Хмельницький національний університет
Факультет інформаційних технологій
Кафедра інженерії програмного забезпечення

КВАЛІФІКАЦІЙНА РОБОТА

Хоменко Вікторії Вадимівни
Прізвище, ім'я, по батькові студента(ки)

на здобуття ступеня вищої освіти Бакалавра

Програмна система для автоматизації аналізу резюме відповідно до вимог вакансії
Назва теми

Галузь знань 12 «Інформаційні технології»
Спеціальність 121 «Інженерія програмного забезпечення»
Освітня програма Освітньо-професійна програма «Інженерія програмного забезпечення»

Шифр КвРІПЗ.2101092.01.18.ПЗ

Виконала студентка IV курсу, група ІПЗ-21-1 _____ Вікторія ХОМЕНКО
Підпис Ім'я, ПРІЗВИЩЕ

Керівник д-р фіз.-мат. наук, проф. _____ Леонід БЕДРАТЮК
Науковий ступінь, вчене звання Підпис Ім'я, ПРІЗВИЩЕ

Нормоконтролер канд. пед. наук, доцент _____ Наталія ПРАВОРСЬКА
Посада Підпис Ім'я, ПРІЗВИЩЕ

До захисту допускаю:
Завідувач кафедри інженерії програмного забезпечення _____ Леонід БЕДРАТЮК
Підпис Ім'я, ПРІЗВИЩЕ

4 червня 2025 р.

Хмельницький 2025

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет Інформаційних технологій
Кафедра Інженерії програмного забезпечення
Рівень вищої освіти Перший (бакалаврський)
Галузь знань 12 «Інформаційні технології»
Спеціальність 121 «Інженерія програмного забезпечення»
Освітня програма Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри ІПЗ
Л. П. Бедратюк
02 01 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Хоменко Вікторії Вадимівні
Прізвище, ім'я, по батькові студента

1. Тема роботи Програмна система для автоматизації аналізу резюме відповідно до вимог вакансії

Керівник роботи Бедратюк Леонід Петрович, д-р фіз.-мат. наук, професор
Прізвище, ім'я, по батькові, науковий ступінь, вчене звання

Затверджена наказом ректора університету від 07.02.2025 р. № 8
2. Строк подання студентом роботи на кафедру 01.06.2025 р.
3. Вихідні дані до роботи Матеріали переддипломної практики
4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)
Дослідження предметної області та постановка завдань, проєктування програмного забезпечення, програмна реалізація програмного продукту.
5. Перелік графічного матеріалу (із зазначенням обов'язкових креслень)
Три креслення:
1. Діаграма варіантів використання
2. Діаграма класів
3. Діаграма зв`язків модулів

1. Консультанти розділів кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Праворська Н. І., канд. пед. наук, доцент	5.05.25	8.05.25
Антиплагіат	Форкун Ю. В., канд. техн. наук, доцент	7.05.25	28.05.25

2. Дата видачі завдання «02» 01 2025р.

КАЛЕНДАРНИЙ ПЛАН

Назва етапів (розділів) кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1 Ознайомлення з тематикою дипломного проєктування, визначення та узгодження індивідуальних тем кваліфікаційних робіт (КвР)	01.12 – 31.12.2024	
2 Збір матеріалу за темою КвР; дослідження предметної області, в якій планується використання програмного забезпечення (ПЗ), визначення задач та вимог, розробка технічного завдання	01.01 – 20.02.2025	
3 Проєктування програмного забезпечення	21.02 – 20.03.2025	
4 Програмна реалізація з використанням відповідних засобів розробки. Тестування ПЗ	21.03 – 30.04.2025	
5 Написання вступу, загальних висновків, оформлення переліку джерел посилання та додатків. Оформлення пояснювальної записки КвР згідно вимог	01.05 – 25.05.2025	
6 Попередній захист КвР	Травень 2025	Згідно графіка
7 Перевірка КвР на плагіат, нормоконтроль, отримання відгуків, рецензій та інших супровідних документів. Брошурування (зшиття) пояснювальної записки.	26.05 – 30.05.2025	
8 Здача КвР на кафедру; підготовка КвР для розміщення у репозитарії ХНУ; підготовка до захисту та захист КвР	з 01.06.2025	

Студент _____ Підпис Хоменко В. В. Ініціали, прізвище

Керівник роботи _____ Підпис Бедратюк Л. П. Ініціали, прізвищ

АНОТАЦІЯ

Тема кваліфікаційної роботи: «Програмна система для автоматизації аналізу резюме відповідно до вимог вакансій».

Автор роботи: Хоменко Вікторія Вадимівна.

Керівник роботи: Бедратюк Леонід Петрович.

Пояснювальна записка: 83 с., 37 рис., 2 табл., 3 дод., 30 джерел.

Мета кваліфікаційної роботи – розробка програмної системи у вигляді Telegram-бота для автоматизації аналізу резюме відповідно до вимог вакансій з використанням хмарного сервісу Azure OpenAI.

У кваліфікаційній роботі проведено аналіз предметної області, досліджено сучасні інструменти автоматизації, а також технології обробки природної мови (NLP) та генерацію штучного інтелекту. Визначено функціональні та нефункціональні вимоги до програмного забезпечення, спроєктовано архітектуру системи, описано основні компоненти, реалізовано Telegram-бота, що виконує адаптацію резюме під конкретну вакансію на основі аналізу її опису.

Для реалізації системи використано мову програмування Python, фреймворк для створення Telegram-ботів – aiogram, бібліотеку FPDF для створення PDF-файлів, а також Azure OpenAI для генерації тексту на основі отриманих вхідних даних. Бот приймає від користувача резюме у форматі PDF та посилання на вакансію, аналізує інформацію з обох джерел, формує адаптований варіант резюме відповідно до вимог вакансії та надсилає результат користувачеві у вигляді нового PDF-файлу.

Практичне значення роботи полягає в автоматизації процесу адаптації резюме під конкретну вакансію. Це значно скорочує час пошуку роботи кандидатам, покращує якість поданих документів та підвищує ймовірність позитивного рішення з боку роботодавців. Система може використовуватись рекрутерами, кар'єрними консультантами, платформами з пошуку роботи та безпосередньо кандидатами.

01.06.2025
Дата

Підпис

ВІДОМІСТЬ ДОКУМЕНТІВ

№ рядка	Формат	Позначення документу	Найменування документу	К-сть	№ екз.	Примітка
			Текстові документи			
1	А4	КвРІПЗ.2101092.01.18.ПЗ	Пояснювальна записка	83		
2	А4		Завдання на кваліфікаційну роботу	1		
3	А4		Анотація	1		
			Графічні документи			
4	А4	КвРІПЗ.2101092.01.18.Е8	Діаграма варіантів використання	1		
5	А4	КвРІПЗ.2101092.01.18.Е8	Діаграма зв'язку модулів	1		
6	А4	КвРІПЗ.2101092.01.18.Е8	Діаграма класів	1		
7	А4		Презентаційні матеріали	12		

Змн.	Арк.	№ докум.	Підпис	Дата	КвРІПЗ.2101092.01.18.ПЗ		
Виконала		Хоменко В.В.		1.05.25	Програмна система для автоматизованого аналізу резюме відповідно до вимог вакансій	Літ.	Арк. 5 / Акрушів 83
Керівник		Бедратюк Л.П.		8.05.25			
Н. Контр.		Праворська Н.І.		8.05.25		ХНУ. ІПЗ-21-1	
Зав. каф.		Бедратюк Л.П.		8.05.25			

ЗМІСТ

ВСТУП

Розвиток автоматизованих систем аналізу резюме значно спрощує процес працевлаштування, роблячи його швидшим, точнішим та менш упередженим. Традиційні методи перегляду резюме займають багато часу, що призводить до втрати ефективності та потенційно якісних кандидатів. Автоматизація цього процесу дозволяє швидко оцінювати відповідність кандидатів до вакансій, підвищуючи якість підбору персоналу. Завдяки технологіям штучного інтелекту, машинному навчанню та обробці природної мови можливо глибше аналізувати текст резюме, виділяти ключові навички та адаптувати зміст під конкретні вакансії. Точність цього аналізу безпосередньо впливає на успішність кандидатів та ефективність рекрутингу, дозволяючи знайти найкращу відповідність між шукачами роботи та роботодавцями.

Даний проект спрямований на створення Telegram-бота, який автоматизує процес адаптації резюме відповідно до конкретної вакансії. Бот дозволяє користувачам завантажити своє резюме у форматі PDF та вказати посилання на вакансію. Використовуючи інструменти аналізу тексту, система витягує ключові вимоги вакансії та співвідносить їх із досвідом і навичками кандидата. Це дає можливість створити персоналізовану версію резюме, яка краще підходить під конкретні вимоги. Інтеграція з Azure OpenAI забезпечує якісний аналіз тексту, допомагаючи не лише адаптувати резюме, а й виявляти можливі слабкі місця та шляхи для їх покращення.

Результатом роботи бота є оновлене резюме у форматі PDF, яке кандидат отримує у відповідь, що значно спрощує процес підготовки заявки на вакансію. Такий підхід сприяє підвищенню шансів на працевлаштування, зменшуючи час, витрачений на адаптацію резюме вручну. Система автоматизованого аналізу проходить тестування, щоб гарантувати відповідність реальним вимогам ринку праці та очікуванням роботодавців. У центрі дослідження – можливості

застосування штучного інтелекту для оптимізації процесу подачі резюме, що відкриває нові перспективи для автоматизації рекрутингу та підвищення його ефективності.

У дослідженні застосовуються численні фундаментальні методології. Воно починається з вивчення поточних галузевих практик, викликів та обмежень через дослідження існуючих підходів до аналізу резюме. Методи обробки природної мови (NLP) є важливими для вилучення та обробки текстової інформації з резюме та посадових інструкцій, що дозволяє системі визначати критичні навички, досвід та кваліфікацію. Впровадження та тестування програмного забезпечення на функціональність, зручність та надійність є основними завданнями етапу розробки. Ефективність системи оцінюється шляхом аналізу точності та продуктивності алгоритму, який визначає, якою мірою ШІ узгоджує облікові дані з посадовими інструкціями. Зрештою, конкурентоспроможність і переваги системи оцінюються шляхом порівняння результатів з існуючими рішеннями.

Розробка Telegram-бота для автоматизованої адаптації резюме під вакансію базується на використанні сучасних технологій, таких як Python, бібліотека FPDF для створення PDF-файлів та фреймворк aiogram для роботи з Telegram API. Процес починається з того, що кандидат завантажує своє резюме у форматі PDF та надсилає посилання на вакансію. Бот аналізує отримані дані, виділяє ключові моменти та адаптує резюме відповідно до вимог вакансії, використовуючи правильний промт для моделі штучного інтелекту. На основі цього формується новий документ, який згодом повертається користувачеві у вигляді оновленого PDF-файлу.

Автоматизований аналіз резюме є важливим інструментом, що підвищує ефективність процесу підбору персоналу. Система дозволяє швидко оцінювати відповідність навичок кандидата до вакансії, усуваючи суб’єктивні фактори, притаманні традиційному аналізу резюме. Завдяки цьому підходу скорочується час на обробку заявок, а точність відповідності кандидатів зростає, що підвищує шанси на успішне працевлаштування.

Фундаментальна частина роботи над ботом включає дослідження існуючих методів аналізу резюме та обробки природної мови (NLP), що дає змогу ефективно вилучати важливу текстову інформацію з документів. Застосування Azure OpenAI для контекстного аналізу дозволяє створити більш точний і релевантний опис професійного досвіду кандидата відповідно до вакансії. У ході розробки система тестується на предмет функціональності, точності та продуктивності, а результати оцінюються шляхом порівняння з традиційними методами скринінгу кандидатів.

Новизна підходу полягає у використанні Telegram-бота як платформи для взаємодії з кандидатом у реальному часі. На відміну від стандартних алгоритмів зіставлення ключових слів, система здатна аналізувати не лише явні, а й приховані зв’язки між компетенціями кандидата та вимогами роботодавця. Взаємодія в режимі реального часу в Telegram робить процес аналізу більш зручним і доступним для користувачів.

Практичне значення цієї роботи полягає в тому, що вона може покращити процес найму на роботу, надаючи зручний та ефективний інструмент для автоматизованого аналізу резюме. Система допомагає скоротити час на перегляд резюме, надаючи структуровану та релевантну інформацію, яка спрощує вибір найбільш відповідних кандидатів. Програмна система прискорює оцінку кандидатів, швидко оцінюючи відповідність їхніх резюме вимогам вакансії, тим самим скорочуючи час і зусилля, необхідні для ручного відбору. Завдяки інтеграції з Azure OpenAI бот забезпечує більш якісний аналіз, що покращує як ефективність підбору, так і досвід кандидатів на роботу. Ця технологія є універсальною і може бути адаптована під різні сфери діяльності, що робить її цінним інструментом для автоматизації підбору персоналу. Персоналізовані рекомендації щодо покращення резюме підвищують ймовірність успіху кандидатів. Завдяки своїй адаптивності та масштабованості рішення є вигідним у різних галузях та для різних вимог до найму. Система також сприяє впровадженню технологій рекрутингу на основі штучного інтелекту, ілюструючи потенціал автоматизації для покращення процесу прийняття кадрових рішень.

У цій роботі в систематизованому вигляді представлено розробку та впровадження автоматизованої системи аналізу та адаптації резюме до вимог вакансій. Дослідження починається зі вступу, в якому розглядається актуальність теми, мета та завдання роботи, об’єкт та предмет дослідження, постановка проблеми та цілі, методи дослідження, практична значущість системи, її новизна та структура роботи.

Перший розділ містить огляд сучасних методологій аналізу та адаптації резюме, основних викликів у сфері рекрутингу, а також можливостей використання штучного інтелекту для автоматизації обробки текстових даних.

Другий розділ присвячений проєктуванню програмного забезпечення. У ньому розглядається вибір архітектури системи, описуються її основні компоненти, проводиться декомпозиція модулів, обґрунтовується вибір алгоритмів обробки тексту та порівнюються методи NLP для аналізу відповідності резюме вакансії. Також у цьому розділі описується взаємодія користувача з Telegram-ботом та вибір технологій для розробки.

Третій розділ містить детальний опис процесу реалізації системи, включаючи алгоритм аналізу тексту, методи вилучення даних із PDF-файлів, генерацію адаптованого резюме у форматі PDF, а також інтеграцію всіх компонентів (Telegram-бота, OpenAI API, PDF-генератора) в єдину систему.

Четвертий розділ присвячений тестуванню розробленої системи. Він охоплює методику тестування, вибір тестових наборів даних, оцінку продуктивності та точності адаптації резюме під вакансію, а також аналіз отриманих результатів.

Робота завершується висновками, де підбиваються підсумки проведеного дослідження, визначаються перспективи розвитку системи та можливі напрямки її вдосконалення. Окремо представлена пояснювальна записка, яка містить детальний опис технічних та програмних рішень, що використовуються в системі.

1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Змістовий аналіз предметної області

Метою цього дослідження є створення програмної системи, яка автоматизує аналіз резюме за допомогою Telegram-боту, використовуючи Azure OpenAI для адаптації кандидата під вимоги вакансій. Ця система вирішує одну з ключових проблем сучасного конкурентного ринку праці: ручне оцінювання резюме рекрутерами та претендентами, що є неефективним і займає багато часу. [1] Використовуючи штучний інтелект і технології обробки природної мови (NLP), система покликана прискорити та підвищити точність співставлення резюме з описами вакансій. Автоматизація аналізу резюме та інтеграція штучного інтелекту є ключовими компонентами цієї системи. Використання NLP дозволяє аналізувати текстову інформацію з вакансій та резюме, а інтеграція Azure OpenAI забезпечує глибше семантичне розуміння змісту. Завдяки цьому система виходить за межі простого порівняння ключових слів і забезпечує більш точну оцінку відповідності кандидата посаді. [2]

Структура резюме має значний вплив на ефективність аналізу. Для коректної роботи системи важливо глибоко розуміти його основні компоненти. Як правило, резюме містить такі розділи: особиста інформація, професійний підсумок, ключові навички, досвід роботи, освіта, сертифікації, волонтерство та контактні дані. Аналізуючи ці розділи, система оцінює, наскільки кваліфікація кандидата відповідає вимогам вакансії та може адаптувати необхідні навички до цих вимог. Використовуючи штучний інтелект, програма порівнює ключові навички та досягнення, створюючи точні й корисні висновки у форматі PDF-файлу для рекрутерів і кандидатів. Основу порівняння резюме з вакансіями становлять їхні вимоги. Вони включають кваліфікацію, навички, досвід та інші характеристики, які залежать від галузі, компанії та конкретної ролі. Наприклад кваліфікація, охоплює рівень освіти, сертифікації або спеціальні ступені, необхідні для посади. Для

розробника програмного забезпечення може вимагатися диплом з комп'ютерних наук, а для маркетолога – освіта у сфері бізнесу або комунікацій.

Навички поділяються на технічні та м'які (soft skills). До технічних належать знання мов програмування, програмних засобів, методологій управління проєктами чи аналізу даних. М'які навички, такі як комунікація, командна робота та вміння вирішувати проблеми, також часто є важливими вимогами.

Досвід вимірюється у роках роботи або у конкретних професійних досягненнях. Це може включати попередні місця роботи, стажування, фриланс або інші проєкти, що відповідають обов'язкам посади. Додатково можуть враховуватися інші характеристики, такі як рівень володіння мовами, географічні вподобання, готовність до релокейту, лідерські якості або знання спеціалізованих інструментів.

Система аналізу та адаптації резюме автоматично витягує ці вимоги з описів вакансій та за допомогою штучного інтелекту порівнює їх з даними кандидатів, адаптуючи рівень відповідності кожного претендента. іШтучний інтелект змінює процес та етапи найму персоналу. Раніше такі завдання, як перегляд резюме, оцінка навичок та перші співбесіди, виконувалися вручну, що вимагало багато часу та ресурсів. Завдяки автоматизації цих етапів ШІ допомагає HR-фахівцям зробити процес найму більш ефективним, орієнтованим на дані та позбавленим суб'єктивних упереджень, а претендентам дає можливість адаптувати свій досвід і навички до вимог вакансії роботи їх мрії. [3]

Сучасні системи можуть швидко аналізувати велику кількість заявок і точно визначати найбільш відповідних кандидатів. Тому аналіз резюме за допомогою ШІ базується на автоматичній обробці тексту. Ця технологія порівнює ключові дані, такі як освіта, досвід та навички, із вимогами вакансії. Завдяки методам обробки природної мови (NLP) комп'ютери можуть розуміти людську мову, виявляти ключові фрази та аналізувати їхній контекст. Це дозволяє точніше оцінювати відповідність кандидата посаді, ніж звичайний пошук за ключовими словами. На відміну від традиційного підходу, що базується на збігу слів, аналіз тексту за допомогою ШІ враховується значення та взаємозв'язки між поняттями. Навіть якщо

у резюме використано інші формулювання, ніж у вакансії, система може розпізнати схожі навички та кваліфікації. Це дозволяє проводити більш об’єктивний і точний аналіз можливостей кандидатів, підвищуючи швидкість та ефективність процесу найму.

Разом із перевагами автоматизації виникають і виклики. Великий потік заявок і складні вимоги до кандидатів ускладнюють процес порівняння резюме з вакансіями. Резюме можуть мати різні структури, стилі оформлення та формулювання, що ускладнює пряме порівняння. Опис вакансій також може містити специфічну термінологію, яка не завжди збігається з формулюваннями кандидатів. Через це кваліфіковані спеціалісти можуть залишитися поза увагою лише тому, що їхнє резюме не містить певних ключових слів. Ще однією проблемою є надлишкова інформація у резюме, наприклад, особисті дані, хобі чи досвід, який не має безпосереднього відношення до вакансії. Це може заважати виявленню найважливіших компетенцій кандидата.

Щоб вирішити ці труднощі, сучасні системи аналізу та адаптації резюме поєднують машинне навчання та NLP. Вони здатні розпізнавати зв’язки між навичками та посадовими обов’язками, а також виділяти ключові дані, навіть якщо їхнє формулювання відрізняється. Завдяки автоматизації можливо ефективно аналізувати велику кількість резюме, зменшуючи навантаження на HR-фахівців.

Однак, навіть з урахуванням усіх переваг, ідеальне співставлення резюме та вакансії залишається складним завданням. ШІ та NLP інколи стикаються з труднощами при обробці мовних нюансів, таких як тональність, наміри кандидата чи нетиповий професійний досвід. Недостатньо якісне навчання моделей або надмірна залежність від певних шаблонів даних можуть призводити до алгоритмічних упереджень. Хоча автоматизований аналіз та адаптація значно підвищує ефективність найму, комплексна оцінка кваліфікації кандидата все ще залишається непростою задачею.

У сфері HR ручна перевірка великої кількості заявок є поширеною практикою, проте зіставлення резюме з вакансіями завжди залишається складним завданням.

Точне порівняння навичок і кваліфікацій ускладнюється через різні формати резюме та відмінності у термінології. Кандидати можуть використовувати різні формулювання для опису своїх компетенцій, що може призводити до того, що навіть найкращі спеціалісти залишаються поза увагою.

Щоб підвищити ефективність цього процесу, технології штучного інтелекту все активніше впроваджуються в HR-процеси. Машинне навчання та NLP дозволяють швидко та точно аналізувати велику кількість резюме, розуміючи не лише окремі слова, а й загальний зміст і контекст. Це дозволяє системам виходити за межі простого пошуку ключових слів і виявляти як очевидні, так і менш явні кваліфікації кандидатів. Такий підхід робить процес найму більш об’єктивним і допомагає знаходити найкращих спеціалістів незалежно від формулювань, які вони використовують у своєму резюме. Попри ці переваги, автоматичний аналіз тексту має і свої виклики. ШІ все ще може некоректно інтерпретувати мовні нюанси та контекст вакансій. Незвичні формулювання чи нестандартні структури резюме можуть вводити систему в оману. Додаткові труднощі створюють терміни, значення яких залежить від контексту. Якщо система не навчена на достатньо різноманітному наборі даних, це може призводити до помилок у зіставленні резюме з вакансіями.

Автоматизований аналіз резюме має очевидні переваги. Він значно прискорює процес рекрутингу, скорочуючи час на ручну перевірку заявок та дозволяючи HR-фахівцям зосередитися на більш важливих етапах – оцінці кандидатів та співбесідах. Такі системи ефективно справляються з великим обсягом заявок, допомагаючи компаніям швидше знаходити відповідних спеціалістів. Завдяки аналізу на основі даних вони не лише зменшують ризик упередженості, а й покращують якість прийнятих рішень у процесі найму.

1.2 Аналіз наявного програмно-технічного забезпечення предметної області

Існує безліч програмних рішень для автоматизації аналізу та адаптації резюме, і кожне з них використовує різні технології, щоб покращити процес рекрутингу. Одним із яскравих прикладів є LinkedIn (рисунок 1.1), який інтегрував алгоритми аналізу резюме у свою платформу. Ці алгоритми допомагають як рекрутерам, так і кандидатам, оцінюючи відповідність резюме вимогам вакансії на основі ключових слів і галузевих тенденцій. Крім того, LinkedIn надає рекомендації щодо оптимізації профілю, допомагаючи кандидатам ефективніше презентувати свої навички та досвід. [4]

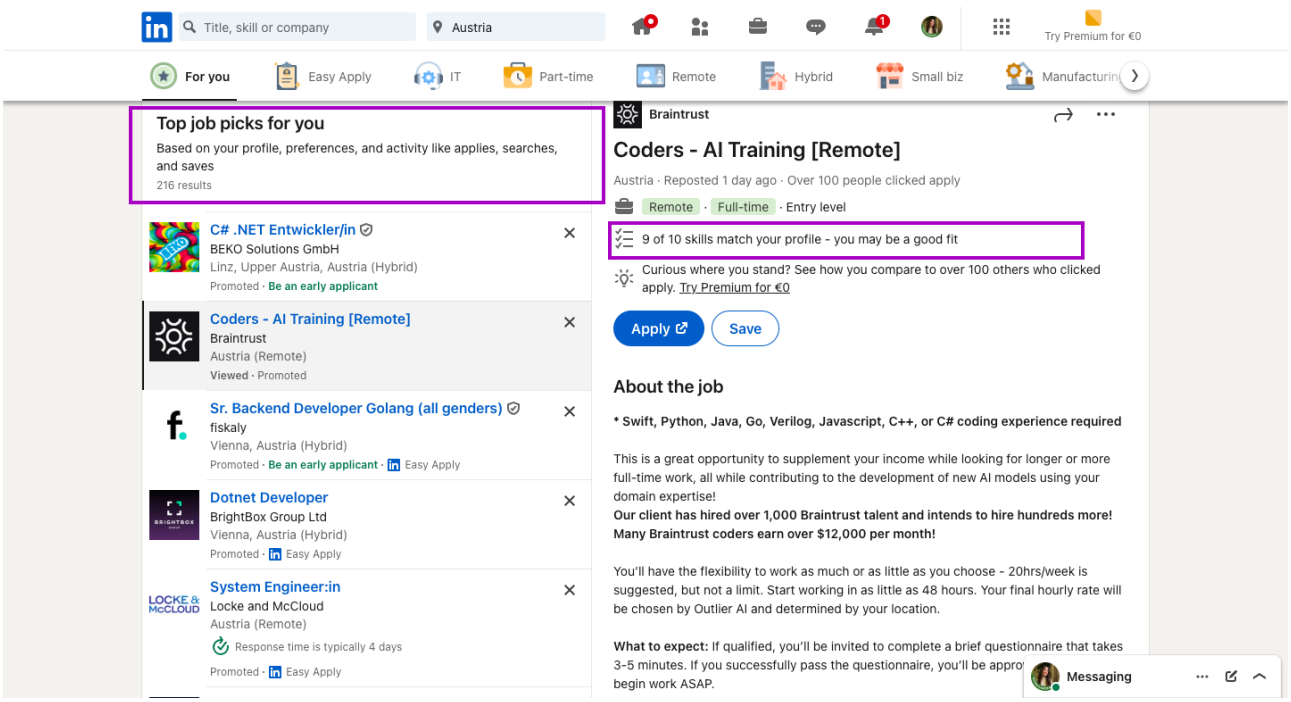


Рисунок 1.1 – Аналіз релевантності на платформі LinkedIn.

Окрім LinkedIn, існують й інші інструменти, такі як Resume Worded і Skillate, що працюють на базі технологій GPT. Вони використовують мовні моделі для глибокого аналізу резюме та описів вакансій, надаючи зворотний зв'язок, щодо релевантності досвіду та зрозумілості формулювань. Завдяки персоналізованим рекомендаціям ці платформи допомагають кандидатам адаптувати свої заявки відповідно до конкретних вимог роботодавців. Такі рішення стають дедалі

популярнішими, адже вони забезпечують глибше розуміння контексту, ефективну обробку даних і високу точність аналізу. [5]

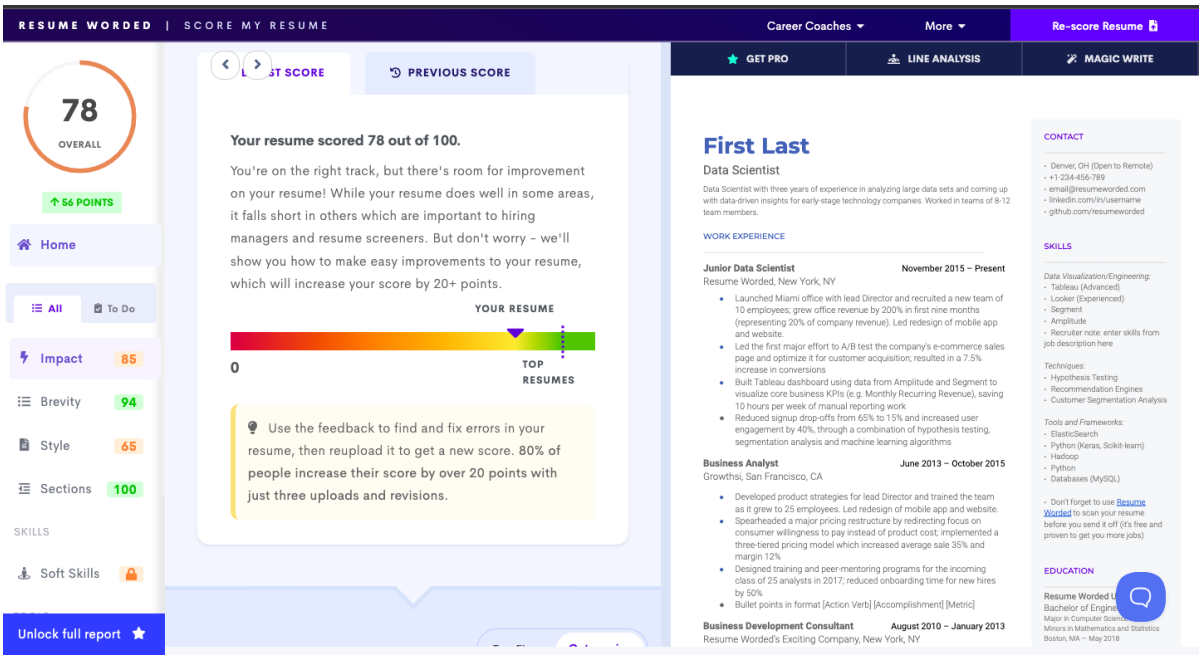


Рисунок 1.2 – Сторінка редагування резюме від Resume Worded.

Важливу роль у процесі підбору персоналу відіграють системи відстеження кандидатів (Applicant Tracking Systems, ATS). Вони автоматично аналізують і ранжують резюме на основі відповідності вимогам вакансії, що значно полегшує рекрутерам роботу з великим потоком заявок. Однак такі системи здебільшого покладаються на пошук ключових слів, що може призвести до відсіювання кваліфікованих кандидатів, чиї резюме не оптимізовані під ці алгоритми, навіть якщо вони відповідають посаді.

Щоб допомогти кандидатам адаптувати свої резюме, існують також онлайн-сервіси аналізу, які доповнюють роботу ATS. Використовуючи складні алгоритми та технології обробки природної мови, вони порівнюють резюме з описами вакансій і надають рекомендації щодо покращення змісту. Такі сервіси роблять процес працевлаштування більш ефективним та об’єктивним, надаючи корисні інструменти як для рекрутерів, так і для шукачів роботи. А ефективність рішень для аналізу резюме безпосередньо залежить від використаних технологій та їхніх можливостей. Системи ATS, що базуються на скануванні та категоризації за ключовими словами,

допомагають великим компаніям керувати потоком заявок. Вони значно прискорюють обробку резюме, проте часто не враховують важливі навички та досвід, якщо вони не виражені у точному формулюванні. Точність зіставлення кандидатів із вакансіями багато в чому визначається можливістю налаштування параметрів пошуку. Незважаючи на те, що сучасні ATS надають гнучкі фільтри для більш точного відбору, у багатьох випадках вони не можуть врахувати контекст навичок або розпізнати релевантний досвід, якщо він описаний інакше, ніж у вакансії.

Рисунок 1.3 – Система ATS.

З цими обмеженнями традиційних ATS систем справляються новітні рішення на базі штучного інтелекту та машинного навчання. Використовуючи сучасні можливості обробки природної мови, такі системи, зокрема ті, що працюють на базі Azure OpenAI, проводять більш детальний аналіз. Вони здатні розуміти контекст і зв’язки між навичками та досвідом, що дозволяє точніше порівнювати кандидатів із вимогами вакансій. Тому сучасні рекрутингові платформи все більше покладаються на технології NLP, які допомагають аналізувати резюме та описи вакансій не лише за ключовими словами, а й з урахуванням їхнього контексту. Це не лише економить час рекрутерів, але й надає кандидатам корисний зворотний зв’язок щодо відповідності їхнього досвіду вимогам ринку праці. Просунуті платформи значно

перевершують традиційні системи, що працюють виключно з ключовими словами. У той час як прості алгоритми можуть не помітити кваліфікованих кандидатів через відмінності у формулюваннях, рішення на базі NLP та AI здатні розпізнавати різноманітні формати опису навичок та досвіду, не обмежуючись жорсткими збігами термінів.

