціональних вимог до нового – якіснішого рішення;
- ❖ спроєктовано та розроблено новий застосунок – поштовий клієнт з сортуванням листів;
- ❖ виконано тестування нової розробки;
- ❖ надано ґрунтовні висновки за виконаною роботою.

Розроблений вебзастосунок був реалізований із використанням мови програмування Python, надбудови Django та середовища розробки PyCharm Free Educational License.

Виконано функціональне тестування застосунку за сценаріями використання застосунку.

Усі завдання дипломної кваліфікаційної роботи бакалавра повністю виконано.

Рисунок Д.12 – Слайд 12