ПОЛТАВСЬКИЙ УНІВЕРСИТЕТ ЕКОНОМІКИ І ТОРГІВЛІ
Навчально-науковий інститут денної освіти
Форма навчання денна
Кафедра комп'ютерних наук та інформаційних технологій

Допускається до захисту
Завідувач кафедри
_____ Олена ОЛЬХОВСЬКА
(підпис)

«___» _____ 2025 р.

КВАЛІФІКАЦІЙНА РОБОТА

на тему
РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ЧАТ-БОТУ ДЛЯ ГЕНЕРАЦІЇ РЕЗЮМЕ

зі спеціальності 122 Комп'ютерні науки
освітня програма «Комп'ютерні науки»
ступеня бакалавра

Виконавець роботи Гусаренко Михайло Олександрович
_____ «____»_____ 2025 р.
(підпис)

Науковий керівник к.пед.н., доцент, Кошова Оксана Петрівна
_____ «____»_____ 2025 р.
(підпис)

Рецензент

ПОЛТАВА 2025 р.

ЗАТВЕРДЖУЮ
Завідувач кафедри _____Олена ОЛЬХОВСЬКА
(підпис)
«___» _____ 2024 р.

ЗАВДАННЯ І КАЛЕНДАРНИЙ ГРАФІК
ВИКОНАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ

на тему «Розробка програмного забезпечення чат боту для генерації резюме**»**
зі спеціальності 122 Комп'ютерні науки
освітня програма «Комп'ютерні науки»
ступеня бакалавр
Прізвище, ім'я, по батькові Гусаренко Михайло Олександрович
Затверджена наказом ректора № 177-Н від «10» вересня 2024 р.
Термін подання студентом роботи «___»_____ 2025 р.
Вихідні дані до кваліфікаційної роботи: статті та документації з теми розробки чат ботів.

Зміст пояснювальної записки (перелік питань, які потрібно розробити)

ВСТУП
РОЗДІЛ 1. ПОСТАНОВКА ЗАДАЧІ
1.1. Мета та задачі дослідження
1.2. Визначення вимог до функціоналу чат-боту для генерації резюме
РОЗДІЛ 2. ОГЛЯД СУЧАСНОГО СТАНУ ПРОБЛЕМИ
2.1. Огляд існуючих інструментів для створення резюме
2.2. Порівняльний аналіз популярних рішень для генерації резюме
2.3. Переваги та недоліки існуючих сервісів для створення резюме
2.4. Технології та методи, що використовуються для автоматизації створення резюме
РОЗДІЛ 3. ТЕОРЕТИЧНА ЧАСТИНА
3.1. Опис обраних технологій і підходів для розробки чат-боту
3.2. Методологія розробки програмного забезпечення та побудова блок-схеми системи
3.3. Архітектура чат-боту для генерації резюме
РОЗДІЛ 4. ПРАКТИЧНА ЧАСТИНА
4.1. Розробка алгоритму збору та обробки даних від користувача
4.2. Побудова діаграми класів чат-боту для створення резюме
4.3. Опис та реалізація інтерфейсу користувача і функціональних можливостей чат-боту
ВИСНОВКИ
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ
ДОДАТОК А
Перелік графічного матеріалу: 1 аркуш блок-схем, 21 ілюстрація.

Консультанти розділів кваліфікаційної роботи

Розділ	Прізвище, ініціали, посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Постановка задачі	Кошова О.П.		
Інформаційний огляд	Кошова О.П.		
Теоретична частина	Кошова О.П.		
Практична частина	Кошова О.П.		

Календарний графік виконання кваліфікаційної роботи

Зміст роботи	Термін виконання	Фактичне виконання
Вступ		
Вивчення методичних рекомендацій та стандартів та звіт керівнику		
Постановка задачі		
Інформаційний огляд джерел бібліотек та інтернету		
Теоретична частина		
Практична частина		
Закінчення оформлення		
Доповідь студента на кафедрі		
Доробка (за необхідністю), рецензування		

Дата видачі завдання «__» _____ 2024 р.

Здобувач вищої освіти _____ Гусаренко Михайло Олександрович
(підпис)

Науковий керівник _____ к.пед.н., доц. Кошова О.П.
(підпис) (науковий ступінь, вчене звання, ініціали та прізвище)

Результати захисту кваліфікаційної роботи

Кваліфікаційна робота оцінена на _____
(балів, оцінка за національною шкалою, оцінка за ECTS)

Протокол засідання ЕК № ____ від «____» _____ 2025 р.

Секретар ЕК_____ _____
(підпис) (ініціали та прізвище)

Затверджую
Зав. кафедрою _____
к.ф.-м.н. Олена ОЛЬХОВСЬКА
«____» _____ 2024 р.

Погоджено
Науковий керівник _____
к.пед.н., Оксана КОШОВА
«____» _____ 2024 р.

План

дипломного проекту з фаху
спеціальності 122 Комп'ютерні науки
освітня програма 122 Комп'ютерні науки
на тему «Розробка програмного забезпечення чат боту для генерації резюме»
Прізвище, ім'я, по батькові Гусаренко Михайло Олександрович

ВСТУП
РОЗДІЛ 1. ПОСТАНОВКА ЗАДАЧІ
1.1. Мета та задачі дослідження
1.2. Визначення вимог до функціоналу чат-боту для генерації резюме
РОЗДІЛ 2. ОГЛЯД СУЧАСНОГО СТАНУ ПРОБЛЕМИ
2.1. Огляд існуючих інструментів для створення резюме
2.2. Порівняльний аналіз популярних рішень для генерації резюме
2.3. Переваги та недоліки існуючих сервісів для створення резюме
2.4. Технології та методи, що використовуються для автоматизації створення резюме
РОЗДІЛ 3. ТЕОРЕТИЧНА ЧАСТИНА
3.1. Опис обраних технологій і підходів для розробки чат-боту
3.2. Методологія розробки програмного забезпечення та побудова блок-схеми системи
3.3. Архітектура чат-боту для генерації резюме
РОЗДІЛ 4. ПРАКТИЧНА ЧАСТИНА
4.1. Розробка алгоритму збору та обробки даних від користувача
4.2. Побудова діаграми класів чат-боту для створення резюме
4.3. Опис та реалізація інтерфейсу користувача і функціональних можливостей чат-боту
ВИСНОВКИ
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ
ДОДАТОК А

Здобувач вищої освіти _____ М. О. Гусаренко
(підпис)
«____» _____ 2024 р.

РЕФЕРАТ

Записка: 86 сторінок, 21 рисунок, 1 додаток, 18 літературних джерел.

Мета дипломної роботи є розробка програмного забезпечення чат боту для генерації резюме.

Об'єкт розробки – чат бот для генерації резюме у форматі PDF.

Методи дослідження – методи проектування телеграм-боту, створеного на основі `aiogram` з використанням вебхуків для обробки API запитів, підтримкою кількох мов (i18n) та можливістю генерації PDF-резюме на основі HTML-шаблонів. Цей бот допомагає користувачам швидко створювати якісні резюме у форматі PDF за допомогою інтерактивних команд у Telegram.

Визначено ключові вимоги до розробки програмного забезпечення чат боту для генерації резюме у форматі PDF; проведено аналіз переваг та обмежень аналогічних застосунків для автоматичної генерації резюме ; проаналізовано та описано підходи та інструменти, використані при розробці ПЗ чат боту; описано алгоритм та методолію створення розробленого чат боту; розроблено мобільний інтерфейс для взаємодії користувача з матеріалом та створено інструкцію користувача для роботи та встановлення і налаштування чат боту; представлено блок схему, use-case діаграму та діаграму класів, що ілюструє особливості програмного коду розробленого ПЗ; розроблено та протестовано чат бот для автоматичної генерації резюме у форматі PDF.

ЗМІСТ

ВСТУП

В умовах сучасного ринку праці успішне працевлаштування значною мірою залежить від якості та змісту резюме, яке представляє потенційного кандидата роботодавцю. Проте процес написання ефективного резюме може стати справжнім викликом для багатьох людей, особливо для тих, хто не має досвіду у формулюванні своїх професійних досягнень і навичок у структурованій та привабливій формі. У зв'язку з цим зростає попит на інструменти, які спрощують процес створення резюме та забезпечують його професійний вигляд.

Одним із перспективних рішень для вирішення цієї задачі є створення чат-боту, який може автоматизувати процес генерації резюме. Чат-боти активно використовуються в різних галузях завдяки можливості інтерактивної взаємодії з користувачем, що робить їх зручними та доступними для широкого кола користувачів. Застосування чат-боту для створення резюме дозволяє не лише автоматизувати процес збору та організації інформації, але й адаптувати формат резюме під вимоги користувача, підвищуючи ефективність цього процесу.

Метою даної роботи є розробка програмного забезпечення чат боту для генерації резюме, який зможе допомогти користувачам створити якісне резюме, відповідно до їхніх потреб та вимог ринку праці. Для досягнення цієї мети необхідно вирішити ряд задач, зокрема провести аналіз існуючих рішень, розробити архітектуру чат-боту, вибрати технології та реалізувати функціональні можливості для збору і обробки інформації від користувача.

Об'єктом дослідження чат бот для генерації резюме у форматі PDF, а *предметом* – розробка програмного забезпечення чат боту для генерації резюме у форматі PDF. Для досягнення поставлених цілей у роботі буде використано методи збору та аналізу даних, а також програмні засоби для розробки програмного забезпечення.

У роботі будуть розглянуті основні принципи та підходи до розробки чат-ботів, проаналізовані технології, що можуть бути застосовані для створення інтерфейсу та логіки чат-боту.

Перелік методів включає проектування телеграм-боту, створеного на основі `aiogram` з використанням вебхуків для обробки API запитів, підтримкою кількох мов (i18n) та можливістю генерації PDF-резюме на основі HTML-шаблонів. Для роботи над ПЗ чат боту було використано середовище розробки PyCharm. Цей бот допомагає користувачам швидко створювати якісні резюме у форматі PDF за допомогою інтерактивних команд у Telegram.

Пояснювальна записка включає чотири основних розділи: постановка задачі, огляд сучасного стану проблеми, теоретична та практична частини. Структура роботи орієнтована на логічне представлення матеріалу та всебічне розкриття теми дослідження.

РОЗДІЛ 1. ПОСТАНОВКА ЗАДАЧІ

1.1. Мета та задачі дослідження

Метою нашої дипломної роботи є розробка програмного забезпечення у вигляді чат-боту для автоматизованого створення резюме у форматі PDF, який допоможе користувачам швидко та ефективно підготувати професійне резюме без необхідності поглиблених знань у складанні документів для працевлаштування. Даний чат-бот має забезпечити користувачеві зручний інтерфейс для введення особистих даних, вибір шаблонів резюме, а також можливість генерувати документ, що відповідатиме сучасним стандартам і вимогам ринку праці.

Для досягнення поставленої мети необхідно вирішити такі задачі:

- Провести аналіз існуючих рішень для автоматизації створення резюме, виявити їх переваги та недоліки з точки зору зручності для користувача, функціональності та ефективності.
- Визначити вимоги до функціоналу чат-боту, включаючи основні та додаткові функції, що забезпечать зручність використання та адаптивність до різних потреб користувачів.
- Розробити архітектуру чат-боту, враховуючи можливість його масштабування та інтеграції з різними платформами для обміну повідомленнями, такими як Telegram, Facebook Messenger тощо.
- Вибрати найбільш відповідні інструменти та технології для реалізації чат-боту, зокрема платформи для розробки чат-ботів, мови програмування та бібліотеки для обробки природної мови (NLP).
- Розробити інтерфейс та функціональні можливості чат-боту для збору, обробки та структурування даних, наданих користувачем.
- Реалізувати функціонал автоматичної генерації резюме в форматі PDF на основі введених даних.

– Провести тестування чат-боту для перевірки його ефективності та зручності використання, а також проаналізувати результати тестування для вдосконалення системи.

Ці задачі забезпечать поетапну розробку, від аналізу до реалізації, що дозволить створити ефективне та зручне програмне рішення для автоматизованого створення резюме.

1.2. Визначення вимог до функціоналу чат-боту для генерації резюме

Для ефективної реалізації чат-боту, що автоматизує процес створення резюме, необхідно визначити основні вимоги до його функціоналу, враховуючи як базові, так і додаткові можливості, що підвищать зручність використання та персоналізацію документа. Вимоги до функціоналу чат-боту можна поділити на функціональні та нефункціональні.

Система чат-боту для створення резюме повинна відповідати визначеним функціональним і нефункціональним вимогам. Зокрема, чат-бот має забезпечувати можливість запиту та отримання даних, необхідних для складання резюме. До таких даних належать особисті відомості користувача, включаючи ім’я та контактну інформацію, інформація про освіту, професійний досвід, основні навички, досягнення та додаткові дані, такі як курси, сертифікати, володіння іноземними мовами або рекомендації. Крім того, система повинна підтримувати динамічний вибір шаблонів резюме, які враховують стилістичні особливості, вимоги різних галузей та вакансій, що дозволяє користувачам адаптувати кінцевий документ до конкретних потреб.

Особливістю функціоналу платформи є можливість адаптації структури резюме відповідно до специфіки вакансії або побажань користувача. Це включає зміну порядку розділів, акцентування уваги на певних досягненнях або модифікацію інших аспектів структури. Чат-бот також повинен автоматично

генерувати документ на основі отриманих даних та обраного шаблону, з можливістю експорту готового резюме у форматах PDF або DOCX для зручного використання. Інтерактивна допомога є невід'ємною частиною функціональності, адже користувачеві надаються рекомендації та підказки під час заповнення кожного розділу, що сприяє кращому розумінню та коректному введенню інформації. Крім того, передбачена можливість збереження проміжних даних, що дозволяє користувачеві продовжити роботу з інформацією у будь-який зручний момент без необхідності повторного заповнення.

Щодо нефункціональних вимог, інтерфейс чат-боту має бути інтуїтивно зрозумілим і доступним для широкого кола користувачів, включаючи осіб з мінімальними комп'ютерними навичками. Безпека даних є важливим аспектом системи, тому забезпечується конфіденційність і захищене зберігання персональної інформації, з обмеженням доступу до неї. Високі вимоги до продуктивності системи передбачають швидку обробку запитів і оперативний зворотний зв'язок із користувачем. Таким чином, система чат-боту має бути комплексним і гнучким інструментом для створення професійних резюме, враховуючи сучасні потреби користувачів.

Підсумовуючи вище викладене, система чат-боту для створення резюме повинна відповідати наступним функціональним вимогам:

1. Збір інформації від користувача.
2. Динамічний вибір шаблонів.
3. Адаптація резюме.
4. Автоматична генерація документа.
5. Інтерактивна допомога.
6. Збереження та редагування.

Система чат-боту повинна відповідати таким нефункціональним вимогам:

1. Зручність використання.
2. Безпека даних.

3. Швидкість роботи.
4. Масштабованість.

Визначення цих вимог є важливим етапом у процесі розробки, оскільки вони задають основу для розробки ефективного і зручного у використанні чат-боту, здатного вирішити задачу автоматизованого створення резюме та відповідати потребам користувачів.

РОЗДІЛ 2. ОГЛЯД СУЧАСНОГО СТАНУ ПРОБЛЕМИ

2.1. Огляд існуючих інструментів для створення резюме

На сьогодні на ринку існує велика кількість платформ та інструментів для створення резюме, що мають різний функціонал, форматування та можливості персоналізації. Більшість таких сервісів орієнтовані на користувачів, які прагнуть створити професійне резюме з мінімальними зусиллями та без необхідності спеціальних знань. До популярних інструментів для створення резюме відносяться онлайн-платформи, такі як Canva, Zety, Resume.com, Novoresume та інші.

Основні функції, які зазвичай пропонують подібні платформи:

- Шаблони резюме - більшість платформ надають користувачам можливість вибору з ряду шаблонів, що відповідають різним стилям та професійним галузям. Шаблони можуть бути простими або більш креативними, орієнтованими на конкретні професії.
- Редагування та налаштування контенту - інструменти для редагування дозволяють користувачам вносити дані про освіту, досвід роботи, навички, сертифікати тощо, а також додавати інші розділи за потреби.
- Форматування та стиль - користувачі можуть змінювати шрифти, кольори, розташування елементів, що допомагає адаптувати резюме до особистих побажань або корпоративних вимог.
- Експорт у різні формати: більшість платформ надають можливість експорту готового резюме у форматах PDF або DOCX, що дозволяє зручно надсилати резюме електронною поштою чи завантажувати на сайти з пошуку роботи.
- Підказки та приклади - деякі сервіси пропонують рекомендації щодо написання різних розділів резюме, що допомагає користувачам коректно описати свій досвід та досягнення.

Огляд кожної з цих платформ показує, що їх основна перевага полягає в простоті використання та готових шаблонах, які дозволяють користувачам створювати привабливі резюме без дизайнерських або редакторських навичок. Проте більшість існуючих рішень не мають достатньої інтерактивності, що ускладнює процес індивідуальної адаптації резюме до різних вакансій.

Іншим типом рішень для автоматизації створення резюме є чат-боти, які наразі лише починають набувати популярності. Чат-боти надають більше можливостей для інтерактивної взаємодії з користувачем, допомагаючи структурувати інформацію в зручному для користувача форматі через діалог. На відміну від класичних інструментів, вони орієнтовані на поступовий збір даних та можуть адаптувати шаблони резюме на основі отриманої інформації.

Цей огляд демонструє потребу в удосконаленні існуючих рішень, особливо в напрямку інтерактивності та персоналізації, що може бути досягнуто за допомогою чат-боту для генерації резюме, який автоматизує процес збору, обробки та організації даних користувача для створення оптимізованого професійного документа.

Сучасний ринок пропонує безліч інструментів для автоматизованого створення резюме, які спрощують процес підготовки професійного документа. Нижче наведено огляд деяких популярних сервісів:

1. Resume.io - це онлайн-платформа, що надає користувачам можливість створювати професійні резюме та супровідні листи за допомогою зручного конструктора. Сервіс пропонує понад 30 настроюваних шаблонів, які відповідають різним галузям та рівням кар’єри. Resume.io також включає функції перевірки граматики та можливість експорту резюме у форматах PDF, DOCX або TXT.

Крім того, платформа використовує технології штучного інтелекту для автоматизованого написання тексту, що полегшує процес створення резюме для користувачів без досвіду. Це особливо корисно для тих, хто шукає роботу за кордоном, оскільки сервіс підтримує багатомовність та локалізацію.