Основна відмінність між цими двома підходами полягає в тому, наскільки добре система розпізнає структуру мови. Традиційні алгоритми часто не враховують синоніми або контекст, у той час як NLP-системи здатні аналізувати широкий спектр мовних конструкцій. Це особливо важливо у сферах, де одна й та сама навичка може бути описана різними термінами. Попри суттєві покращення у сфері автоматизованого аналізу резюме, певні обмеження все ще залишаються. Класичні системи, що орієнтуються на ключові слова, часто не здатні врахувати нюанси опису досвіду й можуть не розпізнати кваліфікованих кандидатів, якщо їхнє резюме сформульоване інакше, ніж текст вакансії. Через це такі системи інколи надають перевагу добре оптимізованим резюме, а не кандидатам із найкращими навичками, оскільки не можуть точно оцінити їхній професійний рівень.

Завдяки аналізу великих обсягів даних алгоритми виявляють закономірності між вимогами вакансій і реальними навичками кандидатів, що робить процес оцінки більш адаптивним і точним. Однак і тут є свої труднощі: необхідність у значних обсягах навчальних даних, ризик алгоритмічних упереджень та складність інтерпретації рішень системи. Ці фактори вказують на важливість постійного вдосконалення AI-рішень для рекрутингу, щоб зробити їх ще більш точними та справедливими.

1.3 Аналіз сучасних методів

В останні роки відбулося суттєве покращення в класифікації та оцінці релевантності резюме завдяки застосуванню як класичних, так і методів глибокого навчання. Аналіз тексту базується на класичних алгоритмах обробки природної мови (NLP), включаючи Word2Vec та TF-IDF. TF-IDF кількісно оцінює значущість слів у документах у відношенні до більш широкого корпусу, тим самим сприяючи ідентифікації критичних навичок у резюме, які відповідають описам вакансій. Word2Vec захоплює семантичне значення та взаємозв'язки через контекст, перетворюючи слова на вектори. Хоча ці методи ефективні для видобутку ознак, вони залежать від попередньо визначених ознак і не здатні захоплювати складні контекстуальні взаємозв'язки. [6]

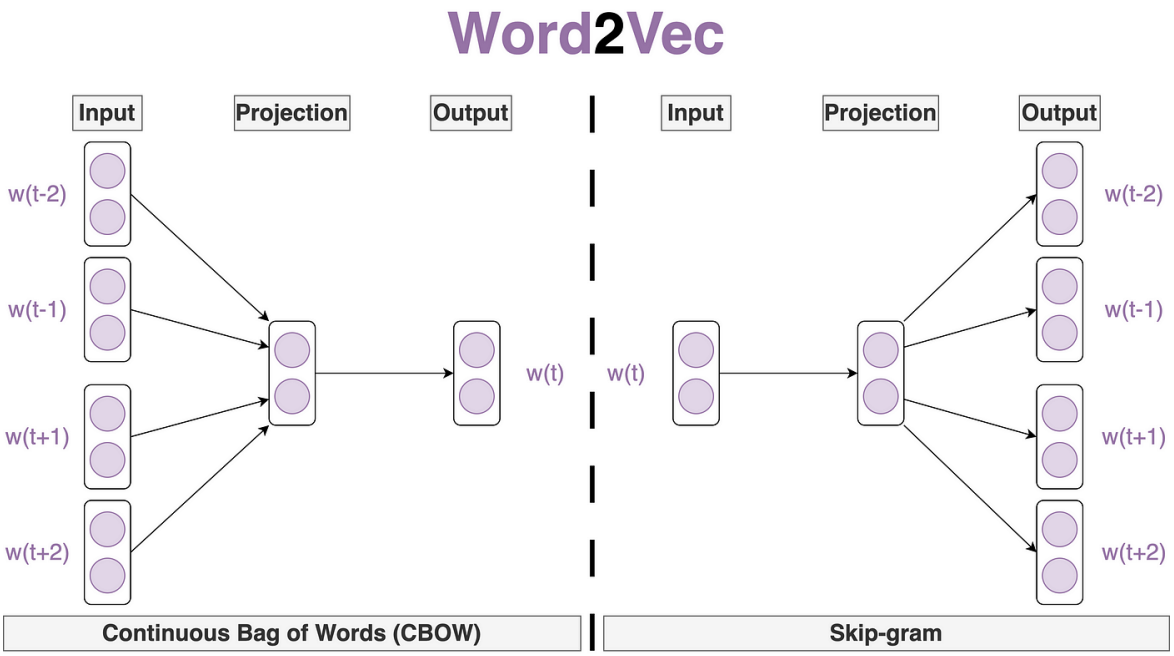


Рисунок 1.4 – Word2Vec схема.

Нейронні мережі та трансформери замінили класичні методи, оскільки глибоке навчання впровадило більш складні техніки. Завдяки розумінню окремих слів та їхніх взаємозв'язків у реченнях і абзацах, трансформери, такі як BERT, змінили процес обробки мови. Ця здатність робить їх особливо ефективними в

оцінці релевантності резюме, оскільки вони можуть проводити більш детальну оцінку кваліфікацій кандидатів. Ці системи є більш адаптивними до складного аналізу тексту завдяки їхній здатності автоматично отримувати відповідні характеристики з даних, на відміну від класичних підходів. Вони винятькові в розумінні нюансів кваліфікацій та управлінні різноманітними термінами резюме. Моделі глибокого навчання пропонують значно точніший аналіз резюме, ніж традиційні методи, незважаючи на їхню обчислювальну інтенсивність і потребу в даних.

Автоматизована оцінка резюме була значно покращена завдяки складним моделям, таким як BERT і GPT-4. Сила BERT полягає в його контекстуальному розумінні слів у реченнях, що дозволяє більш тонко інтерпретувати описи вакансій та кваліфікацій. [7] GPT-4 покращує ці можливості, створюючи контекстуально доречний, зв'язний текст, який не лише порівнює резюме з описами вакансій, але й надає рекомендації щодо покращення. [8] Ці моделі особливо добре справляються з тонкощами природної мови, такими як синоніми та неявні значення, які часто плутають системи, основані на ключових словах.

Ефективність цих моделей у класифікації резюме оцінюється за допомогою різноманітних показників якості. Метрики BLEU, ROUGE та F1-score оцінюють ступінь відповідності виходів моделі очікуваним результатам. ROUGE та BLEU оцінюють відповідність тексту між описами вакансій та резюме, тоді як F1-score збалансовує точність і повноту в ідентифікації відповідних навичок та досвіду. Ці метрики є незамінними для вдосконалення моделей та забезпечення високоякісного аналізу в автоматизованому відборі резюме.

Автоматизований аналіз резюме стикається з унікальними викликами при обробці тексту українською та англійською мовами. Слов'янська мовна структура української мови вимагає спеціалізованих підходів, незважаючи на те, що англійська є домінуючою мовою в глобальному спілкуванні. Розбір тексту підлягає впливу складної граматики, лексики та морфології української мови, яка відрізняється від англійської. Її відмінювані слова можуть набувати різних форм залежно від їхньої

граматичної функції, що вимагає від моделей НЛП розуміння як кореневих значень, так і варіацій у часі, відмінку та числі. Тлумачення значення також залежить від гнучкого порядку слів в українській мові.

Важливо вирішити ці мовні виклики за допомогою алгоритмів обробки природної мови (NLP). Для розбору тексту та ідентифікації ключових компонентів обидві мови використовують такі методи, як розпізнавання іменованих сутностей (NER), маркування частин мови та токенізація. Щоб врахувати унікальні синтаксичні та семантичні принципи кожної мови, ці алгоритми потребують специфічного для мови навчання.

Розвинені моделі, такі як BERT і GPT-4, все більше здатні ефективно обробляти як український, так і англійський текст. Попри структурні відмінності між двома мовами, ці інструменти підвищують точність аналізу резюме, захоплюючи основні мовні зв'язки та контекстуальні значення.

Резюме та описи вакансій перетворюються на формати, зручні для машинного читання, через векторизацію тексту. TF-IDF, Word2Vec та BERT перетворюють текст на числові вектори, які захоплюють семантичне значення. TF-IDF підкреслює значущі терміни, оцінюючи їх частоту та релевантність до корпусу. Word2Vec генерує вектори слів на основі контексту, визначаючи зв'язки між термінами, такими як "розробник" і "програміст". Глибокий навчальний підхід BERT дозволяє детально аналізувати досвід і навички, захоплюючи значення слів у більш широких контекстах речень.

Тоді алгоритми класифікації тексту оцінюють відповідність між резюме та вакансією. Прості зв'язки ознак і структуровані дані добре підходять для традиційних методів, таких як Random Forest і Support Vector Machines (SVM). Нейронні мережі, особливо RNN та трансформери, особливо добре справляються з контекстним аналізом та розпізнаванням шаблонів при роботі зі складними текстовими даними. Ці складні моделі здатні навчатися на неструктурованому тексті

та адаптуватися до мовних нюансів, тим самим сприяючи точному співвідношенню кандидатів і вакансій. [9]

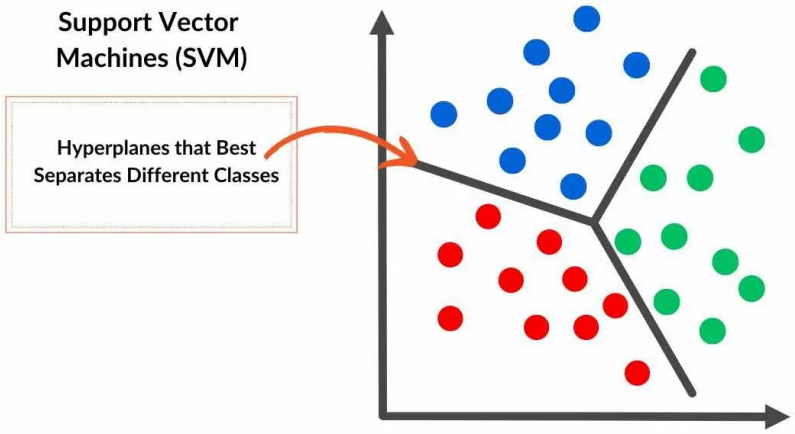


Рисунок 1.5 – Принцип SVM.

Оцінка відповідності резюме вакансії є важливою для процесу набору персоналу. Кількісні показники релевантності забезпечуються за допомогою порівняння контенту та показників відповідності, які варіюються від простого збігу ключових слів до складного машинного навчання.

Ефективність системи оцінюється за допомогою показників продуктивності, таких як точність, повнота та коефіцієнт Дайса. Точність і повнота оцінюють точність і всебічність прогнозів, тоді як показник Дайса кількісно визначає перекриття ключових слів між описами вакансій і резюме. Точне визначення відповідних кваліфікацій підтверджується високими показниками в обох метриках.

F1-оцінка є збалансованим показником продуктивності, який враховує точність і повноту, що робить її особливо корисною для нерівномірного розподілу релевантних і нерелевантних резюме. Ці метрики сприяють вдосконаленню системи та забезпечують складну оцінку резюме, яка перевершує просте співпадіння ключових слів. [10]

1.4 Визначення вимог для програми

Автоматизована система адаптації резюме під конкретну вакансію забезпечує зручний та ефективний процес взаємодії між користувачем і програмним забезпеченням. Головна її функція полягає у прийомі резюме та вакансій через інтерфейс Telegram-бота, обробці отриманих даних і створенні оновленого документа, максимально відповідного вимогам роботодавця.

Кандидат надсилає своє резюме у форматі PDF разом із посиланням на вакансію, після чого система аналізує їх вміст, виокремлюючи ключові аспекти, такі як кваліфікація, досвід, навички та вимоги до посади. Для цього використовуються технології обробки природної мови (NLP), що дозволяють оцінити відповідність інформації у резюме до зазначених критеріїв вакансії.

Застосовуючи спеціалізовані алгоритми аналізу тексту, система автоматично адаптує резюме відповідно до специфікацій вакансії, коригуючи формулювання та розставляючи акценти на найбільш релевантних компетенціях кандидата. Після цього оновлене резюме формується у вигляді нового PDF-документа за допомогою бібліотеки FPDF та надсилається користувачу для подальшого використання.

Такий підхід забезпечує автоматизацію процесу коригування резюме, значно підвищуючи шанси кандидата на успішне проходження відбору, а також скорочуючи час, необхідний для підготовки документів. Інтеграція з Telegram робить систему простою у використанні, а застосування передових алгоритмів гарантує високу якість аналізу та адаптації.

Основна функціональність системи оцінює, наскільки резюме відповідає вимогам опису роботи. Він витягує та порівнює критично важливі компоненти з обох документів, такі як необхідні кваліфікації, досвід, здібності та інші критерії. Завдяки використанню моделей обробки природної мови (NLP) та машинного навчання, система оцінює загальну релевантність досвіду кандидата та визначає, чи резюме ефективно демонструє кваліфікації та навички, які шукає роботодавець. Потім споживачі отримують персоналізований зворотний зв'язок від системи. Цей

зворотний зв'язок допомагає кандидатам покращити свої резюме, пропонуючи способи підкреслити відповідні навички та досвід. Він виявляє розбіжності між вимогами до роботи та резюме, такі як відсутність відповідних навичок або досвіду. Бот допомагає користувачам вдосконалювати свої резюме, щоб підвищити їхні шанси на проходження співбесіди, надаючи прості, дієві рекомендації.

Після аналізу система генерує PDF-файл, який можна завантажити і який містить підсумок результатів та відповідні рекомендації. Цей документ окреслює ступінь відповідності резюме опису роботи, підкреслюючи як його сильні сторони, так і області для покращення. Він надає конкретні рекомендації щодо підкреслення навичок, перегляду розділів та налаштування резюме під конкретну вакансію. Цей PDF-файл слугує практичним ресурсом як для термінових, так і для майбутніх заявок.

Система також веде всебічний журнал процесів, який відстежує всі дії бота, включаючи отримання документів, етапи аналізу та рекомендації. Це забезпечує прозорість і сприяє оптимізації системи, документуючи будь-які помилки або проблеми, які можуть виникнути. Це ведення журналу є необхідним для підвищення точності, моніторингу продуктивності та забезпечення надійності шляхом використання історичних даних. Система може постійно покращувати свою здатність обслуговувати користувачів, ведучи детальні записи.

Для забезпечення ефективної, безпечної та надійної роботи бота з адаптації резюме необхідно враховувати нефункціональні вимоги. Система повинна гарантувати безпеку даних користувачів, включаючи резюме та описи вакансій, впроваджуючи надійні методи зберігання інформації. Щоб уникнути несанкціонованого доступу або змін, критично важливим є шифрування даних як під час передачі, так і під час зберігання. Оскільки резюме містять приватну та конфіденційну інформацію, необхідно дотримуватись протоколів безпеки, які відповідають вимогам конфіденційності та галузевим стандартам. Захист персональних даних користувачів є ключовим аспектом надійності системи та довіри до неї. Ще одним важливим аспектом є продуктивність. Незалежно від

кількості одночасних запитів користувачів чи високих навантажень, система повинна швидко та ефективно обробляти резюме та описи вакансій. Алгоритми аналізу тексту мають бути оптимізовані для високої швидкості, одночасно забезпечуючи точність обробки. Від бота очікується миттєва реакція, швидкий аналіз даних та оперативне формування PDF-документів. Щоб гарантувати стабільну роботу, бекенд повинен мати можливість динамічно масштабуватись під час пікових навантажень, забезпечуючи безперервний доступ до сервісу та якісний користувацький досвід.

Також система повинна мати високу масштабованість, щоб адаптуватися до зростаючого попиту без втрати ефективності. Вона має підтримувати розширення обчислювальних ресурсів і обсягів зберігання відповідно до збільшення кількості оброблюваних резюме та вакансій. Архітектура повинна передбачати можливість розширення через хмарні сервіси або інші рішення, які дозволяють гнучко керувати ресурсами.

Інтеграція Telegram значно підвищує доступність сервісу, дозволяючи користувачам взаємодіяти з ботом у зручному форматі. Це спрощує процес надсилання резюме та посилань на вакансії, забезпечуючи зручну та швидку взаємодію. Телеграм-бот повинен мати інтуїтивно зрозумілий функціонал, який дозволяє користувачам легко надсилати документи, отримувати адаптовані версії резюме та завантажувати готові PDF без складних налаштувань або додаткових інструкцій. Користувацький інтерфейс має бути простим та інтуїтивно зрозумілим, щоб навіть люди без технічних знань могли без зусиль взаємодіяти із системою. Чіткі інструкції, логічно організований процес та корисні підказки допоможуть користувачам швидко адаптувати своє резюме під конкретну вакансію. Надійність, зручність та ефективність роботи системи є основними факторами, що забезпечують її успішне використання. В результаті бот стає потужним інструментом для оптимізації процесу працевлаштування, допомагаючи кандидатам ефективніше відповідати вимогам роботодавців.

1.5 Постановка задачі

Завдання полягає в створенні автоматизованої системи, що здатна оцінювати кваліфікаційні дані та визначати їх відповідність вимогам вакансій. Оскільки ринок праці стає дедалі цифровішим, зростає попит на інструменти, які можуть оптимізувати та прискорити процес найму. Тривала задача ручного перегляду резюме для визначення їх відповідності конкретній посаді є критично важливим елементом цього процесу. Це вимагає значних людських зусиль і може призвести до несуперечностей або упереджень у відборі кандидатів.

Розробка системи у формі Telegram-бота, яка може автономно отримувати резюме, аналізувати та адаптувати їх відповідно до опису вакансії та надавати користувачам практичні рекомендації. Щоб отримати відповідну інформацію як з резюме, так і з вакансій, оцінити їхню сумісність і створити персоналізовані рекомендації для покращення резюме, ця система повинна ефективно використовувати методи обробки природної мови (NLP). Користувачі повинні отримувати цінні інсайти для покращення своїх шансів на отримання роботи, а програма повинна працювати безпечно, ефективно та з високою точністю.

Питання полягає в розробці бота, який буде масштабованим, доступним і надійним. Цей бот повинен бути здатний обробляти численні взаємодії з користувачами, обробляти, адаптувати документи та надавати інтелектуальний, заснований на даних аналіз і рекомендації. Він також повинен безперешкодно інтегруватися з існуючими платформами, такими як Telegram, і використовувати потужні інструменти штучного інтелекту, такі як Azure OpenAI.

Здатність системи ефективно аналізувати кваліфікаційні дані та описи вакансій в основному залежить від використання обробки природної мови (NLP) для обробки тексту. NLP дозволяє системі розуміти, інтерпретувати та маніпулювати людською мовою, що полегшує вилучення критично важливої інформації з заявок, включаючи освітній фон, набір навичок, кваліфікацію та досвід роботи. Це також

дозволяє системі аналізувати опис роботи для визначення критичних вимог, включаючи необхідні навички, обов'язки та кваліфікації, які шукає роботодавець.

Система може оцінити ступінь відповідності резюме вимогам вакансії, порівнюючи зміст резюме з критеріями в описі вакансії, використовуючи методи обробки природної мови (NLP). Це включає такі завдання, як семантичний аналіз, розпізнавання іменованих сутностей, тегування частин мови та токенізація. Система здатна розуміти контекст і значення інформації в документах, використовуючи ці методи, які виходять за межі простого збігу ключових слів. Відповідно, система здатна проводити більш точні оцінки кваліфікацій кандидата для посади та пропонувати рекомендації щодо покращення резюме.

Крім того, NLP дозволяє системі обробляти різноманітні формати тексту, такі як заявки, які можуть бути написані в різних стилях або містити невідповідності. Навіть складні резюме з різноманітним форматуванням можуть бути точно проаналізовані системою, яка здатна обробляти як структуровані, так і неструктуровані дані. Система здатна автоматизувати трудомісткий процес відбору резюме за допомогою NLP, забезпечуючи розумне та ефективне рішення як для кандидатів, так і для роботодавців.

Інтеграція GPT-4 через Azure OpenAI є критично важливим компонентом для покращення системи аналізу резюме. GPT-4, мовна модель, розроблена OpenAI, є надзвичайно ефективною у розумінні та створенні тексту, схожого на людський. Це робить його ідеальним вибором для завдань, таких як оцінка відповідності резюме описам вакансій. Бот може скористатися складними можливостями GPT-4 для обробки природної мови та генерації тексту, підключивши систему до Azure OpenAI.

Завдяки цій інтеграції система здатна проводити всебічний аналіз і порівняння змісту вакансій та резюме, що перевищує межі простого збігу ключових слів. GPT-4 здатний розуміти контекст і нюанси мови, що дозволяє системі оцінювати ступінь відповідності резюме вимогам вакансії. Він здатний витягувати цінні інсайти як з структурованого, так і з неструктурованого тексту, тим самим визначаючи основні навички, досвід і кваліфікації, які відповідають опису роботи. Більше того, GPT-4

має можливість створювати персоналізовані рекомендації для покращення резюме, тим самим гарантуючи, що профіль кандидата адаптований до конкретної посади, на яку він претендує.

Платформа Azure OpenAI пропонує безпечне та масштабоване середовище для роботи GPT-4, що дозволяє системі обробляти кілька запитів від користувачів одночасно без втрати продуктивності. Система зможе забезпечити інтелектуальне, автоматизоване рішення для аналізу резюме, яке буде як точним, так і ефективним, шляхом інтеграції GPT-4 у бот. Це безумовно покращить процес набору як для кандидатів, так і для роботодавців.

Інтеграція механізму зворотного зв'язку в систему аналізу резюме є надзвичайно важливою, оскільки вона гарантує, що користувачі отримують цінні інсайти та рекомендації для покращення своїх резюме. Система генерує зворотний зв'язок, який може допомогти користувачеві вдосконалити свою заявку після аналізу резюме відповідно до опису вакансії за допомогою обробки природної мови та GPT-4. Цей механізм зворотного зв'язку повинен бути налаштований відповідно до унікальних вимог користувача, бути дієвим і прозорим.

Надання рекомендацій та висновок має на меті підкреслити ті сфери, в яких резюме може бути недостатнім або може бути покращеним для більш точного відповідності вимогам роботи. Наприклад, може бути рекомендовано включити певні таланти або досвід, які згадуються в описі роботи, але не включені до резюме. Воно також може запропонувати перефразувати певні розділи для покращення зрозумілості або виділити конкретні досягнення, які відповідають основним вимогам посади.

Користувачі отримують відповіді, які легко зрозуміти через Telegram-бота, який функціонує як механізм зворотного зв'язку. Ці відповіді складаються таким чином, що вони не лише інформативні, але й мотивуючі, запевняючи користувачів у їхній впевненості в здатності покращити свої резюме. Система також повинна надавати пропозиції, які є реалістичними та здійсненними, тим самим запобігаючи впровадженню змін, які є надто складними або нереалістичними. Крім того, система

може запропонувати користувачам переглянути свої резюме відповідно до отриманих відгуків, і вони можуть повторно надіслати свої оновлені резюме для другого етапу аналізу, якщо бажають.

Загалом, механізм зворотного зв'язку є критично важливою функцією, яка покращує процес автоматизованого аналізу резюме. Це підвищує шанси користувачів на отримання бажаної роботи, надаючи їм необхідні інструменти для покращення їхніх резюме, а також сприяючи більш персоналізованому та захоплюючому досвіду.

У розробці автоматизованої системи аналізу резюме важливо тестувати та оцінювати якість роботи, щоб забезпечити правильне функціонування системи та отримання надійних, точних результатів. Процес передбачає оцінку роботи бота на різних етапах, включаючи отримання та обробку резюме, надання зворотного зв'язку та генерацію рекомендацій.

Здатність системи точно аналізувати вміст резюме та порівнювати його з описами вакансій ретельно оцінюється під час тестування. І це передбачає підтвердження того, що алгоритми обробки природної мови (NLP) ефективно витягують відповідні навички, досвід та кваліфікації, а інтеграція з GPT-4 через Azure OpenAI генерує контекстуально точні оцінки та надає доречні рекомендації. Крім того, тестування передбачає забезпечення того, щоб зворотний зв'язок, що генерується, був дієвим, цінним і зрозумілим для користувача.

Також додатковим критичним компонентом тестування є оцінка загальної ефективності бота та часу відповіді. Важливо, щоб система могла ефективно обробляти численні взаємодії з користувачами, швидко обробляти резюме та без затримок повертати результати. Це особливо важливо при обробці великих обсягів резюме або складних описів вакансій. Окрім функціональних можливостей, важливим аспектом розробки та впровадження інтелектуальної системи є забезпечення її зручності для кінцевих користувачів. У цьому контексті особлива увага приділяється оцінці зручності системи, що включає ретельний аналіз інтуїтивності інтерфейсу користувача. Ключовою метою є гарантування того, що

користувачі можуть без зайвих зусиль та плутанини взаємодіяти з асистентом безпосередньо через популярний месенджер Telegram.

Протягом етапу тестування передбачається активне збирання відгуків від реальних користувачів. Ці відгуки є цінним джерелом інформації, що дозволяє виявити будь-які потенційні аспекти, в яких користувацький досвід може бути покращено. До таких аспектів належать, зокрема, ясність наданих асистентом інструкцій, зрозумілість формату зворотного зв'язку, який він надає, а також загальна логічність та плавність потоку взаємодії. Аналіз цих відгуків дасть змогу розробникам внести необхідні корективи для оптимізації зручності використання системи.

Паралельно з оцінкою зручності використання проводиться оцінка загальної якості системи. Цей процес передбачає порівняння продуктивності розробленої системи з попередньо встановленими еталонними показниками або з характеристиками аналогічних інструментів, які вже присутні на ринку. Таке порівняння є важливим для ідентифікації потенційних областей, де система може бути вдосконалена, а також для забезпечення відповідності системи стандартам, які є необхідними для її ефективного застосування в реальних умовах.

На завершальному етапі розробки та тестування ключовим пріоритетом є забезпечення того, щоб фінальна версія системи демонструвала високий рівень якості, точності та надійності у наданні результатів. Для досягнення цієї мети проводиться комплексне тестування та оцінювання, спрямоване на виявлення та своєчасне усунення будь-яких проблем або недоліків, які можуть виникнути в процесі її функціонування. Завдяки такому ретельному підходу користувачі можуть бути впевнені в тому, що система буде стабільно працювати та надавати достовірну інформацію.

2 ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Вибір типу архітектури програмної системи

Архітектура програмної системи буде відігравати вирішальну роль у визначенні її продуктивності, масштабованості та ремонтопридатності, тому при розробці Telegram-бота для автоматизованого аналізу резюме важливо обрати архітектуру, яка ефективно обробляє взаємодію з користувачами, обробляє дані та інтегрується із зовнішніми сервісами, такими як Azure OpenAI. Було розглянуто кілька архітектурних підходів, включаючи монолітну, мікросервісну та серверно-клієнтську моделі, кожна з яких має свої переваги та недоліки.

Спочатку розглянуто монолітну архітектуру через її простоту і легкість розгортання. За такого підходу всі компоненти системи, включно з Telegram-ботом, модулями обробки тексту та генерації PDF-файлів, розроблялися б як єдиний додаток. Такий дизайн дозволив би спростити впровадження та мінімізувати мережеві накладні витрати, оскільки всі функціональні можливості знаходяться в одній кодовій базі. Однак зі зростанням складності системи підтримка та масштабування монолітної архітектури може стати складним завданням. Оновлення будь-якої частини системи вимагає перерозгортання всієї програми, що може призвести до простоїв і збільшення зусиль на розробку. Іншим варіантом, що розглядався, була мікросервісна архітектура. У цій моделі різні функціональні можливості системи, такі як вилучення тексту з PDF-файлів, синтаксичний аналіз описів вакансій, адаптація резюме за допомогою штучного інтелекту та генерація PDF-файлів, розроблялися б як незалежні сервіси. Кожен мікросервіс буде взаємодіяти через API, що забезпечить гнучкість у розгортанні та масштабуванні. Такий підхід дозволяє оновлювати окремі сервіси, не впливаючи на всю систему, покращуючи її влагодження та стійкість. [11] Однак впровадження системи на основі мікросервісів призводить до ускладнення, що вимагає додаткової

інфраструктури для управління міжсервісною взаємодією та ретельного моніторингу для забезпечення надійності. (рисунок 1.6)

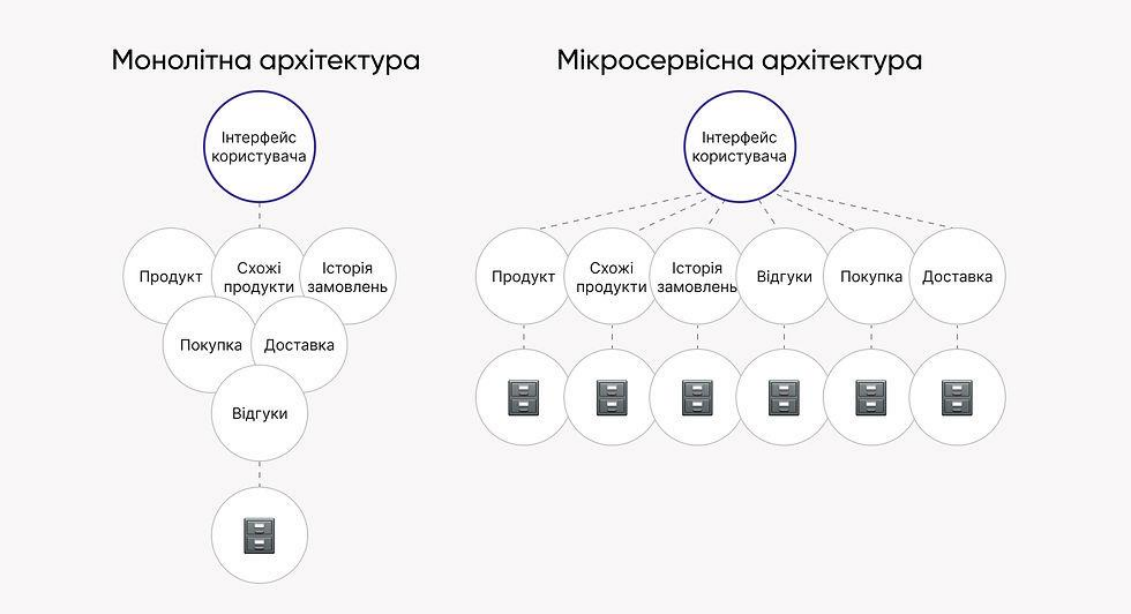


Рисунок 1.6 – Схема монолітної та мікросервісної архітектур.

Також була оцінена модель сервер-клієнт, де бот Telegram виступає в ролі клієнта, обробляючи взаємодію з користувачами, тоді як основна обробка відбувається на віддаленому сервері. У цій архітектурі бот просто збирає вхідні дані, такі як резюме кандидата та посилання на вакансію, і пересилає їх на внутрішній сервер, який обробляє інформацію. Внутрішній сервер взаємодіє з Azure OpenAI для аналізу тексту, адаптує резюме, генерує новий PDF-файл і надсилає кінцевий результат назад боту, який потім доставляє його користувачеві. Такий підхід гарантує, що обчислювально інтенсивні завдання, такі як адаптація резюме на основі штучного інтелекту та генерація PDF, виконуються на більш потужній інфраструктурі, а не в самому боті. Таке розмежування завдань підвищує безпеку, покращує продуктивність і забезпечує кращу масштабованість завдяки одночасній обробці кількох запитів.

