МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Спеціальність: 123 «Комп'ютерна інженерія»

Освітньо-професійна програма: «Безпека комп'ютерних систем і мереж»

Група: 4КБ-02

Дипломний проєкт

здобувача освіти денної форми навчання
КБ.02.20.000.ДП

СТОЯНОВА
ВІТАЛІЯ СТЕПАНОВИЧА

м. Одеса
2025 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Спеціальність: **123 «Комп'ютерна інженерія»**

Освітньо-професійна програма: «**Безпека комп'ютерних систем і мереж»**

Група: **4КБ-02**

ПОЯСНЮВАЛЬНА ЗАПИСКА

до дипломного проєкту на тему:

Розробка віртуального асистента для безпечного формування текстових документів

Проектний матеріал складається з пояснювальної записки на 75 сторінках та графічного (презентаційного) матеріалу на 16 аркушах (слайдах)

Дипломник _____ (Стоянов В.С.)

Керівник _____ (Кривченко А.А.)

Консультанти:

з економічного розділу _____ (Канський М.Ю.)

з розділу охорони праці та техніки безпеки _____ (Чорновол Н.І.)

з нормоконтролю _____ (Петрашова В.І.)

старший консультант _____ (Кривченко Ю.В.)

До захисту допущений

Голова циклової комісії _____ (Кривченко Ю.В.)

Завідувач відділення _____ (Краснокутська К.Г.)

Захист «24» червня 2025 р. Протокол ЕК № 4

Оцінка ЕК 4 (добре) / 80 б.

Секретар ЕК _____

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Відділення *комп'ютерних систем* Комісія *КТ та ПІ*

Спеціальність *123 «Комп'ютерна інженерія»*

Освітньо-професійна програма *«Безпека комп'ютерних систем і мереж»*

ЗАТВЕРДЖУЮ:

Заст. дир. з НВР *Беркань І.В.*

"19" 05 2025 р.

ЗАВДАННЯ

на дипломний проєкт

Здобувачеві освіти *Стоянову Віталію Степановичу*

(прізвище, ім'я, по батькові)

1. Тема проєкту *Розробка віртуального асистента для безпечного формування текстових документів*

затверджена наказом по коледжу від "14" 11 2024 р. № 246

2. Термін здачі закінченого проєкту 16.06.25

3. Вихідні дані до проєкту *1. Реалізувати роботу віртуального асистенту за допомогою чат-боту у месенджері; 2. Реалізувати заповнення шаблонів документів у форматах .csv та .json на основі введених користувачем даних; 3. Передбачити запис сесії з користувачем у базу даних; 4. Передбачити збереження шаблонів; 5. Кожен файл повинен мати унікальний id; 6. Передбачити можливість скачування заповнених файлів; 7. Передбачити видалення непотрібних файлів з бази даних.*

4. Зміст розрахунково-пояснювальної записки (перелік питань, які необхідно розробити)

Аналіз та класифікація віртуальних асистентів; Аналітичний огляд і вибір платформи для віртуального асистента; Огляд засобів формування текстових документів; Визначення форматів текстових документів для обробки; Аналіз засобів безпеки; Аналіз програмних засобів розробки; Вибір і розробка архітектури програмного забезпечення; Вибір платформи хостингу; Розробка та реалізація застосунку віртуального асистента

5. Перелік графічного (презентаційного) матеріалу (з точним зазначенням обов'язкових креслень, кількості слайдів)

Діаграма взаємодії для віртуального асистента; Діаграма процесу формування документів; Модель захисту віртуального асистента; Інтегрована архітектура віртуального асистента; Гексагональна архітектура віртуального асистента; ER-діаграма сутностей і зв'язків у БД віртуального асистента; Діаграма станів віртуального асистента; Блок-схема алгоритму забезпечення безпеки при формуванні текстових документів; Приклади роботи розробленого віртуального асистента для безпечного формування текстових документів

6. Консультанти по проекту, із зазначенням розділів проєкту, що їх стосується

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
Основний розділ	*Кривченко А.А.*		
Економічний розділ	*Канський М.Ю.*		
Розділ охорони праці	*Чорновол Н.І.*		
Нормоконтроль	*Петрашова В.І.*		
Старший консультант	*Кривченко Ю.В.*		

7. Дата видачі завдання 15.05.25р.