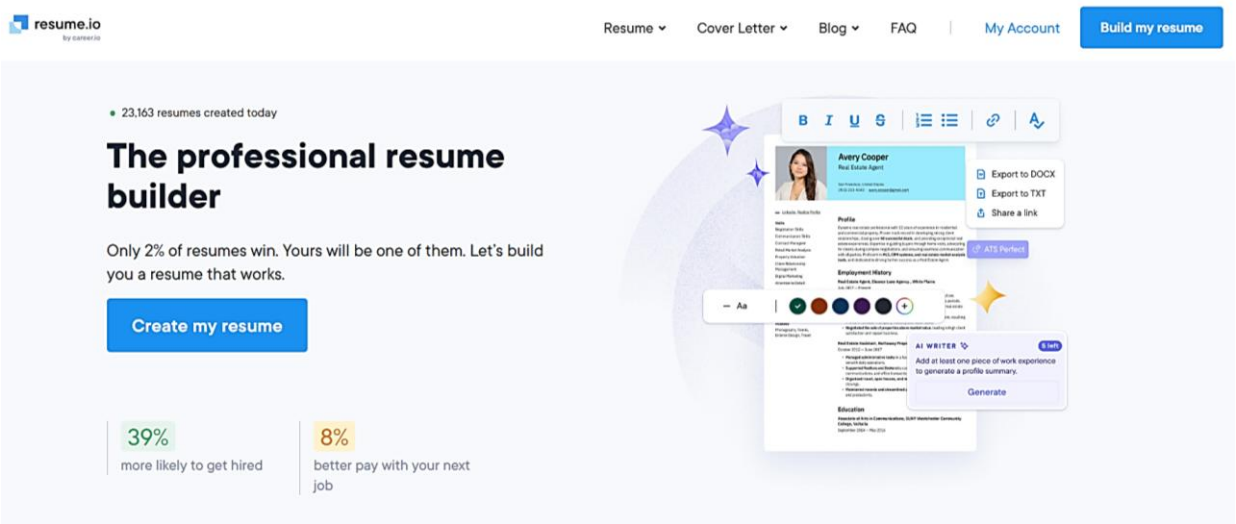

Рисунок 2.1. Онлайн-платформа Resume.io

Платформа пропонує широкий асортимент шаблони супровідних листів для різних посад і дозволяє зберігати файли в різних форматах (PDF, DOCX або TXT), відповідно до ваших потреб. Є функція голосового відтворення, що дозволяє вам говорити, а не вводити введені дані. Крім того, вбудований помічник AI генерує корисні підказки для кожного аспекту вашого резюме, включаючи вашу біографію та доповнення до навичок, опис роботи тощо.

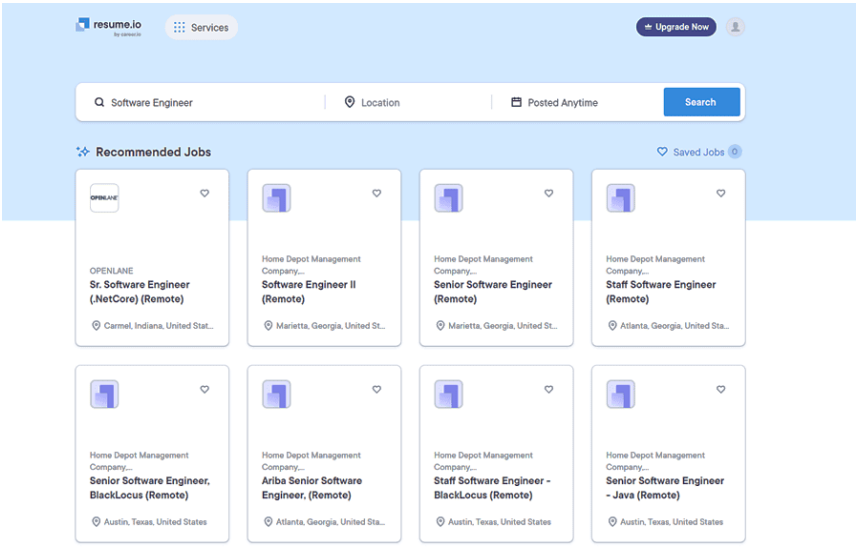

Рисунок 2.2. Інтерфейс онлайн-платформи Resume.io

Resume.io також пропонує функції адаптації резюме під системи

відстеження кандидатів (ATS), що підвищує шанси успішного проходження автоматичних фільтрів при подачі заявок до великих компаній. Платформа має інтуїтивно зрозумілий інтерфейс, що робить процес створення резюме простим та зручним для користувачів. Відгуки користувачів свідчать про високу якість сервісу та зручність його використання [9].

2. Rezi є онлайн-платформою, яка використовує технології штучного інтелекту для автоматизації процесу створення резюме. Основною перевагою сервісу є можливість оптимізації резюме для систем відстеження кандидатів (ATS), що значно підвищує шанси користувачів на успішне проходження автоматичних фільтрів при подачі заявки. Платформа автоматично заповнює дані резюме, надає рекомендації щодо його змісту та забезпечує відповідність ключовим словам і вимогам вакансій [14].

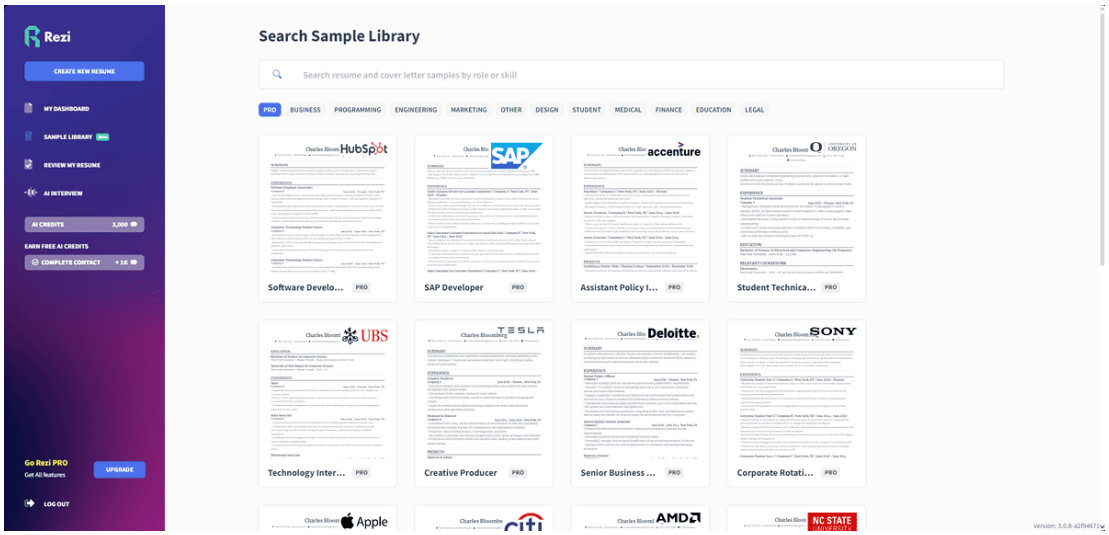

Рисунок 2.3. Онлайн платформа Rezi

Серед основних переваг Rezi варто виділити його здатність автоматизувати процес створення резюме завдяки інтеграції штучного інтелекту. Це значно полегшує завдання користувачів, зокрема тих, які мають недостатній досвід у складанні резюме. Додатковою перевагою є можливість отримання персоналізованих рекомендацій, що сприяє підвищенню якості кінцевого документа. Платформа також забезпечує оптимізацію для ATS, що є критично важливим для здобувачів, які прагнуть подати заявку до великих

компаній.

Rezi аналізує інформацію у вашому резюме в режимі реального часу, а потім пропонує конкретні ключові слова та фрази, які допоможуть вам створити CV, оптимізоване для ATS [14].

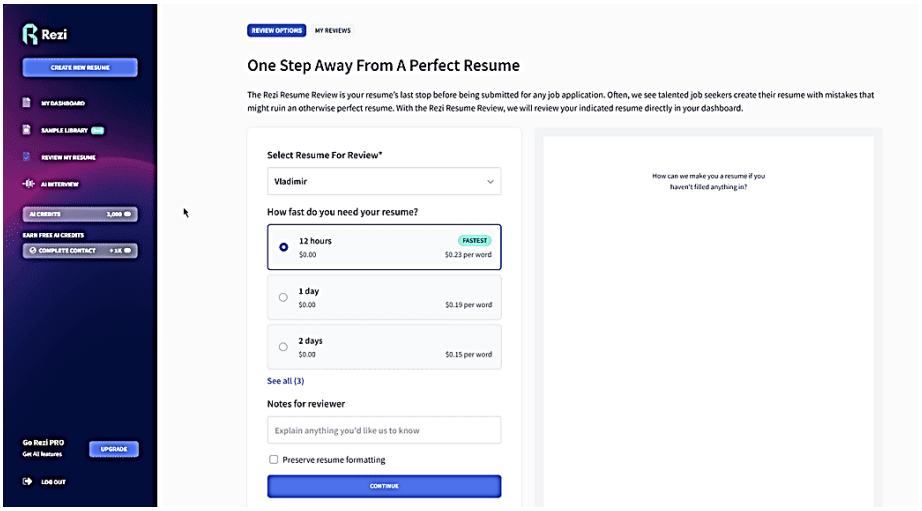

Рисунок 2.4. Функціонал онлайн платформи Rezi

Нарешті, Rezi пропонує функцію перевірки блокчейну додати цифровий підпис до свого резюме, що робить його більш достовірним.

Проте, незважаючи на перелічені переваги, Rezi має й певні недоліки. Зокрема, користувачі відзначають обмежену кількість шаблонів, доступних для налаштування, що може не задовольняти потреби тих, хто прагне унікального оформлення резюме. Додаткові преміум-функції, такі як розширені можливості налаштування та доступ до додаткових шаблонів, доступні лише за умови платної підписки, що може бути недоліком для окремих користувачів. Крім того, як і більшість онлайн-сервісів, Rezi залежить від стабільного інтернет-з'єднання, що може створювати труднощі для користувачів з обмеженим доступом до мережі.

Таким чином, Rezi є ефективним інструментом для створення професійних резюме, який поєднує автоматизацію, рекомендації та оптимізацію для сучасних вимог ринку праці. Проте його ефективність залежить від індивідуальних потреб користувачів та доступу до преміум-функцій [1].

3. Kickresume - це онлайн-платформа для створення резюме та супровідних листів, яка використовує штучний інтелект на базі моделі GPT-4 для автоматичної генерації тексту. Сервіс пропонує понад 50 настроюваних шаблонів, розроблених професійними типографами та рекрутерами, що дозволяє користувачам обрати дизайн, який найкраще відповідає їхнім потребам [14].

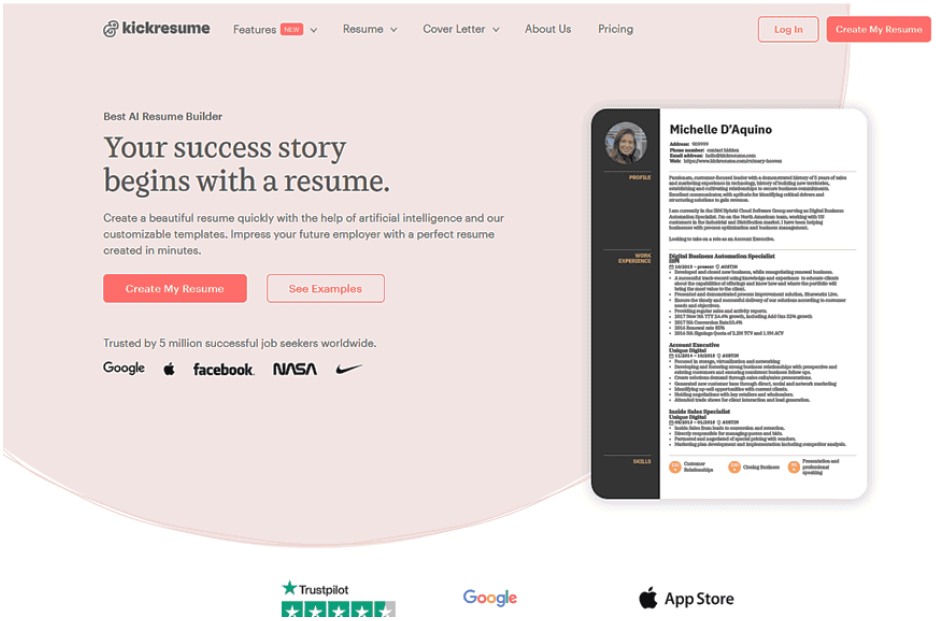

Рисунок 2.5. Онлайн платформа Kickresume

Окрім створення резюме, Kickresume надає можливість перетворити резюме на персональну веб-сторінку за допомогою вбудованого конструктора сайтів. Це дозволяє користувачам створювати професійні онлайн-профілі, які можуть бути легко доступні потенційним роботодавцям.

Для спрощення процесу написання, платформа пропонує понад 20 000 попередньо написаних фраз та пропозицій, які можна використовувати при складанні резюме та супровідних листів. Це особливо корисно для тих, хто має труднощі з формулюванням власних досягнень або обов'язків.

Додатково, Kickresume включає функцію перевірки резюме, яка аналізує документ та надає рекомендації щодо покращення його змісту та оформлення. Це допомагає користувачам створювати більш ефективні та професійні резюме, що підвищує їхні шанси на успішне працевлаштування.

4. Canva є багатофункціональним графічним редактором, що пропонує зручний інструмент для створення стильних і професійних резюме. Цей сервіс має широкий спектр можливостей, зокрема доступ до великої кількості дизайнерських шаблонів, які користувачі можуть налаштовувати відповідно до своїх потреб. Редактор дозволяє змінювати кольори, шрифти, макети та інші графічні елементи, що забезпечує високу гнучкість у створенні індивідуального дизайну. Інтерфейс Canva інтуїтивно зрозумілий і простий у використанні, що робить його доступним навіть для людей без попереднього досвіду роботи з графічними редакторами [4].

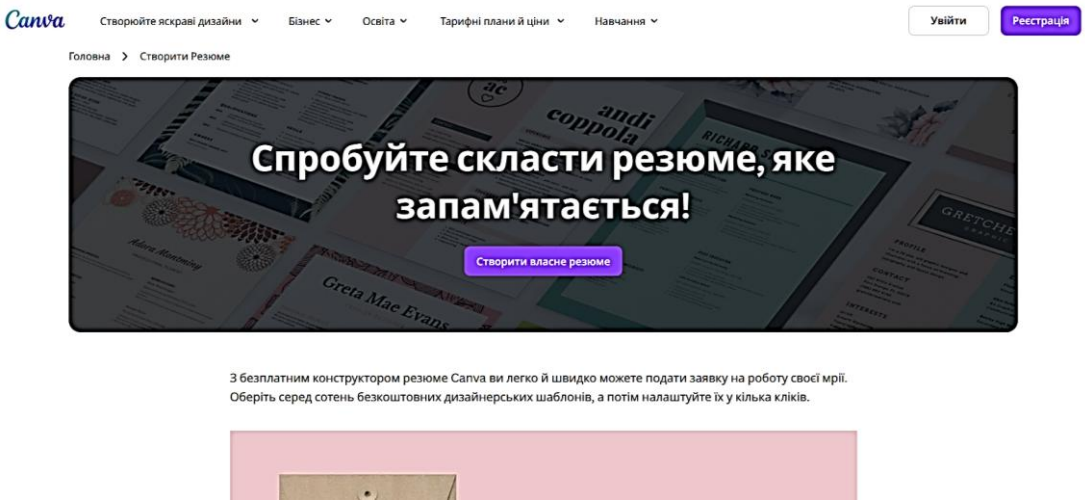

Рисунок 2.6. Графічний редактор Canva

Однією з основних переваг Canva є її орієнтація на візуальну привабливість. Сервіс дозволяє створювати естетично приємні документи, які виділяються на фоні стандартних текстових резюме. Крім того, Canva надає можливість збереження файлів у популярних форматах, таких як PDF, що полегшує подальше використання створеного контенту. Завдяки безкоштовному доступу до основних функцій, Canva є доступною для широкого кола користувачів, зокрема студентів та професіоналів-початківців.

Проте, незважаючи на свої переваги, Canva має певні недоліки. Головним обмеженням є те, що цей сервіс не спеціалізується виключно на створенні резюме, тому деякі функції, важливі для оптимізації резюме під конкретні

вакансії або автоматизовані системи відстеження кандидатів (ATS), відсутні. Це може знижувати ефективність резюме, створених у Canva, у контексті сучасного ринку праці. Додатково, деякі розширені можливості сервісу, як-от доступ до преміум-шаблонів або ексклюзивних графічних елементів, потребують оформлення платної підписки.

Таким чином, Canva є корисним інструментом для створення стильних і привабливих резюме, особливо для користувачів, які прагнуть підкреслити креативність і візуальний стиль. Однак для тих, хто шукає більш спеціалізовані рішення для створення професійного резюме, адаптованого до вимог ATS, Canva може бути менш ефективним вибором.

5. SweetCV є онлайн-платформою для створення сучасних резюме, яка забезпечує швидкий та зручний процес для користувачів різного рівня підготовки. Основною перевагою сервісу є доступ до різноманітних шаблонів, які можуть бути адаптовані до потреб користувачів шляхом зміни дизайну, структури та стилістичних елементів. SweetCV дозволяє створювати резюме, які можна завантажувати у форматі PDF або перетворювати на веб-резюме, доступне через персональний посилання. Така функціональність відкриває нові можливості для представлення резюме в онлайн-просторі, що є актуальним у сучасному цифровому середовищі [13].

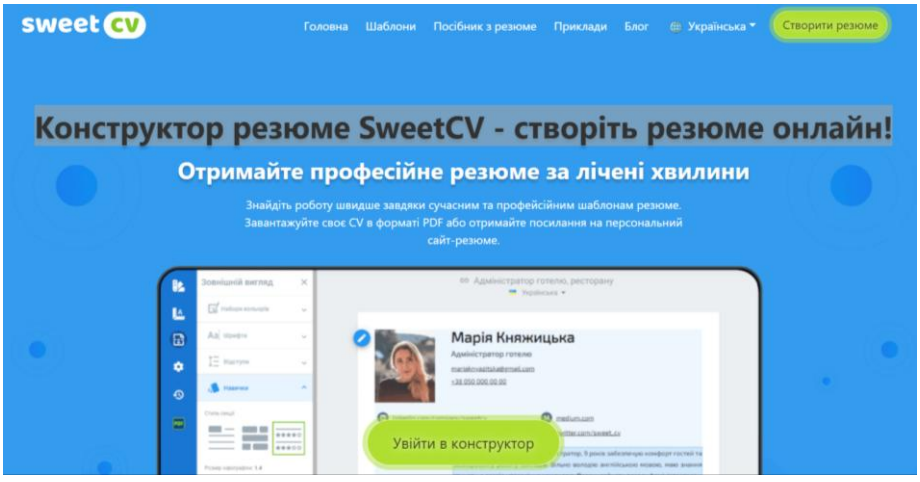

Рисунок 2.7. Онлайн платформа SweetCV

Однією з ключових переваг SweetCV є простота використання та доступність. Інтерфейс платформи інтуїтивно зрозумілий, що дозволяє швидко створювати професійні резюме навіть користувачам з мінімальним досвідом. Водночас можливість створення веб-резюме додає додаткову цінність, дозволяючи кандидатам поширювати свої профілі через інтернет. Окрім того, сервіс надає варіанти налаштування дизайну, що дозволяє користувачам створювати унікальні та індивідуальні документи.

Проте SweetCV має і певні обмеження. Наприклад, сервіс фокусується переважно на візуальній складовій, що може робити його менш придатним для створення резюме, адаптованих до вимог автоматизованих систем відстеження кандидатів (ATS). Крім того, деякі розширені можливості, як-от доступ до ексклюзивних шаблонів або функцій, потребують платної підписки, що може бути недоліком для користувачів з обмеженим бюджетом. Ще одним недоліком є відсутність функції автоматичного генерування тексту чи підказок, що може ускладнювати процес для тих, хто не має досвіду у написанні резюме.