Враховуючи специфічні вимоги проекту, було обрано гібридний підхід, що поєднує елементи серверно-клієнтської моделі з модульним бекенд-дизайном. Telegram-бот виконує роль легкого інтерфейсу для користувачів, тоді як бекенд-сервер, розміщений у хмарному середовищі, ефективно обробляє та адаптує резюме.

Розроблено так, щоб була можливість розширення, що дозволить в майбутньому інтегрувати з додатковими моделями штучного інтелекту, джерелами даних або рекрутинговими платформами. Це гарантуватиме, що програмне забезпечення залишається масштабованим, підтримуваним і здатним обробляти різні робочі навантаження без шкоди для продуктивності. Одним з ключових факторів, що вплинули на вибір архітектури – це, звичайно, потреба у взаємодії в режимі реального часу. Оскільки бот має спілкуватися з багатьма користувачами через Telegram, програмна система має бути високочутливою, щоб кандидати вчасно та швидко отримували адаптоване резюме. І обробка резюме складається з декількох етапів – виділення та видалення тексту з PDF-файлів, розбір описів вакансій, генерування адаптованого тексту на основі ШІ та форматування кінцевого результату в структурований документ, для ефективного управління цими процесами була необхідна добре структурована та модульна архітектура.

Для подолання обмежень монолітної архітектури, в основу системи було остаточно обрано модель «сервер-клієнт». У цій архітектурі Telegram-бот функціонує як клієнт і відповідає за збір даних від користувачів, обробку взаємодії та видачу результатів. Основна обробка та вилучення даних, адаптацію тексту на основі штучного інтелекту та генерацію PDF-файлів, передається на внутрішній сервер, і гарантує, що обчислювально-інтенсивні завдання обробляються ефективно і не перевантажують бота. А завдяки використанню хмарного бекенду, система може динамічно масштабуватися, обробляючи кілька запитів паралельно та інтегрувати додаткові моделі ШІ або джерела даних, коли це буде необхідно.

В архітектурному рішенні відіграла важливу роль – модульність. Система була розроблена з незалежних модулів, які відповідають за окремі завдання. Цей підхід полегшує обслуговування та майбутню модернізацію, бо окремі компоненти можна вдосконалити або замінити, не впливаючи на загальну функціональність.

Важливим пунктом є безпека та конфіденційність даних. Оскільки користувачі подають особисту інформацію у вигляді резюме, важливо було забезпечити безпечну обробку їх даних. Тому завдяки централізації обробки даних

на захищеному внутрішньому сервері – конфіденційна інформація не зберігається у самому боті, що зменшує ймовірні ризики безпеки. Також, для захисту даних користувача під час передачі застосовуються шифрування та безпечні методи зв'язку завдяки API. Бот відповідає за отримання вхідних даних від користувачів, включаючи резюме кандидата у форматі PDF та посилання на опис вакансії. Після того, як бот збирає ці дані, він пересилає інформацію на внутрішній сервер для подальшої обробки. Внутрішній сервер виконує обчислювальну роль, включаючи вилучення та адаптацію тексту з наданого PDF-файлу, синтаксичний аналіз опису вакансії, взаємодію з Azure OpenAI API для створення індивідуальної адаптації резюме та форматування остаточної версії в структурований PDF-документ.

Уся архітектура платформи побудована за модульним принципом, що гарантує її гнучкість, простоту обслуговування та можливість легкого масштабування в майбутньому. Система складається з окремих функціональних блоків, кожен з яких виконує свою роль. Модуль обробки PDF-документів відповідає за перетворення резюме у зручний для подальшого аналізу формат. Модуль аналізу вакансій працює з описами з сайтів працевлаштування, виділяючи звідти вимоги до кандидатів, очікувані навички, досвід та рівень кваліфікації. ШІ формує адаптоване резюме, документ, у якому досвід кандидата подається найбільш виграшно з урахуванням конкретної вакансії. Проєктна гібридна архітектура була спроєктована з урахуванням потреб у масштабованості. Система повинна залишатися стабільною, навіть коли кількість користувачів зростає. Завдяки розділенню функцій між фронтендом (ботом) та бекендом, ключові обчислювальні процеси, як-от генерація тексту, обробка PDF чи аналіз вакансій, виконуються окремо, не впливаючи на швидкодію основного інтерфейсу. Telegram-бот при цьому зберігає свою основну функцію, зручне та інтуїтивно зрозуміле спілкування з користувачем. Тому модульно-сервісний підхід дозволяє не лише підтримувати стабільну роботу системи, а й дає змогу легко додавати нові можливості.

2.2 Опис основних компонентів системи

Система складається з низки взаємопов’язаних компонентів, кожен з яких має чітко визначену роль у забезпеченні безперебійного та ефективного функціонування всього процесу – від першого контакту з користувачем до створення персоналізованого, фінального документа у PDF форматі. У центрі цього рішення є внутрішній сервер на python. Саме тут приймаються запити, обробляються вхідні дані, взаємодіє з зовнішніми сервісами та координується весь робочий процес. Через нього проходить уся інформація, python провідник від надсилання поточного резюме кандидата до отримання адаптованого документу. Одним із ключових інструментів є API Azure OpenAI, що відповідає за надання "інтелектуально-адаптованого" тексту, трансформацію змісту резюме відповідно до вимог вакансії та сучасних вимог.

Зовнішній інтерфейс системи реалізований у вигляді Telegram-бота, що є зручним і зрозумілим рішенням, створеного на базі фреймворку aiogram. Бот виступає посередником між користувачем і внутрішньою логікою сервісу. Користувач проводить адаптацію свого резюме: від завантаження PDF-файлу резюме і надання посилання на вакансію до отримання адаптованого документа і можливістю редагувати поточний документ. Так, тому бот не лише приймаж дані, а й реагує на дії користувача, забезпечуючи зворотний зв’язок, корисні підказки, повідомлення про помилки й інструкції на кожному етапі. Після надсилання резюме ботом, сервер обробляє цей файл, витягуючи текстовий вміст зі збереженням логічної структури. Оскільки багато сучасних резюме містять таблиці, списки, складні форматування або навіть є сканованими копіями, система при потребі використовує методи оптичного розпізнавання тексту (OCR), що дає змогу працювати з широким спектром документів і не втрачати важливу інформацію.

Паралельно з цим, за наданим користувачем посиланням на вакансію, система отримує опис посадової ролі. Залежно від джерела, це може відбуватися або шляхом веб-скрейпінгу, або через API. Далі цей текст ретельно аналізується, і з нього

виокремлюються ключові вимоги – навички, досвід, сертифікації, освіта тощо. Цей контекст слугує фундаментом для подальшої адаптації. І саме на цьому етапі вступає ШІ – модель OpenAI, розгорнута через Azure. Вона має адаптувати поточне резюме кандиата таким чином, щоб воно найкраще відповідало специфіці вибраної вакансії. За допомогою спеціально налаштованих підказок у коді, модель змінює, розширює або уточнює вміст, не порушуючи стилістики, не спотворюючи факти, але акцентуючи увагу на найрелевантнішому для конкретної посади. Такий підхід підвищує шанси кандидата пройти автоматичні фільтри ATS та зацікавити рекрутера вже з першого перегляду. [12] А коли адаптацію вже завершено, система формує новий документ у структурованому форматі. Для генерації готового PDF використовується спеціалізована бібліотека, яка дозволяє створювати акуратні, добре відформатовані документи, що відповідають сучасним стандартам оформлення – бібліотека FPDF. Створене резюме має професійний вигляд, добре читається як людиною, так і автоматизованими системами.

При роботі з персональними даними слід згадати питання конфіденційності інформації, що виходять на перший план. Тому система реалізована з урахуванням сучасних стандартів безпеки: усі передачі шифруються, зберігання даних, тимчасове, а логіка обробки побудована таким чином, щоб інформація не залишалась у системі довше, ніж потрібно. Передбачено також механізми обробки помилок, наприклад, у випадку неправильного формату файлу або недоступності зовнішнього ресурсу система видає коректні повідомлення і не зупиняє роботу.

Telegram-бот у даній реалізації, не просто інтерфейс, а справжній гід, що проводить користувача через усі етапи та забезпечує логічний, послідовний, зрозумілий процес взаємодії з програмою. Якщо кандидат випадково завантажив файл у невірному форматі або залишив неправильне посилання, бот пояснить, у чому помилка, і дасть змогу легко її виправити. Така адаптивність значно покращує і спрощує користувацький досвід.

Після повного циклу обробки, бот надсилає користувачу вже адаптоване резюме – вже у форматі готового до подання на бажану вакансію документа. Таким

чином, кандидат отримує не просто текст, а професійно оформлений файл, готовий до подальшого редагування і використання.

Завдяки такій гармонічній та глибокій інтеграції між усіма елементами та продуманою реалізацією – серверною частиною, штучним інтелектом, інтерфейсом Telegram-бота та засобами безпечної обробки даних – така система стає потужним інструментом для автоматизованої підтримки кандидатів у їх кар'єрних початках. Вона не лише економить час, але й підвищує якість заявки, дозволяючи кожному претендентові показувати себе в найкращому світлі. Сучасні процеси пошуку роботи звертають увагу не лише якість самого резюме, а й те, наскільки воно відповідає конкретним вимогам роботодавця, форматування і сприйняття. Саме для цього в системі був реалізований модуль обробки PDF, який автоматизує адаптацію резюме під задану вакансію, забезпечуючи зручність, правильність формату і ефективність для користувача.

Початкова мета – це отримати повну та структуровану інформацію про кандидата: особисті дані, досвід роботи, освіту, ключові навички й досягнення, для цього і потребується від користувача надати своє власне резюме. А вже завдяки технологіям, можна буде зчитати досвід та навички кандидата з сторінкою та описом до вакансії. Забезпечивши веб-скрепінг, зокрема бібліотекам BeautifulSoup і Requests, система може працювати навіть з динамічними або складно структурованими HTML-сторінками. Вона виділяє назву посади, опис ролі, перелік обов'язків, вимоги до кваліфікації, необхідні навички та іншу релевантну інформацію. Після цього, ці обидва джерела зчитування – резюме кандидата та опис вказаної вакансії опрацьовано, система виконує їх порівняння. Тут використовується інтеграція з Azure OpenAI, що дає змогу глибше аналізувати відповідність між навичками кандидата й вимогами посади. На основі цього аналізу генерується новий зміст резюме: акценти зміщуються на релевантний досвід, підкреслюються саме ті

навички, які мають найбільше значення для цієї конкретної вакансії. Важливо, що зміст змінюється не механічно, а осмислено – із урахуванням контексту.

Завершальним кроком є створення нового PDF-документа з оновленим і адаптованим змістом. Система формує структурований, чистий і професійно оформлений документ, у якому чітко виокремлено ключові розділи, контактна інформація, досвід роботи, освіта, навички, досягнення. Для генерації файлу використовується бібліотека для роботи з PDF, яка дозволяє зберігати послідовність, форматування та візуальну привабливість документа. Такий файл легко читається, зручний для перегляду рекрутерами та виглядає як повноцінне, грамотно складене резюме. Весь процес, від моменту отримання резюме та вакансії до формування адаптованого PDF-документа, відбувається автоматично та інтегровано. Telegram-бот служить інтерфейсом взаємодії з користувачем: саме через нього кандидат надсилає своє резюме і посилання на вакансію, а потім отримує оновлений файл, готовий до подачі роботодавцю.

Під час розробки складних автоматизованих систем, зокрема таких як Telegram-бот для аналізу та адаптації резюме під вимоги вакансій, особливу увагу варто приділяти не лише функціональним можливостям, а й технічним аспектам, що забезпечують стабільну, надійну та передбачувану роботу. Одним із таких аспектів є правильно організована система обробки помилок та ведення журналу подій (логування), яка виконує не другорядну, а фактично критичну роль у загальній архітектурі програмного рішення. Варто враховувати, що будь-яка система, особливо розподілена та взаємодіюча з зовнішніми сервісами, неминуче стикається з непередбачуваними ситуаціями: проблеми з мережею, нестабільність API, некоректні вхідні дані від користувача, внутрішні винятки при обробці запитів – усе це потенційні точки відмови. І завдання розробника не в тому, щоб повністю уникнути помилок (це практично неможливо), а в тому, щоб система вміла на них правильно реагувати: фіксувати, аналізувати, повідомляти, а інколи виправляти самостійно.

У контексті Telegram-бота, який працює з резюме, вакансіями, OpenAI API та іншими компонентами, обробка помилок починається з їхнього передбачення. Ми ретельно аналізуємо усі можливі етапи взаємодії користувача з ботом: від надсилання резюме до отримання адаптованого документа у PDF-форматі. На кожному з цих етапів існують ризики. Наприклад, користувач може надіслати файл у невірному форматі або посилання на вакансію, яке більше неактивне. Також можуть виникати помилки під час парсингу HTML-сторінок, генерації PDF, нестача пам’яті, таймаути при зверненні до стороннього API – кожен із таких випадків повинен бути не лише оброблений, а й грамотно зафіксований.

Саме тут вступає в гру система логування. Вона є своєрідним "чорним ящиком", в якому зберігається історія всіх подій, запитів, помилок та технічних нюансів роботи програми. За допомогою логів можна швидко виявити джерело проблеми, відновити хронологію подій, побачити закономірності у збоях і зрозуміти, як саме користувач взаємодіє з ботом. Це надзвичайно цінна інформація для команди підтримки, розробників і DevOps-фахівців. У Python для цієї мети найчастіше використовують вбудовану бібліотеку logging, яка дозволяє задавати рівні важливості повідомлень – від найдрібніших (DEBUG) до критичних збоїв (CRITICAL). Така градація дозволяє ефективно фільтрувати повідомлення та налаштовувати різні механізми їхнього зберігання: у файл, у базу даних, або ж надсилати до централізованих систем моніторингу, таких як Sentry чи Logstash. Це відкриває широкі можливості для масштабування, віддаленої діагностики та реагування в реальному часі.Однак сам факт наявності логування ще не означає, що користувачеві буде комфортно. Тому важливо також реалізувати грамотне повідомлення про помилки безпосередньо в інтерфейсі бота. Якщо бот стикається з проблемою, наприклад, не може зчитати текст із PDF-файлу або API не відповідає – він має ввічливо і зрозуміло пояснити це користувачу.

2.3 Декомпозиція модулів та опис функціональних блоків

Система взаємодії з користувачем через Telegram-бот, що є частиною UI Layer, відповідає за безпосередню комунікацію з користувачем. [13] Цей модуль, розроблений на базі бібліотеки Aiogram, включає в себе обробку вхідних даних від користувача, таких як PDF-файли з резюме та посилання на вакансії. Важливою частиною є визначення послідовності кроків обробки запиту, що може бути реалізовано за допомогою машини станів (FSM). Процес роботи системи відображається користувачеві через повідомлення про поточний статус аналізу. Після завершення обробки запиту модуль відповідає за надсилання результату користувачеві, що включає виведення адаптованого текстового варіанту та надсилання згенерованого PDF-файлу безпосередньо в чат Telegram. (рисунок 2.1)

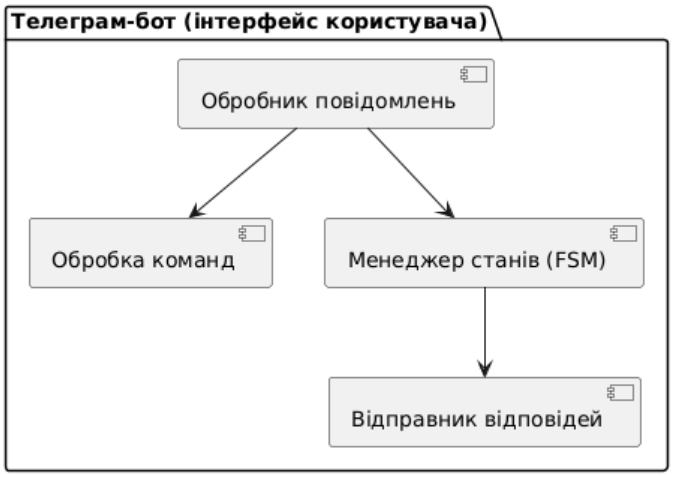


Рисунок 2.1 – Діаграма декомпозиції модуля «Інтерфейс користувача»

Модуль «Обробка резюме» відповідає за отримання вхідного PDF-файлу з резюме та його підготовку до подальшого аналізу. (рисунок 2.2)

Рисунок 2.2 – Діаграма декомпозиції модуля «Обробка резюме»

Процес включає зчитування вмісту PDF-файлу за допомогою спеціалізованих бібліотек, таких як PyMuPDF або pdfminer, з подальшим перетворенням отриманої інформації у текстовий формат. На наступному етапі відбувається структурування цього тексту шляхом виділення ключових розділів, таких як контактні дані, опис досвіду роботи, інформація про освіту та перелік навичок. Результатом роботи модуля є формування структурованих даних у вигляді словника, який потім передається до GPT-моделі для подальшої обробки та аналізу.

Модуль «Парсинг вакансії» призначений для обробки наданого посилання на вакансію. (рисунок 2.3)

Рисунок 2.3 – Діаграма декомпозиції модуля «Парсинг вакансії»

Його функціональність починається з отримання HTML-коду веб-сторінки за допомогою таких інструментів, як requests та BeautifulSoup або їхніх аналогів. Після

успішного отримання HTML відбувається аналіз контенту сторінки з метою виділення ключових блоків інформації, включаючи назву посади, опис обов'язків, перелік вимог до кандидата та необхідні навички. На завершальному етапі оброблені дані структуруються у форматі словника,, який готується для подальшого аналізу та порівняння з інформацією з резюме.

Ядро системи, модуль «Адаптації (GPT-аналітика)», відповідає за формування адаптованого резюме, використовуючи отримані дані з резюме та опису вакансії. В основі його роботи лежить інтеграція з Azure OpenAI API для використання GPT-моделі. Процес починається з формування запиту до GPT, який включає побудову prompt на основі структурованих даних резюме та вакансії, а також налаштування таких параметрів, як температура, максимальна кількість токенів та визначення ролі системи. Після надсилання запиту модуль отримує відповідь від GPT, з якої витягує адаптовані розділи, такі як стислий опис, ключові навички, релевантний досвід та досягнення. На завершення отримані дані форматуються у структурований вигляд, наприклад, у вигляді списку блоків, для подальшого використання. (рисунок 2.4)

Рисунок 2.4 – Діаграма декомпозиції модуля «Адаптація (GPT-аналітика)»

Модуль «Генерація PDF» відповідає за створення PDF-файлу, що містить адаптований зміст резюме. Для цього використовується бібліотека fpdf2. Процес побудови PDF включає форматування окремих розділів адаптованого резюме, таких

як Summary, Skills та Experience, із застосуванням відповідних стилів та шрифтів. Також передбачено додавання заголовків та маркованих списків для кращої структуризації інформації. На етапі побудови документа встановлюються метадані PDF-файлу, генерується структура сторінок та здійснюється збереження готового файлу у визначеній папці cv_files/. (рисунок 2.5)

Рисунок 2.5 – Діаграма декомпозиції модуля «Генерація PDF»

Модуль «Логіка управління (Контролер) » виконує функцію координатора всіх інших модулів системи, визначаючи загальну логіку її роботи та забезпечуючи безперебійний потік даних між ними. (рисунок 2.6)

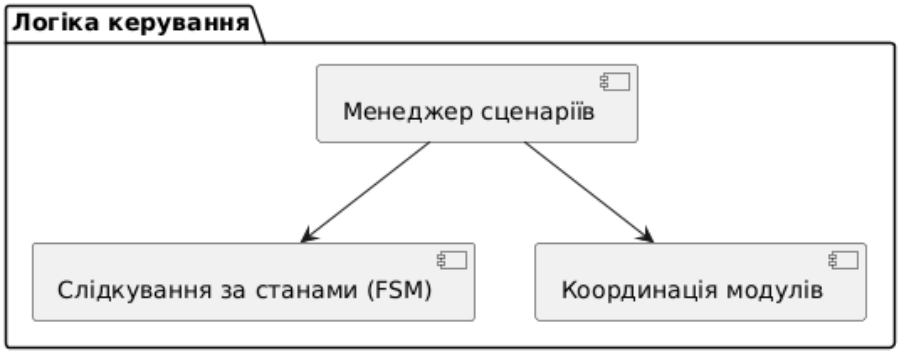


Рисунок 2.6 – Діаграма декомпозиції модуля «Логіка управління (Контролер)»

Важливою частиною його функціональності є реалізація кінцевого автомата (FSM) для управління станами користувача в Telegram-боті. Це включає збереження поточного стану користувача, наприклад, очікування завантаження резюме або введення посилання на вакансію, а також визначення можливих переходів між цими

станами. Крім того, модуль відповідає за послідовний запуск основних етапів обробки запиту: ініціює зчитування PDF-файлу резюме, потім запускає парсинг вакансії, передає дані до модуля GPT-адаптації, після чого активує генерацію PDF-файлу з адаптованим резюме та, нарешті, забезпечує надсилання відповіді користувачеві.

Модуль «Тестування та логування» відповідає за забезпечення стабільності та надійності роботи системи шляхом фіксації помилок, проведення тестування окремих компонентів та ведення журналів для полегшення налагодження. У випадку виникнення виняткових ситуацій модуль здійснює їх запис у лог-файл для подальшого аналізу та інформує користувача про наявність помилки. Для перевірки працездатності окремих частин системи передбачено проведення юніт-тестів основних компонентів, а також використання тестових кейсів, що імітують реальні сценарії з PDF-файлами резюме та описами вакансій. (рисунок 2.7)

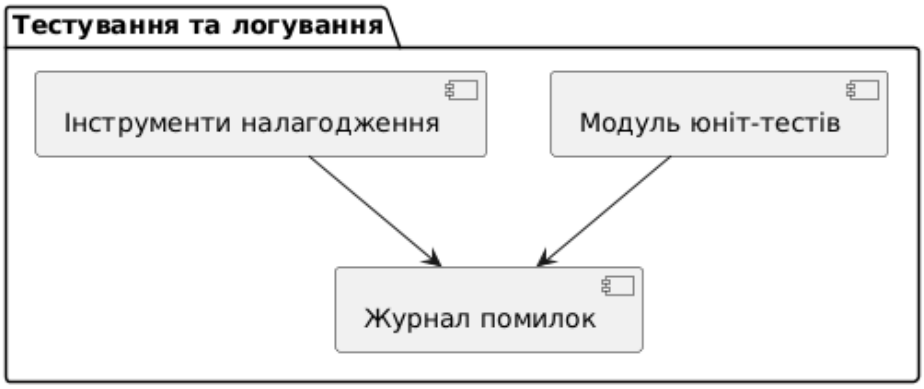


Рисунок 2.7 – Діаграма декомпозиції модуля «Логіка управління (Контролер)»

У процесі обробки резюме та вакансії також можуть виникнути різноманітні помилки, такі як проблеми з форматом файлу, або некоректне кодування символів, або труднощі з вилученням тексту. Тому програмна система повинна бути здатна виявляти та обробляти такі помилки, надаючи користувачу відповідні повідомлення та рекомендації щодо їх усунення. Це забезпечить надійність та стабільність роботи бота.

2.4 Вибір алгоритмів обробки тексту та аналізу відповідності резюме вакансії

Адаптація резюме під конкретну вакансію — це складний процес, що охоплює вилучення тексту, його структурування, аналіз і трансформацію згідно з вимогами роботодавця. Першим кроком є отримання тексту з PDF-файлів, де резюме можуть бути представлені у вигляді таблиць, колонок чи навіть зображень. Для цього використовують бібліотеки Python, зокрема pdfplumber та PyMuPDF: перша краще зберігає структуру, друга – швидше працює й обробляє зображення. Вибір залежить від формату документа, іноді доцільне комбінування підходів. [14] Після вилучення текст потрібно впорядкувати – виокремити блоки на кшталт досвіду роботи, навичок, освіти та контактної інформації. Оскільки ці частини не завжди чітко структуровані, застосовують методи обробки природної мови – наприклад, розпізнавання іменованих сутностей (NER), що дозволяє визначити, де згадуються компанії, посади чи дати. Для цього підходять бібліотеки spaCy або transformers з попередньо навченими мовними моделями. [15]

Далі відбувається аналіз відповідності між резюме й вакансією. Обидва тексти стандартизуються: очищуються від зайвих слів, лематизуються й токенізуються для подальшого порівняння. Щоб оцінити релевантність, використовують методи векторизації – TF-IDF, Word2Vec або BERT – і обчислюють схожість між векторними представленнями текстів, наприклад, за косинусною метрикою. Це дає змогу виявити збіги навіть тоді, коли формулювання різняться або використано синоніми. На основі цього аналізу можна не просто зробити висновок про відповідність кандидата, а й автоматично адаптувати текст резюме. Сучасні мовні моделі, як-от ті, що надаються через Azure OpenAI, здатні переписувати блоки резюме, підкреслюючи релевантний досвід, змінюючи формулювання та підлаштовуючи структуру під вимоги ATS-систем і очікування рекрутерів. [16]

Щоб система була ефективною на практиці, важливо забезпечити швидкість та стабільність. При великій кількості заявок використовують пакетну обробку,

паралельні запити та кешування результатів – це знижує навантаження на модель і прискорює відповідь. Таким чином, ми отримуємо гнучкий і надійний інструмент, що не лише аналізує та адаптує резюме, а й суттєво підвищує шанси кандидата пройти автоматичний відбір та зацікавити роботодавця.

Щоб ефективно адаптувати резюме до конкретної посади, необхідно ретельно вивчити як сам документ кандидата, так і опис вакансії. Обидва документи часто подаються в різних форматах: резюме може бути структурованим або у довільній формі, тоді як оголошення про вакансію є багатослівним, з великою кількістю вимог, критеріїв та описів вакансій. Перший етап – видобування тексту: для обробки PDF-файлів використовуються бібліотеки, що дозволяє видобувати текст з різним рівнем точності залежно від складності структури. Після цього дані систематизуються: система визначає найважливіші блоки – досвід, навички, освіту, зв'язки. Завдяки методам обробки природної мови, таким як NER, можливо точно визначити назви компаній, посади та дати, навіть якщо формат презентації незвичайний. Це дозволяє привести дані до стандартного формату, з яким можна працювати далі.

Наступний крок – порівняти резюме з описом вакансії. Його перетворюють у векторне представлення, що дозволяє оцінити не лише збіг слів, але й схожість значень. Для цієї мети використовуються вбудовування, такі моделі, як BERT, Word2Vec, а також такі міри, як косинусна подібність. Наприклад, якщо у вакансії вказано «розробка архітектури сервера», а в резюме – «створення REST API», система розпізнає схожість, навіть якщо фактичного збігу немає. Це дозволяє глибше зрозуміти, наскільки кандидат відповідає вимогам, та визначити, які частини резюме слід посилити або переробити, щоб справити краще враження.

Технології показують свою справжню цінність на етапі адаптації. Використовуючи мовні моделі, такі як GPT, система не просто трансформує текст – вона адаптує стиль, словниковий запас та оформлення до певного простору. Наприклад, загальну фразу «керував командою розробників» змінюють на «надавав технічне керівництво та координацію команді Scrum», якщо цього вимагає посадова

інструкція. Це важливо не лише для залучення рекрутерів, але й для обходу автоматизованих систем перевірки (АС), які відбирають документи на основі ключових слів. Таким чином, штучний інтелект допомагає резюме не лише добре виглядати, але й бути релевантним змісту. [17]

З технічної точки зору важливо, щоб адаптація відбувалася швидко та масштабно. Під час обробки великої кількості оновлень використовуються паралельні обчислення, кешування та зменшення кількості запитів до API – все це зменшує навантаження та пришвидшує відповідь. Результатом є не просто аналітична система, а інструмент для підвищення шансів кандидата: він отримує професійно підготовлене резюме, яке дійсно відповідає очікуванням роботодавця.

2.5 Проєктування взаємодії користувача з ботом

Створення зручного та зрозумілого процесу взаємодії з користувачем відіграє вирішальну роль у забезпеченні ефективної роботи бота для адаптації резюме. Особливо, коли мова йде про Telegram, який, з одного боку, надає достатньо структуроване середовище для спілкування, а з іншого, залишає простір для гнучкості та зручного сценарію взаємодії. Саме тому надзвичайно важливо, щоб користувач із самого початку чітко розумів, що від нього очікується, а сам процес проходження етапів був максимально прозорим і послідовним. Коли кандидат уперше звертається до бота – це може бути як активний пошук у Telegram, так і перехід за спеціальним посиланням чи qr-кодом. З перших секунд взаємодії бот вітає користувача доброзичливим повідомленням, у якому пояснює свою основну мету та коротко знайомить з тим, як він працює. Тут же надаються інструкції щодо того, які саме дані потрібні для подальшої роботи: PDF-файл з резюме та посилання на вакансію, яка цікавить кандидата.

Коли користувач надав своє резюме, система проводить перевірку, чи відповідає файл очікуваному формату, чи не пошкоджений він, і чи можливо з нього витягнути потрібну інформацію. Якщо ж документ має неправильний формат або

бот не може його обробити, система одразу повідомляє про це користувача, супроводжуючи повідомлення простими й чіткими порадами, як виправити ситуацію, як конвертувати файл у PDF і повторно завантажити. Щойно резюме завантажено успішно, наступним кроком є збір посилання на вакансію. Бот чемно просить користувача надіслати URL-адресу сторінки з описом роботи, яка зацікавила кандидата. Цей етап дуже важливий, адже саме з цього посилання буде витягнута ключова інформація, вимоги до кандидата, обов'язки, бажані навички та очікування роботодавця. Якщо ж посилання виявляється некоректним, наприклад, веде на сторінку, яка не існує, або вимагає авторизації, бот реагує відповідним повідомленням і просить або надіслати правильний URL, або вручну скопіювати текст опису вакансії в чат.

Після того, як ці два необхідні елементи для продовження роботи, резюме та опис вакансії, надано та зібрано, бот переходить до наступного етапу: підготовки даних для аналізу. Тут він обробляє текст з обох джерел, видаляючи зайве форматування, перевіряючи наявність помилок і узгоджуючи структуру. Це потрібно для того, щоб система, яка далі буде працювати з цими текстами, отримала чіткий і зрозумілий вміст без зайвого шуму чи інформації. У цей момент користувач не залишається без зворотнього зв'язку та невіданні, що буде далі. Бот надсилає оновлення про статус кожного етапу, інформуючи, що дані успішно зібрані й аналізуються. Тому користувачам не доведеться чекати невідомо скільки і здогадуватися, чи опрацьовує система їх запити, чи все зламалось.

Далі у справу вступає штучний інтелект, зокрема потужна модель OpenAI на платформі Azure, яка займається адаптацією резюме під конкретну вакансію. На цьому етапі система проводить глибокий аналіз змісту резюме, виокремлює найбільш релевантні моменти, переформульовує речення для більшої зрозумілості і читабельності, замінює загальні фрази на професійну термінологію та підкреслює досягнення, які найбільше відповідають потребам конкретного роботодавця. Наприклад, замість розмитої фрази «допомагав у розробці ПЗ» у фінальному тексті може з'явитися сильніше і чіткіше формулювання: «розробив масштабовані

рішення, що підвищили ефективність системи на 20%». Ці зміни не лише покращують враження від резюме, а й суттєво підвищують шанси пройти автоматизовані системи відбору та зацікавити рекрутера.

Коли процес адаптації завершується, система переходить до формування фінального документа. За допомогою бібліотеки FPDF створюється нове професійно оформлене резюме у форматі PDF. Воно має чітку структуру, зрозумілий візуальний стиль і повністю готове до використання. Щойно документ згенеровано, бот відправляє його користувачу прямо в чат, все просто, швидко й без зайвих дій. За бажанням, система може також запропонувати додаткові опції: створення альтернативної версії резюме, редагування окремих блоків або ручне коригування тексту перед отриманням фінального, готового резюме кандидата.