Керівник *Кривченко А.А.* _____
(підпис)

Завдання прийняв до виконання *Стоянов В.С.* _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/р	Назва етапів дипломного проєкту	Термін виконання етапів дипломного проєкту (роботи)	Відмітка про виконання
1	*Вступ. Постановка задачі проєктування*	15.05.25	Виконано
2	*Аналіз технічного завдання та пошук літератури*	16.05.25	Виконано
3	*Огляд функціональних елементів для реалізації моделі*	17.05.25	Виконано
4	*Огляд засобів формування текстових документів*	18.05.25	Виконано
5	*Вибір технічних та програмних засобів розробки*	22.05.25	Виконано
6	*Аналіз засобів безпеки*	26.05.25	Виконано
7	*Вибір і розробка архітектури програмного забезпечення*	01.06.25	Виконано
8	*Вибір платформи хостингу*	06.06.25	Виконано
9	*Розробка та реалізація застосунку віртуального*	10.06.25	Виконано
	асистента для формування текстових документів		
10	*Розробка модулів застосунку віртуального асистента*	11.06.25	Виконано
11	*Випробування застосунку та аналіз результатів*	12.06.25	Виконано
12	*Виконання економічних розрахунків*	13.06.25	Виконано
13	*Розробка питань з охорони праці та техніки безпеки*	14.06.25	Виконано
14	*Підготовка мультимедійної презентації проекту*	16.06.25	Виконано

Дипломник _____
(підпис)

Керівник _____
(підпис)

Формат	Зона	Поз.	Позначення	Назва	Кіл.	Примітка
				Документація		
			КБ 02. 20 000. 00 ДП	Дипломний проєкт		
А4			КБ 02. 20 000. 00 ДП ПЗ	Пояснювальна записка	1	

Зм.	Арк.	№ докум.	Підпис	Дата	КБ 02. 20 000. 00 ДП		
Розробив		Стоянов В.С.		16.06	Розробка віртуального асистента для безпечного формування текстових документів	Літ.	Аркуш 4 / Аркушів 75
Перевірив		Кривченко А.А.		16.06		Н Д П	
Н. контр.		Петрашова В.І.		16.06.25		ВСП "ОТФК ОНТУ" гр.4КБ-02	
Затверд.		Кривченко Ю.В.		16.06			

ЗМІСТ

ВСТУП

Нині інформаційні технології стрімко розвиваються, а автоматизація рутинних операцій стає запорукою ефективності у багатьох сферах діяльності.

Однією з актуальних задач є безпечне формування текстових документів, що набуває особливого значення в умовах зростаючих вимог до цифрової безпеки та зручності користувацького досвіду. Саме тому дипломний проект, присвячений розробці віртуального асистента для безпечного формування текстових документів, є своєчасним і перспективним дослідженням у галузі сучасних інформаційних систем.

Метою даної роботи є створення чат-бота у Telegram, який забезпечуватиме інтерактивну взаємодію з користувачем для заповнення шаблонів документів, а також реалізацію механізмів безпечного збереження, обробки і маніпулювання файлами, що містять як дані, так і самі шаблони. Проект передбачає підтримку роботи з файлами різних форматів — зокрема, .csv, .xlsx, .json та .xml, що дозволить забезпечити гнучкість у роботі з даними та адаптацію до різних типів вихідних документів.

Безпека інформації є пріоритетним завданням як для корпоративних користувачів, так і для окремих осіб.

Особливу увагу в проекті має бути приділено питанням безпеки даних та збереження історії сесій з користувачами, що може бути реалізовано через збереження кожного файлу з унікальним ідентифікатором, а також ведення журналу взаємодій у базі даних. Такий підхід дозволить не лише забезпечити належний рівень безпеки, а й сприятиме подальшій аналітиці використання системи, оптимізації її роботи та інтеграції з іншими сервісами.

Розробка віртуального асистента для безпечного формування текстових документів є важливим кроком у напрямку підвищення якості взаємодії користувача з цифровими сервісами, автоматизації рутинних операцій та забезпечення високого рівня безпеки інформації. Реалізація цього проекту дозволить зробити вагомий внесок у вдосконалення процесів управління інформацією, відповідно до сучасних вимог до якості та безпеки даних.

1 ОСНОВНИЙ РОЗДІЛ

1.1 Аналіз та класифікація віртуальних асистентів

Сучасні віртуальні асистенти – це програмні агенти, розроблені для автоматизованої взаємодії з користувачем. Вони відіграють важливу роль як у загальному спілкуванні, так і у вирішенні спеціалізованих завдань у бізнес-сфері, освіті, медицині та інших галузях. Розглянемо основні класифікаційні параметри, доповнені прикладами, таблицями та ілюстраціями.

Віртуальні асистенти можуть бути орієнтовані на ведення неформального діалогу або ж забезпечення спеціалізованої допомоги. Приклади:

- Асистенти загального призначення: чат-боти, що ведуть невимушену розмову та надають широке коло інформаційних послуг (наприклад, боти для спілкування у соціальних мережах);
- Спеціалізовані асистенти: боти, призначені для виконання конкретних завдань, як-от консультація клієнтів у банківській сфері або підтримка при бронюванні послуг.