Загалом SweetCV є ефективним інструментом для створення стильних і сучасних резюме, який підходить для користувачів, які шукають зручне рішення з можливістю створення веб-версій. Однак його обмеження в адаптації до ATS та залежність від платних функцій можуть вплинути на вибір сервісу серед професіоналів, які орієнтовані на більш комплексні рішення.

Існує іще значна кількість онлайн-сервісів для створення резюме, кожен з яких має свої унікальні переваги, функціональність та орієнтацію на різні категорії користувачів. Нижче наведемо опис іще декількох популярних платформ, які можуть бути корисними для створення професійного резюме.

Zety є онлайн-конструктором резюме, який відомий своєю гнучкістю та простотою у використанні. Сервіс пропонує безліч шаблонів, адаптованих під різні професійні галузі. Крім того, Zety надає рекомендації щодо оптимізації змісту резюме, дозволяючи користувачам створювати документи, які відповідають сучасним вимогам роботодавців. Платформа також забезпечує

можливість створення супровідних листів, що спрощує процес подання заявок.

Novoresume орієнтований на створення як базових, так і розширених резюме, які відповідають високим професійним стандартам. Його ключовою особливістю є інтуїтивно зрозумілий інтерфейс, що дозволяє швидко адаптувати структуру документа до специфіки вакансії. Платформа також пропонує преміум-шаблони, які підходять для різних рівнів кар'єри.

VisualCV є платформою, яка акцентує увагу на візуальній складовій резюме. Вона дозволяє створювати резюме з індивідуальним дизайном, що виділяються серед стандартних шаблонів. VisualCV також пропонує можливість експорту документів у різних форматах та створення онлайн-профілів, які можуть бути легко доступними для роботодавців.

Resume Genius надає широкий вибір шаблонів і допоміжні інструменти для створення резюме. Однією з ключових переваг цього сервісу є інтегровані текстові підказки для кожного розділу, що спрощує процес написання навіть для користувачів без попереднього досвіду. Resume Genius також забезпечує відповідність документів сучасним вимогам роботодавців.

Enhancv є платформою, що спеціалізується на створенні креативних резюме. Цей сервіс дозволяє користувачам виділяти свої досягнення та навички за допомогою унікальних дизайнерських елементів. Enhancv пропонує широкий вибір шаблонів, які підходять для творчих професій і спеціалістів, які прагнуть підкреслити свою індивідуальність.

Resumeble забезпечує як стандартні інструменти для створення резюме, так і додаткові послуги з професійного написання документів. Платформа також пропонує аналіз створених резюме на відповідність вимогам вакансій, що робить її ефективним інструментом для пошуку роботи.

Creddle є простим і сучасним інструментом для створення резюме, який дозволяє інтегрувати дані з LinkedIn, що значно спрощує процес заповнення. Ця платформа орієнтована на користувачів, які потребують швидкого та зручного рішення для створення професійних документів.

CakeResume пропонує унікальну функціональність, що дозволяє створювати резюме у форматі портфоліо. Цей підхід особливо підходить для представників творчих професій, таких як дизайнери, фотографи або художники. Платформа підтримує індивідуальні налаштування шаблонів, що робить її привабливою для креативних користувачів.

MyPerfectResume забезпечує зручний редактор резюме, який допомагає користувачам на кожному етапі написання документа. Сервіс містить широкий вибір шаблонів та функціональність для створення професійних супровідних листів, що робить його універсальним інструментом для пошуку роботи.

TopResume спеціалізується на професійному редагуванні резюме. Платформа надає послуги з аналізу існуючих документів та їх оптимізації відповідно до вимог конкретних вакансій. Цей сервіс орієнтований на користувачів, які прагнуть отримати конкурентоспроможний документ, створений за участю експертів.

Ці інструменти значно спрощують процес створення резюме, надаючи користувачам можливість швидко та ефективно підготувати професійний документ, що відповідає сучасним стандартам та вимогам ринку праці.

Мобільні застосунки та чат-боти для створення резюме стають все більш популярними завдяки їхній зручності та доступності. Ці інструменти дозволяють користувачам створювати резюме безпосередньо зі смартфона або через інтеграцію з месенджерами. Мобільні застосунки, такі як Canva, пропонують адаптовану для мобільних пристроїв версію з доступом до великої кількості дизайнерських шаблонів. Інші програми, такі як Resume Builder by PathSource та Resume Star, забезпечують покроковий процес створення резюме з можливістю експорту готового документа у форматі PDF або надсилання на електронну пошту. Аналогічно, CV Engineer і LinkedIn пропонують додаткові можливості, такі як інтеграція з професійними профілями та збереження декількох шаблонів резюме для різних вакансій.

Чат-боти також стають важливим інструментом у цьому сегменті.

Наприклад, Rezi AI Chatbot використовує штучний інтелект для створення резюме, адаптованого до систем відстеження кандидатів (ATS).

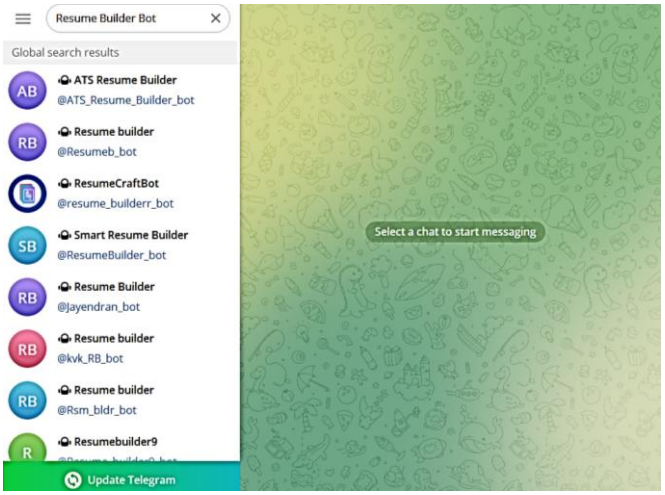

Рисунок 2.8. Приклади чат ботів для генерації резюме

Інші боти, такі як Resume Builder Bot у Telegram, дозволяють користувачам заповнювати базові дані та автоматично генерувати резюме у вигляді PDF. Деякі чат-боти, зокрема у системах, таких як MyPerfectResume або Kickresume, інтегрують текстові підказки та шаблони для полегшення процесу написання резюме. GPT-4 базовані боти, що працюють у месенджерах, як-от WhatsApp або Telegram, використовують можливості штучного інтелекту для автоматичного генерування тексту на основі відповідей користувача.

2.2. Порівняльний аналіз популярних рішень для генерації резюме

Сучасні платформи для створення резюме значно відрізняються за функціональністю, інтерфейсом та адаптацією під потреби користувачів. Щоб визначити переваги та обмеження різних рішень для генерації резюме, проведемо порівняльний аналіз найбільш популярних платформ: Resume.io, Rezi, Kickresume, Canva та SweetCV. Основними критеріями аналізу є

доступність шаблонів, зручність використання, підтримка штучного інтелекту, можливість експорту резюме та адаптація під системи відстеження кандидатів (ATS) [2].

Платформа	**Кількість шаблонів**	**Зручність використання**	**Підтримка AI**	**Формати експорту**	**Адаптація під ATS**
Resume.io	30+	Висока	Ні	PDF, DOCX, TXT	Так
Rezi	Обмежена	Висока	Так	PDF	Так
Kickresume	50+	Висока	Так (GPT-4)	PDF, DOCX, веб-сторінка	Часткова
Canva	1000+	Висока	Ні	PDF	Ні
SweetCV	Обмежена	Середня	Ні	PDF, веб-сторінка	Часткова

Критерії порівняння платформ для створення резюме включають кількість доступних шаблонів, зручність використання, підтримку штучного інтелекту, формати експорту та адаптацію під системи відстеження кандидатів (ATS). Наприклад, такі платформи, як Canva і Kickresume, пропонують великий вибір шаблонів, що забезпечує широкий спектр стилів для різних потреб. Однак надмірна кількість варіантів іноді ускладнює вибір для користувача.

З точки зору зручності використання, Resume.io, Rezi і Kickresume мають інтуїтивно зрозумілі інтерфейси, що робить процес створення резюме простим. У той час SweetCV вимагає більше часу на налаштування, а Canva, хоча й має значну кількість шаблонів, не є спеціалізованим інструментом для створення резюме, що може ускладнити роботу.

Щодо підтримки штучного інтелекту, Rezi і Kickresume виділяються своїми можливостями. Rezi оптимізує резюме для ATS, тоді як Kickresume застосовує GPT-4 для автоматичного генерування тексту, полегшуючи написання ключових розділів резюме.

Формати експорту також є важливим фактором. Усі платформи підтримують PDF як основний формат, однак Resume.io та Kickresume додатково пропонують експортування у форматі DOCX, а SweetCV і Kickresume дозволяють створювати веб-сторінки резюме, забезпечуючи більшу гнучкість для різних ситуацій.

Нарешті, адаптація під ATS є критичною для успішного подання заявок до великих компаній. Rezi спеціалізується на створенні резюме, які легко проходять через такі системи, і Resume.io також забезпечує цю функцію. Canva і SweetCV, навпаки, мають обмежену відповідність вимогам ATS, що може знижувати їхню ефективність для пошуку роботи в корпоративному середовищі.

Порівняльний аналіз показує, що кожна платформа має свої сильні та слабкі сторони. Resume.io та Rezi підходять для створення структурованих резюме, що проходять ATS, але обмежені в кількості шаблонів. Canva надає найбільше шаблонів, проте не підтримує адаптацію під ATS, а Kickresume пропонує розширений функціонал завдяки використанню GPT-4, що робить його зручним для створення індивідуалізованих резюме.

Використання мобільних застосунків і чат-ботів для створення резюме має ряд переваг. Вони забезпечують зручність у користуванні, оскільки дозволяють створювати резюме у будь-який час і в будь-якому місці. Ці інструменти відзначаються простотою у використанні завдяки інтуїтивно зрозумілим інтерфейсам, що робить їх доступними навіть для користувачів з мінімальними технічними навичками. Крім того, швидкість обробки даних і можливість експорту готового документа значно спрощують процес складання резюме. Деякі з цих інструментів інтегруються з іншими платформами, такими як хмарні сервіси або професійні мережі, що дозволяє зберігати або поширювати створені документи з мінімальними зусиллями.

Загалом, мобільні застосунки та чат-боти є перспективними інструментами для створення резюме, які спрощують цей процес завдяки своїй

доступності, гнучкості та інтеграції з сучасними технологіями. Вибір конкретного інструменту залежить від потреб користувача та бажаних функціональних можливостей.

2.3. Переваги та недоліки існуючих сервісів для створення резюме

Аналізуючи існуючі сервіси для створення резюме, можна виділити кілька важливих аспектів щодо використаного програмного забезпечення та інструментів, які впливають на функціональність, зручність і адаптивність цих сервісів. Серед найбільш популярних технологій, що застосовуються в розробці таких платформ, є фреймворки для веб-розробки, бібліотеки для обробки тексту та генерації PDF-документів, а також алгоритми штучного інтелекту для аналізу даних і побудови рекомендацій.

Веб-фреймворки, такі як React, Vue та Angular, забезпечують сучасні можливості для швидкої та зручної розробки адаптивних інтерфейсів, що легко масштабуються та підтримуються. Вони дозволяють створювати інтуїтивно зрозумілі інтерфейси, доступні з будь-якого пристрою, включаючи комп’ютери, планшети та смартфони. Водночас використання таких технологій вимагає висококваліфікованих розробників, що може збільшувати витрати на розробку, а також може створювати проблеми з оптимізацією для повільних інтернет-з’єднань, що впливає на користувацький досвід.

Бібліотеки для генерації PDF, такі як PDFKit та jsPDF, сприяють зручності експорту резюме в PDF-форматі, забезпечуючи високоякісне форматування та простоту створення індивідуальних шаблонів. Однак вони мають обмеження у відтворенні складних графічних елементів, а їх інтеграція може впливати на продуктивність системи, потребуючи додаткової оптимізації.

Технології штучного інтелекту, включаючи GPT-4 та BERT, автоматизують написання тексту для резюме, генеруючи вміст на основі

введених даних. Це особливо корисно для користувачів без досвіду створення резюме, адже система надає рекомендації та приклади текстів. Водночас такі технології потребують значної обчислювальної потужності для роботи в реальному часі, а генерований текст може бути шаблонним, що потребує додаткового редагування.

Адаптація резюме для систем відстеження кандидатів (ATS) здійснюється через аналіз ключових слів і тексту. Це значно підвищує ймовірність успішного проходження автоматичних фільтрів, що є важливим для великих компаній. Однак такі алгоритми часто обмежують можливості дизайну резюме, що може знижувати його привабливість для рекрутерів, а також вимагають інтеграції спеціалізованих інструментів.

Онлайн-конструктори, що використовують редактори типу WYSIWYG, надають можливість візуального редагування резюме в реальному часі, що забезпечує зручність і спрощує налаштування формату та стилю. Проте вони можуть не підтримувати складні формати та інтеграції, а також обмежувати користувачів у створенні унікальних стилів.

Існуючі сервіси для створення резюме мають низку недоліків. По-перше, вони часто не забезпечують достатньої гнучкості для адаптації під специфічні потреби користувачів, пропонуючи лише стандартні шаблони, що може бути недоліком для представників нетипових професій. По-друге, обмежена підтримка багатомовності ускладнює використання сервісів на світовому ринку, що знижує їх універсальність. По-третє, відсутність інтеграції з іншими платформами, такими як LinkedIn або системи подачі вакансій, створює додаткові незручності для користувачів, які змушені переносити дані вручну. Нарешті, залежність від стабільного інтернет-з’єднання є значним бар’єром для користувачів з обмеженим доступом до мережі або низькою швидкістю інтернету.

Попри численні переваги, існуючі сервіси для автоматизованого створення резюме мають певні обмеження, пов'язані з інструментами та

програмним забезпеченням, які використовуються для їхньої розробки. Розуміння цих аспектів дозволяє визначити напрями вдосконалення та допомагає створити більш адаптивні, зручні та ефективні платформи, які відповідатимуть специфічним потребам користувачів.

2.4. Технології та методи, що використовуються для автоматизації створення резюме

Автоматизоване створення резюме стає все більш популярним завдяки розвитку сучасних технологій, які спрощують процес підготовки документа та адаптації його до вимог ринку праці. Використання різних технологій дозволяє сервісам збирати дані, структурувати їх, форматувати, а також забезпечувати інтерактивну взаємодію з користувачем. Розглянемо основні технології та методи, які застосовуються для розробки платформ автоматизованого створення резюме.

1. Технології для розробки інтерфейсу користувача (Front-end)

Для зручної взаємодії з користувачем розробники використовують різні фреймворки та бібліотеки для створення інтуїтивно зрозумілих інтерфейсів:

- React.js, Vue.js, Angular — ці фреймворки дозволяють створювати динамічні інтерфейси, що полегшує взаємодію користувача з інструментом для створення резюме. Вони дозволяють легко налаштовувати шаблони, редагувати текстові поля та переглядати зміни в режимі реального часу.
- Bootstrap, Tailwind CSS — CSS-фреймворки, які використовуються для стилізації інтерфейсу. Вони забезпечують зручну верстку елементів, роблячи інтерфейс більш естетичним та легким для розуміння.

2. Технології для обробки та зберігання даних (Back-end)

Основою роботи будь-якої платформи для автоматизованого створення резюме є серверна частина, яка відповідає за обробку та зберігання даних користувача:

- Node.js, Django, Ruby on Rails — популярні серверні фреймворки, що дозволяють швидко обробляти запити та керувати даними, наданими користувачем. Ці фреймворки забезпечують обробку даних для подальшого форматування та експорту у вигляді резюме.
- MongoDB, PostgreSQL, MySQL - бази даних, які використовуються для зберігання персональних даних, інформації про досвід роботи, освіту та навички користувачів. Використання надійних реляційних або NoSQL баз даних дозволяє забезпечити швидкий доступ до даних та їх безпеку.

3. Бібліотеки для створення та експорту документів

Важливим аспектом автоматизованого створення резюме є можливість експорту документа у зручному форматі, такому як PDF або DOCX:

- pdfkit, jsPDF - бібліотеки для JavaScript, які дозволяють генерувати PDF-документи на стороні клієнта або сервера. Вони забезпечують високоякісне відображення тексту та графіки, що важливо для створення професійних резюме.
- Docx.js — бібліотека для генерації файлів DOCX, яка дозволяє експортувати резюме у форматі, що сумісний з Microsoft Word. Це підвищує зручність для користувачів, які потребують редагованих версій документа.

4. Технології штучного інтелекту для аналізу та генерації контенту

Застосування штучного інтелекту (AI) в інструментах для створення резюме дозволяє забезпечити більше персоналізованих рекомендацій та автоматизувати написання тексту:

- GPT-4, BERT — моделі обробки природної мови (NLP), які використовуються для генерації тексту. Вони можуть запропонувати користувачу текст для опису досвіду роботи або навичок на основі введених даних, що значно спрощує процес написання резюме.

- Аналіз ключових слів для ATS (Applicant Tracking Systems) — алгоритми, що використовують машинне навчання для підбору ключових слів у резюме, відповідно до опису вакансій. Це підвищує ймовірність успішного проходження автоматичних фільтрів у системах відстеження кандидатів.

5. Інструменти для інтерактивної взаємодії з користувачем

Для зручного збору інформації та налаштування резюме на різних платформах використовуються різні інструменти інтерактивної взаємодії:

- Чат-боти (Dialogflow, Rasa) — інструменти для створення чат-ботів, що дозволяють збирати дані від користувача у форматі діалогу. Вони надають можливість динамічно змінювати питання та зберігати інформацію, яку потім використовують для генерації резюме.
- WYSIWYG редактори (Trix, Quill) — текстові редактори, що дозволяють користувачу самостійно редагувати текст у зручному форматі. WYSIWYG редактори спрощують форматування резюме, забезпечуючи перегляд змін в режимі реального часу.

Для створення мобільних застосунків та чат-ботів використовуються різноманітні технології та методи, які охоплюють програмні платформи, архітектурні підходи, інструменти розробки та алгоритми обробки даних. Однією з ключових технологій для створення мобільних застосунків є кросплатформні фреймворки, такі як React Native, Flutter і Xamarin. Ці інструменти дозволяють розробляти застосунки для різних операційних систем, таких як Android та iOS, використовуючи спільний код. Це забезпечує ефективність розробки та зменшує витрати часу і ресурсів. Для нативної розробки застосунків використовуються мови програмування Swift для iOS та Kotlin або Java для Android [16].

У створенні чат-ботів важливу роль відіграють технології штучного інтелекту, зокрема обробка природної мови (NLP) та машинне навчання. Інструменти, такі як Dialogflow, IBM Watson Assistant, Rasa та Microsoft Bot Framework, дозволяють інтегрувати алгоритми аналізу тексту та розпізнавання

намірів користувачів. Для реалізації основної логіки чат-ботів застосовуються популярні мови програмування, такі як Python, JavaScript або Node.js. Важливою складовою є алгоритми обробки тексту, що забезпечують розпізнавання запитів користувача, генерацію відповідей та навчання системи на основі зібраних даних.