Протягом усього процесу взаємодії бот спроєктований так, щоб адекватно й ввічливо реагувати на будь-які непередбачені ситуації. Якщо виникає проблема, наприклад, посилання на вакансію не працює, файл резюме пошкоджено або обробка тексту затримується, бот завжди надає змістовне повідомлення з поясненням проблеми та порадами щодо її вирішення. Також, у таких випадках, у системі передбачено надання не посилання на вакансію, а просто передачу тексту самим кандидатом, з потрібної вакансії. Що дозволяє уникнути плутанини й забезпечити комфорт користувача на кожному етапі і що саме важливе, продовжити процес адаптації та успішно завершити його. Таким способом, завдяки грамотно побудованому процесу взаємодії, бот мінімізує зусилля з боку користувача й водночас забезпечує високу точність і ефективність усього механізму. Інтеграція з Telegram дозволяє зробити цей сервіс доступним для широкої аудиторії без потреби завантажувати додаткові додатки або вивчати нові інтерфейси. Кожна дія, кожне повідомлення та кожен етап логічно продумані і вписуються в загальну структуру, забезпечуючи не просто технічну функціональність, а повноцінний досвід, орієнтований на користувача. У підсумку, адаптація резюме стає не складним завданням, а простим і зрозумілим процесом, який дозволяє кожному кандидату максимально ефективно представити себе перед роботодавцем.

2.6 Вибір середовища розробки та технологій

У розробці програмного забезпечення для автоматичного аналізу резюме було обрано Python як основну мову програмування завдяки її універсальності, простоті використання, широкій підтримці бібліотек та засобів ШІ. Python ідеально підходить для обробки даних на серверній стороні, а також для інтеграції з зовнішніми сервісами, що робить її найкращим вибором для створення Telegram-бота, який взаємодіє з сервісами штучного інтелекту та генерує адаптовані файли. Багатий вибір бібліотек, доступних у Python, дозволяє підібрати найбільш корисні, для програмного продукту. Такі бібліотеки, як aiogram для розробки Telegram-ботів і FPDF для генерації PDF-документів, гарантує, що процес розробки буде ефективним та зручним. Python сумісний з фреймворками машинного навчання та здатність безперешкодно взаємодіяти із зовнішніми API також стали важливими факторами при виборі цієї мови для інтеграції з Azure OpenAI. Бот покладається на потужні можливості обробки природної мови (NLP) від Azure для аналізу описів вакансій і відповідного адаптування резюме користувача. Ця інтеграція потребує безперебійного зв'язку через API, і Python, завдяки своїй надійній підтримці HTTP-запитів та обробки JSON-даних, гарантує, що бот зможе ефективно взаємодіяти з Azure OpenAI без жодних проблем. Також, перевагою Python є його підтримка асинхронного програмування, оскільки бот має обробляти великі обсяги даних та одночасно взаємодіяти з численними користувачами (наприклад, аналізуючи резюме, опис вакансій або генеруючи PDF-документи), асинхронне програмування дозволяє ефективно виконувати ці завдання без блокування чи затримки відповідей користувачам. Завдяки цьому система може обробляти великий обсяг запитів одночасно, забезпечуючи безперервну та чуйно реагуючу роботу бота. [18]

Для розробки самого Telegram-бота було обрано бібліотеку aiogram, яка завдяки своїй простоті, гнучкості та підтримці асинхронної роботи дозволяє боту обробляти вхідні повідомлення, управляти взаємодією з користувачем та надсилати відповіді без складнощів. Бібліотека розроблена для безперешкодної роботи з

Telegram Bot API, що полегшує впровадження всіх необхідних функцій, від обробки введених даних до надсилання кінцевого адаптованого резюме у форматі PDF. Для генерації PDF-документів було вибрано бібліотеку FPDF, оскільки вона є легкою та гнучкою у використанні, дозволяючи створювати професійно оформлені документи. Ця бібліотека забезпечує простоту використання навіть для розробників, які не мають досвіду у створенні документів. Вона дозволяє боту формувати нове резюме, адаптуючи його вміст на основі вакансії, а потім перетворювати цю інформацію в акуратно оформлений PDF.

Середовище для цього проєкту базується на Python та кількох бібліотеках, які ідеально підходять для виконання специфічних вимог системи. Вибір цих технологій гарантує, що програмне забезпечення буде легко підтримуватися, масштабованим і з можливістю подальшого розширення новими функціями за потребою та вдосконаленням. Інтеграція цих технологій забезпечує ефективну розробку та дозволяє системі впоратися з поточними вимогами та можливими майбутніми покращеннями, наприклад, додання нових функцій. Python, разом з бібліотеками aiogram, FPDF та інтеграцією з Azure OpenAI, створює стійку та надійну основу для побудови потужного, масштабованого та ефективного програмного рішення, яке автоматизує процес адаптації резюме та відповідає потребам користувачів, які шукають спрощений досвід подачі на вакансії.

Для обробки текстових даних у системі було обрано API OpenAI через Azure, що стало важливим компонентом для системи аналізу та адаптації резюме кандидатів відповідно до вимог вакансії. Інтеграція API OpenAI через Azure дає можливість боту ефективно розуміти та обробляти природну мову, що є вирішальним етапом у роботі системи. Це важливо, оскільки бот не лише має здатність розбирати сирий текст з резюме, а й здатний розпізнавати всі тонкощі описів вакансій, виокремлюючи ключові навички, кваліфікації та обов'язки, що мають значення для конкретної посади.

Azure OpenAI надає доступ до мовних моделей, таких як GPT, які здатні виконувати різноманітні завдання з обробки природної мови, такі як підсумовування

інформації, витягування ключових деталей і переформатування тексту відповідно до конкретних вказівок. Завдяки використанню цієї технології, бот може інтелектуально аналізувати резюме та коригувати його зміст, щоб він відповідав конкретній мові та ключовим словам, що зустрічаються в описі вакансії. Це забезпечує адаптацію резюме до вимог роботи і збільшує шанси кандидата на відповідність очікуванням рекрутерів або автоматизованих систем підбору персоналу, які також можуть використовувати подібні методи оцінки заявок, засновані на штучному інтелекті. Вибір сервісу OpenAI від Azure має кілька переваг. По-перше, він дозволяє безперешкодно інтегруватися з іншими компонентами системи, зокрема з Python, оскільки API OpenAI можна легко викликати через HTTP-запити, що Python обробляє ефективно за допомогою таких бібліотек, як requests та http.client. Використання Azure надає системі переваги у масштабованості, безпеки та надійності хмарної інфраструктури, що є необхідним для обробки змінних обсягів запитів, а також для забезпечення конфіденційності даних та відповідності вимогам регулювання. Система використовує API OpenAI не тільки для розуміння та перетворення змісту резюме, але й для створення відповідних адаптацій на основі аналізу опису вакансії, наданого користувачем. Інфраструктура Azure підтримує швидку та надійну комунікацію з API, що гарантує швидке і точне виконання завдань обробки тексту навіть за умови, коли кілька користувачів одночасно взаємодіють з ботом. Інтегруючи API OpenAI через Azure в роботу бота, система отримує доступ до гнучкості та потужності передових інструментів штучного інтелекту, що є необхідним для надання точних та чутливих до контексту змін у резюме.

Для розгортання необхідно вибрати відповідні інструменти та технології, які забезпечать стабільну роботу, масштабованість і підтримуваність системи. Оскільки завдання бота включають обробку текстових даних, взаємодію з Azure OpenAI і генерацію PDF-файлів, важливо правильно підібрати інструменти для розгортання, щоб забезпечити високу продуктивність і надійність системи. Одним з основних інструментів для розгортання є Docker. Цей інструмент дозволяє контейнеризувати

додаток, що означає, що бот разом з усіма його залежностями та налаштуваннями може бути упакований у легкий і портативний контейнер. Використання Docker дозволяє уникнути потенційних проблем, пов'язаних з несумісністю середовищ та конфліктами версій, оскільки контейнер інкапсулює всі залежності, такі як Python, бібліотеки та конфігураційні файли, в одному пакеті. Що значно спрощує процес розгортання, масштабування системи та оперативне виправлення проблем без турбот про сумісність різних частин інфраструктури. [19]

Для хостингу бота природним вибором є хмарні сервери, які пропонують високу масштабованість і гнучкість. Хостинг що пропонує Azure, дозволяє зробити бота доступним у будь-який час з будь-якого місця і забезпечує необхідні обчислювальні ресурси для обробки запитів користувачів. Оскільки навантаження на систему може змінюватися, важливо використовувати рішення для хостингу, яке дозволяє легко масштабувати ресурси, гарантуючи, що бот зможе ефективно обробляти одночасно кілька запитів.

Хмарні хостинг-платформи зазвичай надають різні рівні функцій безпеки, такі як зашифровані канали зв'язку, фаєрволи та послуги резервного копіювання даних, що є необхідними для захисту чутливої інформації, наприклад, резюме користувачів та персональних даних. Завдяки безпечному хмарному хостингу бот може забезпечити високу доступність і мінімізувати час простоїв, гарантуючи безперервну роботу для користувачів. Поєднуючи Docker для контейнеризації з хмарними серверами для хостингу, середовище розробки можна оптимізувати для ефективного розгортання і масштабування.

Впровадження системи контролю версій є важливим кроком для управління вихідним кодом. Контроль версій дозволяє відстежувати зміни, внесені в код протягом часу. Git, одна з найбільш популярних систем контролю версій, надає ефективний спосіб управління і співпраці над кодом, зберігаючи детальну історію змін. Завдяки Git кожна зміна коду фіксується, що дозволяє розробникам повернутися до попередніх версій за потреби, порівняти різні версії файлів і побачити, хто вніс кожну зміну. Це зменшує ризик виникнення помилок і допомагає

уникати конфліктів при роботі кількох людей над одним проєктом. GitHub, хмарна платформа, побудована навколо Git, часто використовується для хостингу репозиторіїв і полегшення співпраці в команді. Вона надає централізовану локацію для зберігання вихідного коду проєкту і дозволяє розробникам завантажувати свої зміни до спільного репозиторію. Функції співпраці на GitHub, такі як pull-запити, задачі та перевірка коду, дозволяють командам працювати ефективно, навіть якщо вони знаходяться в різних частинах світу. Розробники можуть створювати гілки для нових функцій або виправлень помилок, а коли код готовий, вони можуть подавати pull-запити для злиття цих змін з основною кодовою базою. Такий робочий процес гарантує, що всі зміни будуть перевірені перед тим, як вони будуть інтегровані, що підтримує високу якість коду та знижує ймовірність внесення помилок у систему. Крім того, GitHub надає ряд інструментів для управління проєктами. Наприклад, задачі можна використовувати для відстеження помилок, запитів на нові функції та завдань, що забезпечує всіх учасників проєкту інформацією про поточний стан і майбутні етапи роботи. Крім того, GitHub Actions можна використовувати для автоматизації деяких процесів, таких як тестування коду, деплой оновлень або генерування документації.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Опис процесу розробки основних модулів

Процес розробки програмного забезпечення, яке автоматизує аналіз резюме кандидата відповідно до вимог конкретної вакансії, починається з ретельного налаштування середовища розробки. Саме на цьому етапі визначаються всі інструменти, що забезпечуватимуть стабільність, ефективність і масштабованість системи. У центрі цього середовища стоїть мова програмування Python, яка є не лише популярною в галузі розробки, а й надзвичайно гнучкою для реалізації задач із обробки текстів, інтеграції з API та роботи з PDF-файлами. Встановлення Python є базовим кроком, за яким іде створення віртуального середовища, ізольованого простору, в якому можна безпечно керувати залежностями, уникаючи конфліктів із бібліотеками з інших проєктів. Основні модулі розроблялися у такій послідовності (табл. 3.1):

ТАБЛИЦЯ 3.1 — Структура модулів Telegram-бота

№	Назва модуля	Призначення
1	Налаштування середовища	Ініціалізація Python-середовища, встановлення залежностей
2	Модуль обробки повідомлень	Прийом файлів резюме та посилань на вакансії від користувача
3	Модуль аналізу контенту	Витяг тексту з PDF та парсинг вакансії, попередня обробка текстів
4	Модуль адаптації резюме	Формування запиту, передача тексту до сервісу обробки, отримання результату
5	Модуль генерації PDF	Побудова структурованого резюме в PDF
6	Модуль взаємодії з Git	Контроль версій, рев'ю, командна робота через GitHub

Розробка Telegram-бота розпочинається із налаштування середовища, що забезпечує надійну основу для подальшої роботи. Коли встановлено Python, створено віртуальне середовище та додано всі необхідні бібліотеки, серед яких

aiogram для обробки повідомлень, fpdf для генерації PDF-файлів, pdfplumber для витягу тексту з документів та requests для взаємодії з вебресурсами, вдається досягти ізольованості від зовнішніх змін і гарантувати стабільну роботу всієї системи. Після успішної технічної підготовки розпочалась реалізація механізму взаємодії користувача з ботом. Через асинхронний функціонал бібліотеки aiogram користувач може надсилати резюме у форматі PDF та посилання на конкретну вакансію. Бот оперативно реагує на отримані дані, дотримуючись логіки сценарію, що забезпечує зрозумілу та послідовну взаємодію усіх етапів. Інтерфейс при цьому залишився простим і зручним, на відмінну від більшості сайтів, з подібними функціями, що особливо важливо для користувача, який не має великого досвіду.

Отримані документи обробляються автоматично: з PDF-файлів витягується текст за допомогою бібліотек pdfplumber або PyPDF2, після чого інформація очищується від зайвих елементів, таких як номери сторінок чи службові символи, чи зовсім непотрібної інформації. Тексти вакансій завантажуються з посилання, або вставляються вручну користувачем шляхом копіювання, після чого також проходять попередню обробку. Це дозволяє забезпечити чистий і структурований матеріал для подальшої роботи. А коли всі дані підготовлені, формується спільна структура, в якій резюме співвідноситься з вимогами з вакансії. Такий підхід дозволяє порівняти досвід кандидата з очікуваннями роботодавця й зробити висновки щодо їхньої відповідності. Обробка здійснюється локально, тому це зменшує залежність від сторонніх сервісів і дає найкращі результати та повний контроль, зберігаючи конфіденційність.

Заключним кроком є формування підсумкового документа. За допомогою бібліотеки fpdf2 створюється адаптоване резюме у форматі PDF, де чітко виділені ключові блоки та структура документу: контактна інформація, навички, професійний досвід і освіта. Документ виглядає професійно й готовий до надсилання роботодавцю без жодних додаткових змін. А для підтримки якості коду та ефективної командної роботи використовувалась система контролю версій Git разом із платформою GitHub. Для кожного з модулів створювались окремі гілки, що

дало змогу ретельно перевіряти зміни, уникати конфліктів і зберігати цілісність проєкту. Вся історія комітів зберігається, що дозволяє легко простежити хід розробки, повернутись до попередніх рішень або проаналізувати внесені покращення.

3.2 Реалізація алгоритму аналізу тексту резюме та вакансії

Процес реалізації алгоритму аналізу тексту резюме та вакансії є надзвичайно важливим і продуманим. Бот, що використовує передові можливості GPT-4 через інтеграцію з Azure OpenAI – це рішення, що дозволяє автоматизувати аналіз резюме та співвідносити його з вимогами вакансії, використовуючи потужний мовний інтелект, здатний глибоко аналізувати текст, розуміти контекст і навіть знаходити невидимі на перший погляд зв’язки між навичками кандидата та очікуваннями роботодавця. Розроблена система функціонує як Telegram-бот, що приймає на вхід два основних джерела інформації: резюме кандидата у форматі PDF та посилання на вакансію. Перший етап реалізації алгоритму передбачає організацію середовища та інтеграцію необхідних бібліотек, таких як aiogram (для роботи з Telegram API), fpdf2 (для генерації PDF-документів), pdfminer.six або PyPDF2 (для витягування тексту з PDF), а також інструментів для веб-скрейпінгу – BeautifulSoup або requests-html. Центральним елементом є інтеграція з Azure OpenAI, яка забезпечує доступ до моделей GPT-4, здатних здійснювати глибокий контекстуальний аналіз тексту.

Алгоритм для даної системи є досить простим, але діючим. Після завантаження файлу резюме система виконує його попередню обробку. Зокрема, реалізовано модуль розпізнавання та витягування структурованого тексту з PDF-документа, незалежно від формату його оформлення. Паралельно система отримує текст вакансії, обробляючи HTML-вміст відповідної веб-сторінки та витягуючи лише релевантні блоки. Для обробки тексту вакансії, розміщеного на веб-сторінці, реалізовано модуль автоматизованого витягування інформації на основі технологій

веб-скрейпінгу. Використовуючи бібліотеки requests або httpx у поєднанні з парсерами BeautifulSoup чи lxml, система надсилає HTTP-запит на вказану URL-адресу вакансії та отримує HTML-вміст сторінки. Після чого, здійснюється синтаксичний аналіз DOM-структури документа. [20] Локалізація та вилучення саме тих блоків контенту, які мають найвищу інформативну цінність у контексті адаптації резюме є головним моментом. До таких блоків, належать:

– вимоги до кандидата (hard skills, soft skills, досвід роботи);

– посадові обов’язки;

– умови праці (графік, тип зайнятості, локація, можливість віддаленої роботи тощо).

Алгоритм визначає цільові секції за допомогою ключових слів і шаблонів, поширених у вакансіях. Ключові слова та фрази дозволяють автоматично сегментувати контент і очистити його від вторинної інформації, наприклад, загальних описів компанії або маркетингових вставок. На виході формується компактний текстовий фрагмент, який репрезентує зміст вакансії у структурованому вигляді. (рис. 3.1)

Рисунок 3.1 – Процес витягання даних про вакансію з HTML-документу.

Наступним етапом є передача тексту вакансії та тексту резюме кандидата до мовної моделі GPT-4 через API Azure OpenAI. [21] Алгоритм аналізу побудований

так: модель вивчає зміст резюме, визначає ключові навички, досвід, досягнення, а також структуру подачі інформації. Після цього здійснюється зіставлення отриманої інформації та характеристик із описом вакансії. Модель не обмежується синтаксичним порівнянням, а вона застосовує семантичний аналіз, виявляючи потенційні збіги навіть у випадках, коли формулювання у двох джерелах різняться. Особливістю алгоритму є контекстуальна обробка, де GPT-4 здатна ідентифікувати імпліцитні зв'язки між текстами, виявляючи релевантність навіть тоді, коли певна навичка не вказана явно. Наприклад, досвід з певною технологією може бути витлумачений як релевантний на основі додаткових термінів або опису проєктів. У випадках, коли відсутні ключові компоненти, модель генерує рекомендації щодо редагування тексту резюме: пропонує уточнити або переформулювати окремі фрагменти, щоб підвищити відповідність вакансії. На основі отриманого аналізу створюється адаптована версія резюме, що відповідає вимогам конкретної позиції. Вона генерується у текстовому вигляді, після чого за допомогою бібліотеки fpdf2 формується фінальний документ у форматі PDF. Цей документ автоматично надсилається користувачеві у Telegram.

Також, з технічного боку важливою частиною реалізації стало забезпечення стабільної взаємодії між усіма модулями системи. Проведено кілька типів тестування: функціональне, що дозволяє перевірити роботу на всіх етапів обробки, навантажувальне тестування, що оцінює стійкість під час одночасної обробки кількох запитів, а також UX-тестування в реальному середовищі Telegram.

Ключовим результатом є формування персоналізованих, якісно оформлених резюме, які значно підвищують шанси кандидата відповідати очікуванням роботодавця. Усі етапи – від обробки PDF та HTML до семантичного аналізу й генерації адаптованого документа працюють як єдиний цілісний алгоритм. Тому розроблений підхід демонструє ефективність використання сучасних моделей штучного інтелекту для вирішення прикладних задач на ринку праці. Інтеграція мовної моделі GPT-4 у процес адаптації резюме дозволяє не лише автоматизувати рутинні операції, але й забезпечити глибокий аналіз контексту, що наближає

результати системи до рівня експертної оцінки. Перспективи подальшого розвитку включають розширення на багатомовні ринки, врахування галузевих особливостей, а також інтеграцію з системами рекрутингу для формування повного циклу пошуку та підбору персоналу.

3.3 Модуль обробки текстових запитів.

Взаємодія з користувачем у рамках системи починається одразу, з моменту, коли кандидат звертається до телеграм-бота, який виступає не просто як інтерфейс, а як повноцінний посередник між користувачем і функціоналом, закладеним у програму. У своїй основі цей інтерфейс базується на сучасному, гнучкому та асинхронному фреймворку, бібліотеці aiogram, яка дозволяє ботам обробляти вхідні повідомлення у реальному часі, реагувати на різні типи запитів та підтримувати складну логіку діалогів. Саме завдяки цій бібліотеці вдалося реалізувати дійсно комфортну для користувача систему взаємодії. Важливо також було не лише створити просту форму прийому даних, але й подбати про те, щоб бот реагував на кожну дію користувача послідовно, логічно і без затримок. Завдяки використанню FSM-контролера, механізму керування станами діалогу, вдалось забезпечити чітку структуру взаємодії: бот послідовно веде користувача через етапи, починаючи від отримання початкових даних, продовжуючи аналізом, та фінальною генерацією адаптованого резюме. [22] Якщо ж у процесі трапляються виняткові ситуації, наприклад, завантажено не той тип файлу, або посилання виявилося недійсним, бот швидко реагує, інформує про проблему й пропонує інструкції для її вирішення. Це стало можливим завдяки гнучким інструментам обробки помилок, які підтримуються aiogram і вдало реалізовані в логіці проєкту.

Окрему увагу було приділено тому, щоб комунікація залишалась для користувача максимально зрозумілою й передбачуваною. У будь-який момент бот інформує про поточний етап обробки, пояснює, що саме відбувається, та попереджає про завершення певної дії. Вкінці система надсилає вже готове, адаптоване під

вакансію резюме у форматі PDF прямо в чат, і користувач одразу бачить результат своєї взаємодії з системою та може корегувати далі якісь особисті моменти. І тут варто наголосити, що бот працює не лише із повідомленнями, але й з callback кнопками та іншими елементами Telegram API, що дозволяє реалізовувати більш складні сценарії взаємодії, але більш зручні для використання користувачем, з простотою користування.

Ще одним важливим аспектом взаємодії з ботом стала реалізація системи команд, які значно спрощують навігацію по функціоналу. Завдяки можливості вводити прості текстові команди користувач може швидко ініціювати адаптацію резюме, повторно запустити процес, перевірити статус виконання або звернутись до інструкції. Це дозволяє не тільки зекономити час, а й зробити взаємодію більш зрозумілою та інтуїтивною навіть для тих користувачів, які не мають технічного бекграунду. Проте слід сказати, що команди – це не просто "тригери", що запускають певні функції. Вони взаємодіють із контекстом діалогу, зберігають проміжні стани і дають можливість повертатися до попереднього етапу, якщо це необхідно. Це особливо корисно у випадках, коли користувач, наприклад, змінив резюме й хоче адаптувати вже оновлений документ без початку всього процесу з нуля. У технічному сенсі команди також допомогли структурувати код у межах проєкту, кожна команда відповідає за окремий функціональний блок, що значно спростило розробку, тестування та подальший розвиток системи. Якщо виникає потреба додати нову функцію або змінити логіку взаємодії, це можна зробити швидко і без ризику порушити інші частини системи.

Кожна дія бота продумана таким чином, щоб бути логічною, своєчасною та доречною, а весь функціонал побудовано навколо ідеї забезпечення простоти, швидкості та індивідуального підходу. Користувачеві не потрібно вивчати складні інтерфейси чи проходити реєстрації, достатньо відкрити чат, надіслати необхідні матеріали та дочекатися результату, який можна одразу використовувати для подачі заявки.

3.4 Реалізація інтерфейсу користувача

Враховуючи сучасні тенденції та зручність для користувачів, було прийнято рішення створити чат-бота, який значно спрощує процес комунікації і робить сервіс максимально доступним для широкої аудиторії. Платформою вибору став Telegram. Це цілком логічне рішення, адже Telegram – одна з найпопулярніших месенджерів у світі, а отже велика кількість користувачів вже знайома з його інтерфейсом та принципами роботи.

Взаємодія користувача з ботом побудована за чітким і зрозумілим покроковим сценарієм, що робить процес максимально простим, комфортним та швдким. Основна ідея такого підходу, допомогти користувачу легко надати необхідні дані, отримати адаптоване резюме і при цьому відчувати, що весь процес проходить під контролем і без зайвих складнощів. Спершу бот вітає користувача і коротко розповідає про свої можливості. Це важливий момент, адже завдяки вітальному повідомленню користувач розуміє, що від нього очікується і які кроки будуть далі. Такий підхід сприяє кращій залученості та знижує ризик непорозумінь або помилкових дій під час роботи з ботом. (рис. 3.2)

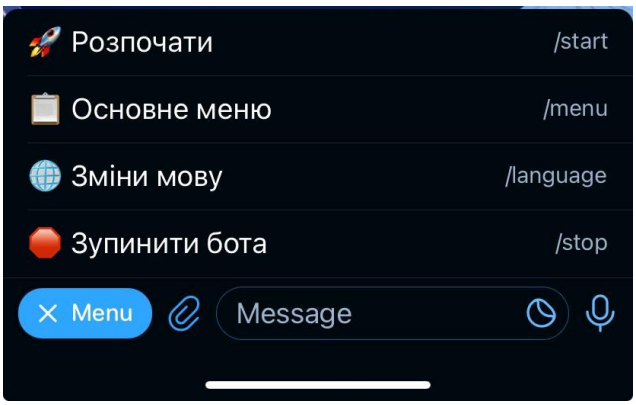


Рисунок 3.2 – Основне меню кнопок, при роботі з ботом.

Після цього бот запрошує надіслати PDF-файл із резюме кандидата. PDF-формат є найбільш поширеним та зручним для збереження всіх важливих даних та резюме. Важливо, що система перевіряє отриманий файл на відповідність вимогам,

щоб переконатися у його придатності для подальшої обробки. Якщо щось не так, користувачу надається інструкція, як виправити ситуацію, що допомагає уникнути зайвих труднощів.

Наступним етапом користувач має надати посилання на вакансію, під яку потрібно адаптувати резюме. Саме завдяки цьому посиланню система отримує точний опис вимог роботодавця, що дозволяє якісно налаштувати текст резюме під конкретну позицію. Такий підхід значно підвищує релевантність результату і робить адаптацію більш ефективною. Отримавши обидва необхідні елементи – резюме і посилання, бот надсилає підтвердження про прийняття даних і повідомляє, що починає обробку. У цей момент запускається процес генерації адаптованого тексту за допомогою технологій Azure OpenAI. Під час очікування користувач отримує повідомлення про статус, що допомагає підтримувати інтерес і позбавляє від невизначеності щодо стану обробки. (рис. 3.3)

Рисунок 3.3 – Основні етапи адаптації резюме користувача.

Після завершення роботи формується новий PDF-файл із адаптованим резюме, який бот автоматично відправляє у чат. Це означає, що користувачу не

потрібно виконувати жодних додаткових дій, він отримує готовий, зручний для використання документ. Загалом, запропонований сценарій взаємодії забезпечує простоту, швидкість і зрозумілість, що суттєво підвищує якість користувацького досвіду і робить роботу системи ефективною та приємною. (рис. 3.4)

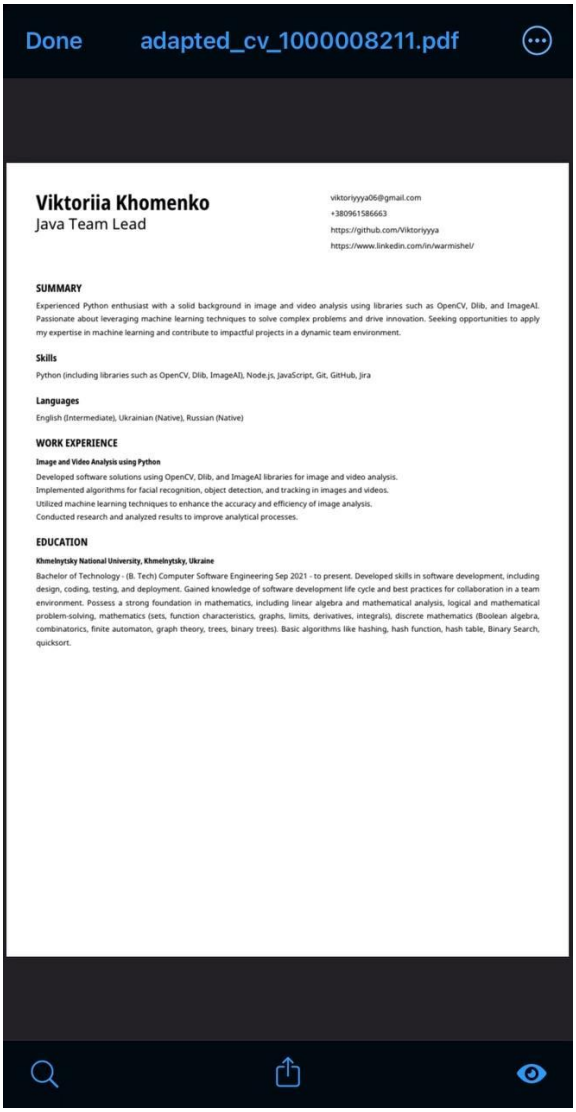


Рисунок 3.4 – Адаптоване під вакансію резюме кандидата.

Після того, як відбувається аналіз і генерація адаптованого резюме, система надає користувачу зручний і швидкий доступ до отриманих результатів. Також, при бажанні, користувач може вносити зміни у згенерований файл і тоді, бот надсилає

нове, відредаговане резюме і пропонує завершити процес, або знов, додати додаткові зміни. (рис. 3.5)

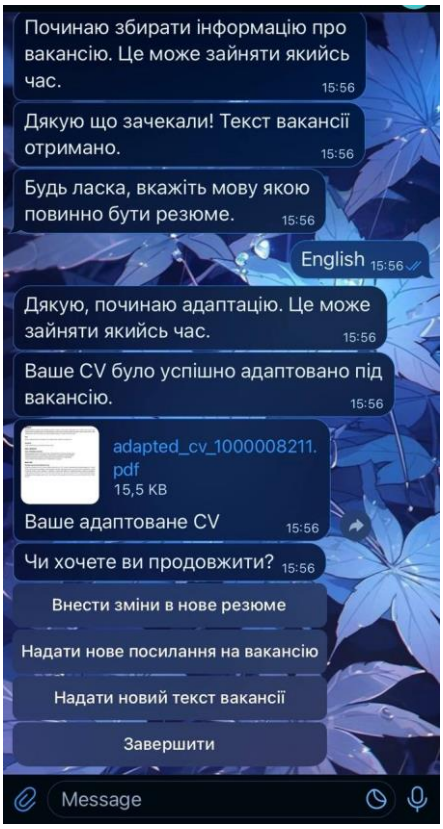


Рисунок 3.5 – Вибір додаткових функцій адаптації резюме.

Повноцінний PDF-файл із адаптованим резюме, який надсилається користувачу у чат. Вибір формату PDF зумовлений його універсальністю та зручністю для подальшого використання: його легко роздрукувати або надіслати роботодавцю. Ще однією важливою функцією системи є можливість повторної генерації адаптованого резюме за бажанням користувача. Це дає змогу внести додаткові зміни або скоригувати вже створений документ без потреби проходити весь процес заново. Такий підхід робить взаємодію з ботом більш гнучкою і максимально орієнтованою на індивідуальні потреби користувача.