Архітектурно мобільні застосунки та чат-боти використовують підхід клієнт-сервер. У мобільних застосунках клієнтська частина відповідає за інтерфейс користувача та взаємодію, тоді як серверна частина забезпечує обробку даних, автентифікацію, інтеграцію з базами даних та зовнішніми сервісами. У чат-ботух клієнтська частина зазвичай представлена платформами для обміну повідомленнями, такими як Telegram, WhatsApp або Facebook Messenger, тоді як серверна частина забезпечує виконання логіки відповіді на запити .

Для розробки інтерфейсу мобільних застосунків застосовуються інструменти проєктування та прототипування, такі як Figma або Adobe XD. Щодо чат-ботів, інтерфейси часто інтегруються з веб- або мобільними платформами за допомогою API. API також використовуються для інтеграції з іншими сервісами, такими як платіжні системи, CRM або аналітичні інструменти [16].

Методи DevOps та CI/CD забезпечують ефективність у створенні, тестуванні та розгортанні як мобільних застосунків, так і чат-ботів. Ці методи дозволяють автоматизувати процеси розробки, скорочуючи час доставки продукту до кінцевого користувача. Водночас засоби контейнеризації, такі як Docker, сприяють масштабованості та легкій інтеграції з різними середовищами.

Таким чином, створення мобільних застосунків і чат-ботів базується на синергії сучасних інструментів розробки, архітектурних підходів і алгоритмів штучного інтелекту, що забезпечує ефективність і функціональність цих рішень у сучасному цифровому середовищі. Отже, сучасні технології та методи

дозволяють значно спростити процес створення резюме, забезпечуючи при цьому персоналізований підхід для кожного користувача. Використання інструментів для інтерактивної взаємодії, технологій штучного інтелекту та бібліотек для експорту документів підвищує зручність і ефективність платформ. Оптимальний вибір технологій забезпечує гнучкість у налаштуванні та можливість інтеграції з іншими системами, що робить процес створення резюме швидким, зручним та більш пристосованим до потреб ринку праці.

РОЗДІЛ 3. ТЕОРЕТИЧНА ЧАСТИНА

3.1. Опис обраних технологій і підходів для розробки чат боту

Метою нашої дипломної роботи була розробка програмного забезпечення телеграм-бота, створеного на основі `aiogram` з використанням вебхуків для обробки API запитів, підтримкою кількох мов (i18n) та можливістю генерації PDF-резюме на основі HTML-шаблонів. Цей бот допомагає користувачам швидко створювати якісні резюме за допомогою інтерактивних команд у Telegram.

Для роботи над ПЗ чат боту було обрано наступні технології та інструменти:

1. Середовище розробки PyCharm.
2. Основні бібліотеки та фреймворки: aiogram, Quart.
3. aiogram_i18n - модуль для підтримки інтернаціоналізації (i18n) у aiogram.
4. pdfkit та Jinja2 - інструменти для генерації PDF-файлів з HTML-шаблонів.
5. Метод API-запитів: вебхуки.

Середовище розробки PyCharm

PyCharm є інтегрованим середовищем розробки (IDE), розробленим компанією JetBrains, спеціально для програмування мовою Python. Цей інструмент надає широкий спектр функціональних можливостей, що сприяють підвищенню продуктивності розробників, забезпечуючи зручність і ефективність у роботі з кодом. PyCharm пропонує розширені засоби редагування, аналізу та навігації в коді, що робить його одним із найбільш популярних середовищ розробки серед Python-програмістів.

Основною перевагою PyCharm є його потужний аналізатор коду, який автоматично визначає потенційні помилки та пропонує їх виправлення. Інтеграція засобів перевірки стилю коду та підтримка стандартів PEP 8 дозволяє розробникам писати чистий і зрозумілий код. PyCharm також підтримує автозавершення, що значно прискорює написання складних конструкцій і забезпечує зручність навігації по проєкту. Окрім цього, середовище має інтегровані інструменти для роботи з Git, базами даних, Docker і тестовими фреймворками, що дозволяє виконувати всі основні операції без необхідності використовувати сторонні програми [17].

PyCharm забезпечує повну інтеграцію з популярними фреймворками та бібліотеками Python, включаючи Django, Flask, NumPy, Pandas і TensorFlow. Це особливо важливо для розробників, які працюють у галузі веб-розробки, наукових обчислень або машинного навчання. Інтегроване середовище також підтримує розробку у віртуальних середовищах Python, таких як virtualenv або conda, що забезпечує ізоляцію проєктів і полегшує управління залежностями.

Порівняно з іншими IDE, PyCharm виділяється своїм інтуїтивно зрозумілим інтерфейсом і гнучкими можливостями налаштування. Розробники можуть адаптувати середовище до власних потреб, налаштовуючи комбінації клавіш, теми та панелі інструментів. PyCharm також має розширені можливості для налагодження, які дозволяють проводити детальний аналіз виконання програми, включаючи роботу з потоками, змінними та точками зупинки. Ці можливості роблять процес виявлення та виправлення помилок значно ефективнішим.

Щодо порівняння з іншими середовищами розробки, такими як Visual Studio Code або Jupyter Notebook, PyCharm пропонує комплексний набір інструментів для повного циклу розробки. Visual Studio Code є легшим і більш універсальним редактором, але потребує встановлення численних розширень для досягнення функціональності PyCharm. У випадку з Jupyter Notebook, це середовище більше підходить для інтерактивного аналізу даних, але не

забезпечує такого рівня підтримки розробки великих програмних проєктів [8].

Таким чином, PyCharm є потужним і функціональним інструментом для Python-розробників, який забезпечує всі необхідні засоби для створення, тестування та розгортання програм. Його можливості дозволяють зосередитися на основних аспектах розробки, зменшуючи час на виконання рутинних завдань і підвищуючи загальну ефективність роботи. Це середовище особливо рекомендується для розробників, які працюють над великими або складними проєктами, завдяки інтегрованим інструментам і широким можливостям налаштування.

Основні бібліотеки та фреймворки

aiogram

Aiogram є однією з найпопулярніших бібліотек для розробки чат-ботів у месенджері Telegram, створеною на основі мови програмування Python. Головною особливістю цієї бібліотеки є асинхронна архітектура, яка забезпечує високу продуктивність та дозволяє обробляти значну кількість запитів користувачів одночасно. Завдяки цьому Aiogram є ефективним інструментом для створення як простих, так і складних масштабованих ботів.

Бібліотека повністю підтримує Telegram Bot API, включаючи всі останні оновлення, такі як інтерактивні кнопки, мультимедійний контент, опитування та вбудовані повідомлення. Інтерфейс програмування Aiogram відзначається своєю простотою, що дає змогу навіть недосвідченим розробникам швидко освоїти процес створення ботів. Однією з ключових функцій є можливість гнучкої маршрутизації вхідних повідомлень за допомогою системи хендлерів, яка дозволяє розробляти складні логічні сценарії для взаємодії з користувачами [3].

Aiogram також забезпечує підтримку концепції проміжного програмного забезпечення (middleware), що дозволяє інтегрувати додаткову логіку між запитами користувача та їх обробкою. Ця функціональність корисна для

впровадження таких механізмів, як авторизація, обмеження доступу або збір аналітичних даних. Крім того, бібліотека включає інструменти для управління станами, що дозволяє створювати багатокрокові сценарії, зокрема для форм або діалогів.

Асинхронна природа Aiogram робить її надзвичайно ефективною для роботи у високонавантажених системах. Бібліотека інтегрується з сучасними інструментами розробки, такими як Docker та CI/CD, що сприяє швидкому тестуванню і розгортанню проєктів. Проте Aiogram має певні обмеження, такі як відсутність офіційної підтримки інших месенджерів, окрім Telegram. Крім того, розробка на базі цієї бібліотеки вимагає знання асинхронного програмування, що може бути викликом для початківців.

Aiogram часто використовується для створення різноманітних Telegram-ботів, включаючи системи автоматизації служби підтримки, інформаційні боти, інтерактивні опитування, торговельні платформи з інтеграцією платіжних систем та інші автоматизовані рішення. Завдяки своїм функціональним можливостям Aiogram є універсальним інструментом для розробників, які прагнуть створювати ефективні та сучасні Telegram-боти.

Переваги: Висока продуктивність завдяки асинхронності та підтримка розширених функцій (як-от FSM), що дозволяє будувати складну логіку з обробки запитів.

Причина вибору: Забезпечує зручну роботу з FSM та підтримку вебхуків, що робить її ідеальною для створення бота з високим рівнем взаємодії.

Роль у проєкті: Використовується Dispatcher для маршрутизації команд, Bot для управління ботом, а також MemoryStorage для зберігання сесій користувачів.

Quart

Quart є сучасним веб-фреймворком для Python, який базується на асинхронній архітектурі та підтримує протокол HTTP/1.1, WebSocket і HTTP/2.

Асинхронний фреймворк для створення веб-додатків на Python, сумісний з asyncio. Його головною метою є забезпечення розробникам гнучкого, масштабованого та продуктивного інструменту для створення веб-додатків. Quart розроблений як асинхронна альтернатива популярному фреймворку Flask, використовуючи ті ж принципи і синтаксис, що робить його зручним для переходу користувачів Flask до більш сучасної парадигми розробки [5].

Quart відзначається підтримкою асинхронних викликів, що дозволяє розробникам повністю використовувати потенціал асинхронного програмування, інтегрованого в Python через asyncio. Це дає можливість обробляти велику кількість одночасних запитів без значних втрат продуктивності, що робить Quart ідеальним для високонавантажених веб-додатків. Його асинхронна архітектура дозволяє легко інтегрувати інструменти, такі як асинхронні бази даних або веб-сервіси.

Синтаксис Quart є практично ідентичним Flask, що забезпечує швидке освоєння фреймворка для розробників, які знайомі з Flask. Він підтримує такі ж концепції, як маршрутизація, обробка запитів і робота з шаблонами. Ця сумісність дозволяє легко переносити проєкти з Flask на Quart, при цьому отримуючи всі переваги асинхронної обробки. Крім того, Quart забезпечує підтримку WebSocket, що робить його підходящим для розробки сучасних реального часу веб-додатків, таких як чати, системи повідомлень чи інтерактивні панелі моніторингу.

Однією з ключових переваг Quart є його гнучкість у роботі з сучасними протоколами, такими як HTTP/2, який дозволяє покращити швидкість завантаження веб-ресурсів завдяки мультиплексуванню. Фреймворк також підтримує інтеграцію з іншими популярними бібліотеками, такими як SQLAlchemy для роботи з базами даних або Jinja2 для створення HTML-шаблонів. Завдяки цьому Quart може слугувати основою як для невеликих додатків, так і для великих розподілених систем.

Порівняно з іншими асинхронними фреймворками, такими як FastAPI або

Sanic, Quart пропонує баланс між простотою використання та функціональністю. FastAPI, хоча і має більш високу продуктивність для REST API, більше орієнтований на генерацію OpenAPI-документації та інтеграцію з Pydantic, тоді як Quart більше підходить для розробки традиційних веб-додатків. Sanic, у свою чергу, має схожу асинхронну архітектуру, але менш сумісний із традиційними бібліотеками Flask, що може вимагати додаткових зусиль під час міграції.

Quart забезпечує високу гнучкість завдяки підтримці асинхронних middleware та розширень, що дозволяє створювати кастомізовані рішення для специфічних вимог проєктів. Крім того, його архітектура спрощує роботу з великими командами розробників завдяки модульності та можливості масштабування. Це робить Quart вибором для тих, хто прагне створювати сучасні веб-додатки, використовуючи переваги асинхронного програмування та сумісності з уже відомими інструментами. У підсумку, Quart поєднує в собі зручність Flask із перевагами асинхронного підходу, забезпечуючи ефективність і сучасність розробки веб-додатків.

Переваги: Підтримка асинхронної обробки запитів, що дозволяє використовувати webhook для обробки подій Telegram в реальному часі.

Причина вибору: Налаштування вебхуків дозволяє обробляти події Telegram без затримок, без необхідності постійного полінгу.

Роль у проєкті: Використовується для обробки вхідних POST-запитів з Telegram API та налаштування роутів вебхука.

aiogram_i18n

aiogram_i18n є розширенням для бібліотеки Aiogram, яке забезпечує функціональність локалізації та інтернаціоналізації (i18n) у процесі розробки Telegram-ботів. Цей інструмент дозволяє створювати багатомовні боти, які можуть автоматично визначати мову користувача або змінювати її відповідно до налаштувань, заданих у боті. Основною метою aiogram_i18n є спрощення

процесу додавання підтримки кількох мов, забезпечуючи високий рівень інтеграції з основним функціоналом Aiogram [7].

Бібліотека aiogram_i18n базується на популярному пакеті aiogram і використовує стандартні підходи до інтернаціоналізації, такі як зберігання текстів у форматах, сумісних із gettext (.po і .mo файли). Це забезпечує гнучкість у роботі з текстовими ресурсами та дозволяє розробникам легко керувати перекладами. Крім того, aiogram_i18n підтримує автоматичне визначення мови за допомогою Telegram API, що дозволяє використовувати мовні налаштування профілю користувача для вибору відповідного перекладу.

Інтеграція aiogram_i18n у проєкт є зручною завдяки модульній архітектурі Aiogram. Розробник може додати middleware для роботи з мовними налаштуваннями, що забезпечує автоматичний вибір текстових ресурсів залежно від контексту. Такий підхід дозволяє легко створювати багатомовні сценарії взаємодії, у яких кожен користувач отримує контент своєю мовою без необхідності додаткового налаштування.

Головною перевагою aiogram_i18n є його зручність та висока продуктивність у процесі локалізації ботів. Інструмент дозволяє використовувати динамічні значення в перекладах, що особливо корисно для створення текстів із змінними, такими як імена користувачів, дати або числа. Крім того, бібліотека забезпечує можливість тестування локалізації, дозволяючи розробникам швидко перевіряти коректність текстів та їх відображення в інтерфейсі бота.

У порівнянні з іншими підходами до інтернаціоналізації, такими як власна реалізація локалізації або використання окремих бібліотек, aiogram_i18n пропонує глибоку інтеграцію з архітектурою Aiogram. Це дозволяє уникнути складнощів, пов'язаних із налаштуванням обробки запитів, і зосередитися на створенні якісного користувацького досвіду. Інші фреймворки для локалізації можуть не мати такої зручної підтримки middleware або інтеграції з Telegram API, що робить aiogram_i18n більш ефективним для специфічних потреб

розробників Telegram-ботів.

Таким чином, aiogram_i18n є незамінним інструментом для розробників, які прагнуть створювати багатомовні Telegram-боти. Його простота інтеграції, підтримка стандартів локалізації та глибока інтеграція з основними функціями Aiogram забезпечують високу продуктивність і якість взаємодії з користувачами на різних мовах. Це рішення рекомендовано для проєктів, орієнтованих на глобальну аудиторію, де підтримка кількох мов є критично важливою.

Переваги: Спрощує локалізацію ботів, дозволяючи легко додавати багатомовну підтримку.

Причина вибору: Надає можливість підтримувати декілька мов через I18nMiddleware та FluentRuntimeCore.

Роль у проєкті: Забезпечує мультимовну підтримку, що підвищує зручність користування ботом.

pdfkit та Jinja2

pdfkit є бібліотекою для Python, яка дозволяє створювати PDF-документи на основі HTML і CSS. Цей інструмент надає зручний спосіб конвертації HTML-сторінок у PDF-файли, використовуючи механізм WebKit через утиліту wkhtmltopdf. Завдяки підтримці сучасних стандартів HTML5 і CSS3, pdfkit дозволяє генерувати документи з високою якістю форматування, що робить його незамінним для створення звітів, рахунків, брошур і інших документів, які потребують друку або поширення в PDF-форматі [6].

Однією з ключових особливостей pdfkit є його здатність обробляти HTML-файли, URL-адреси або рядки HTML-коду як джерело для генерації PDF-документів. Ця універсальність робить бібліотеку гнучкою в різних сценаріях, включаючи динамічну генерацію документів у веб-додатках або автоматизацію звітів у бекенд-системах. pdfkit підтримує розширені функції форматування, такі як стилізація тексту, використання шрифтів, додавання

зображень, таблиць і навіть інтерактивних елементів, таких як посилання.

Серед переваг pdfkit варто відзначити його простоту використання та підтримку різних платформ, включаючи Windows, macOS і Linux. Інтеграція бібліотеки з іншими Python-інструментами дозволяє легко поєднувати її з фреймворками, такими як Flask або Django, для генерації PDF-документів на основі веб-шаблонів. Однак одним із недоліків є залежність від зовнішнього компонента wkhtmltopdf, який потребує окремого встановлення. Незважаючи на це, pdfkit залишається потужним рішенням для проєктів, що вимагають високоякісної конвертації HTML у PDF.

wkhtmltopdf є потужним інструментом командного рядка, який дозволяє конвертувати HTML-документи у формат PDF. Цей інструмент базується на рендерингу HTML за допомогою WebKit, що забезпечує високу якість відображення сторінок і підтримку сучасних веб-стандартів, включаючи HTML5, CSS3 та JavaScript. Використання wkhtmltopdf є популярним підходом у проєктах, де потрібна автоматизована генерація друкованих документів, таких як рахунки, звіти або інструкції [11].

Однією з основних переваг wkhtmltopdf є його здатність обробляти HTML-документи так, як це робить веб-браузер. Це означає, що будь-який HTML-файл або веб-сторінка, яка виглядає добре у браузері, може бути точно відтворена у PDF-документі. Інструмент підтримує складні стилі CSS, включаючи шрифти, кольори, таблиці, зображення та фонові елементи. wkhtmltopdf також дозволяє використовувати інтерактивні елементи, такі як посилання та закладки, які зберігаються у фінальному PDF-файлі.

Інструмент wkhtmltopdf забезпечує гнучкість у налаштуванні виводу. Він дозволяє змінювати розміри сторінок, поля, орієнтацію, додавати колонтитули та заголовки, а також налаштовувати параметри рендерингу, такі як якість зображень і використання векторної графіки. Крім того, wkhtmltopdf підтримує функції, що дозволяють об’єднувати кілька HTML-документів у єдиний PDF-файл, а також автоматично створювати зміст на основі заголовків сторінок.

Однією з ключових особливостей wkhtmltopdf є його простота інтеграції з іншими інструментами та програмами. Він може використовуватися як самостійний інструмент через командний рядок або бути інтегрованим у веб-додатки за допомогою бібліотек для мов програмування, таких як Python, PHP, Ruby чи Node.js. Це робить wkhtmltopdf універсальним рішенням для розробників, які створюють системи з автоматизованою генерацією PDF-документів.

Попри численні переваги, wkhtmltopdf має кілька обмежень. По-перше, він залежить від WebKit, що може спричиняти відмінності у рендерингу порівняно з іншими браузерами, такими як Chrome або Firefox. По-друге, для складних HTML-сторінок із великим обсягом JavaScript може знадобитися додаткове налаштування або оптимізація. Крім того, wkhtmltopdf є ресурсомістким інструментом, що слід враховувати при роботі з великими документами або при виконанні численних конверсій на сервері.

Загалом, wkhtmltopdf є потужним і надійним рішенням для перетворення HTML у PDF, яке широко використовується в системах автоматизації та генерації документів. Його можливості у поєднанні з гнучкістю та підтримкою сучасних веб-стандартів роблять його популярним вибором серед розробників, які працюють із веб-додатками або потребують створення високоякісних друкованих матеріалів. У поєднанні з бібліотеками, такими як Python pdfkit або PHP Laravel Snappy, wkhtmltopdf стає ще більш зручним у використанні для створення складних PDF-документів.