Отже, виведення результатів у цій системі організоване так, щоб користувач мав швидкий і зручний доступ до адаптованого резюме, а також міг легко взаємодіяти з ботом на кожному етапі роботи.

3.5 Інтеграція компонентів системи

Для того щоб повною мірою зрозуміти принцип функціонування та ефективність даної системи – слід детально описати інтеграцію її компонентів. Цей пункт присвячений аналізу взаємодії окремих елементів, що утворюють цілісну архітектуру, та висвітленню механізмів, які забезпечують їхню злагоджену роботу. Особливістю Telegram API є те, що він дозволяє обробляти як текстові повідомлення, так і різні типи файлів, включаючи резюме в PDF-форматі. Ця гнучкість забезпечує високий рівень адаптивності системи, оскільки вона здатна працювати з різними сценаріями користувацької поведінки. Уся логіка бота зроблена так, що після отримання вхідних даних він автоматично починає ланцюг внутрішніх процесів від зчитування тексту з документа, до глибокого аналізу його змісту. І тут ключову роль відіграє інтеграція з OpenAI API, яка надає системі інтелектуальну функцію. Саме це відповідає за змістовну обробку інформації: аналізує подане резюме в контексті конкретної вакансії, виявляє релевантні навички, професійний досвід, досягнення та трансформує в змістовно насичене, стилістично правильне та орієнтоване на конкретного роботодавця резюме.

OpenAI API, інтегрований через хмарну платформу Azure, виконує основну аналітичну роботу за допомогою потужної мовної моделі GPT-4. Весь процес побудований таким чином, що після отримання тексту з резюме та вакансії система формує структуровані запити, промпти, які включають як зміст документа, так і вимоги до посади. На основі цього бот отримує від моделі змістовну відповідь, що містить нову версію резюме, вже адаптовану до вакансії. Вона враховує ключові очікування роботодавця, використовуючи відповідні формулювання, підкреслюючи необхідні компетенції, досягнення, стиль викладу та навіть тональність, властиву професійному середовищу. А щоб уся ця система працювала злагоджено й без збоїв, було впроваджено ефективний механізм логування, невід’ємну частину кожного етапу взаємодії між користувачем і ботом. Логування охоплює всі ключові моменти, як: надходження резюме, отримання посилання на вакансію, час звернення,

проміжні та фінальні результати генерації тексту, створення PDF-документа та відправлення його назад користувачу. Завдяки цьому механізму можливо не лише підтримувати стабільність сервісу, але й швидко виявляти технічні проблеми, наприклад, некоректний формат файлу або недійсне посилання. Усі такі випадки фіксуються в логах і можуть бути проаналізовані для подальшого вдосконалення системи. Аналітичною складовою логування можна покращити та відкрити можливості для аналізу поведінки користувачів, наприклад, які функції використовуються найчастіше, які вакансії цікавлять найбільше, як змінюється активність у різні періоди. Це дозволяє забезпечити прозорість функціонування всієї системи та зробити її гнучкішою, чутливішою до змін та запитів реальних користувачів.

Формування фінального документа, який користувач отримує у вигляді адаптованого PDF-файлу – етап, не просто синтаксично правильне оформлення, хоча це є досить суттєвим моментом. А підсумкове представлення всіх попередніх дій системи. Саме цей файл кандидат може одразу прикріпити до вакансії і податися на неї і тому до його якості висуваються особливі вимоги. Щоб реалізувати цей функціонал, у Telegram-бот було інтегровано PDF-генератор на основі бібліотеки FPDF, яка дозволяє точно структурувати інформацію, використовувати різні стилі, дотримуватись логіки сучасного резюме та створювати візуально привабливий документ без необхідності додаткового редагування. Після того як система аналізує текст резюме та зіставляє його з вакансією, створюється структурована версія, вона включає персональні дані, узагальнений опис професійного профілю, ключові навички, досвід, досягнення та освіту. [23] Ця інформація надходить до генератора, який перетворює її у фінальний файл. При цьому користувачеві не потрібно нічого додатково робити, у Telegram-чаті він просто отримує повідомлення з прикріпленим готовим PDF-документом. Така автоматизована подача результату не лише заощаджує час, а й дозволяє зробити весь процес максимально простим та доступним для кожного. Користувач не залишає межі звичного месенджера, і водночас отримує документ, який готовий до професійного використання. А для

того, щоб гарантувати стабільність роботи всієї системи, велике значення має ще один невидимий, але критично важливий механізм як, логування. У середовищі, де взаємодія з користувачем відбувається в режимі реального часу, а система використовує зовнішні API, необхідно мати інструмент, який дозволяє відстежувати хід виконання операцій та своєчасно виявляти потенційні збої. Тому при розробці бота було впроваджено систему логування, яка фіксує надходження повідомлень, кожен етап обробки, запити до OpenAI, відповіді сервера, процес створення PDF та надсилання його користувачу. Також визначено і описано основні компоненти системи, наведення яких, зображено в таблиці. (таблиця 3.2)

ТАБЛИЦЯ 3.2 – Основні компоненти системи.

№	Компонент	Бібліотеки / Технології	Опис функції
1	Середовище розробки	Python, venv	Базова платформа для створення і запуску проєкту
2	Telegram-інтерфейс	aiogram	Обробка запитів користувача, передача файлів і текстів
3	Робота з PDF	PyPDF2, pdfplumber	Витягнення тексту з резюме у форматі PDF
4	Взаємодія з API	requests	Надсилання запитів до сервісу обробки тексту, отримання результатів
5	Формування документа	fpdf2	Генерація фінального адаптованого резюме у PDF-форматі
6	Контроль версій	Git, GitHub	Управління змінами, командна робота, майбутні вдосконалення.

Це все дозволяє забезпечити повний цикл взаємодії з кожним користувачем, у простому та швидкому форматі. Не нехтуючи якістю системи, надійністю, можливості масштабування. Система має технічну базу для майбутніх оновлень і нових функцій. Завдяки такій архітектурі система залишається стабільною, безпечною та водночас гнучкою та здатною адаптуватися.

3.6 Методика проведення тестування

Тестування програмного забезпечення є важливою складовою процесу розробки, яка забезпечує правильну роботу системи та її здатність відповідати вимогам користувачів. Telegram-бот, який взаємодіє з зовнішніми API, зокрема з OpenAI, та обробляє документи, тому тестування стає ще більш критичним і вражаючим. Воно охоплює всі етапи взаємодії користувача з ботом, починаючи від отримання першого повідомлення і закінчуючи наданням адаптованого резюме у форматі PDF. І кожен етап потребує окремої уваги та перевірки. Одним із перших таких етапів тестування є перевірка отримання та обробки вхідних даних. Для системи важливо, щоб правильно зчитувалися PDF-файли, отримувалася з них потрібна інформація та коректно адаптувалась під вимоги вакансії. Це включає в себе не тільки текстову інформацію та ключові фрагменти, а й саме форматування документа.

Далі тестувалася інтеграція бота з OpenAI API. Важливо було перевірити, чи модель правильно аналізує текст вакансії, чи коректно адаптує резюме відповідно до вимог роботодавця. Під час тестування перевірялися й інші аспекти роботи API. Чи забезпечує модель належну якість аналізу та чи відповідає результат високим стандартам точності, якості та професіоналізму. Також, нее менш важливим саме тестування функції генерації PDF-документа. Для цього була використана бібліотека FPDF, яка дозволяє створювати структуровані документи з різними стилями та форматуванням. Тестування включало багаторазову перевірку правильності форматування документа, наявності всіх необхідних розділів і забезпечення точності у кінцевому результаті. Важливо також було перевірити, чи коректно генерується PDF-файл і чи надсилається він користувачу без помилок.

Для більш якісного кінцевого результату, тестування проводилося на різних сценаріях, включаючи випадки з некоректними даними, що дозволило перевірити, як система реагує на помилки, як генерується зворотний зв'язок та чи коректно інформує користувача у разі виникнення неполадок. Наприклад, якщо користувач

надіслав своє резюме не у форматі PDF-файлу, система одразу повідомить і вкаже, що потрібно змінити формат подачі. Це важливо для того, щоб бот був стабільним у роботі і не допускав помилок під час обробки запитів. Тестування таких сценаріїв гарантує, що програма здатна реагувати на різноманітні ситуації, забезпечуючи користувачеві зрозумілий і коректний зворотний зв'язок.

Тестування було виконано як вручну, так і за допомогою автоматизованих тестів, що дозволило своєчасно виявляти помилки на ранніх етапах розробки та коригувати їх. Це дає можливість значно покращити якість програмного продукту, забезпечити стабільність роботи та зменшити ризики, пов'язані з помилками, що можуть виникнути в процесі активного використання бота. Завдяки комплексному підходу до тестування, включаючи перевірку функціональності, інтеграцій, форматування та обробки помилок, вдалося створити надійний продукт, який задовольняє вимоги користувачів і працює без збоїв. [24] Таке тестування дає впевненість, що бот буде ефективно працювати в реальних умовах, допомагаючи користувачам автоматизувати процес адаптації резюме під конкретні вакансії.

У випадку з ботом, що автоматизує адаптацію резюме під вакансії, важливо перевірити, як система справляється з одночасним обробленням кількох запитів, що можуть містити великі PDF-файли та вимагати обробки складних текстових запитів через OpenAI API. Це тестування не тільки визначає, чи витримує система високий рівень навантаження, а й дозволяє оцінити, чи зберігається якість обробки кожного запиту, навіть коли кількість одночасних користувачів зростає. Для виконання навантажувального тестування були створені умови, що імітують одночасну обробку кількох десятків запитів. Використовувалися спеціальні інструменти, такі як Apache JMeter та Locust, що дозволяють моделювати реальні умови роботи бота та вимірювати час реакції системи, наявність затримок, а також перевіряти, чи не виникають помилки при високих навантаженнях. [25] Важливою частиною тестування стало перевірка того, як система обробляє саме PDF-документи, які можуть вимагати значних ресурсів для їх зчитування та обробки. Бот мав успішно працювати навіть з такими великими обсягами даних, не допускаючи зниження

швидкості обробки запитів або появи затримок. У цих ситуаціях необхідно було перевірити, чи здатна система ефективно керувати кількома паралельними процесами без значного уповільнення або виникнення збоїв у роботі. Тестування також включало у себе і перевірку поведінки бота при максимальному навантаженні, щоб переконатися, що система працює стабільно навіть у найскладніших умовах. Результати тестування стали основою для подальшої оптимізації системи. Виявлені проблеми, такі як надмірне навантаження на сервери, потреба в поліпшенні алгоритмів обробки запитів або оптимізація роботи з бібліотеками для генерації PDF-документів, були вирішені.

Іншим важливим видом тестування є тестування безпеки, яке забезпечує захист особистих даних користувачів. Оскільки бот взаємодіє з зовнішніми системами та передає дані через інтернет, важливо забезпечити їхню конфіденційність. Перевірка шифрування даних, таких як SSL, гарантує, що дані не можуть бути перехоплені під час передачі. Це включає в себе перевірку безпеки з'єднань, щоб не допустити витоків даних через неналежно налаштовані канали зв'язку. Тестування також включає перевірку захищеності API, щоб переконатися, що ключі доступу до зовнішніх систем, таких як OpenAI, не можуть бути вкрадені. Ще додатковим рішенням у програмному продукті було впровадження тимчасового зберігання файлу, що надає користувач, для адаптації.

Бот може отримувати текстові запити від користувачів, тому важливо вжити заходів, щоб був захист від спаму або зловмисних спроб отримати доступ до чужих даних. Система є здатною коректно обробляти всі вхідні повідомлення, щоб не дати можливості вплинути на її роботу зловмисникам. Всі виявлені уразливості було виправлено, а сама система відповідає найкращим стандартам безпеки, гарантуючи, що всі особисті дані залишаються захищеними.

Ручне тестування відіграє важливу роль на початкових етапах розробки, коли тестувальник перевіряє, як бот взаємодіє з користувачами, чи правильно він обробляє запити, завантажує резюме у форматі PDF, аналізує їх і генерує адаптовані версії для вакансій. Систему вручну перевірено на зв'язок з API, коректність та

бачення аналізу вакансій та резюме, а також відправку результатів користувачеві. Це дозволило виявити недоліки, які були не помічені під час автоматизованих перевірок. Автоматичне тестування, в свою чергу, дозволило перевірити основні функції бота, забезпечуючи стабільність його роботи при обробці запитів, взаємодії з API та генерації PDF-файлів. Автоматизація тестування дозволяє скоротити час перевірок, знизити ймовірність людських помилок та прискорити виявлення проблем на ранніх етапах. Ідеальний підхід до тестування поєднує ручне та автоматичне тестування, що дає можливість забезпечити всебічну перевірку програмного забезпечення. Тестування має здійснюватися на всіх етапах розробки, від початкових перевірок до більш глибоких інтеграційних тестів.

Аналіз помилок – виявлення та вивчення помилок, які можуть виникати в результаті неправильних дій користувачів або неочікуваних ситуацій. [26] Це включає в себе неправильний ввід даних, такі як порожні файли чи некоректно заповнені резюме, а також проблеми з зовнішніми API, наприклад, випадки, коли Azure OpenAI не відповідає або не може обробити запит. Протестовано всі можливі варіанти введення, щоб система могла коректно реагувати на різні типи помилок, запобігаючи можливим перешкодам у роботі. Один із ключових етапів аналізу помилок тут є тестування на рівні інтеграції з API. Адже якщо зовнішні сервіси, такі як Azure OpenAI, не надають очікуваних результатів або відбуваються проблеми з підключенням, це може призвести до збоїв у роботі системи. Бот було налаштовано на обробку таких ситуацій: якщо API не відповідає або виникає помилка при отриманні даних, бот не має зупиняти свою роботу, а повинен повідомляти користувача про проблему, даючи йому можливість скоригувати дії чи спробувати знову. Після виявлення помилок важливо проводити їх детальний аналіз і виправлення. Помилки не тільки потребують оперативного усунення, але й мають бути документовані, щоб у майбутньому запобігти їх повторенню.

3.7 Вибір тестових даних

Правильний вибір тестових даних є важливим для досягнення ефективності та коректності роботи системи. Враховуючи, що Telegram-бот використовує Azure OpenAI для аналізу вакансій та адаптації резюме, важливо підготувати тестові набори, які максимально відображатимуть реальні ситуації, з якими користувачі можуть зіткнутися. Це дозволить переконатися, що система працює правильно і надійно, а також враховує всі можливі варіанти використання. Структура тестування самого резюме повинна бути різноманітною та відповідати вимогам, які система повинна обробляти. Тестові дані включають різні формати та стилі резюме, що відрізняються за структурою, рівнем деталізації, досвідом роботи, кваліфікацією, навичками та освітою. Було враховано, що тестові резюме можуть варіюватися від простих варіантів з мінімальними даними до більш складних форм з розширеними описами досягнень, проектів, спеціальних навичок та інших аспектів, що допомагають оцінити здатність системи працювати з різними рівнями складності. Завдяки різноманітності тестувань резюме, це дозволило перевірити, як система адаптує ці документи до конкретних і у той самий час різних вакансій. Наприклад, тестувалося резюме, що містять лише основну інформацію про кандидата, а також резюме, яке включає велику кількість деталей про професійний досвід, досягнення чи участь у великих проектах, що було враховано. Таким чином в результаті, система коректно визначає важливі дані навіть у найбільш спрощених резюме, а також працює з більш складними документами, де є потреба в глибшому аналізі. [27]

Не менш важливою було. Перевірити резюме в різних форматах, що стосуються як технічних, так і нетехнічних спеціалізацій. Так це дозволило оцінити, як добре система здатна розпізнавати специфічні навички та досвід, які важливі для тих чи інших вакансій, і як точно адаптує резюме під ці вимоги. Крім того, тестування з використанням різноманітних помилок або неструктурованих даних дало змогу перевірити, як бот справляється з неповними чи некоректно

оформленими даними, чи здатен він обробляти такі ситуації без помилок чи збоїв. Загалом, мета була у тому, щоб тестові дані охоплювали широкий спектр можливих ситуацій, з якими бот може зіткнутися під час адаптації резюме до вакансії. Тільки так можна провести всебічне тестування, яке забезпечить надійність і точність системи в реальних умовах. Тестування різноманітних варіантів резюме, як з коректними, так і з помилковими чи неповними даними, дозволило не лише перевірити основні функції бота, але й оцінити його здатність до адаптації в умовах, наближених до реальних.

Другим фрагментом тестових даних є самі вакансії, що йдуть важливою складовою процесу тестування програмного забезпечення, яке займається адаптацією резюме під конкретні вимоги робочих місць. У разі Telegram-бота, що використовує Azure OpenAI для аналізу вакансій та адаптації резюме, підготовка відповідних прикладів вакансій стає необхідною для оцінки, як система справляється з різними типами посад і професій. Тестування вакансій дозволило перевірити, наскільки добре працює система в умовах реальних сценаріїв, коли потрібно адаптувати резюме під специфічні вимоги, що можуть варіюватися залежно від посади чи галузі. Приклади вакансій були різноманітними, щоб охопити якомога більше типів робочих місць та вимог. Такий підхід дав змогу оцінити, як система справляється з різними рівнями кваліфікації, вимогами до досвіду, а також з вакансіями з різних сфер діяльності. Всі ці вакансії були репрезентативними для сучасного ринку праці та включали як базові вимоги, наприклад, необхідні знання мов програмування чи досвід роботи в певній галузі, так і більш складні запити, що потребують детальнішої адаптації резюме з урахуванням специфіки вакансії. Завдяки цьому система зможе ефективно працювати в умовах, де необхідно враховувати не тільки базові вимоги, але й більш глибокі особливості професії.

Тестування включало в себе вакансії для різних професійних напрямків. Це можуть бути не тільки такі як технічні спеціальності, як програмісти, інженери чи розробники, так і нетехнічні ролі, наприклад, менеджери проектів, маркетологи, HR-спеціалісти, лікарі, юристи та увесь набір існуючих професій, які потребують

резюме. Важливо враховувати різні рівні досвіду – від вакансій для початківців до більш кваліфікованих посад, що вимагають багаторічного досвіду та специфічних знань. Вакансії включають різні вимоги до освіти, кваліфікацій, навичок та досвіду роботи, а також додаткові побажання, наприклад, знання конкретних інструментів, вміння працювати в команді чи володіння певними мовами. Це допомогло перевірити, як система справляється з різними запитами та адаптує резюме таким чином, щоб воно максимально відповідало вимогам роботодавця. Ці вакансії відображали реальні умови, з якими зустрічаються кандидати на ринку праці, адже тільки в такому випадку можна досягти максимальної ефективності в адаптації резюме. Врахування специфіки вакансій та вимог до кандидатів дали змогу перевірити, чи здатна система правильно інтерпретувати ключові вимоги, знаходити важливі навички та досвід, а також надавати резюме, яке є не лише грамотно адаптованим, але й орієнтованим на конкретного роботодавця. Це дозволило протестувати здатність бота до визначення ключових слів, вимог та критеріїв, на основі яких має бути побудоване резюме. Перевірено, що система може створити чудове резюме, який відповідає вимогам конкретної вакансії.

Щодо аналізу відповідності, що є ще однією важливою складовою тестування програмного забезпечення, що відповідає за адаптацію резюме кандидата до вимог конкретної вакансії. Цей процес передбачає перевірку того, наскільки ефективно система, яка використовує ШІ, може адаптувати надане резюме, аби воно точно відповідало специфікаціям вакансії, підвищуючи таким чином шанси кандидата на успіх. Завдяки цьому етапу було оцінено, як здатен бот правильно розпізнавати ключові вимоги, такі як навички, досвід роботи, освіта, а також інші важливі фактори, що можуть впливати на вибір кандидата для вакансії. Один із головних аспектів аналізу відповідності – це здатність системи правильно ідентифікувати важливі елементи вакансії. Це стосувалося як базових вимог, таких як знання певних мов програмування чи досвід роботи в певній галузі, так і більш тонких критеріїв, таких як здатність працювати в команді, комунікаційні навички чи лідерські якості. Важливо було, щоб система не лише розпізнавала ці вимоги, але й могла точно

вказати, які елементи резюме потрібно адаптувати, аби воно максимально відповідало вимогам роботодавця. Це дозволило збільшити ймовірність того, що кандидат буде відібраний для вакансії, оскільки його резюме точно відповідає всім вимогам.

Оцінка результатів GPT-4 є важливим етапом тестування програмного забезпечення, яке використовує штучний інтелект для адаптації резюме під вимоги конкретних вакансій. У випадку бота, що працює на основі Azure OpenAI та моделі GPT-4, цей етап стає вирішальним для визначення ефективності процесу адаптації резюме. Метою було оцінити, наскільки точними та релевантними є результати, що генерує модель, і чи відповідають вони вимогам вакансії, що вказуються роботодавцем. Процес оцінки результатів було проведено в реальних умовах. Кожен тестовий запит, який подається ботом, піддався ретельному аналізу, щоб з'ясувати, наскільки коректно система витягує інформацію з вакансії та резюме. Важливо було розібратися, як вона застосовує свої знання для створення адаптованого тексту, який максимально відповідає вимогам.

Оцінкою результатів передбачено використання різних тестових даних, що включали резюме з різними форматами, структурами та змістом, а також вакансії з різними вимогами до кандидатів. GPT-4 здатен правильно розпізнавати ключові навички, досвід і кваліфікацію, які підкреслені в адаптованому резюме для успішної подачі на вакансію. GPT-4 добре генерує текст, що відповідає вимогам вакансії. Створений текст не лише відповідає вимогам, але й передає важливу інформацію про кандидата, що може бути цікавою роботодавцю. Передбачено перевірку точності адаптованого резюме. Моделі вдалося вірно сформулювати ключові моменти в резюме, та правильно підкреслити необхідні навички та досягнення кандидата. Текст відповідає вимогам конкретної вакансії і є логічно послідовним та природним. Модель змогла створити текст, що звучить природно, зберігаючи одночасно всю необхідну точність і деталі.

3.8 Аналіз результатів тестування

Аналіз результатів тестування було проведень за кількома важливими метриками. Оцінка точності та швидкодії є одними з ключових етапів аналізу результатів для ПП, оскільки ці фактори безпосередньо впливають на ефективність роботи програмного забезпечення та його здатність задовольняти потреби користувачів. Перш за все, оцінка точності полягала не лише в перевірці того, як добре система розпізнає та інтегрує ключові аспекти вакансії в адаптоване резюме, але й в тому, наскільки грамотно система враховує досвід, кваліфікацію та навички кандидата, щоб створити відповідний документ. Система не тільки правильно виокремлювала важливі моменти, такі як навички та досвід, але й щоб вона зберігала професіоналізм і коректність у стилістиці, формулюваннях та форматуваннях, забезпечуючи цим не тільки точність, а й презентабельність тексту. Це дозволило резюме не тільки відповідати вимогам вакансії, але й бути грамотно складеним, що підвищує шанси кандидата на успіх. Оцінка точності також включала перевірку відповідності тексту резюме професійному стандарту, що важливо для роботодавців, які шукають добре структуровані документи, що чітко відображають досвід та навички кандидата. Важливим аспектом є те, що система адаптує текст під різні вакансії та здатна врахувати специфіку кожної конкретної посади, а не загальні сфери, створюючи максимально релевантний результат. Система може створювати персоналізовані тексти, що відповідають вимогам роботодавців, і добре справляється з аналізом вимог та переведенням їх в адаптовану форму, яка буде зрозуміла та приваблива для потенційного роботодавця.

Тестування швидкодії бота прямо впливає на досвід користувача у використанні. Швидкодія визначає, скільки часу система витрачає на обробку запиту та генерування адаптованого резюме у форматі PDF. У випадку, коли система використовує Azure OpenAI для аналізу текстів, було враховано час, що

витрачається на запит до API та обчислення, пов’язані з адаптацією тексту до вимог вакансії. Час обробки запиту безпосередньо залежить від складності вакансії та резюме, тому передбачено, що система здатна обробляти навіть найскладніші запити швидко і ефективно.

Система справляється з великими обсягами даних і високими навантаженнями, а також була здатна адаптуватися до різних типів запитів і працювати стабільно при високому трафіку, забезпечуючи ефективність навіть при великій кількості одночасних користувачів. Оптимізація швидкодії має на меті не лише зниження часу обробки запитів, але й адаптацію до різних умов навантаження, що є важливим для забезпечення безперебійної роботи системи. В цілому, поєднання точності та швидкодії є основою для створення конкурентоспроможної системи, яка може успішно конкурувати на ринку різних автоматизованих рішень для адаптації резюме. Система не тільки забезпечує високу точність у відображенні навичок та досвіду кандидата, але й швидко обробляє запити, що забезпечить високий рівень задоволеності користувачів та підвищить ефективність процесу подачі резюме.

Рекомендації користувачів є одним із основних критеріїв для оцінки ефективності програмного забезпечення. В даному контексті, система надала точні, релевантні та зрозумілі рекомендації, щоб кожен кандидат міг успішно адаптувати своє резюме для конкретної вакансії. Саме на основі таких рекомендацій кандидат має можливість коригувати своє резюме, щоб воно відповідало вимогам роботодавця. А для оцінки якості рекомендацій було необхідно врахувати кілька основних аспектів. Перш за все, система може правильно інтерпретувати вимоги вакансії та порівняти їх із резюме кандидата. Зосередженість є на найважливіших елементах, таких як досвід роботи, навички та досягнення кандидата, щоб вони відповідали вимогам, зазначеним в описі вакансії. Це дозволило зробити адаптоване резюме більш конкретним і релевантним, а також підвищує шанси кандидата пройти відбір. Крім того, важливо, щоб результат адаптації був ясним, чітким і без зайвих узагальнень, що допомагає кандидату краще зрозуміти, на яких аспектах слід зосередитися для покращення свого резюме.

ВИСНОВКИ

У межах цієї кваліфікаційної роботи було розроблено Telegram-бота для автоматизованої адаптації резюме під конкретну вакансію з використанням сучасних інформаційних технологій. Система дозволяє користувачеві завантажити своє резюме у форматі PDF, вказати посилання на вакансію, після чого на основі інструментів обробки природної мови та інтеграції з Azure OpenAI виконується аналіз відповідності професійного досвіду вимогам заданої користувачем вакансії. Кінцевим результатом є оновлений варіант резюме, сформований у вигляді PDF-файлу, який повертається користувачеві через Telegram-інтерфейс і здатен до редагування, якщо у цьому є потреба.

Під час реалізації проєкту було здійснено низку взаємопов’язаних етапів: проведено аналіз існуючих підходів до адаптації резюме, спроєктовано архітектуру системи, обрано відповідні та чіткі інструменти (Python, aiogram, FPDF, Azure OpenAI), розроблено окремі модулі для аналізу тексту та генерації PDF, протестовано функціональність бота, а також оцінено якість результатів на основі порівняння з традиційними методами підбору кандидатів на роботу. Кожен з етапів виконувався з урахуванням чіткої послідовності дій і дозволив досягти поставленої мети.

Отримані результати демонструють високу ефективність підходу до персоналізованої адаптації резюме. Запропоноване програмне рішення скорочує час на підготовку та подання заявки, знижує потребу в ручному редагуванні документа, а також підвищує точність відповідності між кандидатом і вакансією. Це, у свою чергу, сприяє підвищенню шансів на працевлаштування, покращенню досвіду користувачів та оптимізації процесу рекрутингу. Впровадження такого ПЗ дає користувачам можливість заощадити час і зусилля, мінімізує суб’єктивний фактор під час оцінювання резюме та дозволяє зосередитися на найрелевантніших навичках та досвіді. Система є масштабованою та може бути адаптована для використання в різних сферах: кадрових агентствах, HR-відділах компаній,

платформах для пошуку роботи, освітніх проєктах із підготовки молодих спеціалістів тощо. Окрім цього, важливо підкреслити, що рішення не лише підвищує ефективність взаємодії між шукачами роботи та роботодавцями, а й сприяє формуванню більш прозорого та об’єктивного підходу до оцінювання кандидатів. Його застосування дозволяє зменшити ризики дискримінації на основі суб’єктивних вражень або несистемного аналізу, забезпечуючи рівні умови для всіх претендентів. Сервіс може бути корисним як для досвідчених фахівців, так і для початківців, які ще не сформували чітке професійне портфоліо, адже система допомагає виділити сильні сторони та підкреслити їх у контексті конкретної вакансії.

Перспективи подальшого розвитку системи є багатообіцяючими. Вони включають інтеграцію багатомовної підтримки, що значно розширить географію застосування рішення, а також створення інтуїтивно зрозумілого веб-інтерфейсу для максимально зручного доступу користувачів. Планується розширення алгоритмів оцінки відповідності, зокрема, шляхом врахування так званих soft skills (м'яких навичок), які часто відіграють ключову роль у успішному працевлаштуванні та адаптації в колективі. Крім того, передбачається впровадження можливості зворотного зв’язку з користувачем для додаткового навчання моделі та постійного покращення її точності й ефективності.

Результати цієї роботи можуть стати основою для подальших наукових досліджень і практичних рішень у сфері автоматизації підбору персоналу на основі штучного інтелекту. Крім того, накопичені дані можуть бути використані для виявлення тенденцій ринку праці, аналізу попиту на певні професії та формування рекомендацій для освітніх установ щодо актуалізації навчальних програм відповідно до поточних потреб індустрії. Таким чином, це рішення не лише оптимізує індивідуальний процес пошуку роботи, а й має потенціал для глобального впливу на розвиток ринку праці та системи професійної освіти.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. How HR Automation Can Optimize the Candidate Experience. URL: https://www.hrtechnologist.com/articles/recruitment-onboarding/how-hr-automation-can-optimize-the-candidate-experience/ (дата звернення: 03.01.2025).
2. Natural Language Processing (NLP) in Human Resources – Azure OpenAI Documentation. URL: https://azure.microsoft.com/en-us/products/ai-services/openai-service/ (дата звернення: 17.02.2025).
3. Gartner Says 75% of Organizations Will Have Adopted AI for HR by 2025. URL: https://www.gartner.com/en/newsroom/press-releases/2023-09-12-gartner-says-75-percent-of-organizations-will-have-adopted-ai-for-hr-by-2025 (дата звернення: 09.01.2025).
4. LinkedIn. URL: https://www.linkedin.com (дата звернення: 12.02.2025).
5. Resume Worded – AI-Powered Resume Review. URL: https://resumeworded.com (дата звернення: 05.02.2025).
6. Speech and Language Processing (3rd ed. draft) by Daniel Jurafsky and James H. Martin. URL: https://web.stanford.edu/~jurafsky/slp3/ (дата звернення: 14.02.2025).
7. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. URL: https://arxiv.org/abs/1810.04805 (дата звернення: 06.01.2025).
8. GPT-4 Technical Report. URL: https://openai.com/research/gpt-4-technical-report (дата звернення: 20.02.2025).
9. The Elements of Statistical Learning by Trevor Hastie, Robert Tibshirani, and Jerome Friedman. URL: https://web.stanford.edu/~hastie/ElemStatLearn/ (дата звернення: 02.01.2025).

10. Evaluating Machine Learning Models – Classification Metrics. URL: https://developers.google.com/machine-learning/crash-course/classification/metrics (дата звернення: 15.02.2025).

11. Microservices – Martin Fowler. URL: https://martinfowler.com/articles/microservices.html (дата звернення: 23.02.2025).

12. How to Beat the Applicant Tracking System (ATS) in 2024 – The Muse. URL: https://www.themuse.com/advice/how-to-beat-the-applicant-tracking-system-ats-resume (дата звернення: 28.02.2025).

13. How to design chatbots – The Ultimate Guide | UXDesign.cc. URL: https://uxdesign.cc/how-to-design-chatbots-the-ultimate-guide-9ddc7b28669e (дата звернення: 27.02.2025).