Jinja2 є популярним шаблонізатором для Python, який використовується для створення динамічних HTML-сторінок, документів і текстів. Його ключовою особливістю є можливість відділення логіки від представлення, що сприяє більшій структурованості й підтримуваності коду. Завдяки простому синтаксису, схожому на інші шаблонізатори, такі як Django Templates або Twig, Jinja2 є зручним для розробників із різним рівнем підготовки.

Jinja2 підтримує потужну систему змінних, що дозволяє динамічно

передавати дані з Python-коду в шаблони. Шаблони можуть містити фільтри, цикли, умовні оператори та макроси, що робить їх надзвичайно гнучкими та придатними для створення складних інтерфейсів. Наприклад, Jinja2 дозволяє автоматично відображати списки даних, динамічно змінювати елементи сторінки залежно від контексту або включати частини інших шаблонів за допомогою наслідування.

Серед переваг Jinja2 виділяється його інтеграція з популярними веб-фреймворками, такими як Flask і Django. Завдяки цьому розробники можуть створювати динамічні веб-додатки, де шаблони генеруються на сервері та передаються клієнту як статичний HTML. Крім того, Jinja2 підтримує розширення, що дозволяє додавати кастомні фільтри та функції, адаптуючи його до специфічних потреб проєкту. Шаблонізатор також забезпечує високий рівень безпеки завдяки вбудованій системі автоматичного екранування, що запобігає виконанню небажаного коду.

Порівняно з іншими шаблонізаторами, Jinja2 виділяється своєю продуктивністю та гнучкістю. Він здатний ефективно обробляти великі обсяги даних і створювати складні шаблони з мінімальними витратами ресурсів. Проте, як і будь-який інструмент, Jinja2 має свої обмеження. Наприклад, він не підходить для рендерингу на клієнтській стороні або роботи в інтерактивних середовищах, таких як React або Vue.

Загалом, pdfkit і Jinja2 представляють потужну комбінацію інструментів для генерації динамічних документів і веб-сторінок. Використання Jinja2 для створення HTML-шаблонів, які потім конвертуються в PDF за допомогою pdfkit, є популярним підходом у багатьох веб-додатках, забезпечуючи зручність і гнучкість у створенні високоякісного контенту. Це рішення рекомендується для проєктів, що вимагають поєднання динамічної генерації контенту та його подальшого експорту у зручний формат.

Переваги: pdfkit і wkhtmltopdf дозволяють створювати PDF із HTML, Jinja2 спрощує динамічне налаштування шаблонів.

Причина вибору: pdfkit надає потужний інструмент для генерації PDF, а Jinja2 дозволяє динамічно налаштовувати HTML.

Роль у проєкті: Бот створює PDF-резюме на основі обраного користувачем стилю, що зберігається у вигляді HTML-шаблонів із CSS.

Метод API-запитів: вебхуки

Метод API-запитів: вебхуки є одним із сучасних підходів до обробки подій та передачі даних між клієнтськими та серверними додатками в реальному часі. Вебхуки (webhooks) дозволяють серверу автоматично надсилати повідомлення або дані іншому додатку, коли відбувається певна подія. Цей механізм працює за принципом зворотного виклику (callback) через HTTP-запити, які ініціюються сервером-ініціатором і надсилаються до URL-адреси, вказаної отримувачем [18].

Основною перевагою використання вебхуків є їхня ефективність у передачі даних у реальному часі без необхідності постійного опитування серверів клієнтом. У традиційній моделі клієнт періодично надсилає запити для перевірки, чи відбулася подія, що може призводити до надмірного навантаження на систему. Натомість вебхуки дозволяють серверу одразу повідомляти клієнта про події, що суттєво знижує затримки та оптимізує використання ресурсів.

Вебхуки застосовуються у багатьох галузях, включаючи інтеграцію платіжних систем, систем керування контентом, CRM та аналітичних платформ. Наприклад, при обробці платежів через платформи, такі як Stripe або PayPal, вебхуки можуть автоматично інформувати про статус транзакції або успішну оплату. У сфері розробки веб-додатків вебхуки використовуються для відстеження змін у базах даних, автоматичного оновлення інтерфейсів або запуску певних процесів у відповідь на дії користувачів.

Однією з важливих особливостей вебхуків є їхня залежність від стабільності серверного середовища. Для коректної роботи необхідно, щоб

сервер отримувача був доступний і правильно обробляв вхідні запити. Зазвичай сервер отримувача повинен бути налаштований для обробки HTTP POST-запитів, які містять дані у форматі JSON або іншому популярному форматі. У випадку тимчасової недоступності отримувача сервер-ініціатор може зберігати запити для повторної доставки, але це залежить від конкретної реалізації.

Перевагами використання вебхуків є їхня простота в реалізації та висока продуктивність. Вони дозволяють мінімізувати затримки передачі даних, зменшити навантаження на сервери та покращити загальну інтеграцію між системами. Водночас вебхуки мають і певні недоліки. Наприклад, відсутність стандартизованих протоколів безпеки може робити їх уразливими до атак, таких як підробка запитів (request forgery). Для захисту вебхуків зазвичай використовуються механізми підписів запитів, автентифікації через токени або шифрування даних.

Порівняно з іншими методами інтеграції, такими як опитування серверів (polling) або довготривалі з'єднання (long polling), вебхуки є більш ефективним рішенням у багатьох сценаріях, особливо для обробки подій у реальному часі. Однак їхнє впровадження потребує ретельного налаштування серверів та врахування аспектів безпеки.

Загалом вебхуки є потужним інструментом для інтеграції та автоматизації процесів у системах, які потребують передачі даних у реальному часі. Їх використання сприяє покращенню продуктивності та швидкості роботи додатків, забезпечуючи зручний спосіб зв'язку між серверами без зайвого навантаження на ресурси. Це робить їх популярним вибором для сучасних веб-додатків та сервісів, орієнтованих на інтерактивність і швидкість обробки даних.

У розробленому нами чат боті використовується вебхук для отримання запитів від Telegram API замість звичайного полінгу. Вебхуки дозволяють зменшити затримку обробки повідомлень, оскільки дані надходять одразу, як тільки Telegram має подію для відправки боту.

Переваги вебхуків:

• Менша затримка: Дані надходять моментально, без потреби перевіряти нові події постійно.

• Економія ресурсів: Сервер обробляє запити лише коли вони надходять, замість безперервного опитування, що економить ресурси.

• Покращена безпека: Можливість встановлення перевірок автентифікації (як-от перевірка токена) для підтвердження запитів.

Основні завдання та проблеми, які вирішує бот:

1. Створення резюме – чат бот дозволяє користувачам створювати резюме з чіткою структурою та вибором стилів.

2. Мультимовність - завдяки i18n користувачі можуть обирати мову інтерфейсу, що робить бот доступним для більш широкої аудиторії.

3. Автоматизація процесу обробки запитів - вебхуки замість полінгу дозволяють економити ресурси та зменшити затримку.

4. Конвертація резюме у PDF - користувач отримує готовий PDF-документ, що спрощує процес створення професійного резюме.

Для налаштування та запуску бота нами було виконано наступне:

1. Встановлення необхідних залежностей з файлу `requirements.txt`:

```bash
pip install -r requirements.txt
```

2. Створення віртуального середовища:

```bash
python3.10 -m venv venv
```

3. Активація віртуального середовища:

- Linux/macOS:

```bash
source venv/bin/activate
```

```
- Windows:
```bash
.\venv\Scripts\activate
```

4. Встановлення wkhtmltopdf:

- Linux:

```bash
sudo apt install wkhtmltopdf
```

- macOS:

```bash
brew install wkhtmltopdf
```

- Windows:

Завантаження `wkhtmltopdf` [https://wkhtmltopdf.org/downloads.html] із вказанням шляху до `wkhtmltopdf.exe`.

5. Встановлення Ngrok** та запуск на порту 5000:

```bash
ngrok http 5000
```

6. Налаштування конфігурації у `cfg.py`:

```python
BOT_TOKEN = "YOUR_BOT_TOKEN"
URL_SERVER_WEBHOOK = "YOUR_NGROK_URL"  # приклад: https://f6aa-91-246-142-249.ngrok-free.app
WKHTMLTOPDF = "PATH_TO_WKHTMLTOPDF"
```
```

6. Запуск сервера для роботи з вебхуками:

```bash
python main.py
```

3.2. Методологія розробки програмного забезпечення та побудова блок схеми системи

Методологія Waterfall (водоспадна модель) є однією з класичних моделей управління проектами, яка широко застосовується у сфері розробки програмного забезпечення. Ця модель передбачає лінійний та послідовний підхід до розробки, де кожен етап повинен бути завершений перед тим, як розпочнеться наступний. Waterfall є детально структурованою методологією, яка забезпечує чітке розуміння вимог, планування і реалізації проекту. Розробка Android-застосунку для агрегації вакансій за допомогою методології Waterfall може мати свої переваги і недоліки, які слід враховувати в залежності від специфіки проекту [10].

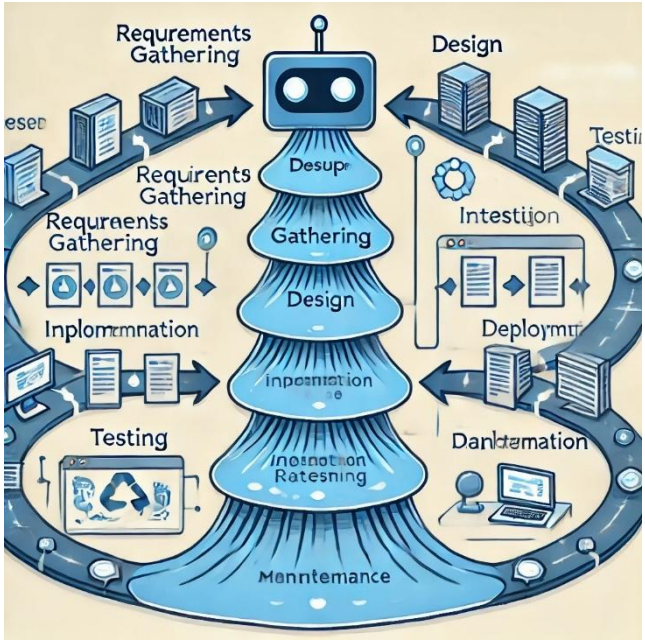

Рисунок. 3.1. Модель розробки застосунку

Опис усіх етапів розробки чат-боту для автоматичної генерації резюме у форматі PDF за методологією Waterfall:

1. Збір вимог (Requirements Gathering)

Етап необхідний, щоб визначити основні функціональні можливості, які потрібні для чат-боту, включаючи взаємодію з користувачем, поля введення, вибір формату та можливість генерувати PDF. І задля співпраці з зацікавленими сторонами для визначення ключових функцій, створення користувацьких історій та формулювання очікувань щодо функціональності чат-боту.

2. Проєктування системи (System Design)

Цей етап призначений для розробки технічного плану, який описує архітектуру чат-боту, інструменти для генерації PDF та користувацький інтерфейс.

Завданнями цього етапу є проектування потоку діалогів, обробку на бекенді та макет PDF. Вибір інструментів та технологій, які будуть використовуватись, наприклад, модулі для обробки природної мови та бібліотеки для створення PDF.

3. Реалізація (Implementation)

Призначень для написання коду функціоналу чат-боту відповідно до проєктних специфікацій.

Серед завдань видіяємо насутпні: розробити та інтегрувати діалоговий механізм бота, компоненти для генерації PDF та обробку введених даних користувачем; провести тестування кожного модуля окремо для забезпечення стабільності.

4. Тестування (Testing)

Етап призначений для перевірки, що кожна частина чат-боту працює коректно та відповідає вимогам. Він необхідний для проведення

функціонального, юзабіліті та продуктивного тестування. Переконатись, що бот генерує точний зміст резюме та правильний формат PDF. Усунути всі виявлені помилки або недоліки.

5. Розгортання (Deployment)

Даний етап призначений для запуску чат-боту для кінцевих користувачів.

Тут важливо розгорнути чат-боту на обраній платформі (веб, мобільний додаток тощо) і налаштувати необхідну підтримку на бекенді для обробки даних користувачів та генерації файлів PDF.

6. Обслуговування (Maintenance)

Даний етап призначений для забезпечення довгострокової функціональності і врахування відгуків користувачів. Він необхідний для моніторингу продуктивності бота, оновлення функцій за необхідності, виправлення нових помилок та оптимізації систему для обробки збільшеної кількості користувачів.

Цей структурований підхід дозволяє завершити кожен етап перед переходом до наступного, забезпечуючи чітку покрокову розробку чат-боту для генерації резюме [10].

Для того щоб побудувати діаграму роботи чат-боту на основі наданого коду, використаємо блок-схему, яка продемонструє основні етапи обробки запитів та основні функціональні блоки чат-боту. Діаграма відображає, як бот отримує та обробляє оновлення (updates) від Telegram, а також взаємодію між компонентами:

Основні блоки чат-боту:

1. Запуск сервера та налаштування бота - процес налаштування вебхука, зберігання сесій, та налаштування локалізації.
2. Обробка запитів вебхука - приймає вхідні оновлення від Telegram.
3. Передача оновлення до диспетчера - диспетчер отримує оновлення та направляє їх до відповідного обробника.

4. Обробка оновлень (Dispatcher) - визначає необхідний обробник для кожного типу оновлення (наприклад, текстові повідомлення, команди, кнопки).

5. Завершення роботи - процес закриття сесії, видалення вебхука та збереження стану.

Опис блоків:

1. Старт сервера - початок роботи Quart сервера, встановлення вебхука на вказаний URL і підключення до Telegram Bot API.

2. Встановлення локалізації - ініціалізація та налаштування I18nMiddleware для підтримки мультимовності.

3. Чекання запитів від Telegram - бот знаходиться у стані очікування запитів на кінцеву точку вебхука /webhook/<bot_token>.

4. Обробка запитів вебхука - отримує JSON-повідомлення, перевіряє токен і розпаковує дані оновлення.

5. Передача оновлення в диспетчер - Dispatcher (dp) отримує об’єкт Update і вибирає відповідний обробник, зареєстрований у main_dp.router.

6. Обробка оновлень - на основі типу оновлення диспетчер викликає відповідний обробник (наприклад, обробка команд, тексту, кнопок).

7. Відправка відповіді через Telegram API - після обробки оновлення бот відповідає користувачеві через Telegram API.

8. Очікування нових оновлень - бот продовжує чекати наступних оновлень.

9. Завершення роботи - при завершенні бота відбувається очищення вебхука та закриття сесії.

Ця схема описує основні кроки роботи чат-боту і взаємодію компонентів під час обробки запитів.

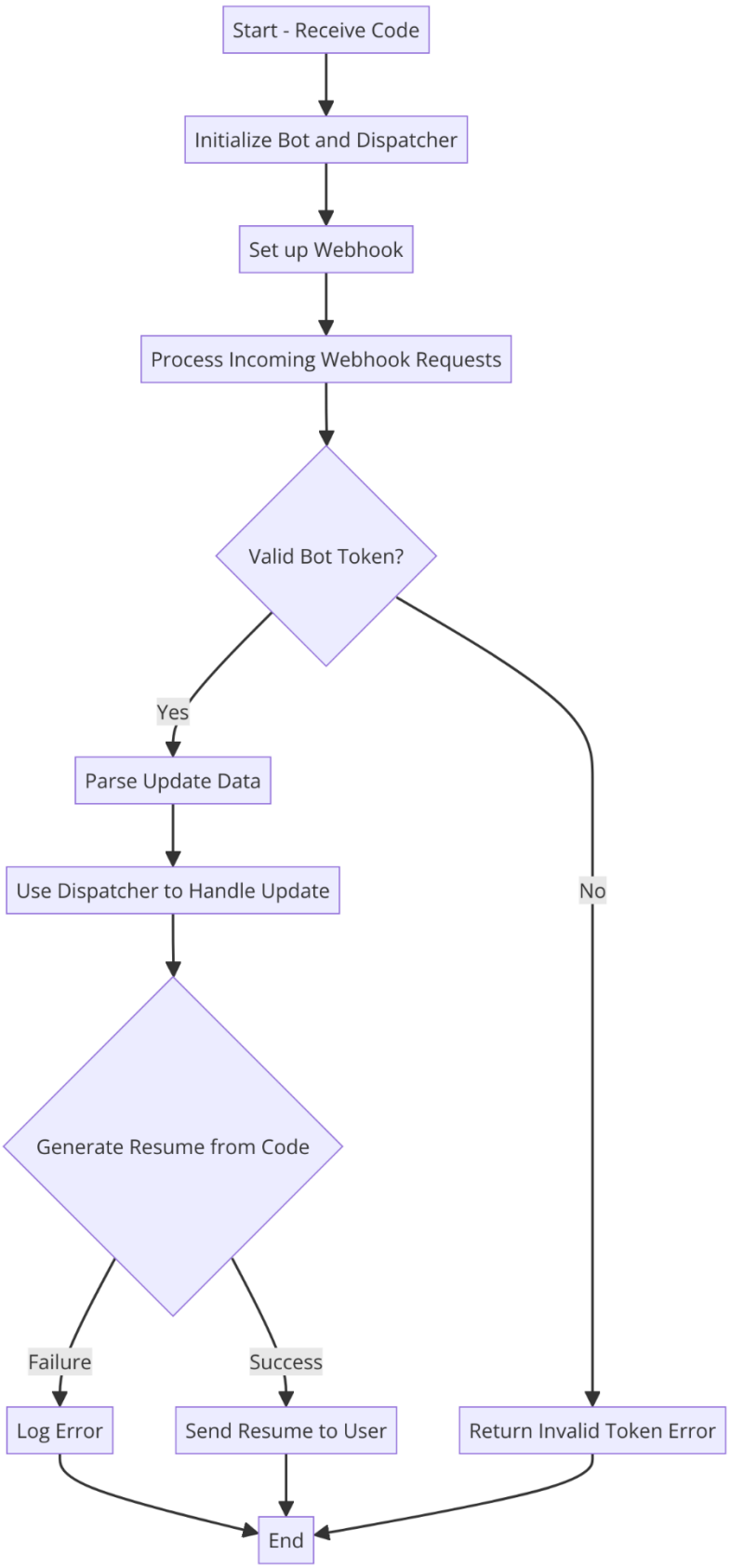

Рисунок 3. 2. Блок схема ПЗ розробленого чат боту

3.3. Архітектура чат боту для генерації резюме

Архітектура програмного забезпечення цього чат-боту ґрунтується на багаторівневій (модульній) структурі з компонентами, що виконують чітко визначені функції. Вона може бути описана як асинхронна архітектура з використанням вебхуків та диспетчера подій, яка включає кілька основних модулів та шарів, що взаємодіють один з одним. Така структура забезпечує гнучкість, масштабованість і простоту підтримки коду. Давайте розглянемо архітектуру детальніше.