14. PyMuPDF Documentation. URL: https://pymupdf.readthedocs.io/en/latest/ (дата звернення: 03.03.2025).

15. Hugging Face – Transformers Documentation. URL: https://huggingface.co/docs/transformers/ (дата звернення: 05.03.2025).

16. Azure OpenAI Service – Microsoft Learn. URL: https://learn.microsoft.com/en-us/azure/ai-services/openai/ (дата звернення: 09.03.2025).

17. How to Beat the Applicant Tracking System (ATS) in 2024 – The Muse. URL: https://www.themuse.com/advice/how-to-beat-the-applicant-tracking-system-ats-resume (дата звернення: 11.03.2025).

18. Python Documentation – json Library. URL: https://docs.python.org/3/library/json.html (дата звернення: 17.03.2025).

19. Docker Documentation – Get Started. URL: https://docs.docker.com/get-started/overview/ (дата звернення: 19.03.2025).

20. Web Scraping with Python – Ryan Mitchell. O'Reilly Media. (дата звернення: 22.03.2025).

21. Azure OpenAI Service – Models Overview. URL: https://learn.microsoft.com/en-us/azure/ai-services/openai/concepts/models (дата звернення: 25.03.2025).

22. Aiogram FSM – aiogram documentation. URL: https://docs.aiogram.dev/en/latest/dispatcher/fsm.html (дата звернення: 29.03.2025).

23. How to Format a Resume – The Balance Careers. URL: https://www.thebalancecareers.com/how-to-format-your-resume-2062332 (дата звернення: 01.04.2025).

24. Software Testing Basics: A Guide to Types, Principles, and Methods. (дата звернення: 05.04.2025).

25. What is Load Testing? – k6 Blog. URL: https://k6.io/blog/what-is-load-testing/ (дата звернення: 09.04.2025).

26. Effective Bug Management and Error Analysis in Software Testing. (дата звернення: 15.04.2025).

27. Types of Resumes – Indeed Career Guide. URL: https://www.indeed.com/career-advice/resumes-cover-letters/types-of-resumes (дата звернення: 28.04.2025).

28. Debugging Techniques in Python – Real Python. URL: https://realpython.com/python-debugging-pdb/ (дата звернення: 23.03.2025).

29. Load Testing vs. Stress Testing – Apache JMeter Blog. URL: https://jmeter.apache.org/usermanual/glossary.html#load_test (дата звернення: 12.04.2025).

30. Бедратюк Л. П. Types of Resumes – Indeed Career Guide URL: https://www.indeed.com/career-advice/resumes-cover-letters/types-of-resumes (дата звернення: 28.04.2025).

31. Бедратюк Л. П. Debugging Techniques in Python – Real Python URL: https://realpython.com/python-debugging-pdb/ (дата звернення: 23.03.2025).

ДОДАТОК А

(обов’язковий)

ГРАФІЧНІ МАТЕРІАЛИ

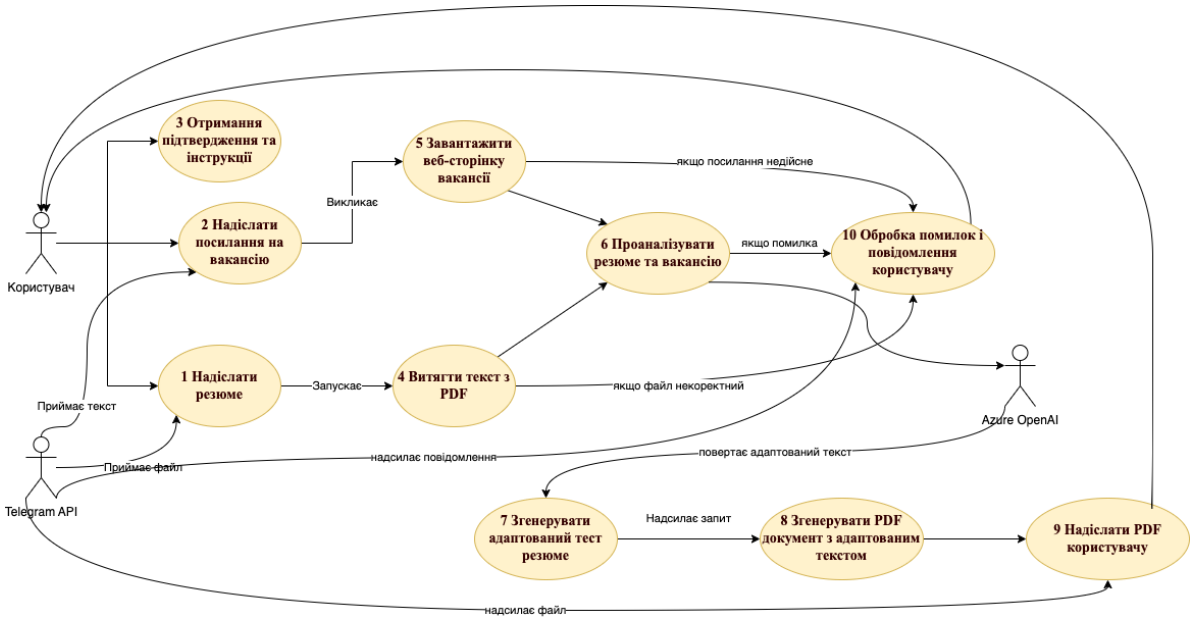


Рисунок А1 – Діаграма варіантів використання.

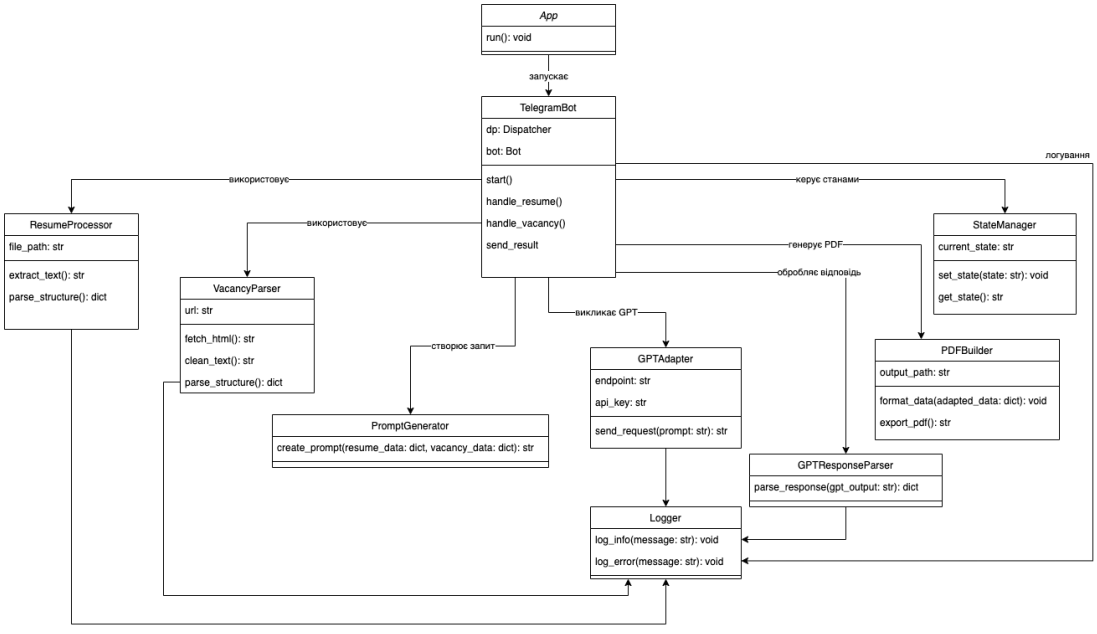


Рисунок А2 – Діаграма класів.

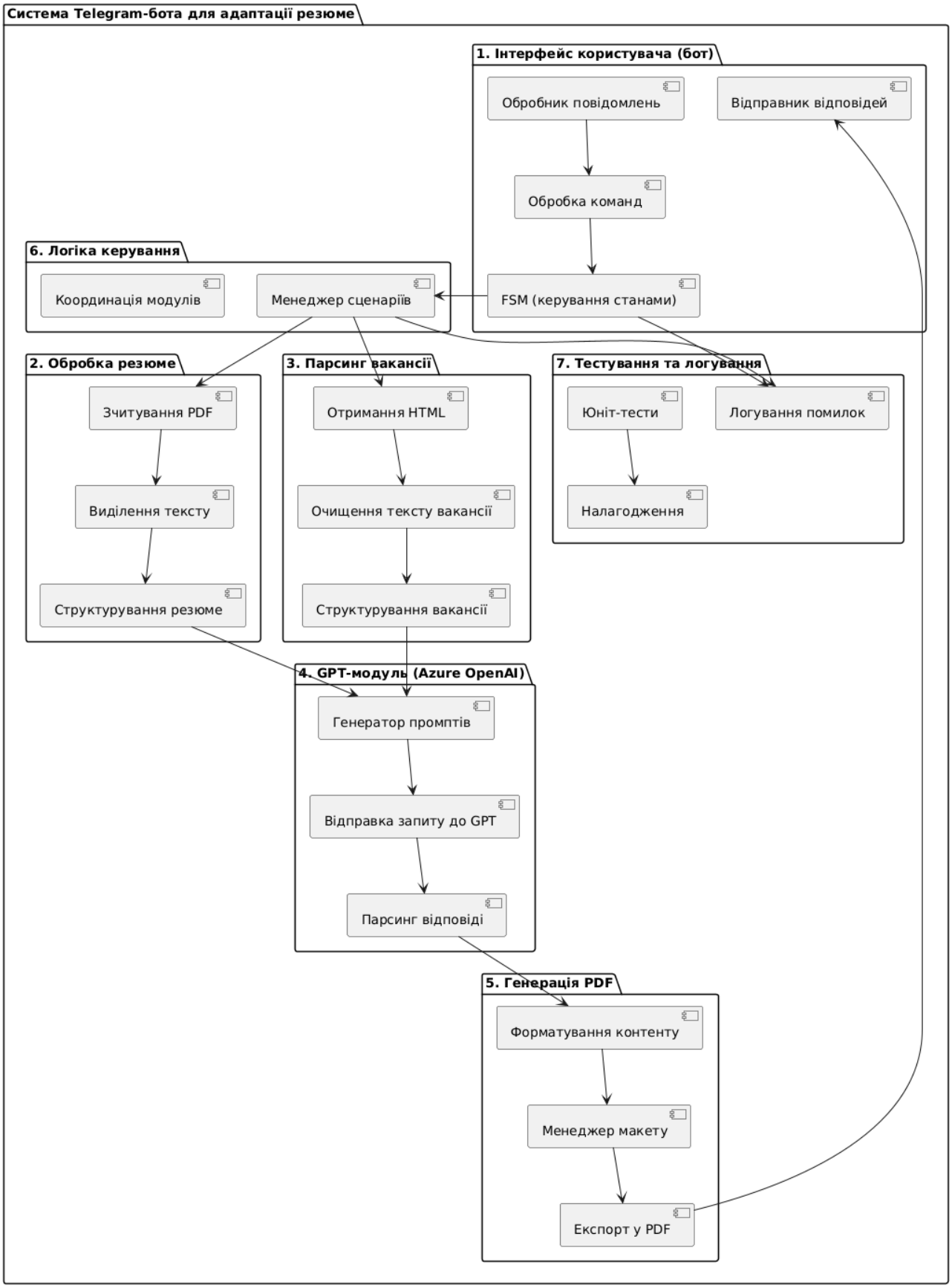


Рисунок А3 – Діаграма зв’язків модулів.

Рисунок А4 – Діаграма DFD (рівень 0) – Контекстна діаграма.

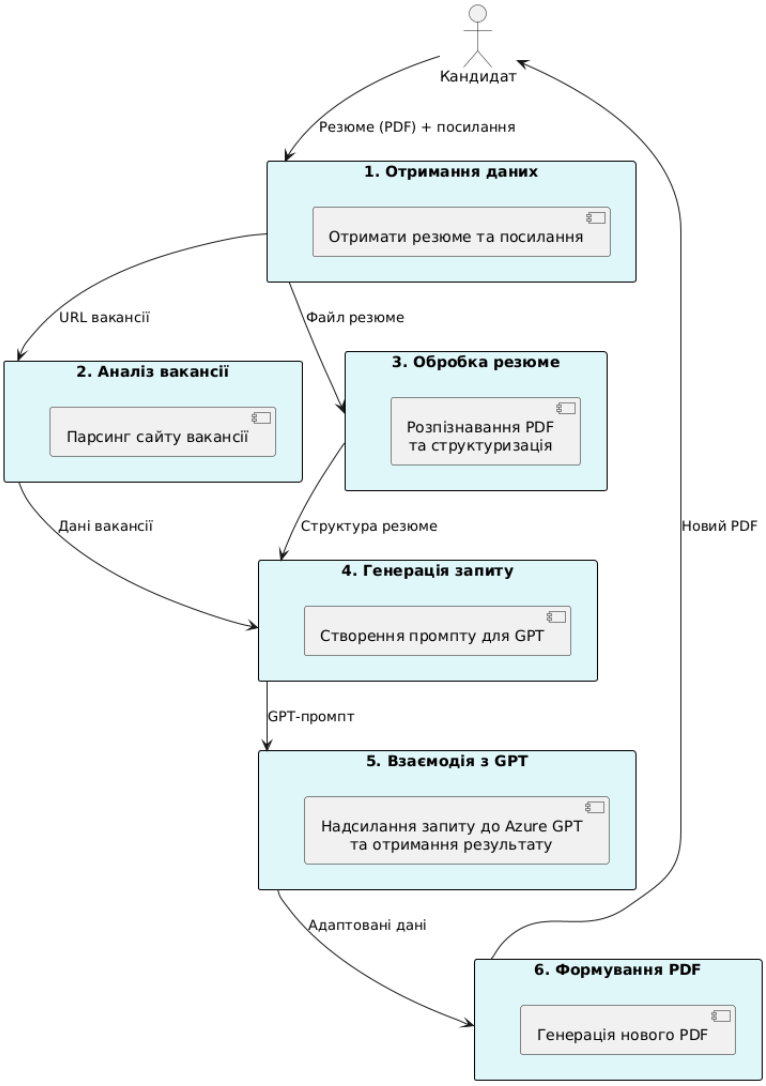


Рисунок А5 – Діаграма DFD (рівень 1) – Основні процеси в системі.

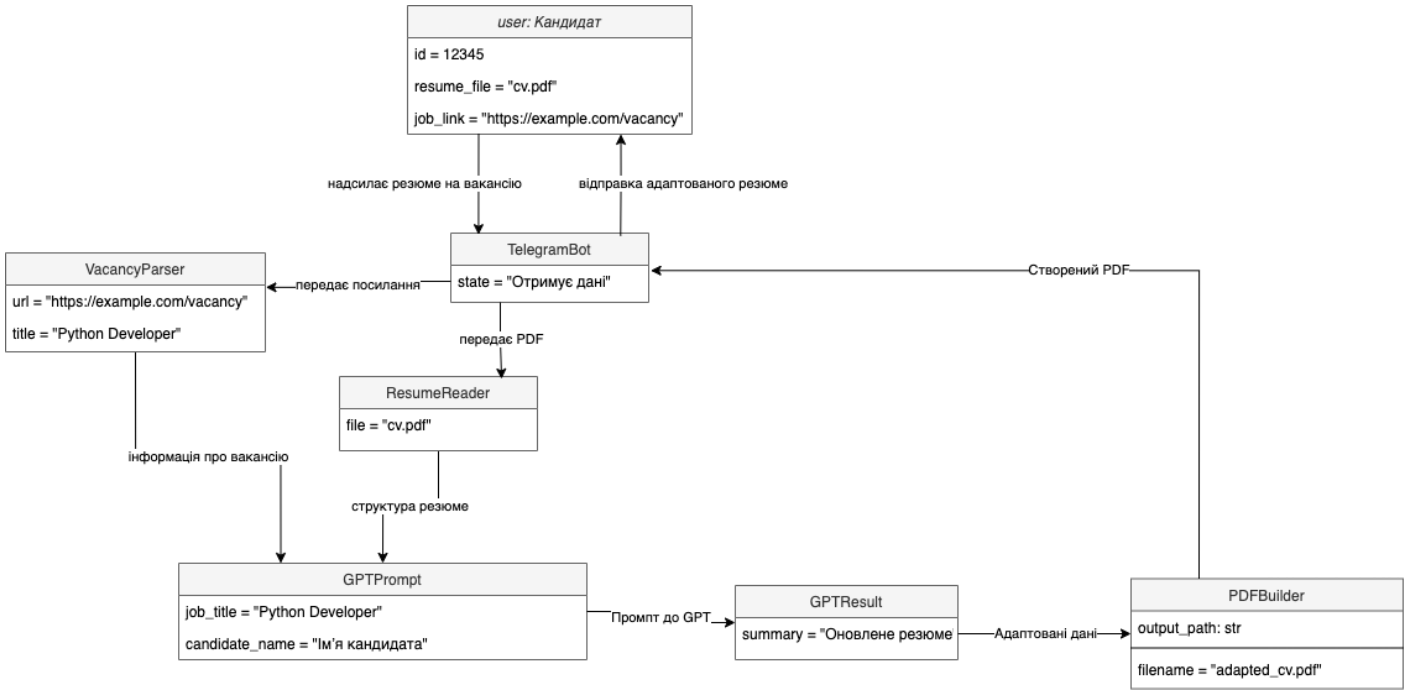


Рисунок А6 – Діаграма об’єктів.

Jon Dou
Senior Java Softwareentwickler

email@example.com
+123456789

SUMMARY

Experienced Senior Full Stack Developer with over 7 years of expertise in designing, developing, and supporting complex applications. Skilled in both frontend and backend technologies, with a strong focus on Java and related frameworks. Adept at working in dynamic, agile environments and committed to delivering top-notch software solutions.

SKILLS

Java, Spring Boot, Spring, Angular, RESTful APIs, Microservices Architecture, CI/CD, AWS Technologies, Docker, Kubernetes, RDBMS, NoSQL

LANGUAGES

English, German

WORK EXPERIENCE

Senior Full Stack Developer
- Designed and developed scalable automation solutions in a fast-paced, agile environment.
- Led the design and implementation of various business process automation initiatives.
- Developed applications using Java, Spring Boot, Spring, and Angular frameworks.
- Integrated RESTful APIs and employed Microservices Architecture for application development.
- Utilized RDBMS such as Postgres/SQL and NoSQL databases like MongoDB and DynamoDB.
- Implemented AWS technologies including SQS, SNS, Lambda, API Gateway, and S3.
- Developed and maintained CI/CD pipelines and containerized applications using Docker and Kubernetes.

Java Developer
- Participated in all phases of the software development cycle from design to post-production support.
- Created detailed application specifications and technical designs.
- Contributed to continual improvement by suggesting enhancements to user interfaces and software architecture.

EDUCATION

XYZ University
Bachelor's Degree in Computer Science

CERTIFICATIONS

AWS Certified Solutions Architect, Certified Kubernetes Administrator

ACHIEVEMENTS

Led a project that reduced application deployment time by 30% through implementation of CI/CD pipelines.
Improved application performance by optimizing database queries and microservices communication.

EMPLOYEE BENEFITS

Access to diverse professional development opportunities including conferences and training sessions.
Mentorship program to support rapid onboarding and continuous growth.
Various employee benefits including meal subsidies, job tickets, and profit-sharing based on company success.

Рисунок А7 – Макет формату резюме.

ДОДАТОК Б

(обов'язковий)

ПРОГРАМНИЙ КОД ОСНОВНИХ МОДУЛІВ

Б.1 Код ініціалізації бота

```
import os
import logging
import sys
from aiogram import Bot, Dispatcher
from aiogram import F
from routes import router
from apps.write_cv.write_cv_routes import write_cv_router
from apps.feedback_insights import feedback_routes

API_TOKEN = os.getenv("BOT_TOKEN")

# Ініціалізація бота та диспетчера
bot = Bot(token=API_TOKEN)
dp = Dispatcher()

# Ініціалізація роутеру
dp.include_router(router)
dp.include_router(write_cv_router)

async def main():
    # Видалення вебхука, якщо він існує
    await bot.delete_webhook()

    # Запуск polling
    await dp.start_polling(bot)

if __name__ == '__main__':
    import asyncio
    logging.basicConfig(level=logging.INFO, stream=sys.stdout)
    asyncio.run(main())
```

Б.2 Код для налаштування системи

```
import os
import sys

# if os.environ.get('AWS_LAMBDA_FUNCTION_NAME'):
#     # Додаємо шлях до проєкту в sys.path тільки в середовищі Lambda
#     current_dir = os.path.dirname(os.path.abspath(__file__))

def get_env_variable(name, default=None):
    """ Отримати змінну середовища або завершити роботу, якщо її немає і default
не вказаний """
    value = os.environ.get(name, default)
    if value is None:
        print(f"Помилка: не знайдено змінну середовища {name}")
        sys.exit(1)
    return value

# Змінні для Telegram бота та веб-сервера
BOT_TOKEN = get_env_variable("BOT_TOKEN")
WEB_SERVER_HOST = get_env_variable("WEB_SERVER_HOST", "127.0.0.1")
```

```
WEB_SERVER_PORT = get_env_variable("WEB_SERVER_PORT", 8080)
WEBHOOK_PATH = get_env_variable("WEBHOOK_PATH", "/webhook")
WEBHOOK_SECRET = get_env_variable("WEBHOOK_SECRET", "my-secret")
BASE_WEBHOOK_URL             =             get_env_variable("BASE_WEBHOOK_URL",
"https://yourdomain.com")

# Змінні для Azure OpenAI
AZURE_API_ENDPOINT = get_env_variable("AZURE_API_ENDPOINT")
AZURE_API_KEY = get_env_variable("AZURE_API_KEY")
AZURE_API_VERSION = get_env_variable("AZURE_API_VERSION")
AZURE_DEPLOYMENT_NAME = get_env_variable("AZURE_DEPLOYMENT_NAME")

# Додавайте додаткові змінні за аналогією

def load_domains(file_path):
    with open(file_path, 'r', encoding='utf-8') as file:
        return [line.strip() for line in file.readlines()]

WELCOME_MESSAGE = r"""
    Добридень\! 
    Ласкаво просимо до нашого Телеграм\-бота\.

     Бот може адаптувати Ваше резюме до вакансії\.
     Команда для виклику адаптації CV до вакансії: \/set\_vacancy
"""

NEW_WELCOME_MESSAGE = r"""
    Добрий день\! 
    Ласкаво просимо до нашого Телеграм\-бота — вашого надійного помічника\!

    Наш бот вміє:

    Допомагати адаптувати ваше резюме до вакансії\.
    Команда для виклику адаптації CV до вакансії: /set_vacancy

    Допомагає отримати зворотний зв'язок за записом вашого інтерв’ю\.
    Команда для виклику генерації зворотнього зв’язку: /feedback
"""
```

Б.3 Код для розгортання на AWS lambda

```
import os
import logging
import sys
import json
import asyncio
from aiohttp import web
from aiogram import Bot, Dispatcher, Router, types
from aiogram.client.default import DefaultBotProperties
from aiogram.enums import ParseMode
from aiogram.webhook.aiohttp_server import SimpleRequestHandler,
setup_application
from aiogram.fsm.storage.memory import MemoryStorage
from aiogram.filters import CommandStart, Command
from aiogram.fsm.context import FSMContext

from apps.write_cv.write_cv_routes import write_cv_router
from apps.feedback_insights.feedback_routes import feedback_router
from apps.config import user_data


# Перевірка середовища виконання
```

```
if os.getenv("AWS_LAMBDA_FUNCTION_NAME"):
    # from lambda_bot.routes import router
    from lambda_bot.config import BOT_TOKEN, WEB_SERVER_HOST, WEB_SERVER_PORT,
WEBHOOK_PATH, WEBHOOK_SECRET, BASE_WEBHOOK_URL, NEW_WELCOME_MESSAGE
    # Додаємо роутер у диспетчер тільки для AWS Lambda
    storage = MemoryStorage()
    bot = Bot(token=BOT_TOKEN,
default=DefaultBotProperties(parse_mode=ParseMode.HTML))
    dp = Dispatcher(storage=storage)
    # dp.include_router(router)
else:
    # from routes import router
    from config import BOT_TOKEN, WEB_SERVER_HOST, WEB_SERVER_PORT,
WEBHOOK_PATH, WEBHOOK_SECRET, BASE_WEBHOOK_URL, NEW_WELCOME_MESSAGE
    storage = MemoryStorage()
    bot = Bot(token=BOT_TOKEN,
default=DefaultBotProperties(parse_mode=ParseMode.HTML))
    dp = Dispatcher(storage=storage)
    # dp.include_router(router)

new_start_router = Router()

@new_start_router.message(CommandStart())
async def command_start_handler(message: types.Message) -> None:
    """
    Обробник команди /start.
    Надсилає нове вітальне повідомлення користувачу.
    """
    await message.answer(NEW_WELCOME_MESSAGE, parse_mode='MarkdownV2')


@new_start_router.message(Command("stop"))
async def stop_handler(message: types.Message, state: FSMContext) -> None:
    """
    Обробник команди /stop.
    Зупиняє бота та очищає стан користувача.
    """
    await message.answer("Бот зупинено. Щоб почати знову, надішліть /start")
    user_id = message.from_user.id
    user_data.pop(user_id, None) # Видаляємо дані користувача
    await state.clear()


dp.include_routers(
    new_start_router,
    write_cv_router,
    feedback_router
)

logging.getLogger().setLevel(logging.INFO)
logger = logging.getLogger()

# Отримуємо значення IS_ENABLED зі змінних середовища
IS_ENABLED = os.getenv("IS_ENABLED", "True").lower() == "true"
logger.info(f"Значення IS_ENABLED: {IS_ENABLED}") # Змінено текст логу

async def handle_event(event):
    """
    Обробник подій для AWS Lambda.
    Перевіряє, чи увімкнено бота, парсить оновлення та передає їх диспетчеру.
    """
    if not IS_ENABLED:
```

```
        logger.info("Бот вимкнено. Оновлення ігнорується.") # Змінено текст
логу
        return 'Бот вимкнено' # Змінено текст

    update = types.Update.model_validate(json.loads(event['body']))
    logger.info("Серіалізований JSON: " + str(update)) # Змінено текст логу
    await dp.feed_update(bot=bot, update=update)
    return 'OK'

def lambda_handler(event, context):
    """
    Точка входу для AWS Lambda.
    Ініціалізує цикл подій та обробляє вхідну подію.
    """
    logger.info("Отримана подія: " + str(event)) # Змінено текст логу
    loop = asyncio.get_event_loop()
    if loop.is_closed():  # Виправлено тут (коментар)
        loop = asyncio.new_event_loop()
        asyncio.set_event_loop(loop)
    result = loop.run_until_complete(handle_event(event))
    return {
        'statusCode': 200,
        'body': json.dumps({'message': result})
    }


async def on_startup(bot: Bot) -> None:
    """
    Функція, що виконується під час запуску програми.
    Встановлює вебхук для бота.
    """
    if IS_ENABLED:
        await bot.set_webhook(f"{BASE_WEBHOOK_URL}{WEBHOOK_PATH}",
secret_token=WEBHOOK_SECRET)
        print("Вебхук встановлено") # Змінено текст

def main() -> None:
    """
    Основна функція для запуску бота у веб-режимі (не Lambda).
    """
    if os.getenv("AWS_LAMBDA_FUNCTION_NAME"):
        return  # Роутер вже додано на початку для AWS Lambda (коментар)

    if IS_ENABLED:
        # Реєстрація middleware для передачі об'єкта бота в обробники
        new_start_router.message.middleware()(lambda handler, event, data:
handler(event, {**data, 'bot': bot}))
        write_cv_router.message.middleware()(lambda handler, event, data:
handler(event, {**data, 'bot': bot}))
        feedback_router.message.middleware()(lambda handler, event, data:
handler(event, {**data, 'bot': bot}))

        dp.startup.register(on_startup)
        app = web.Application()
        webhook_requests_handler = SimpleRequestHandler(dispatcher=dp, bot=bot,
secret_token=WEBHOOK_SECRET)
        webhook_requests_handler.register(app, path=WEBHOOK_PATH)
        setup_application(app, dp, bot=bot)
        web.run_app(app, host=WEB_SERVER_HOST, port=WEB_SERVER_PORT)

if __name__ == "__main__":
    logging.basicConfig(level=logging.INFO, stream=sys.stdout)
    main()
```

Б.4 Код для парсингу резюме та вакансії

```
import asyncio
import logging
import os
from bs4 import BeautifulSoup
import aiohttp
from ..utils import azure_openai

AZURE_DEPLOYMENT_NAME = os.getenv("AZURE_DEPLOYMENT_NAME")

async def fetch_html(url: str) -> str:
    try:
        async with aiohttp.ClientSession() as session:
            async with session.get(url) as response:
                response.raise_for_status()
                return await response.text()
    except aiohttp.ClientError as e:
        print(f"Error fetching the URL: {e}")
        return None

async def parse_job_listing(html):
    soup = BeautifulSoup(html, 'html.parser')

    return " ".join(soup.get_text().split('\n')).strip()

async def parsing_vacancy_details(url):
    html = await fetch_html(url)

    if html:
        job_description = await parse_job_listing(html)
        return await check_vacancy_with_openai(job_description)

async def check_vacancy_with_openai(job_description):
    MAX_CHARS = 4096
    chunks = [job_description[i:i + MAX_CHARS] for i in range(0,
len(job_description), MAX_CHARS)]

    structured_text = []

    for chunk in chunks:
        combined_prompt = [
            {
                "role": "user",
                "content": (
                    "Check if there is a job description in the
[text]text[/text]."
                    "If so, return the structured text that relates to the
vacancy (and delete the excess that does not relate to the vacancy), "
                    "if the text does not relate to the description of the
vacancy, then you need to give the answer \"not a vacancy\": "
                    f"[text]{chunk}[/text]"
                )
            },
        ]

        try:
            response = azure_openai.ChatCompletion.create(
                deployment_id=AZURE_DEPLOYMENT_NAME,
                messages=combined_prompt,
            )
```

```
            result = response.choices[0].message.content
            if not result in ["not a vacancy", "Not a vacancy"]:
                structured_text.append(result)
            #set timeout for 10 seconds
            await asyncio.sleep(15)
        except azure_openai.error.RateLimitError as e:
            logging.error(f"RateLimitError: {e}")
            return None
        except Exception as e:
            logging.error(f"OpenAIError: {e}")
            return None

    final_result = " ".join(structured_text)

    return final_result
```