Основні шари та компоненти архітектури:

1. Шар API-зв’язку (Communication Layer):
    - Компоненти: aiogram, Quart
    - Функції:
        - Використання вебхуків (/webhook/<bot_token>), щоб отримувати події від Telegram API.
        - Отримання, валідація та обробка запитів від Telegram (в тому числі перевірка токену).
        - Формування відповідей у форматі JSON для Telegram.
    - Інструменти: фреймворк Quart для створення асинхронного веб-сервера, який приймає запити вебхука.
2. Шар бізнес-логіки (Business Logic Layer):
    - Компоненти: Dispatcher та зареєстровані хендлери (обробники подій)
    - Функції:
        - Обробка основної логіки бота, що включає обробку команд, повідомлень, кнопок та інших взаємодій.
        - Використання Dispatcher як центру керування подіями, що спрямовує отримані оновлення (updates) до відповідних хендлерів.
        - Реалізація бізнес-логіки (наприклад, обробка конкретних

команд, введення тексту, кнопок тощо).

- Інструменти: модуль aiogram для асинхронної обробки подій, а також domain.handler для зберігання логіки обробників (хендлерів).

3. Шар керування станом (State Management Layer):
   - Компоненти: MemoryStorage, Finite State Machine (FSM)
   - Функції:
     - Зберігання станів користувачів під час багатокрокових діалогів або форм.
     - Використання FSM для управління послідовністю кроків у діалозі, де стан зберігається в пам’яті під час виконання бота.
   - Інструменти: MemoryStorage як сховище станів, FSM для покрокового керування логікою чатів.
4. Шар локалізації (Localization Layer):
   - Компоненти: I18nMiddleware, LocaleManager
   - Функції:
     - Відповідальність за мультимовність і підтримку локалізації контенту, який бот відправляє користувачам.
     - Використання FluentRuntimeCore для завантаження файлів локалізації з визначеної директорії.
   - Інструменти: aiogram_i18n для інтернаціоналізації та LocaleManager для динамічного вибору мови користувача.
5. Шар конфігурації та налаштувань (Configuration Layer):
   - Компоненти: cfg (конфігураційний файл)
   - Функції:
     - Зберігання конфігураційних налаштувань, таких як токен бота, URL вебхука, параметри з’єднання та інші глобальні параметри.
   - Інструменти: Звичайний Python-модуль cfg.py, що забезпечує зручний доступ до змінних.

6. Шар взаємодії з Telegram API (Telegram API Interaction Layer):
    - Компоненти: Bot, AiohttpSession
    - Функції:
        - Забезпечення асинхронного підключення до API Telegram для надсилання відповідей та взаємодії з Telegram-серверами.
        - Ініціалізація Bot з токеном і налаштуванням сесії.
    - Інструменти: AiohttpSession для асинхронного HTTP-запиту, що забезпечує з’єднання з API Telegram.

Особливості архітектури:

1. Вся архітектура побудована на асинхронній обробці запитів (використовуючи asyncio), що робить бота ефективним і дозволяє одночасно обробляти багато запитів без блокувань.
2. Кожен компонент виконує чітко визначену роль (отримання повідомлень, обробка бізнес-логіки, управління станом тощо), що полегшує підтримку і розширення коду.
3. Використання диспетчера подій та системи хендлерів дозволяє легко додавати або змінювати функціональні можливості, додаючи нові обробники подій або змінюючи існуючі.
4. Завдяки використанню вебхука бот працює безперервно, отримуючи запити у вигляді HTTP-запитів від Telegram, що оптимізує його роботу на серверах з обмеженими ресурсами.
5. Шар локалізації забезпечує динамічну зміну мови відповідей бота на основі налаштувань користувача, завдяки чому бот може адаптуватися під різні регіони.

Архітектура такого типу має кілька суттєвих переваг. Вона характеризується гнучкістю, що дозволяє легко додавати нові команди, локалізації або змінювати існуючу логіку без необхідності внесення значних змін до загальної структури. Крім того, така архітектура забезпечує високу масштабованість, оскільки асинхронна робота дозволяє ефективно обробляти

значну кількість запитів одночасно. Ще однією важливою особливістю є зручність підтримки: поділ на окремі модулі дає змогу протестувати та підтримувати кожен компонент незалежно. У цілому, така архітектура створює надійну основу для розробки, розширення та масштабування Telegram-боту, що робить її оптимальним вибором для вирішення відповідних завдань.

РОЗДІЛ 4. ПРАКТИЧНА ЧАСТИНА

4.1. Розробка алгоритму збору та обробки даних від користувача

Розглянемо далі алгоритм, який описує процес збору та обробки даних від користувача при роботі чат-боту. Цей алгоритм орієнтований на асинхронну роботу бота з вебхуком, використанням станів і локалізації, як у вашій архітектурі:

Алгоритм збору та обробки даних від користувача у Telegram-боті передбачає низку послідовних етапів, що забезпечують ефективну взаємодію з користувачем. Спочатку відбувається налаштування вебхука і запуск сервера, де ініціалізуються вебсервер, бот та диспетчер. Вебхук налаштовується на кінцевій точці, що дозволяє Telegram надсилати оновлення боту. Бот перебуває у стані очікування запитів від Telegram-сервера, таких як повідомлення, команди або натискання кнопок. Після отримання оновлення у вигляді JSON-даних, вебсервер передає їх диспетчеру для подальшої обробки. На цьому етапі перевіряється валідність токену, і, якщо перевірка успішна, починається обробка.

Диспетчер перетворює вхідні дані у формат, передбачений API Telegram, та визначає тип оновлення, наприклад, текстове повідомлення, команда чи взаємодія з кнопками. Потім оновлення передається відповідному обробнику події залежно від її типу. У разі використання багатокрокового процесу діалог з користувачем продовжується з урахуванням його поточного стану, що дозволяє зберігати контекст взаємодії.

Під час збору даних бот ініціює інтерактивний діалог із користувачем, надсилаючи запити та отримуючи відповіді, які зберігаються у тимчасовому сховищі станів. Це забезпечує послідовність збору інформації між різними етапами взаємодії. Після кожного кроку здійснюється перевірка, чи всі необхідні дані були отримані. Якщо даних недостатньо, діалог продовжується.

Коли всі дані зібрані, відбувається їх валідація, наприклад, перевірка формату електронної пошти. У разі невідповідності дані запитуються повторно.

Після успішної перевірки даних бот виконує бізнес-логіку, яка може включати реєстрацію, створення запису або підключення до зовнішніх API чи баз даних для збереження інформації. Користувачу надсилається відповідь, яка залежить від результатів обробки. Якщо операція успішна, бот інформує про це, а у разі помилки надсилає відповідне повідомлення.

Завершуючи діалог, бот очищує тимчасовий стан користувача у сховищі, забезпечуючи початок нової сесії без збережених даних. Нарешті, бот повертається у режим очікування, готовий до обробки нових запитів від Telegram. Ця послідовність дій забезпечує структуровану, надійну та ефективну обробку даних від користувачів.

Щоб краще продемонструвати, як збір і обробка даних від користувача реалізується у розробленому нами ПЗ чат боту, покажемо основні етапи, такі як запуск бота, отримання та обробка запитів, керування станом користувача і локалізацією із демонстрацією частин коду, що відповідає етапам алгоритму.

1. Запуск сервера та налаштування вебхука

```
@app.before_serving
async def on_startup():
    # Ініціалізація середовища локалізації для мультимовної підтримки
    i18n_middleware = I18nMiddleware(
        core=FluentRuntimeCore(path='presentation/locales'),
        default_locale='en',
        manager=LocaleManager()
    )
    await i18n_middleware.core.startup()
    i18n_middleware.setup(dp)  # Налаштування локалізації в диспетчері

    # Видалення попереднього вебхука та встановлення нового для
```

прийому подій

```
await bot.delete_webhook()
await bot.set_webhook(

url=f"{cfg.URL_SERVER_WEBHOOK}/webhook/{cfg.BOT_TOKEN}",
    drop_pending_updates=True,
    allowed_updates=dp.resolve_used_update_types()
)
print("Webhook setup complete")
```

1) Ініціалізація локалізації: цей код налаштовує мультимовну підтримку за допомогою I18nMiddleware.
2) Налаштування вебхука: видалення старого вебхука і встановлення нового на сервері. Це дозволяє Telegram надсилати повідомлення боту.

2. Обробка запитів від Telegram

```
@app.route('/webhook/<bot_token>', methods=['POST'])
async def webhook(bot_token):
    # Перевірка токену для забезпечення безпеки
    if bot_token != cfg.BOT_TOKEN:
        return jsonify({"status": "Invalid token"})

    # Отримання JSON-даних запиту та перетворення їх у формат `Update`
    json_data = await request.get_json()
    update = Update(**json_data)

    # Передача оновлення диспетчеру для визначення та обробки події
    await dp.feed_update(bot, update)

    return jsonify({"status": "ok"})
```

1) Перевірка токену: цей крок гарантує, що запити надходять від

Telegram.

2) Отримання та обробка оновлення: дані запиту конвертуються у формат Update, який передається диспетчеру для подальшої обробки.

3. Визначення обробника події та управління станами

Обробка оновлень відбувається через диспетчер dp, який визначає потрібний хендлер (обробник) на основі події від користувача. Тут диспетчер може використовувати хендлери, що обробляють різні етапи збору даних у багатокроковому діалозі.

Наприклад, додамо обробники для збору імені та електронної пошти:

```
from aiogram import types
from aiogram.fsm.context import FSMContext
from aiogram.fsm.state import StatesGroup, State

# Створюємо FSM для керування станами збору даних
class Registration(StatesGroup):
    waiting_for_name = State()
    waiting_for_email = State()

# Хендлер для старту та запиту імені
@dp.message_handler(commands=['start'], state="*")
async def start_registration(message: types.Message, state: FSMContext):
    await message.reply("Привіт! Як тебе звати?")
    await state.set_state(Registration.waiting_for_name)

# Хендлер для обробки імені
@dp.message_handler(state=Registration.waiting_for_name)
async def process_name(message: types.Message, state: FSMContext):
    # Збереження імені у стані
    await state.update_data(name=message.text)
```

```
    # Запит на введення електронної пошти
    await message.reply("Дякую! Тепер, будь ласка, введи свою електронну пошту.")
    await state.set_state(Registration.waiting_for_email)

# Хендлер для обробки електронної пошти
@dp.message_handler(state=Registration.waiting_for_email)
async def process_email(message: types.Message, state: FSMContext):
    email = message.text
    # Валідація електронної пошти
    if "@" not in email:
        await message.reply("Це виглядає як неправильна електронна пошта. Спробуй ще раз.")
        return

    # Збереження електронної пошти та завершення реєстрації
    user_data = await state.get_data()
    await message.reply(f"Дякуємо, {user_data['name']}! Реєстрацію завершено.")

    # Очищення стану
    await state.clear()
```

1) Старт діалогу і запит імені: коли користувач вводить команду /start, бот переходить до стану waiting_for_name і запитує ім'я.
2) Збір імені та перехід до наступного стану: після отримання імені бот переходить до стану waiting_for_email.
3) Збір електронної пошти та її валідація: бот перевіряє електронну пошту на правильність формату. Якщо дані коректні, реєстрацію завершується.

4. Завершення роботи та очищення стану

Використання await state.clear() в кінці реєстрації дозволяє очистити дані користувача після завершення процесу. Це гарантує, що наступна сесія з користувачем почнеться без залишкових даних.

5. Закриття сесії та завершення роботи бота

```
@app.after_serving
async def on_shutdown():
    await bot.delete_webhook()  # Видалення вебхука при завершенні роботи
    await bot.session.close()    # Закриття сесії бота
    print("Bot shutdown complete")
```

6. Закриття сесії та завершення роботи: при завершенні роботи бота видаляється вебхук, а сесія бота закривається для безпечного завершення.

Використання механізму Finite State Machine (FSM) є ключовим для управління різними етапами взаємодії з користувачем. FSM дозволяє організувати багатокроковий процес збору даних, зберігаючи стан кожного користувача та забезпечуючи логічний перехід між етапами. Це особливо корисно у складних сценаріях, коли потрібно збирати інформацію поступово, реагуючи на кожен крок користувача.

Хендлери в диспетчері виконують важливу функцію вибору відповідного обробника для кожного стану або події. Диспетчер аналізує тип оновлення, отриманого від Telegram (наприклад, текстове повідомлення, команда або натискання кнопки), і направляє його до потрібного хендлера. Це дозволяє боту обробляти різноманітні запити користувачів, забезпечуючи гнучкість і модульність системи.

Очищення стану після завершення діалогу з користувачем є критично важливим для коректної роботи бота. Видалення проміжних даних зі сховища станів гарантує, що нова взаємодія почнеться «з чистого аркуша». Це запобігає виникненню помилок через залишкові дані від попередніх сесій і забезпечує безперервність та прозорість у взаємодії з користувачем.

Цей алгоритм і відповідний код забезпечують інтуїтивний збір та обробку даних від користувача через багатокроковий діалог.

4.2. Побудова діаграми класів чат боту для створення резюме

Діаграма класів, побудована на основі розробленого нами програмного забезпечення чат боту для генерації резюме (див. рисунок 4.1.), відображає структуру основних класів, інтерфейсів та їх взаємодію.

Діаграма класів ілюструє основну структуру компонентів чат-боту та їх взаємозв’язки, що можна побачити у наведеному вами коді. Основні класи та їх функціональні зв’язки представлені нижче:

1. Клас Bot:
   - Відповідає за основну роботу з ботом. Має властивості token (токен для доступу до бота) та session (сесія AiohttpSession для роботи з HTTP-запитами).
   - Основні методи:
     - delete_webhook(): Видаляє встановлений webhook.
     - set_webhook(url: String): Встановлює webhook за вказаним URL.
2. Клас Dispatcher:
   - Основна функція Dispatcher — це обробка вхідних оновлень для бота.
   - Властивість storage зберігає стан бота в пам'яті (MemoryStorage).
   - Методи:
     - include_routers(): Підключає маршрутизатори для обробки різних типів запитів.

- feed_update(bot: Bot, update: Update): Передає оновлення боту для обробки.

3. Клас LocaleManager:

- Відповідає за локалізацію, керуючи мовними налаштуваннями бота.
- Властивість core визначає основну локалізацію за допомогою FluentRuntimeCore.
- Метод set_locale(locale: String) встановлює необхідну локалізацію.

4. Клас I18nMiddleware:

- Забезпечує багатомовність бота, працюючи разом з Dispatcher.
- Має core, що вказує на FluentRuntimeCore.
- Метод setup(dispatcher: Dispatcher) інтегрує мультимовну підтримку з Dispatcher.

5. Клас QuartApp (реалізований як app у вашому коді):

- Це веб-додаток, що використовує Quart для обробки HTTP-запитів.
- Основні методи:
  - before_serving(): Виконується перед запуском додатка для ініціалізації.
  - after_serving(): Виконується після завершення роботи додатка для очищення ресурсів.
  - route(path: String, methods: List[String]): Налаштовує маршрути, наприклад, обробник для /webhook/<bot_token>.

Зв’язки між класами:

- QuartApp пов'язаний з Dispatcher для обробки запитів.
- Dispatcher напряму пов'язаний з Bot, використовуючи його методи.
- LocaleManager взаємодіє з Bot для налаштування локалізації.
- Dispatcher також пов'язаний з I18nMiddleware для мультимовної підтримки.

Ця діаграма показує, як окремі компоненти (класи) працюють разом, щоб забезпечити повний цикл обробки запитів, генерації відповіді та управління мовними налаштуваннями.

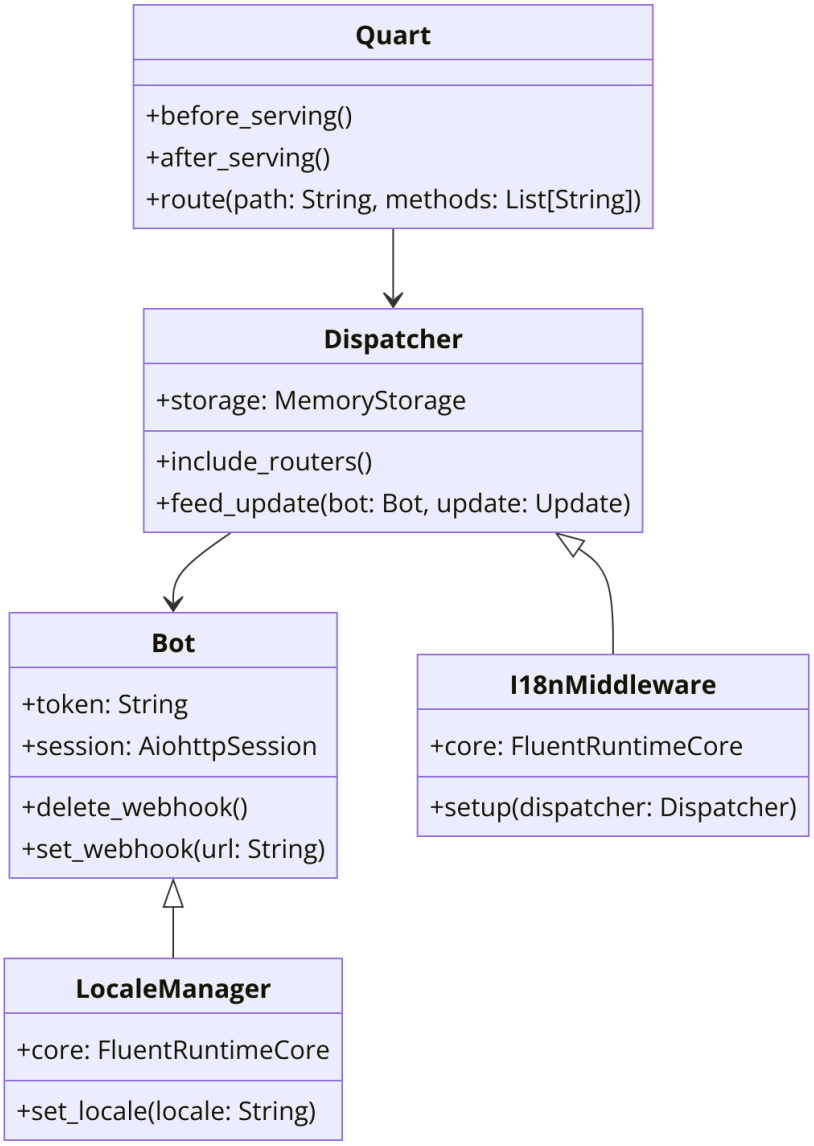

Рисунок 4.1. Діаграма класів ПЗ чат боту для генерації резюме

Діаграма прецедентів ілюструє роботу чат-боту для генерації резюме, наведемо основні етапи взаємодії:

1. Взаємодія з користувачем:
    - Користувач надсилає код до API чат-боту для генерації резюме.

2. Налаштування Webhook:
    - API чат-боту ініціює запит до Webhook для налаштування механізму обробки вхідних оновлень.
    - Webhook підтверджує налаштування і повертає підтвердження до API чат-боту.
3. Генерація резюме:
    - API чат-боту обробляє наданий код, передаючи його до модуля Генерації резюме.
    - Модуль генерації резюме повертає готове резюме до API чат-боту.
4. Локалізація:
    - API чат-боту використовує Менеджер локалізації для застосування відповідних мовних налаштувань, щоб резюме відповідало перевагам мови користувача.
5. Відповідь користувачу:
    - Нарешті, API чат-боту надсилає згенероване резюме назад Користувачу.

Ця послідовність організовує основні функції бота: обробку вхідних даних, генерацію резюме, управління локалізацією та повернення фінального результату користувачу.

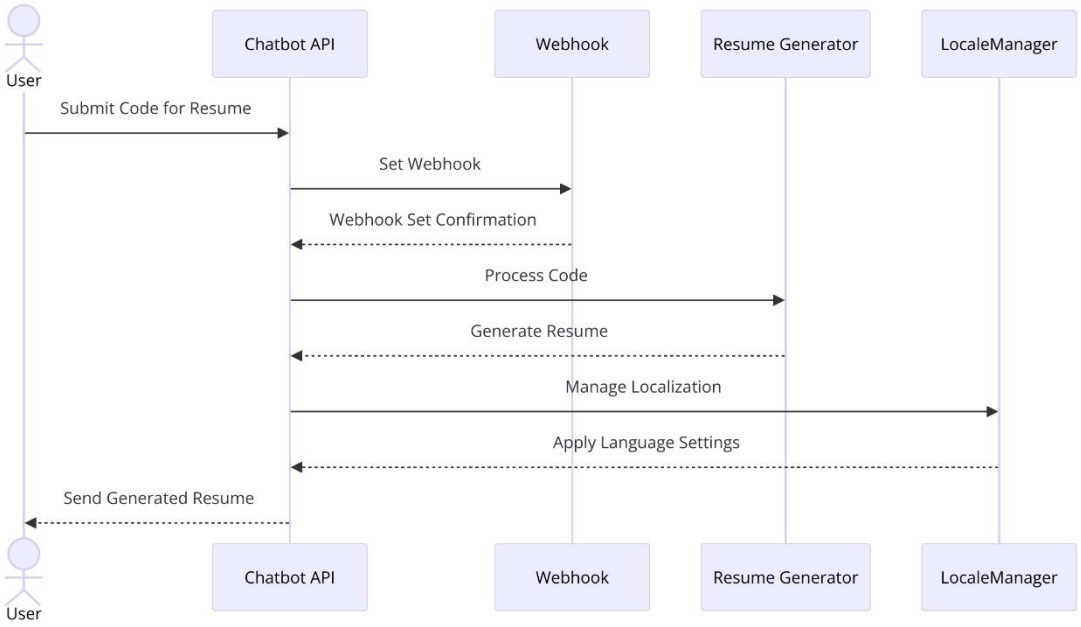

Рисунок 4.2. Use Case діаграма для розробленого чат боту

4.3. Опис та реалізація інтерфейсу користувача і функціональних можливостей чат боту

На рисунку ми бачимо інтерфейс користувача чат боту для генерації резюме у форматі PDF (Рисунок 4.3.). Робота починається із команди «Старт», далі заповнюються основні позиції резюме як от: ім’я, посада, контактні дані, наявні навички та досвід роботи в інших компаніях.

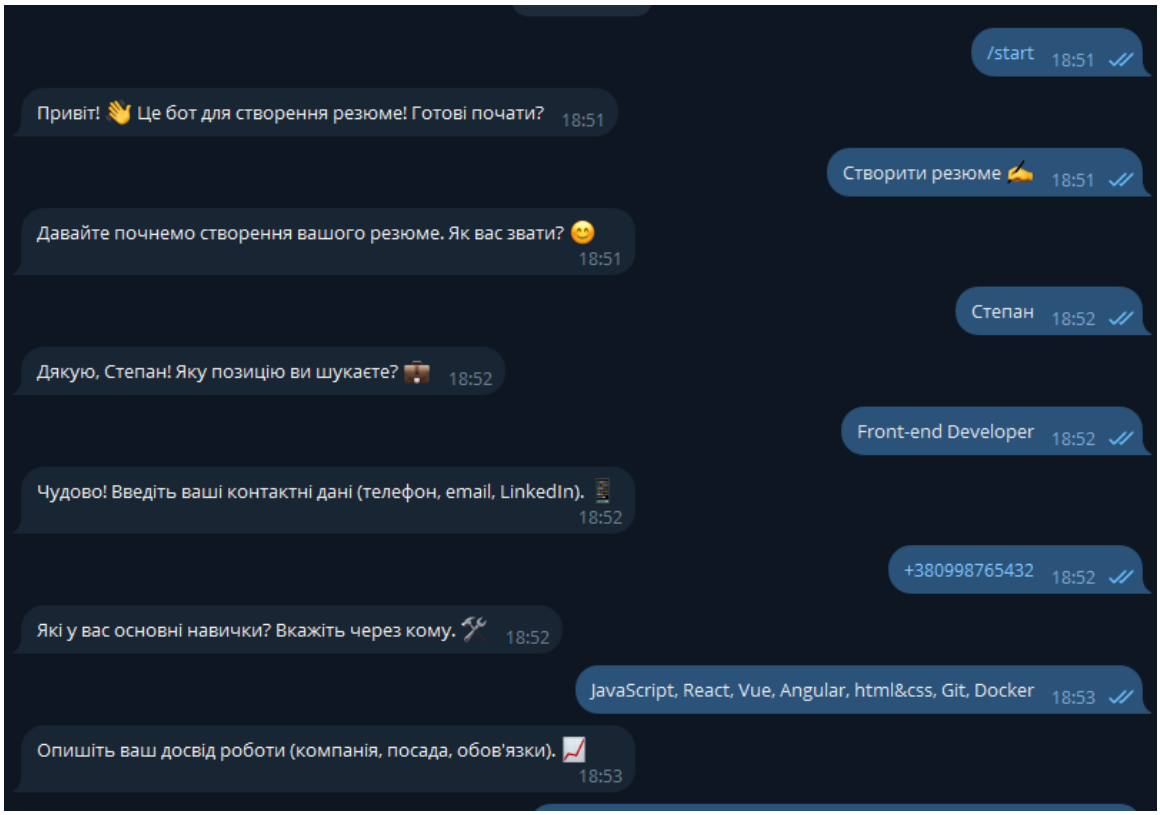

Рисунок 4.3. Початок роботи з чат ботом

Далі на рисунках 4.4.-4.6. представлені етапи роботи із чат ботом при заповненні своїх даних і можливі відповіді чат боту.

Рисунок 4.4. Робота зі створення резюме у чат боті

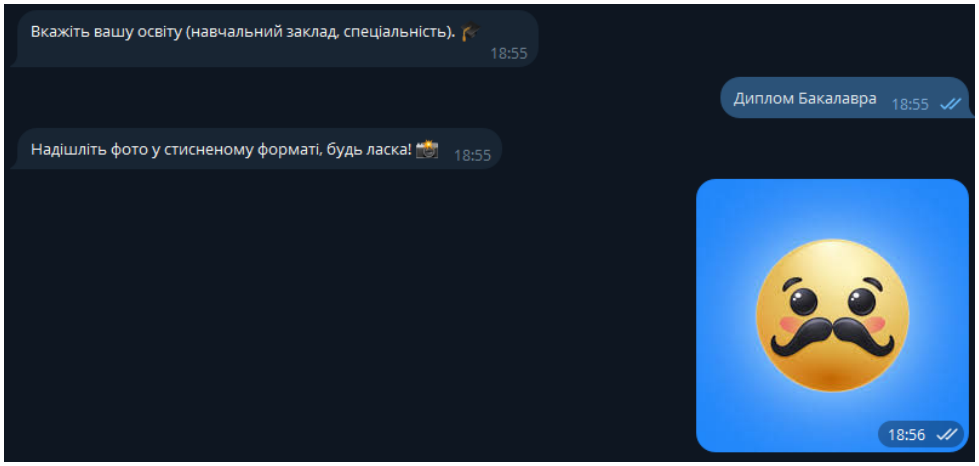

Рисунок 4.5. Робота зі створення резюме у чат боті

Рисунок 4.6. Робота зі створення резюме у чат боті

Після вибору стилю резюме, його одразу можна згенерувати.

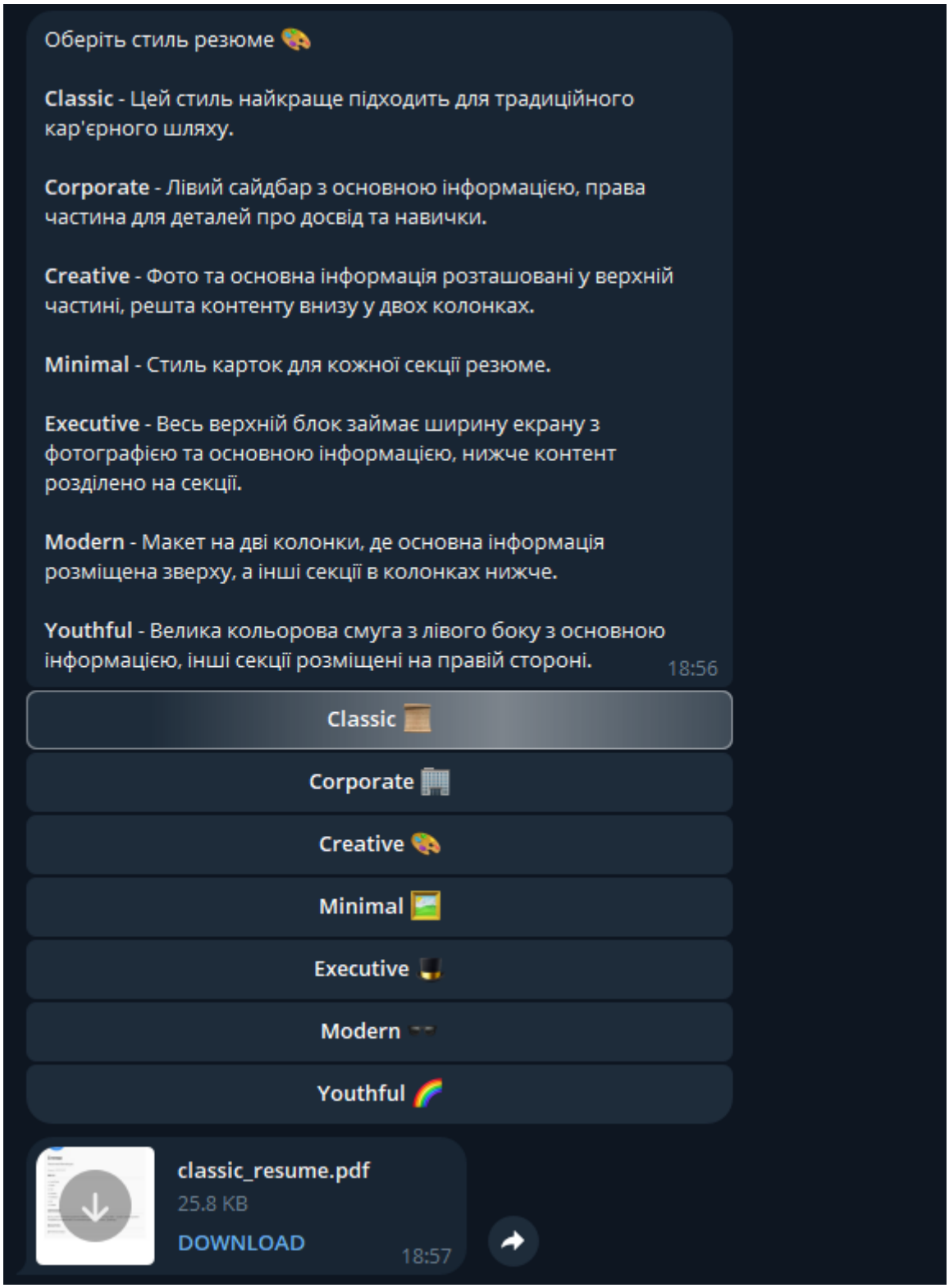

Рисунок 4.7. Генерація резюме у форматі PDF у класичному стилі

Можна одразу згенерувати і інші стилі.

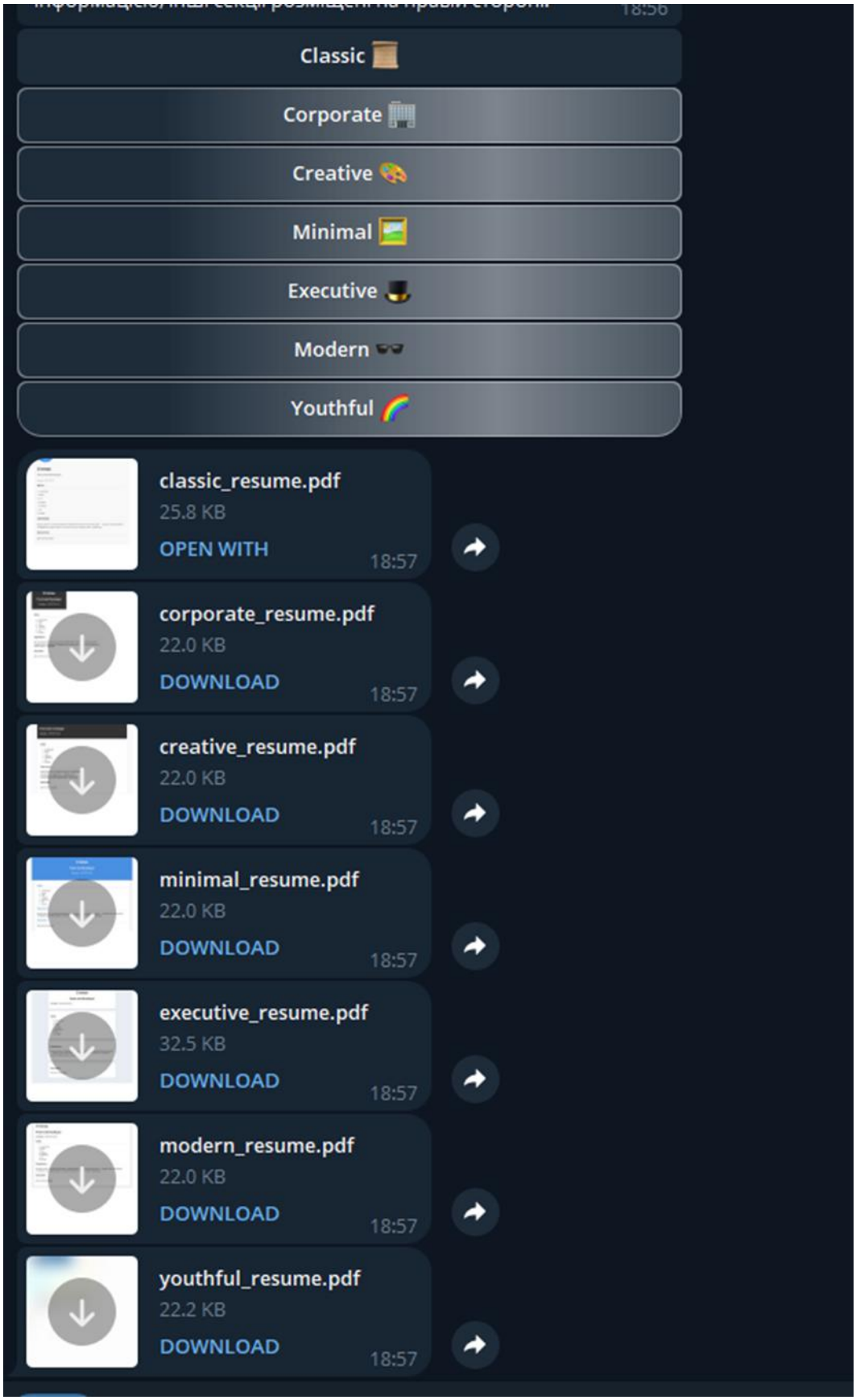

Рисунок 4.8. Генерація резюме у декількох стилях

Нижче представлені декілька варіантів згенерованого резюме, для віртуального користувача «Степан» (див. рисунок 4.9-4.11).