Б.5 Код для адаптації резюме під вакансію

```
import json
import logging
import re
from .pdf_builder import generate_pdf
from config import AZURE_DEPLOYMENT_NAME
from ..utils import azure_openai
from ..config import user_data

async def from_string_to_dict(json_string: str) -> dict:
    json_string_cleaned = re.sub(r'(?<!\\)\n', ' ', json_string)
    json_compatible_string = json_string_cleaned.replace('```python', '')
    json_compatible_string = json_compatible_string.replace('```json', '')
    json_compatible_string = json_compatible_string.replace("```", '')
    json_compatible_string = json_compatible_string.replace('\n', '\\n')

    json_compatible_string = json_compatible_string.strip()

    try:
        result_dict = json.loads(json_compatible_string)
        return result_dict
    except json.JSONDecodeError as e:
        logging.error(f"Error parsing JSON: {e}")

async def adapt_cv_to_vacancy(user_id: int):
    """
    Function for adapting a resume to a vacancy using the Azure OpenAI API.

    :param cv_text: CV text.
    :param vacancy_text: Vacancy text.
    :return: Adapted resume and path to PDF file.
    """
    change_text = user_data[user_id].get('change_text', '')
    vacancy_text = user_data[user_id].get('vacancy_text', '')
    if change_text:
        cv_text = user_data[user_id].get('adapted_cv', '')
    else:
        cv_text = user_data[user_id].get('cv_text', '')

    if change_text:
        amendments = f"also please make changes using
[amendments]amendments[/amendments]:\n[amendments]{change_text}[/amendments]"
    else:
```

```
        amendments = ''
    try:
        language = user_data[user_id].get('language_for_cv', 'english')
        messages = [
            {
                "role": "system",
                "content": (
                    "You are an expert in editing resumes according to job
descriptions. "
                    "If any information does not fit into the provided
categories, create new headings to organize it appropriately. "
                    "Return only the edited resume in the form of a Python
dictionary, without explanations or additional comments."
                )
            },
            {
                "role": "user",
                "content": (
                    "Please help me adapt my [resume]resume[/resume] to
[vacancy]vacancy[/vacancy] resume must be in [language]language[/language], "
                    f"all text must match the selected language, including
section titles, {amendments}. "
                    "so that the output data of the new resume is in the
following format:\n"
                    '{"full_name": "Jon Dou", "position": "Name of the
position", "contact_information": ["email@example.com", "+123456789", "..."], '
                    '{"sections": [{"section_name": "summary", "content":
"text"}, {"section_name": "skills", "content": "Skill 1, Skill 2"},
{"section_name": "languages", "content": "English, Spanish"}, '
                    '{"section_name": "work experience", "content": ([{"title":
"Software Engineer, "content": ["text", "text", "text"]}] or "text")},
{"section_name": "education", "content": ([{"title": "College Name", "content":
"text"}] or "text")}, '
                    '{"section_name": "other name of section", "content":
"text"}]}.\n'
                    f"[language]{language}[/language]\n"
                    f"[resume]{cv_text}[/resume]\n"
                    f"[vacancy]{vacancy_text}[/vacancy]"
                )
            }
        ]

        response = azure_openai.ChatCompletion.create(
            deployment_id=AZURE_DEPLOYMENT_NAME,
            messages=messages
        )
        adapted_cv_text = response.choices[0].message.content
        dict_info = await from_string_to_dict(adapted_cv_text)
        user_data[user_id]['adapted_cv'] = adapted_cv_text

        output_filename = f'adapted_cv_{user_id}.pdf'
        name = dict_info.pop("full_name")
        contact_info = dict_info.pop("contact_information")
        position = dict_info.pop("position", "")
        sections = dict_info.pop("sections")

        try:
            await generate_pdf(name, position, contact_info, sections,
output_filename)
            return output_filename
        except Exception as e:
            logging.error(f"Error during PDF generation: {e}")
            if not user_data[user_id].get('error_creating_pdf', None):
```

```
                user_data[user_id]['error_creating_pdf'] = True
                return await adapt_cv_to_vacancy(cv_text, vacancy_text,
user_id)
            else:
                return None

    except Exception as e:
        logging.error(f"Error during OpenAI call: {e}")
        return None
```

Б.6 Код для створення PDF-файлу

```
import logging
from fpdf import FPDF
import os

FONT_BOLD = 'NotoSans-Bold'
FONT_REGULAR = 'NotoSans'
SIZE_TITLE = 10
SIZE_CONTENT = 8
LINE_HEIGHT_TITLE = 6
LINE_HEIGHT_CONTENT = 5

class PDF(FPDF):
    def __init__(self):
        super().__init__()
        self.set_auto_page_break(auto=True, margin=15)
        self.add_font(FONT_REGULAR, '', 'apps/fonts/NotoSans.ttf', uni=True)
        self.add_font(FONT_BOLD, 'B', 'apps/fonts/NotoSans_Bold.ttf', uni=True)

    def add_client_info(self, name, position, contact_info):
        self.set_font(FONT_BOLD, 'B', 22)
        self.cell(0, 10, name, ln=True, align='L')
        y_name = self.get_y() - 10

        # Calculate available width for the position text
        position_max_width = 120

        # Display position, accounting for text wrapping
        self.set_font(FONT_REGULAR, '', 16)
        if self.get_string_width(position) <= position_max_width:
            self.cell(0, 6, position, ln=True, align='L')
        else:
            self.multi_cell(0, 6, position, align='L')

        y_position = self.get_y()

        # Display contact info to the right, aligned with the name
        self.set_xy(120, y_name)
        self.set_font(FONT_REGULAR, '', 8)

        for contact in contact_info:
            self.cell(0, 6, contact, ln=True, align='L')
            y_name += 6
            self.set_xy(120, y_name)

        self.ln(10)

    def add_section(self, title, content):
        self.set_font(FONT_BOLD, 'B', SIZE_TITLE)
        if title.lower() in ['summary', 'experience', 'education',
'work_experience', 'work experience']:
```

```
            self.set_font(FONT_BOLD, 'B', SIZE_TITLE + 1)
            self.cell(0, LINE_HEIGHT_TITLE, title.upper().replace('_', ' '),
ln=True)
        else:
            self.cell(0, LINE_HEIGHT_TITLE, title.title().replace('_', ' '),
ln=True)
        self.ln(1)
        self.set_font(FONT_REGULAR, '', SIZE_CONTENT)

        if isinstance(content, list):
            for item in content:
                if isinstance(item, dict):
                    self.set_font(FONT_BOLD, 'B', SIZE_CONTENT)
                    self.cell(0, LINE_HEIGHT_TITLE, item['title'], ln=True)

                    self.set_font(FONT_REGULAR, '', SIZE_CONTENT)
                    if isinstance(item['content'], list):
                        self.multi_cell(0, LINE_HEIGHT_CONTENT,
'\n'.join(item['content']))
                    else:
                        self.multi_cell(0, LINE_HEIGHT_CONTENT,
f"{item['content']}")
                    self.ln(2)
                else:
                    if isinstance(item, list):
                        self.multi_cell(0, LINE_HEIGHT_CONTENT,
'\n'.join(item))
                        self.ln(2)
                    else:
                        self.multi_cell(0, LINE_HEIGHT_CONTENT, item)
                        self.ln(2)
        else:
            self.multi_cell(0, LINE_HEIGHT_CONTENT, content)
            self.ln(2)
        self.ln(2)

    def add_multiple_sections(self, sections):
        for section in sections:
            self.add_section(section['section_name'], section['content'])

async def generate_pdf(name, position, contact_info, sections,
output_filename):
    pdf = PDF()
    try:
        output_dir = os.path.dirname('cv_files/')
        if not os.path.exists(output_dir):
            os.makedirs(output_dir)

        pdf.add_page()
        pdf.add_client_info(name, position, contact_info)
        pdf.add_multiple_sections(sections)

        pdf.output(os.path.join(output_dir, output_filename))
        print(f"PDF generated successfully: {output_filename}")
    except Exception as e:
        logging.error(f"Error generating PDF: {e}")
```

Б.7 Код для форматування PDF-файлу

```
from fpdf import FPDF
import os
```

```
class PDF(FPDF):
    def header(self):
        if self.page_no() == 1:
            self.set_font('notosans', 'B', 12)
            self.cell(0, 10, 'CV and Interview Analysis', ln=True, align='C')

    def chapter_title(self, title):
        self.set_font('notosans', 'B', 14)
        self.cell(0, 10, title, ln=True)

    def chapter_body(self, body):
        self.set_font('notosans', '', 12)
        self.multi_cell(0, 10, body)
        self.ln()

    def add_section(self, title, content):
        self.chapter_title(title)
        self.chapter_body(content)

def format_text(text):
    text = text.replace('**', '').replace('###', '').replace('####',
'').replace('#', '')
    return text

def create_pdf(content, file_name):
    pdf = PDF()

    font_path = os.path.join('apps', 'fonts')
    try:
        pdf.add_font('notosans', '', os.path.join(font_path, 'NotoSans.ttf'),
uni=True)
        pdf.add_font('notosans', 'B', os.path.join(font_path,
'NotoSans_Bold.ttf'), uni=True)
    except Exception as e:
        print(f"Font loading error: {e}")

    pdf.add_page()

    for section_title, section_body in content.items():
        formatted_title = format_text(section_title)
        formatted_body = format_text(section_body)
        pdf.add_section(formatted_title, formatted_body)

    pdf.output(file_name)
```

Б.8 Код для обробки запитів

```
import os
import asyncio
import logging

from aiogram import Router, Bot
from aiogram.filters import Command
from aiogram.types import Message, FSInputFile, CallbackQuery,
InlineKeyboardButton, InlineKeyboardMarkup
from aiogram.fsm.context import FSMContext
from aiogram.fsm.state import State, StatesGroup

from ..utils import download_file, extract_text_and_replace_links
from .adapt_cv import adapt_cv_to_vacancy
from .parsing import check_vacancy_with_openai, parsing_vacancy_details
```

```
from ..config import user_data, url_pattern

logging.basicConfig(level=logging.INFO)

write_cv_router = Router()

class VacancyStates(StatesGroup):
    waiting_for_link_of_vacancy = State()
    waiting_for_text_of_vacancy = State()
    waiting_for_cv = State()
    waiting_for_change_text = State()
    waiting_for_final_step = State()
    waiting_language = State()

@write_cv_router.message(Command(commands=['set_vacancy']))
async def receive_send_cv(message: Message, state: FSMContext) -> None:
    await state.set_state(VacancyStates.waiting_for_cv)
    user_id = message.from_user.id
    user_data[user_id] = {}
    await message.answer('Будь ласка, надайте CV в форматі PDF.')

@write_cv_router.message(VacancyStates.waiting_for_cv)
async def receive_cv(message: Message, state: FSMContext, bot: Bot) -> None:
    try:
        file_id = message.document.file_id
        user_id = message.from_user.id
        file_name = f"{user_id}_cv"
        save_path = await download_file(file_id, file_name, bot)
        cv_text = await extract_text_and_replace_links(save_path)
        user_data[user_id]['cv_text'] = cv_text
        await message.answer("Дякую! Ваше CV отримано.")
        await state.clear()
        await receive_vacancy_type_inline(message)
    except Exception as e:
        logging.error(f'Error reading the CV: {e}')
        await message.answer("Не вдалося прочитати файл. Будь ласка,
переконайтеся, що це правильний PDF документ.")
    finally:
        if os.path.exists(save_path):
            os.remove(save_path)

@write_cv_router.message(Command(commands=['set_vacancy_type']))
async def receive_vacancy_type_inline(message: Message) -> None:
    buttons = [
        [InlineKeyboardButton(text='Надати посилання на вакансію',
callback_data='vacancy_link')],
        [InlineKeyboardButton(text='Надати текстовий опис вакансії',
callback_data='vacancy_text')],
    ]
    keyboard_markup = InlineKeyboardMarkup(inline_keyboard=buttons)
    await message.answer("Будь ласка, оберіть в меню, як бажаєте надати вимоги
до вакансії:", reply_markup=keyboard_markup)

@write_cv_router.callback_query(lambda callback_query: callback_query.data in
['vacancy_link', 'vacancy_text'])
async def set_vacancy_type(callback_query: CallbackQuery, state: FSMContext) ->
None:
    if callback_query.data == 'vacancy_link':
        await callback_query.message.edit_text("Ви обрали надати посилання на
вакансію.")
        await callback_query.message.answer("Будь ласка, введіть посилання на
вакансію.")
        await state.set_state(VacancyStates.waiting_for_link_of_vacancy)
```

```
    elif callback_query.data == 'vacancy_text':
        await callback_query.message.edit_text("Ви обрали надати текстовий опис
вакансії.")
        await callback_query.message.answer("Будь ласка, введіть текстовий опис
вакансії\n(з обмеженням в 4096 символів, має бути одне повідомлення).")
        await state.set_state(VacancyStates.waiting_for_text_of_vacancy)

@write_cv_router.message(VacancyStates.waiting_for_link_of_vacancy)
async def receive_vacancy_link(message: Message, state: FSMContext, bot: Bot) -
> None:
    vacancy_link = message.text.strip()
    user_id = message.from_user.id
    if url_pattern.match(vacancy_link):
        await message.answer('Починаю збирати інформацію про вакансію. Це може
заиˇняти якийсь час.')

        try:
            vacancy_text = await parsing_vacancy_details(vacancy_link)
            if not vacancy_text:
                raise Exception('Не вдалося виділити текст вакансії.')
            user_data[user_id]['vacancy_text'] = vacancy_text
            await message.answer('Дякую що зачекали! Текст вакансії отримано.')
            await message.answer("Будь ласка, вкажіть мову якою повинно бути
резюме.")
            await state.set_state(VacancyStates.waiting_language)
        except Exception as e:
            logging.error(f'Error parsing the URL: {e}')
            await message.answer('Не вдалося отримати текст вакансії.')
            await message.answer('Якщо помилка повториться, будь ласка,
спробуйте обрати "Надати текстовий опис вакансії" /set_vacancy_type')
            await message.answer("Будь ласка, перевірте правильність посилання
і відправте повторно.")
    else:
        await message.answer("Будь ласка, введіть дійсне посилання на
вакансію.")

@write_cv_router.message(VacancyStates.waiting_for_text_of_vacancy)
async def receive_vacancy_text(message: Message, state: FSMContext) -> None:
    vacancy_text = await check_vacancy_with_openai(message.text.strip())
    user_id = message.from_user.id

    if vacancy_text:
        user_data[user_id]['vacancy_text'] = vacancy_text
        await message.answer("Дякую! Текст вакансії отримано.")
        await message.answer("Будь ласка, вкажіть мову якою повинно бути
резюме.")
        await state.set_state(VacancyStates.waiting_language)
    else:
        await message.answer("Не вдалося виділити текст вакансії. Перевірте
правильність тексту. Спробуйте знову надати текст.")

@write_cv_router.message(VacancyStates.waiting_language)
async def receive_language(message: Message, state: FSMContext, bot: Bot) ->
None:
    user_id = message.from_user.id
    if message.text:
        user_data[user_id]['language_for_cv'] = message.text
    else:
        return await message.answer("Будь ласка, надайте в повідомленні мову
якою повинно бути резюме.")
    await message.answer("Дякую, починаю адаптацію. Це може заиˇняти якийсь
час.")
    await get_adapted_cv_text_and_create_pdf(message, state, bot, user_id)
```

```
async def get_adapted_cv_text_and_create_pdf(message: Message, state:
FSMContext, bot: Bot, user_id, change_text='') -> None:
    output_filename = None
    try:
        output_filename = await adapt_cv_to_vacancy(user_id)
        if output_filename is None:
            raise Exception("Помилка адаптації CV, будь ласка, спробуйте ще
раз.")

        if not change_text:
            await message.answer("Ваше CV було успішно адаптовано під
вакансію.")
        else:
            await message.answer("Ваше CV було успішно оновлено відповідно до
ваших змін.")

        success = await send_with_retry(bot, user_id, output_filename)
        if not success:
            await message.answer("Не вдалося відправити PDF файл. Спробуйте
пізніше.")
        await state.clear()
        await set_final_step(message)
    except Exception as e:
        logging.error(f'Error reading CV: {e}')
        await message.answer("Не вдалося згенерувати PDF файл. Будь ласка,
спробуйте ще раз.")
        await state.clear()
        await receive_vacancy_type_inline(message)
    finally:
        if os.path.exists(f'cv_files/{output_filename}'):
            os.remove(f'cv_files/{output_filename}')

async def send_with_retry(bot: Bot, user_id: int, output_filename: str,
retry_count: int = 5, delay: int = 5) -> bool:
    for attempt in range(retry_count):
        try:
            file_path = os.path.join('cv_files/', output_filename)
            if not os.path.exists(file_path):
                logging.error(f"File {file_path} does not exist.")
                return False

            input_file = FSInputFile(file_path)
            await bot.send_document(chat_id=user_id, document=input_file,
caption="Ваше адаптоване CV")
            logging.info("Document sent successfully.")
            return True
        except Exception as e:
            logging.error(f"Attempt {attempt + 1}: Error sending document -
{e}")
            if attempt < retry_count - 1:
                await asyncio.sleep(delay)

    logging.error("Failed to send document after several attempts.")
    return False

async def set_final_step(message: Message) -> None:
    buttons = [
        [InlineKeyboardButton(text='Внести зміни в нове резюме',
callback_data='edit_cv')],
        [InlineKeyboardButton(text='Надати нове посилання на вакансію',
callback_data='vacancy_link')],
```

```
        [InlineKeyboardButton(text='Надати новий текст вакансії',
callback_data='vacancy_text')],
        [InlineKeyboardButton(text='Завершити', callback_data='finish')]
    ]
    keyboard_markup = InlineKeyboardMarkup(inline_keyboard=buttons)
    await message.answer("Чи хочете ви продовжити?",
reply_markup=keyboard_markup)

@write_cv_router.callback_query(lambda callback_query: callback_query.data ==
'edit_cv')
async def edit_cv(callback_query: CallbackQuery, state: FSMContext) -> None:
    await callback_query.message.edit_text("Ви обрали внести зміни в нове
резюме.")
    await callback_query.message.answer("Будь ласка, напишіть пункт резюме та
свої пропозиції що до покращення.")
    await state.set_state(VacancyStates.waiting_for_change_text)

@write_cv_router.message(VacancyStates.waiting_for_change_text)
async def receive_change_text(message: Message, state: FSMContext, bot: Bot) ->
None:
    user_id = message.from_user.id
    if message.text:
        change_text = message.text
        user_data[user_id]['change_text'] = change_text
        await get_adapted_cv_text_and_create_pdf(message, state, bot, user_id,
change_text)

@write_cv_router.callback_query(lambda callback_query: callback_query.data ==
'finish')
async def finish(callback_query: CallbackQuery) -> None:
    user_id = callback_query.from_user.id
    user_data.pop(user_id, None)
    await callback_query.message.edit_text("Дякую! Ви завершили процес
створення CV.")
```

ДОДАТОК В

(обов’язковий)

ПРЕЗЕНТАЦІЙНІ МАТЕРІАЛИ

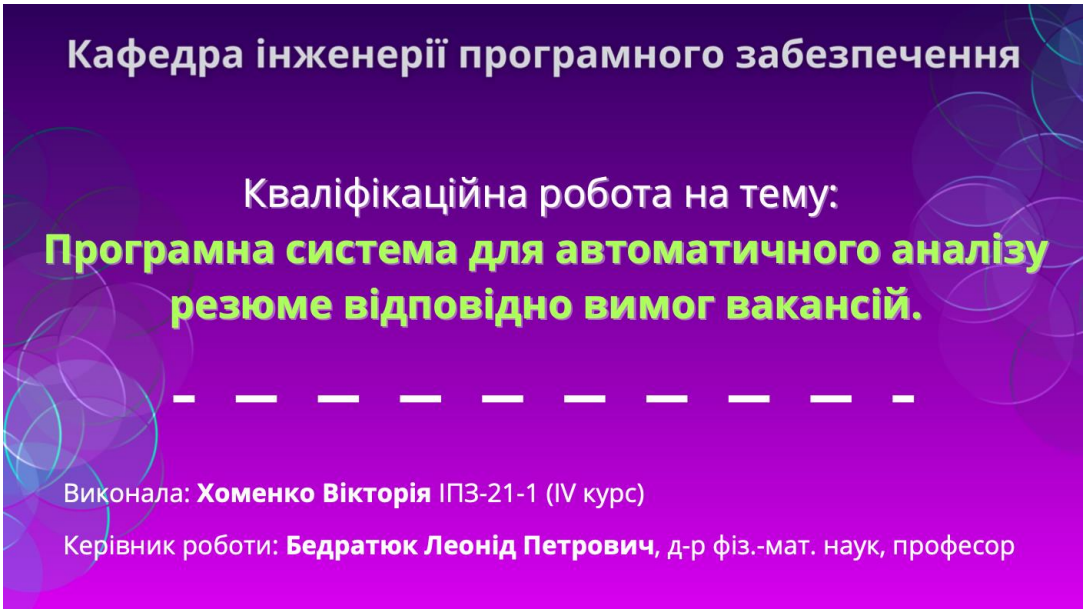


Рисунок В.1 – Слайд 1

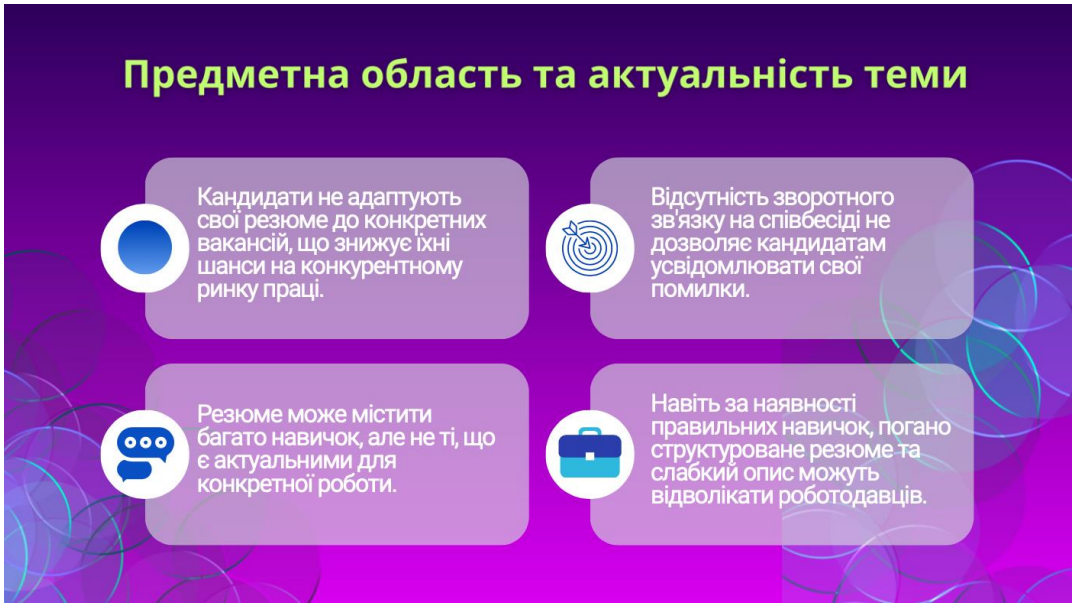


Рисунок В.2 – Слайд 2

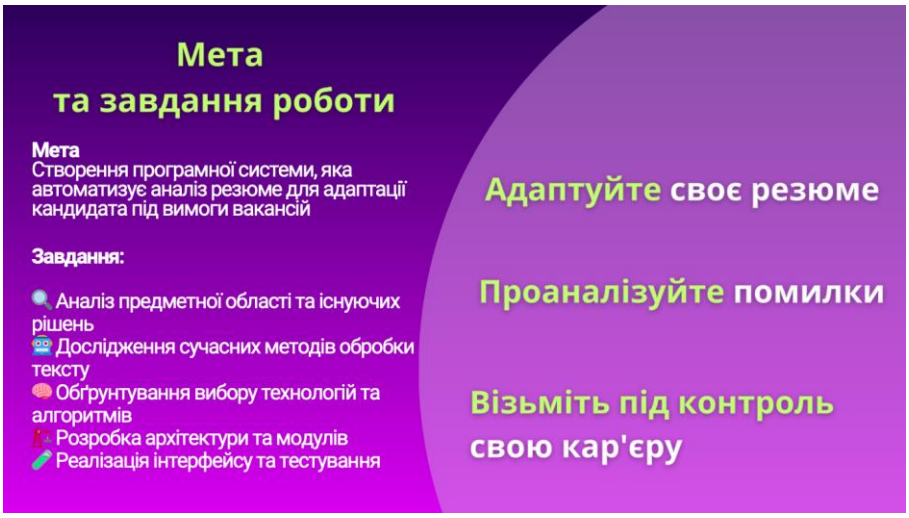


Рисунок В.3 – Слайд 3

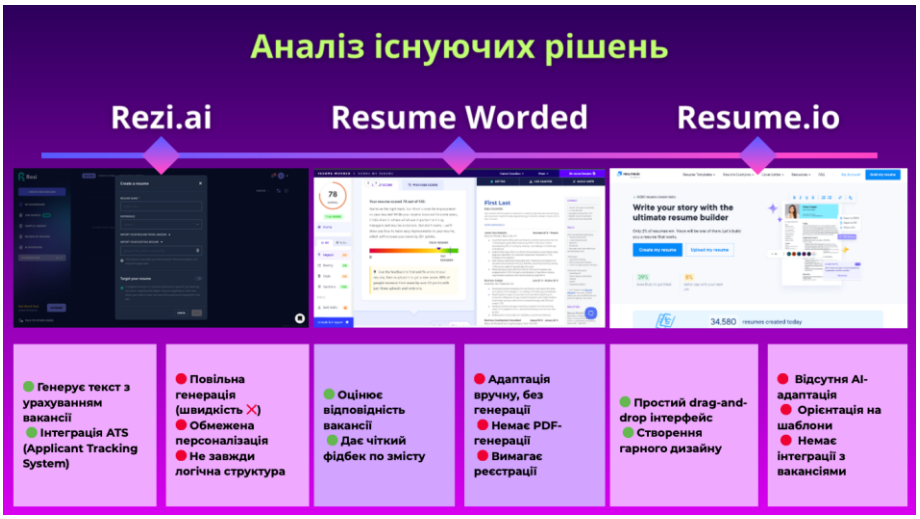


Рисунок В.4 – Слайд 4

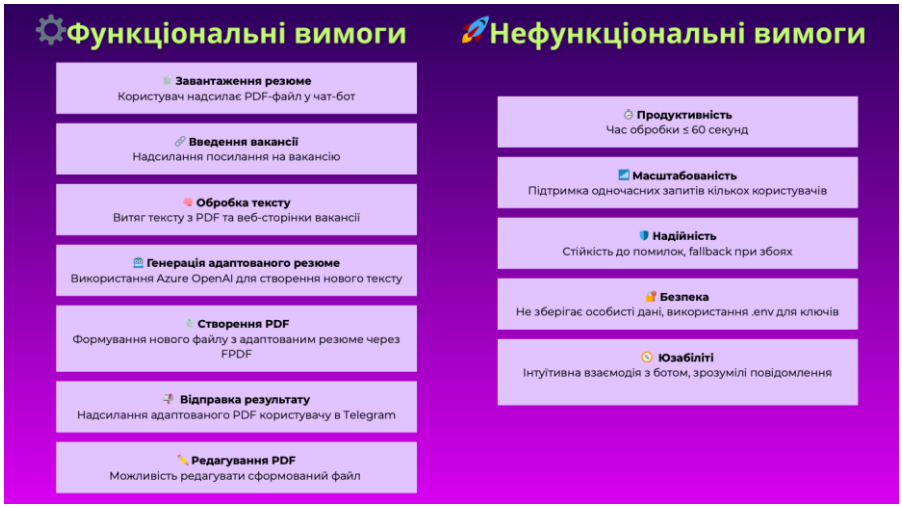


Рисунок В.5 – Слайд 5

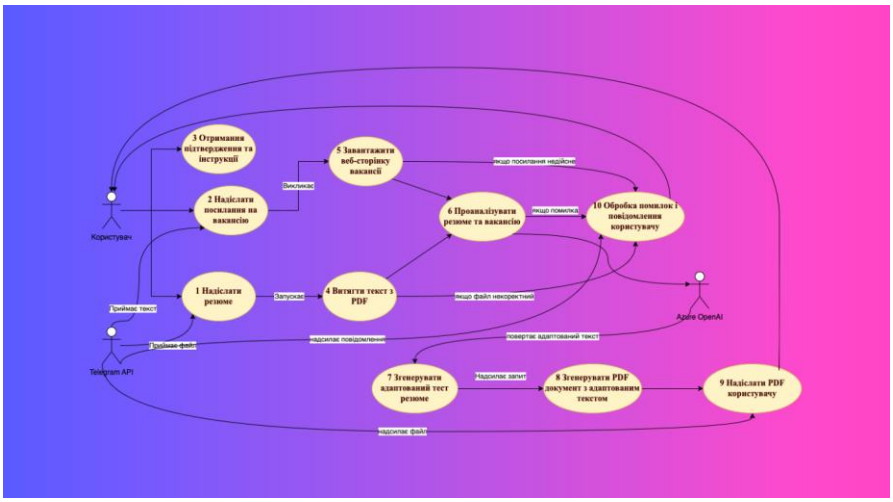


Рисунок В.6 – Слайд 6

Рисунок В.7 – Слайд 7

Рисунок В.8 – Слайд 8

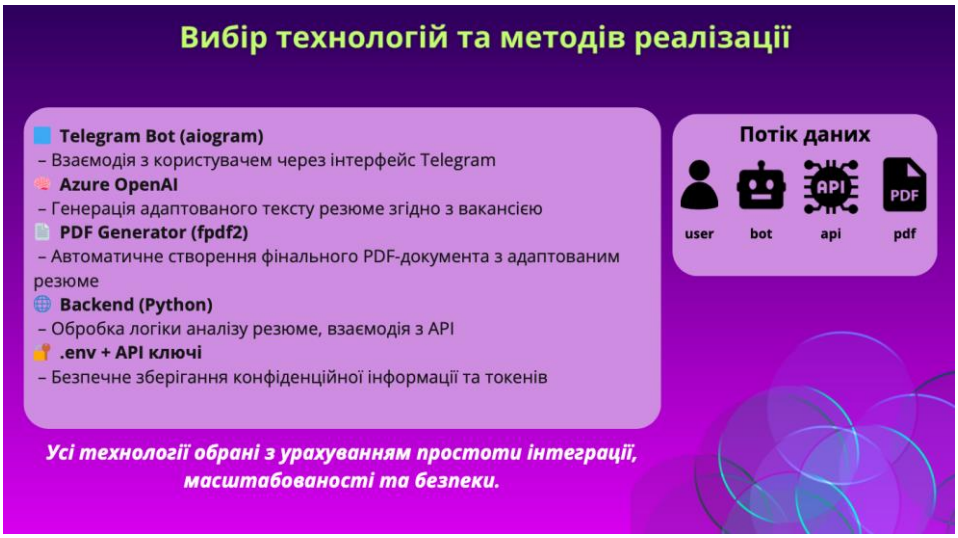


Рисунок В.9 – Слайд 9

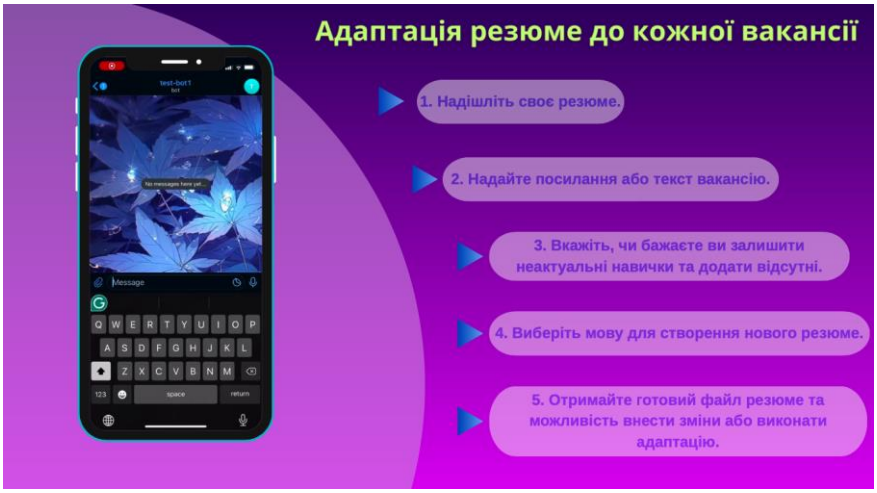


Рисунок В.10 – Слайд 10

Рисунок В.11 – Слайд 11

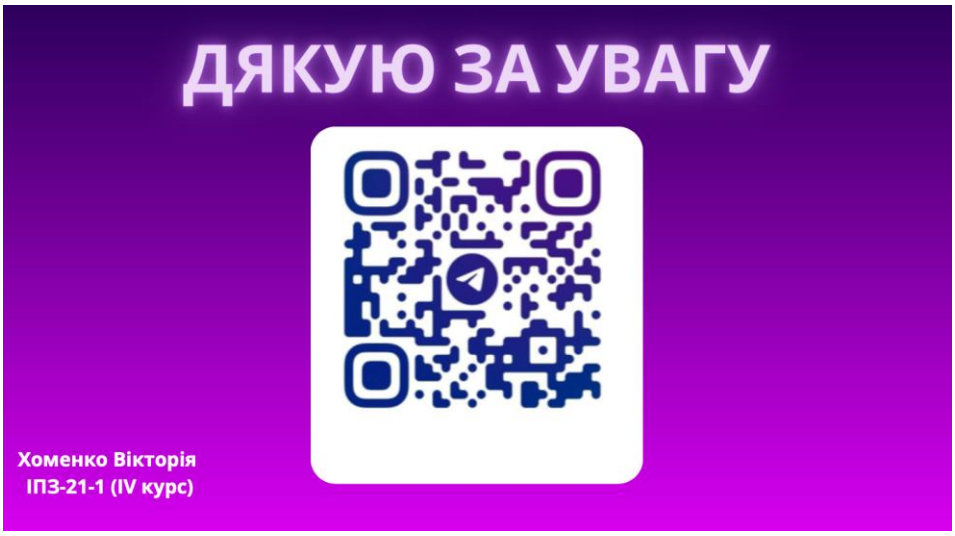
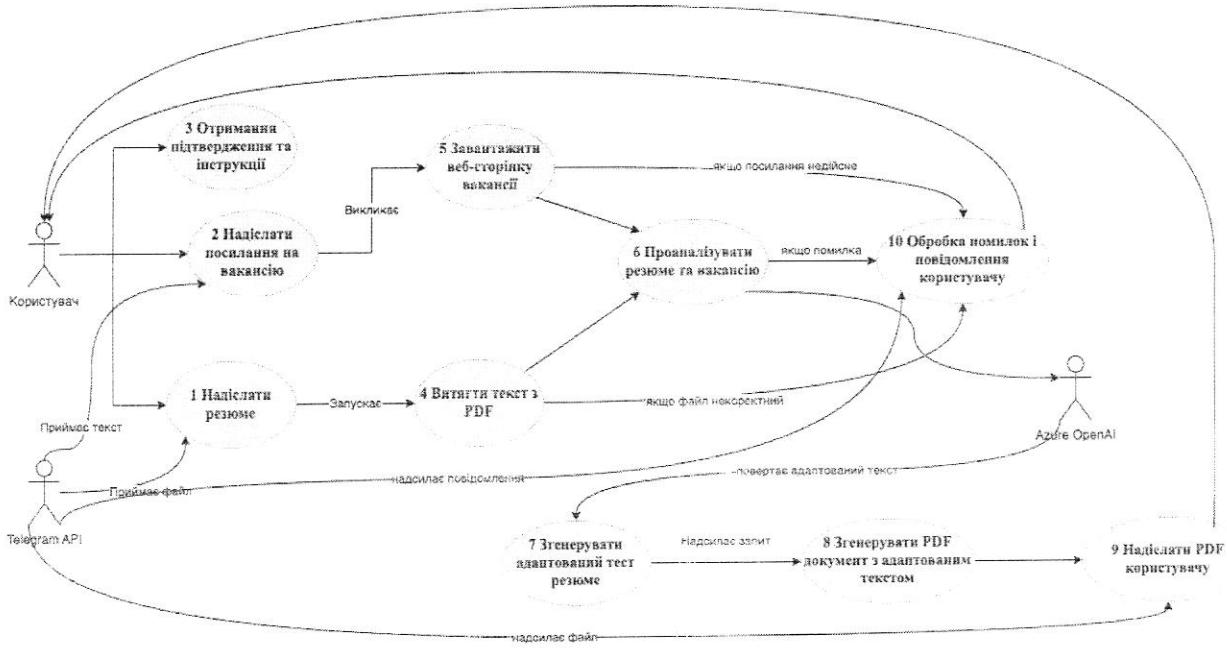
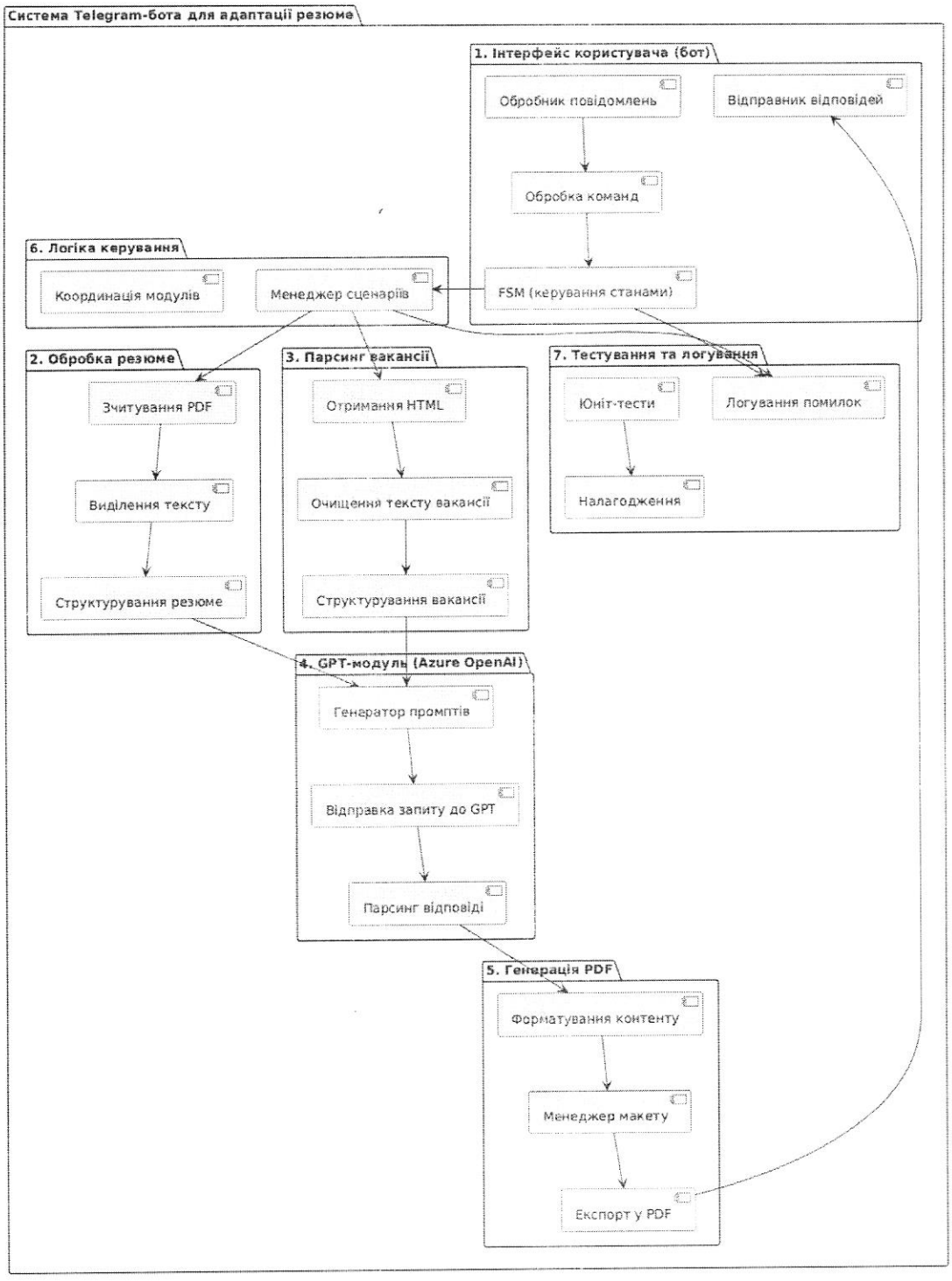


Рисунок В.12 – Слайд 12

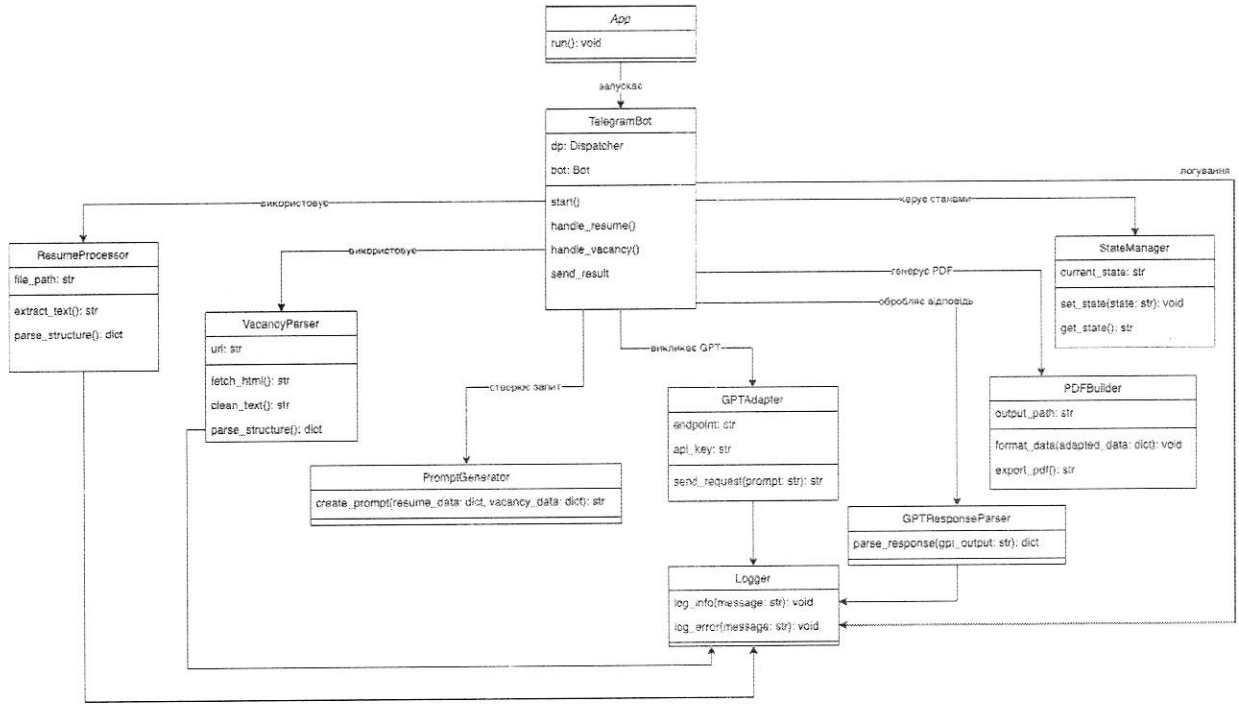
ГРАФІЧНА ЧАСТИНА



					КвРІПЗ.2101092.01.18.Е8	
					Програмна система для автоматизованого аналізу резюме відповідно до вимог вакансій	Літ. / Маса. / Масштаб
Змн.	Арк.	№ докум.	Підпис	Дата		
Виконала		Хоменко В. В.		1.06.25		
Керівник		Бедратюк Л. П		1.06.25	Діаграма варіантів використання	Аркуш 1 / Аркушів 3
Перевірив						
Н. Контр.		Праворська Н. І.		10.05.25		ХНУ, ІПЗ-21-1
Зав. каф.		Бедратюк Л. П.		10.06.25		



					КвРПЗ.2101092.01.18.Е8	
Змн.	Арк.	№ докум.	Підпис	Дата	Програмна система для автоматизованого аналізу резюме відповідно до вимог вакансій	Літ. / Маса. / Масштаб
Виконала		Хоменко В. В.		04.06		
Керівник		Бедратюк Л. П		05.06	Діаграма зв`язків модулів	Аркуш 2 / Аркушів 3
Перевірив						
Н. Контр.		Праворська Н. І.		4.05.25		ХНУ, ІПЗ-21-1
Зав. каф.		Бедратюк Л. П.		05.06		



					КвРІПЗ.2101092.01.18.Е8	
					Програмна система для автоматизованого аналізу резюме відповідно до вимог вакансій Діаграма класів	Літ. / Маса. / Масштаб
Змн.	Арк.	№ докум.	Підпис	Дата		
Виконала		Хоменко В. В.		04.06		Аркуш 3 / Аркушів 3
Керівник		Бедратюк Л. П		05.06		
Перевірив						ХНУ, ІПЗ-21-1
Н. Контр.		Праворська Н. І.		05.06.25		
Зав. каф.		Бедратюк Л. П.		05.06		

СУПРОВІДНІ ДОКУМЕНТИ

Завідувачу кафедри інженерії програмного забезпечення проф. Леоніду БЕДРАТЮКУ
здобувача вищої освіти
Хоменко Вікторії Вадимівни
факультет ІТ, IV курс, група ІПЗ-21-1

ЗАЯВА

З правилами чинного Положення про систему забезпечення академічної доброчесності в Хмельницькому національному університеті, згідно з яким виявлення академічного плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту і застосування заходів дисциплінарної та академічної відповідальності, ознайомлений. Про використання програмно-технічних засобів для перевірки кваліфікаційних робіт здобувачів вищої освіти на наявність академічного плагіату оповіщений та надаю свою згоду на обробку й збереження університетом моєї роботи в інституційному репозитарії Хмельницького національного університету.

Також надаю університету право на передачу моєї роботи для обробки та збереження в базах даних програмно-обчислювального комплексу StrikePlagiarism та/або програмно-технічного засобу AntiPlagiarism і використання роботи для виявлення академічного плагіату в інших роботах, які перевіряються програмно-технічними засобами та користувачами, що мають доступ до цих програмно-технічних засобів, виключно в обмежених цілях для виявлення текстових збігів у роботах.

Робота надається для перевірки в електронному варіанті. Електронна версія моєї роботи збігається (ідентична) з друкованою.

26.05.2025
дата

підпис

Протокол аналізу звіту подібності науковим керівником

Заявляю, що я ознайомився (-лась) з Повним звітом подібності, який був згенерований Системою виявлення і запобігання плагіату щодо роботи:

Автор: Хоменко Вікторія

Співавтор:

Назва: БКР_Програмна система для автоматизаціїаналізу резюме відповідно до вимог вакансії
Науковий керівник:
Підрозділ: Кафедра інженерії програмного забезпечення

Коефіцієнт подібності 1:2.3%

Коефіцієнт подібності 2:1%

Мікропробіли: 0

Заміна букв: 0

Інтервали: 0

Білі знаки: 0

Дата створення звіту: 2025-05-28 19:45:54.0

Після аналізу Звіту подібності констатую наступне:

☐ **Запозичення, виявлені в роботі є законними і не є плагіатом. Рівень подібності не перевищує допустимої межі. Таким чином робота незалежна і приймається.**

☐ **Запозичення не є плагіатом, але перевищено граничне значення рівня подібностей. Таким чином робота повертається на доопрацювання.**

☐ **Виявлено запозичення і плагіат або навмисні текстові спотворення (маніпуляції), як передбачувані спроби укриття плагіату, які роблять роботу невідповідною вимогам законодавства (Ст. 32. ЗУ Про вищу освіту, пункт 3.1, Ст. 42. ЗУ Про освіту) та вимог НАЗЯВО (Критерій 5), а також кодексу етики і процедурам. Таким чином робота не приймається.**

Обгрунтування:

Дата 28.05.25

експерт

Anti-Plagiarism (UA) v-15.281 Educational

The maximum coincidence with one document 2.0%

Dictionaries check: en_US, ru_RU, ua_UA. **Errors in the documents: 6%**

	Document		Sum coincidence on the DB	
ID: 242384 Title: БКР_Програмна система для автоматизації аналізу резюме відповідно до вимог вакансії Added in a DB: 2025-05-29 Authors: Хоменко Вікторі Heads: Бедратюк Леонід Петрович, д-р фіз.-мат. наук, професор Consultants: Opponents:	Symbols	Lexemes	Symbols	Lexemes
	129881	1902	3494 (3%)	39 (2%)

Plagiarism sources

ID	Description	Plagiarism presence in the document	
		Symbols	Lexemes

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

освітнього ступеня «Бакалавр»

Дипломник Хоменко Вікторія Вадимівна
Тема Програмна система для автоматизації аналізу резюме відповідно до вимог вакансії
Спеціальність 121 – Інженерія програмного забезпечення

Обсяг кваліфікаційної роботи:

Кількість листів креслень 3; кількість сторінок записки 83.
1.Короткий зміст пояснювальної записки та прийнятих рішень У кваліфікаційній роботі розглянуто створення боту для адаптації резюме під вакансію. Проведено аналіз предметної області, сформульовано вимоги, обґрунтовано архітектурні та технологічні рішення. Проведено тестування, що підтвердило коректну роботу розробленого програмного забезпечення.

2. Висновок про відповідність роботи поставленому завданню Кваліфікаційна робота виконана відповідно до поставленого завдання та з дотриманням всіх вимог.

3. Характеристика виконання кожного розділу роботи, ступінь використання останніх досягнень науки і техніки та передових методів роботи У межах цієї роботи було обґрунтовано доцільність створення програмного засобу для автоматизованого аналізу та адаптації резюме відповідно до конкретних вимог вакансії. Система реалізована у вигляді Telegram-бота з використанням сучасних інструментів, зокрема Python, бібліотеки aiogram для обробки подій чату, Azure OpenAI для аналізу та генерації текстів, а також FPDF для формування вихідного PDF-документа. На початковому етапі було проведено аналіз предметної області, визначено актуальність задачі автоматизації процесу адаптації резюме та сформульовано функціональні вимоги до системи. У подальшому виконано проєктування архітектури рішення, яка передбачає послідовну обробку введених користувачем даних – резюме у форматі PDF та посилання на вакансію. Було реалізовано механізм парсингу вакансії, витягнення ключових характеристик, побудову промптів для взаємодії з мовною моделлю Azure OpenAI, а також адаптацію змісту резюме відповідно до отриманого опису вакансії. Всі проміжні результати інтегруються у фінальну PDF-версію адаптованого резюме, яка автоматично надсилається користувачу. Особливу увагу приділено логіці обробки вхідних помилок, стабільності роботи окремих модулів, а також оптимізації як швидкодії, так і точності генерації адаптованого змісту. Під час реалізації використано найновіші підходи в галузі обробки природної мови, API-запитів та створення чат-ботів. Завершальний етап включав тестування системи на прикладах реальних вакансій і резюме, під час якого було перевірено відповідність реалізованої функціональності початковим вимогам, стабільність взаємодії з OpenAI API та зручність користування ботом. Отримані результати підтверджують ефективність створеного рішення, його актуальність для ринку праці та потенціал до подальшого розвитку.

4. Позитивні сторони роботи Проєкт демонструє сучасний підхід до автоматизації процесу адаптації резюме до вимог конкретної вакансії за допомогою інтеграції Telegram-бота з мовною моделлю Azure OpenAI. Архітектура рішення забезпечує зручну взаємодію з користувачем, стабільну обробку вхідних даних та якісне формування результату. Візуальне й функціональне оформлення системи відповідає сучасним вимогам до інтерфейсів чат-ботів. У процесі реалізації застосовано передові інструменти Python-розробки, включаючи aiogram та FPDF, що дозволило досягти високого рівня автоматизації, точності аналізу та відповідності адаптованого резюме заявленим вимогам вакансій.

5. Негативні сторони роботи Обмеженість функціоналу полягає в тому, що наразі реалізовано лише базову логіку обробки резюме та вакансії без додаткової персоналізації чи розширеної аналітики. Система не зберігає історію запитів користувача та не передбачає глибокого налаштування параметрів адаптації. Взаємодія з користувачем обмежується текстовим інтерфейсом Telegram-бота без розгалужених сценаріїв або інтеграції з зовнішніми базами даних. Крім того, точність аналізу значною мірою залежить від якості вхідного PDF-документа, а візуальне оформлення результатів у PDF-файлі поки не підтримує гнучке налаштування стилів або шаблонів.

6. Оцінка графічного оформлення та пояснювальної записки Графічне оформлення відповідає тематиці програмного проєкту та включає ключові діаграми, що відображають архітектуру системи, логіку послідовної обробки даних, структуру модулів, взаємодію об'єктів і потоки інформації. Усі візуальні матеріали розроблено з урахуванням сучасних підходів до документування програмного забезпечення. Пояснювальна записка оформлена відповідно до встановлених вимог, має чітку та логічну структуру, послідовно розкриває всі етапи розробки – від постановки задачі до аналізу отриманих результатів.

7. Відгук про кваліфікаційну роботу в цілому Кваліфікаційна робота є завершеною, цілісною та технічно грамотною. У пояснювальній записці чітко прослідковується логіка побудови рішення – від аналізу проблеми до її практичної реалізації. Застосовані сучасні технології та архітектурні підходи забезпечили створення хорошого продукту. Робота добре структурована та наповнена ілюстративним матеріалом.

8. Інші зауваження

9. Оцінка кваліфікаційної роботи Кваліфікаційна робота виконана, відповідає поставленій задачі.

РЕЦЕНЗЕНТ Лисенко Сергій Миколайович, доктор технічних наук, професор, заступник декана факультету інформаційних технологій з наукової та міжнародної роботи.

" 4 " червня 2025 р. _____
(підпис)

SemanticAI for Education

РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

першого освітнього рівня «Бакалавр»

Студента: Хоменко Вікторії Вадимівни
Група: ІПЗ-21-1

Тема: *«Програмна система для автоматизації аналізу резюме відповідно до вимог вакансії»*

Спеціальність: 121 – Інженерія програмного забезпечення

Короткий зміст пояснювальної записки

У вступі обґрунтовано актуальність автоматизації аналізу резюме для підвищення ефективності та точності процесу працевлаштування, зменшення часу та упередженості традиційного ручного відбору. Метою роботи є створення Telegram-бота, який автоматично адаптує резюме кандидата під конкретну вакансію, використовуючи технології штучного інтелекту, машинного навчання та обробки природної мови, зокрема інтеграцію з Azure OpenAI. Завдання включають дослідження предметної області, аналіз існуючих рішень, проєктування та реалізацію програмного забезпечення, тестування системи та оцінку її ефективності. Практична значущість полягає у спрощенні та прискоренні процесу підбору персоналу, підвищенні якості адаптації резюме та зменшенні часу на ручну обробку заявок.

Відповідність отриманих результатів роботи поставленим завданням

Завдання, сформульовані у вступі, включають створення Telegram-бота для автоматизованої адаптації резюме, використання технологій штучного інтелекту та NLP, розробку програмного забезпечення, тестування та оцінку ефективності системи. У висновках зазначено, що система дозволяє автоматизувати процес адаптації резюме, працює як єдиний цілісний алгоритм, пройшла тестування на функціональність і продуктивність, а також готова до використання. Таким чином, отримані результати повністю відповідають поставленим завданням, оскільки

реалізовано Telegram-бота з описаними функціями, проведено тестування і зроблено оцінку системи.

Оцінка розділів

Перший розділ систематизує предметну область, досліджує сучасні методи аналізу резюме та виклики рекрутингу, інтегрує технології штучного інтелекту для автоматизації обробки тексту. Опис предметної області є методологічно обґрунтованим і репрезентативним, висвітлює ключові компоненти резюме та проблематику ручного відбору. Аналіз наявного програмно-технічного забезпечення охоплює перелік аналогів із загальними характеристиками, хоча порівняння та висновки щодо доцільності нового ПЗ подані поверхнево. Визначення функціональних та нефункціональних вимог є загальним, без детальної конкретизації, а моделі варіантів використання та специфікації сценаріїв відсутні, що ускладнює повне розуміння системи. В цілому, розділ є вичерпним у теоретичному аспекті, але потребує більш глибокої деталізації та структурованості.

Другий розділ проєктує архітектуру системи, розглядає варіанти архітектурних моделей, описує модульну структуру, взаємодію компонентів та інтерфейс користувача. Архітектурні рішення є технічно обґрунтованими і відповідають вимогам Telegram-бота, проте відсутнє чітке обґрунтування вибору конкретної архітектури та шаблонів проектування. Декомпозиція модулів і об'єктів описана на базовому рівні, без UML-діаграм та візуалізацій, що знижує прозорість структури. Взаємодія компонентів сформульована логічно, але без глибокої деталізації патернів. Інтерфейс користувача охоплює ключові взаємодії, але описано загально, без макетів чи схем. Вибір технологій відповідає завданню, хоча відсутній аналіз альтернатив. Загалом, розділ є архітектурно виваженим і функціонально орієнтованим, але потребує більшої деталізації та ілюстрацій.

Третій розділ реалізує Telegram-бота, описує програмну реалізацію основних модулів, логіку обробки повідомлень, інтеграцію з Azure OpenAI та генерацію PDF-документів. Реалізація відповідає архітектурі, використані сучасні технології Python та бібліотеки. Тестування проведено як вручну, так і автоматизовано, охоплює функціональність, продуктивність та інтеграцію, проте відсутня візуалізація результатів тестування. Інтерфейс користувача є простим і зручним, хоча опис UI є узагальненим без прикладів. Підсумки реалізації підтверджують готовність системи до використання, але не містять детального аналізу перспектив розвитку. Розділ є функціонально повним і технічно досконалим, проте потребує більшої деталізації тестування та UI.

Позитивні сторони

Робота демонструє інноваційний підхід до автоматизації адаптації резюме через інтеграцію Telegram-бота з технологіями штучного інтелекту, що є сучасним і перспективним рішенням. Вступ і перший розділ систематизують предметну область, підкреслюючи актуальність і практичну значущість теми. Архітектурні рішення другого розділу є технічно обґрунтованими, модульна структура відповідає функціональним вимогам, а використання Python і бібліотек забезпечує гнучкість і масштабованість. Третій розділ реалізує повноцінний функціональний продукт, який пройшов тестування, має зручний інтерфейс і відповідає поставленим завданням. Загалом, робота є методологічно обґрунтованою, технологічно доцільною та практично орієнтованою.

Недоліки

Опис предметної області та функціональних вимог у першому розділі є загальним і потребує більшої деталізації, зокрема відсутні UML-діаграми, конкретні сценарії

використання та специфікації. Аналіз аналогів не містить глибокого порівняння та чітких висновків щодо доцільності нового ПЗ. Другий розділ не містить обґрунтування вибору архітектури та шаблонів проектування, а також не має візуалізацій структури та взаємодії компонентів. Опис інтерфейсу користувача є поверхневим без прикладів чи макетів. Третій розділ не надає детального опису алгоритмів реалізації, не містить візуалізації результатів тестування та не розкриває перспектив розвитку системи. Загалом, робота потребує більшої структурованості, деталізації та ілюстративного матеріалу.

✉ Відгук в цілому

Робота є актуальною та має високу практичну значущість у контексті цифровізації ринку праці та автоматизації рекрутингу. Зміст відповідає темі і поставленим завданням, демонструє новизну у використанні Telegram-бота та інтеграції з Azure OpenAI для глибокого аналізу резюме. Ідеї та рішення обґрунтовані, хоча деякі аспекти потребують більшої деталізації. Програмний продукт є працездатним, функціональним і зручним для користувачів, реалізований із застосуванням сучасних технологій. Водночас, відсутність детальних моделей, візуалізацій та глибокого аналізу тестування дещо знижує загальне враження. Проте, робота демонструє високий рівень аналітичного мислення та самостійності здобувача.

✉ Оцінка кваліфікаційної роботи

Кваліфікаційна робота виконана в повному обсязі з дотриманням основних методичних і технічних вимог. Пояснювальна записка є логічно структурованою, змістовною, розкриває тему, містить обґрунтовані висновки та нові ідеї. Розроблений Telegram-бот є функціональним, реалізований із застосуванням сучасних технологій, пройшов тестування і готовий до використання. Здобувач демонструє компетентність, послідовність викладу та аналітичний підхід. Водночас, робота має недоліки у деталізації моделей, описі алгоритмів та візуалізації результатів тестування, що не дозволяє оцінити її як відмінну. Враховуючи це, робота заслуговує оцінки «добре».

✉ Рекомендації

Роботу рекомендовано до захисту з урахуванням необхідності доопрацювання окремих аспектів. Рекомендується доповнити пояснювальну записку UML-діаграмами варіантів використання, класів та взаємодії, деталізувати функціональні та нефункціональні вимоги, а також надати приклади інтерфейсу користувача. Варто розширити опис алгоритмів реалізації та додати візуалізацію результатів тестування для підвищення наочності. Також доцільно сформулювати перспективи розвитку системи та можливі напрямки оптимізації. У разі успішного врахування цих рекомендацій робота може бути впроваджена як ефективний інструмент автоматизації рекрутингу.

РІШЕННЯ ЕКСПЕРТНОЇ КОМІСІЇ КАФЕДРИ ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Назва кваліфікаційної роботи Програмна система для автоматизації аналізу резюме відповідно до вимог вакансі
Автор Хоменко Вікторія Вадимівна
Освітня програма Освітньо-професійна програма «Інженерія програмного забезпечення»
Рівень вищої освіти Перший (бакалаврський)
Спеціальність 121 «Інженерія програмного забезпечення»
Науковий керівник: Бедратюк Леонід Петрович, д-р фіз. -мат. наук, професор

На основі аналізу кваліфікаційної роботи на дотримання вимог академічної доброчесності (у т.ч. відсутності ознак академічного плагіату) з урахуванням результатів перевірки роботи спеціалізованим програмним засобом(ами) комісія зробила такий висновок:

№	Висновок	Позначка про відповідність
1	Ознаки академічного плагіату	
1.1	Запозичення, виявлені в роботі, є законними і не є академічним плагіатом (далі – зазначаються підстави віднесення запозичень до правомірних, якщо потрібно). Робота приймається до захисту.	**відповідає**
1.2	Виявлені запозичення не є академічним плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи (далі – зазначаються детальні та аргументовані підстави віднесення запозичень до правомірних). Робота приймається до захисту, але має бути відкоригована.	
1.3	Виявлені запозичення не є академічним плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. Робота може бути допущена до захисту після того як буде відкоригована та доопрацьована і успішно пройде повторну перевірку на академічний плагіат.	
1.4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття текстових запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	
2	Інші види порушень академічної доброчесності	

Підтвердження:

Запозичення, виявлені в роботі, є законними і не є плагіатом, оскільки:

1) у тексті кваліфікаційної роботи системами перевірки на плагіат виявлено схожість з деякими документами в частині загальновживаних обов'язкових словосполучень у стандартних бланках (титулка, завдання, анотація, відомість документів), у структурі змісту, назвах розділів/підрозділів тощо, у назвах публікацій та у переліку джерел посилання;

2) в якості запозичень системою було зафіксовано деякі послідовності вихідного коду і посиланання на бібліотеки, які є стандартними мовними конструкціями програмування та не можуть розглядатися як об'єкт авторських прав і, відповідно, їх порушення;

3) усі запозичення є фрагментарними або мають належним чином оформленні посилання;

4) виявлені модифікації тексту не впливають на відсоток схожості.

Сумарний обсяг всіх запозичень, визначений системою виявлення збігів ідентичності/ схожості, складає 2,3%, що, з урахуванням наведених обгрунтувань, відповідає характеру теми і свідчить на користь кваліфікаційної роботи.

Дата 3.06.2021

Завідувач кафедри _____ Підпис Леонід БЕДРАТЮК Ім'я, ПРІЗВИЩЕ

Гарант освітньої програми _____ Підпис Леонід БЕДРАТЮК Ім'я, ПРІЗВИЩЕ

Керівник кваліфікаційної роботи _____ Підпис Леонід БЕДРАТЮК Ім'я, ПРІЗВИЩЕ