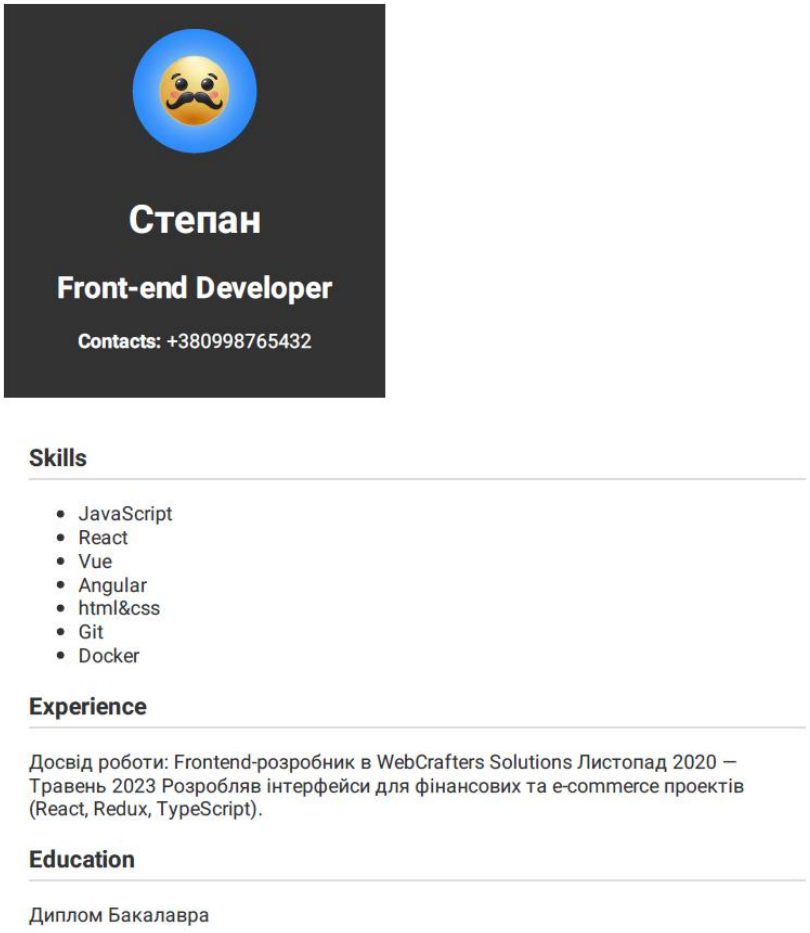

Рисунок 4.9. Резюме у корпоративному стилі

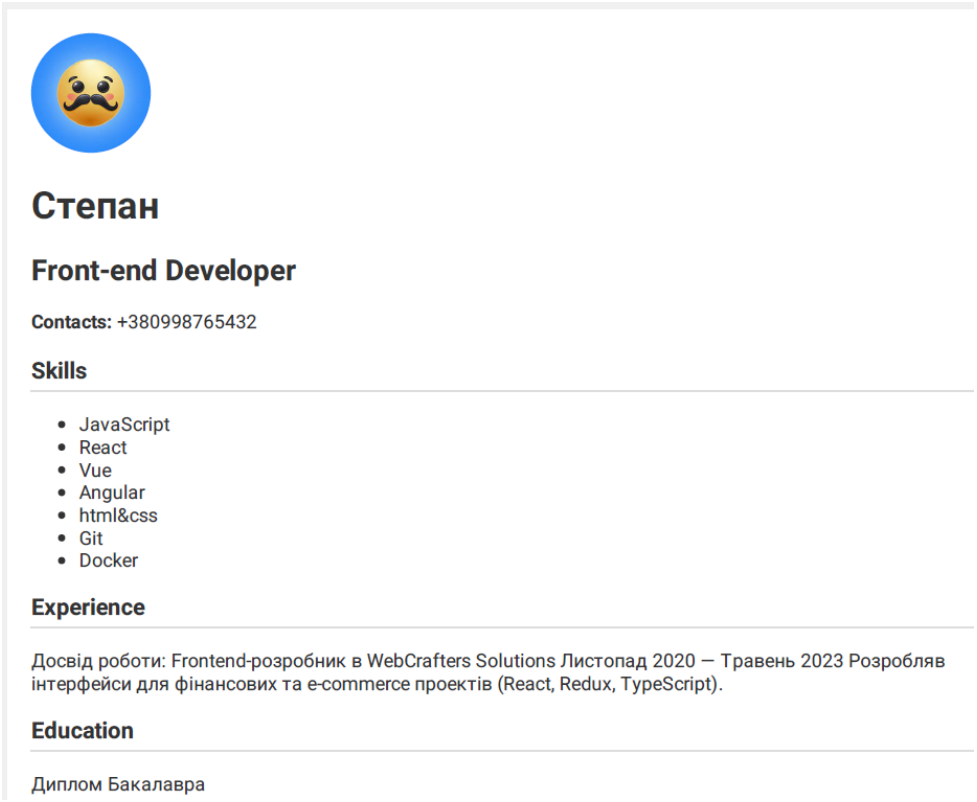

Рисунок 4.10. Резюме у креативному стилі

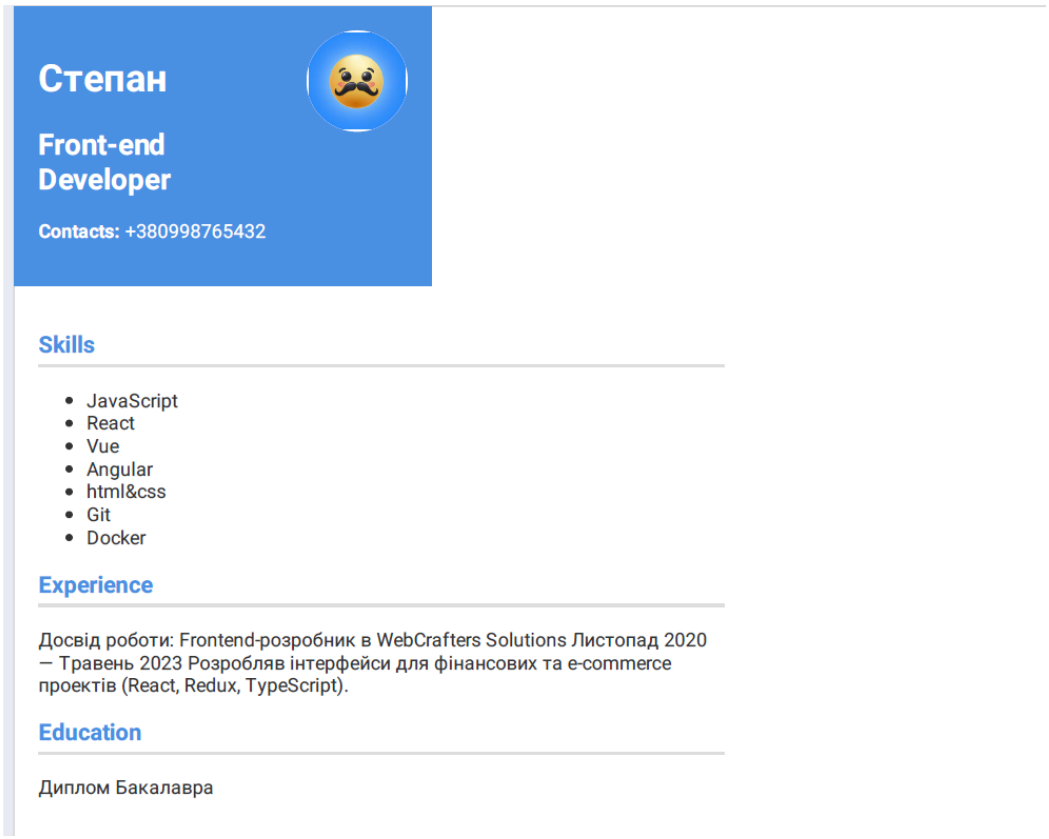

Рисунок 4.11. Резюме у молодіжному стилі

ВИСНОВКИ

Розробка чат-боту для генерації резюме у форматі PDF є актуальною через зростання попиту на автоматизовані інструменти для підготовки якісних документів. У сучасних умовах ринку праці швидке створення структурованого резюме, яке відповідає стандартам роботодавців, є важливим фактором успішного працевлаштування. Використання чат-ботів дозволяє економити час, зменшувати кількість помилок і підвищувати загальну якість документів завдяки інтеграції сучасних стандартів оформлення.

Такий інструмент особливо корисний для широкого кола користувачів, включаючи студентів, молодих спеціалістів і досвідчених фахівців. Чат-бот з можливістю персоналізації забезпечує зручність, доступність і адаптацію до індивідуальних потреб користувачів, сприяючи ефективному пошуку роботи, тому актуальність нашої дипломної роботи є досить високою.

Отже, у результати виконання дипломного проєкту нами було повністю виконано поставлені завдання дипломної роботи, а саме:

- Проведено аналіз існуючих рішень для автоматизації створення резюме, виявлено їх переваги та недоліки з точки зору зручності для користувача, функціональності та ефективності.
- Визначено вимоги до функціоналу чат-боту, включаючи основні та додаткові функції, що забезпечать зручність використання та адаптивність до різних потреб користувачів.
- Розроблено архітектуру чат-боту, його алгоритм, та методологію розробки ПЗ.
- Обрано найбільш відповідні інструменти та технології для реалізації чат-боту (середовище розробки PyCharm; основні бібліотеки та фреймворки: aiogram, Quart; aiogram_i18n; pdfkit та Jinja2 - інструменти для генерації PDF-файлів з HTML-шаблонів; метод API-запитів: вебхуки), зокрема

платформи для розробки чат-ботів, мови програмування та бібліотеки для обробки природної мови.

– Розроблено блок схему чат боту, представлено use-case діаграму та діаграму класів ПЗ розробленого чат боту.

– Розроблено інтерфейс та функціональні можливості чат-боту для збору, обробки та структурування даних, наданих користувачем.

– Реалізовано функціонал автоматичної генерації резюме в форматі PDF (у декількох стилях), на основі введених даних.

– Проведено тестування чат-боту для перевірки його ефективності та зручності використання.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. 10 найкращих інструментів для створення резюме AI. Режим доступу: URL: https://hashdork.com/uk/найкращі-конструктори-резюме-штучного-інтелекту/ - Назва з екрану

2. 10 сервісів для складання резюме, які вас здивують функціональністю. Режим доступу: URL: https://www.imena.ua/blog/10-services-of-resume/ - Назва з екрану

3. aiogram. Режим доступу: URL: https://github.com/aiogram/aiogram - Назва з екрану

4. Canva. Режим доступу: URL: https://www.canva.com/uk_ua/stvoryty/reziume/ - Назва з екрану

5. Developing a web application with Quart. Режим доступу: URL: https://pythonic.rapellys.biz/articles/developing-a-web-application-with-quart/ - Назва з екрану

6. How to Generate PDFs from HTML with Python-PDFKit (HTML string, HTML File and URL). Режим доступу: URL: https://apitemplate.io/blog/how-to-generate-pdfs-from-html-with-python-pdfkit/ Назва з екрану

7. I18n example. Режим доступу: URL: https://aiogram-birdi7.readthedocs.io/en/latest/examples/i18n_example.html - Назва з екрану

8. Jupyter vs. VS Code vs. PyCharm: Which One Should You Choose? Режим доступу: URL: https://dev.to/dazevedo/jupyter-vs-vs-code-vs-pycharm-which-one-should-you-choose-37jg - Назва з екрану

9. Resume.io Review: The Honest Breakdown to Pros & Cons. Published on September 5th, 2024. Режим доступу: URL: https://www.hirequotient.com/blog/resume.io-review - Назва з екрану

10. Waterfall Model for Software Development and Testing Teams. Режим доступу: URL: https://qatestlab.com/resources/knowledge-center/waterfall-process/ - Назва з екрану

11. wkhtmltopdf. Режим доступу: URL: https://wkhtmltopdf.org/- Назва з екрану

12. Біловодська О. А. Системне дослідження використання чат-боту в комунікації з клієнтами / О. А. Біловодська, К. І. Лагута // Формування ринкових відносин в Україні. – 2020. – № 5 (228). – С. 62-68. Режим доступу: https://er.knutd.edu.ua/handle/123456789/15953 - Назва з екрану

13. Конструктор резюме SweetCV - створіть резюме онлайн! Режим доступу: URL: https://sweetcv.com/ua/ - Назва з екрану

14. Найкращий конструктор резюме зі штучним інтелектом: 10 комплексних програм. Режим доступу: URL: https://mspoweruser.com/uk/best-ai-cv-builder/ - Назва з екрану

15. Ольховська О. В. Методичні рекомендації до виконання кваліфікаційної роботи для студентів спеціальності 122 Комп'ютерні науки освітня програма «Комп'ютерні науки» ступеня бакалавра / О. В. Ольховська, О. О. Черненко. – Полтава: ПУЕТ, 2024. – 67 с. Режим доступу: URL: http://dspace.puet.edu.ua/bitstream/123456789/14768/1/%D0%9C%D0%B5%D1%82%D0%BE%D0%B4%D0%B8%D1%87%D0%BA%D0%B0%20%D0%91%D0%A0%20%D0%B4%D0%BB%D1%8F%20%D0%9A%D0%9D%202024_%D0%B1%D0%B0%D0%BA%D0%B0%D0%BB%D0%B0%D0%B2%D1%80.pdf - Назва з екрану

16. Топ трендів розробки мобільних додатків у 2024 році. Режим доступу: URL: https://whileweb.com/uk/blog/top-trendiv-rozrobki-mobilnih-dodatkiv-u-2024-roci/ - Назва з екрану

17. Топ-5 Кращих IDE для Python розробки у 2024. Режим доступу: URL: https://university.sigma.software/best-python-ide-and-code-editors/ - Назва з екрану

18. Що таке Webhook? Режим доступу: URL: https://apix-drive.com/ua/blog/marketing/scho-take-webhook - Назва з екрану

ДОДАТОК А
Код програми

main.py

```
import asyncio
import logging

from aiogram import Bot, Dispatcher
from aiogram.client.default import DefaultBotProperties
from aiogram.client.session.aiohttp import AiohttpSession
from aiogram.enums import ParseMode
from aiogram.fsm.storage.memory import MemoryStorage
from aiogram.types import Update
from aiogram_i18n import I18nMiddleware
from aiogram_i18n.cores import FluentRuntimeCore
from quart import Quart, jsonify, request

import cfg
from domain.LocaleManager import LocaleManager
from domain.handler import main_dp

storage = MemoryStorage()
dp = Dispatcher(storage=storage)

default_properties = DefaultBotProperties(parse_mode=ParseMode.HTML)
bot = Bot(token=cfg.BOT_TOKEN, default=default_properties, session=AiohttpSession())

app = Quart(__name__)
logging.basicConfig(level=logging.INFO)

dp.include_routers(
    main_dp.router
)
```

```
@app.before_serving
async def on_startup():
    i18n_middleware = I18nMiddleware(
        core=FluentRuntimeCore(path='presentation/locales'),
        default_locale='en',
        manager=LocaleManager()
    )

    await i18n_middleware.core.startup()
    i18n_middleware.setup(dp)

    await bot.delete_webhook()
    await bot.set_webhook(
        url=f"{cfg.URL_SERVER_WEBHOOK}/webhook/{cfg.BOT_TOKEN}",
        drop_pending_updates=True,
        allowed_updates=dp.resolve_used_update_types()
    )
    print("Webhook setup complete")


@app.after_serving
async def on_shutdown():
    await bot.delete_webhook()
    await bot.session.close()
    print("Bot shutdown complete")


@app.route('/webhook/<bot_token>', methods=['POST'])
async def webhook(bot_token):
    if bot_token != cfg.BOT_TOKEN:
        return jsonify({"status": "Invalid token"})

    json_data = await request.get_json()
    update = Update(**json_data)
```

```
    await dp.feed_update(bot, update)

    return jsonify({"status": "ok"})
```

main_dp.py

```
import base64
from io import BytesIO

import requests
from aiogram import Router, F
from aiogram.filters import Command
from aiogram.fsm.context import FSMContext
from aiogram.types import Message, CallbackQuery, InputFile, BufferedInputFile, PhotoSize, File
from aiogram_i18n import I18nContext, L

from domain.PDFGenerator import PDFGenerator
from domain.states.Resume import Resume
from presentation.keyboard.main_kb import kb_menu, kb_style_resume, StyleResume

router = Router()

# Ініціалізація PDFGenerator
pdf_generator = PDFGenerator(template_path="data/templates")

# Ліміти для кожного поля
LIMITS = {
    "name": 50,          # Ім'я
    "position": 100,     # Позиція
    "contacts": 150,     # Контактні дані
    "skills": 500,       # Навички
    "experience": 500,   # Досвід роботи
    "education": 500     # Освіта
}
```

```
@router.message(Command("start"))
async def start_(message: Message, state: FSMContext, i18n: I18nContext):
    await message.answer(text=i18n.COMMAND.START(), reply_markup=kb_menu)


@router.message(F.text == L.RESUME.CREATE())
async def start_resume(message: Message, state: FSMContext, i18n: I18nContext):
    await state.clear()

    await message.answer(text=i18n.RESUME.NAME())
    await state.set_state(Resume.name)


@router.message(Resume.name)
async def get_name(message: Message, state: FSMContext, i18n: I18nContext):
    if len(message.text) > LIMITS["name"]:
        await message.answer(i18n.RESUME.NAME.ERROR(limit=LIMITS["name"],
size=len(message.text)))
        return
    await state.update_data(name=message.text)
    await message.answer(text=i18n.RESUME.POSITION(name=message.text))
    await state.set_state(Resume.position)


@router.message(Resume.position)
async def get_position(message: Message, state: FSMContext, i18n: I18nContext):
    if len(message.text) > LIMITS["position"]:
        await message.answer(i18n.RESUME.POSITION.ERROR(limit=LIMITS["position"],
size=len(message.text)))
        return
    await state.update_data(position=message.text)
    await message.answer(text=i18n.RESUME.CONTACTS())
    await state.set_state(Resume.contacts)
```

```
@router.message(Resume.contacts)
async def get_contacts(message: Message, state: FSMContext, i18n: I18nContext):
    if len(message.text) > LIMITS["contacts"]:
        await message.answer(i18n.RESUME.CONTACTS.ERROR(limit=LIMITS["contacts"],
size=len(message.text)))
        return
    await state.update_data(contacts=message.text)
    await message.answer(text=i18n.RESUME.SKILLS())
    await state.set_state(Resume.skills)

@router.message(Resume.skills)
async def get_skills(message: Message, state: FSMContext, i18n: I18nContext):
    if len(message.text) > LIMITS["skills"]:
        await message.answer(i18n.RESUME.SKILLS.ERROR(limit=LIMITS["skills"],
size=len(message.text)))
        return
    await state.update_data(skills=message.text)
    await message.answer(text=i18n.RESUME.EXPERIENCE())
    await state.set_state(Resume.experience)


@router.message(Resume.experience)
async def get_experience(message: Message, state: FSMContext, i18n: I18nContext):
    if len(message.text) > LIMITS["experience"]:
        await message.answer(i18n.RESUME.EXPERIENCE.ERROR(limit=LIMITS["experience"],
size=len(message.text)))
        return
    await state.update_data(experience=message.text)
    await message.answer(text=i18n.RESUME.EDUCATION())
    await state.set_state(Resume.education)

@router.message(Resume.education)
```

```
async def get_education(message: Message, state: FSMContext, i18n: I18nContext):
    if len(message.text) > LIMITS["education"]:
        await message.answer(i18n.RESUME.EDUCATION.ERROR(limit=LIMITS["education"],
size=len(message.text)))
        return
    await state.update_data(education=message.text)
    await message.answer(text=i18n.RESUME.PHOTO())
    await state.set_state(Resume.photo)


@router.message(Resume.photo, F.photo)
async def get_photo(message: Message, state: FSMContext, i18n: I18nContext):
    photo = message.photo[-1]

    file: File = await message.bot.get_file(photo.file_id)

    if file.file_path:
        file_url = f"https://api.telegram.org/file/bot{message.bot.token}/{file.file_path}"
        response = requests.get(file_url)
        response.raise_for_status()
        photo_base64 = base64.b64encode(response.content).decode('utf-8')
        await state.update_data(photo=photo_base64)

    await message.answer(text=i18n.RESUME.STYLE(), reply_markup=kb_style_resume)
    await state.set_state(Resume.style)

@router.callback_query(StyleResume.filter(), Resume.style)
async def get_style(callback: CallbackQuery, state: FSMContext, i18n: I18nContext):
    style_choice = callback.data.split(":")[1]
    data = await state.get_data()

    # Конвертируем HTML в PDF
    pdf_content = pdf_generator.generate_pdf(data, style_choice)
```

```
    # Создаем BufferedInputFile с PDF
    buffered_file = BufferedInputFile(pdf_content, filename=f"{style_choice}_resume.pdf")
    await callback.message.answer_document(buffered_file)
```

PDFGenerator.py

```
import tempfile

import pdfkit
from jinja2 import Environment, FileSystemLoader

from cfg import WKHTMLTOPDF

class PDFGenerator:
    def __init__(self, template_path: str,
                 wkhtmltopdf_path: str = WKHTMLTOPDF):
        self.env = Environment(loader=FileSystemLoader(template_path))
        # Указываем путь к wkhtmltopdf
        self.config = pdfkit.configuration(wkhtmltopdf=wkhtmltopdf_path)

    def generate_pdf(self, data: dict, style: str) -> bytes:
        # Чтение содержимого CSS файла
        with open(f"data/static/css/{style}.css", "r") as css_file:
            css_content = css_file.read()

        # Рендеринг шаблона с динамическим вставлением CSS
        template = self.env.get_template(f"{style}.html")
        html_content = template.render(data, css=css_content)

        # Генерация PDF
        with tempfile.NamedTemporaryFile(suffix=".pdf", delete=False) as tmp_file:
            pdfkit.from_string(html_content, tmp_file.name, configuration=self.config)
            tmp_file.seek(0)
            pdf_data = tmp_file.read()
```

```
        return pdf_data
```

LocalManager.py

```
from aiogram.types import User
from aiogram_i18n.managers import BaseManager

class LocaleManager(BaseManager):

    async def set_locale(self, locale: str) -> str:
        pass

    async def get_locale(self, event_from_user: User) -> str:
        return event_from_user.language_code
```

Resume.py

```
from aiogram.fsm.state import StatesGroup, State

class Resume(StatesGroup):
    name = State()
    position = State()
    contacts = State()
    skills = State()
    experience = State()
    education = State()
    photo = State()
    style = State()
```

cfg.py

```
BOT_TOKEN = ""
URL_SERVER_WEBHOOK = "https://f6aa-91-246-142-249.ngrok-free.app"
WKHTMLTOPDF = "C:\\Program Files\\wkhtmltopdf\\bin\\wkhtmltopdf.exe"
if __name__ == '__main__':
    app.run(host="0.0.0.0", port=5000)
```