Форма доступу визначає, де та як користувач взаємодіє з ботом (рис.1.1):

- Веб-інтерфейси: інтеграція чат-ботів безпосередньо на сайтах для надання швидких відповідей на питання відвідувачів;
- Месенджери: використання ботів у додатках типу Telegram, Facebook Messenger, Slack, Viber – що дозволяє використовувати переваги мобільного зв’язку, навіть при обмеженому доступі до Інтернету;
- Групові чати та підписка: розсилки та автоматизовані повідомлення через групові канали або за допомогою підписки.

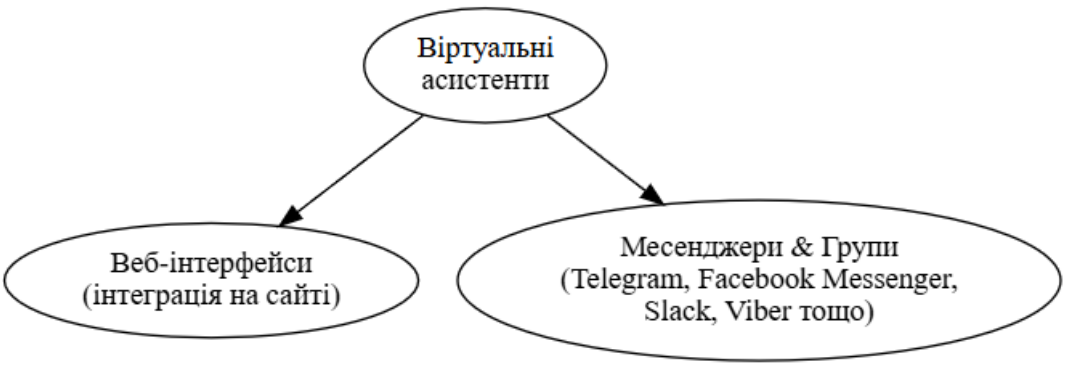


Рисунок 1.1. Блок-схема розташування віртуальних асистентів за формою доступу

Асистенти можуть використовувати різні типи взаємодії, що визначають зручність користування:

- Кнопковий інтерфейс: взаємодія здійснюється через заздалегідь підготовлені варіанти відповіді, що підходить для стандартних операцій;
- Текстовий інтерфейс: надає можливість вільного введення тексту користувачем для формування більш складних запитів;
- Голосовий інтерфейс: дозволяє задавати запитання або команди за допомогою голосу; використовує технології розпізнавання та синтезу мови.

Приклади для різних типів взаємодії:

- Текстовий бот: користувач надсилає запитання у вигляді текстового повідомлення – відповідь формується на основі алгоритмів NLP;
- Голосовий бот: Google Assistant або Apple Siri, що розуміють усне мовлення та генерують голосові відповіді.

За алгоритмом обробки вхідних даних можна розрізнити:

- Сценарні (детерміновані) системи: взаємодія за заздалегідь визначеним деревом рішень;
- Інтелектуальні рішення: базуються на методах штучного інтелекту (NLP, NLU, NLG) і можуть адаптуватися до запитів користувача;
- Гібридні системи: комбінують фіксовані сценарії з можливостями самонавчання, дозволяючи розпізнавати наміри користувача та оперативно реагувати.

Приклади ботів з різними алгоритмами обробки вхідних даних:

- Бот підтримки клієнтів банку, відповідає за типові запити (сценарний тип);
- Watson Assistant від IBM – інтелектуальний асистент, що використовує машинне навчання для підвищення якості відповідей.

Остаточну класифікацію можна проводити за цільовою аудиторією:

- Особисті асистенти: орієнтовані на індивідуальних користувачів (наприклад, Siri, Google Assistant);
- Бізнес-асистенти: спрямовані на автоматизацію процесів в корпоративному середовищі (наприклад, чат-боти для CRM, ERP систем).

Таблиця 1.1. Основні класифікаційні критерії віртуальних асистентів

Параметр класифікації	**Опис**	**Приклади**
За призначенням	Загальна комунікація або виконання вузькоспеціалізованих завдань	@ChatBuddy (розмовний), @BankSupportBot (спеціалізований)
За формою доступу	Середовище реалізації: веб-сайт, месенджери або групові чати	Веб-чат на сайті, Telegram-бот, Facebook Messenger-бот
За типом інтерфейсу	Методи взаємодії: кнопковий, текстовий, голосовий	Кнопковий бот Viber, текстовий бот у Slack, голосовий бот Google Assistant
За алгоритмом роботи	Спосіб обробки запитів: сценарний, інтелектуальний або гібридний	FAQ-бот (сценарний), Watson Assistant (інтелектуальний)
За цільовою аудиторією	Призначення для окремих користувачів або корпоративного сегмента	Siri (особистий асистент), chatbot для CRM (бізнес-асистент)

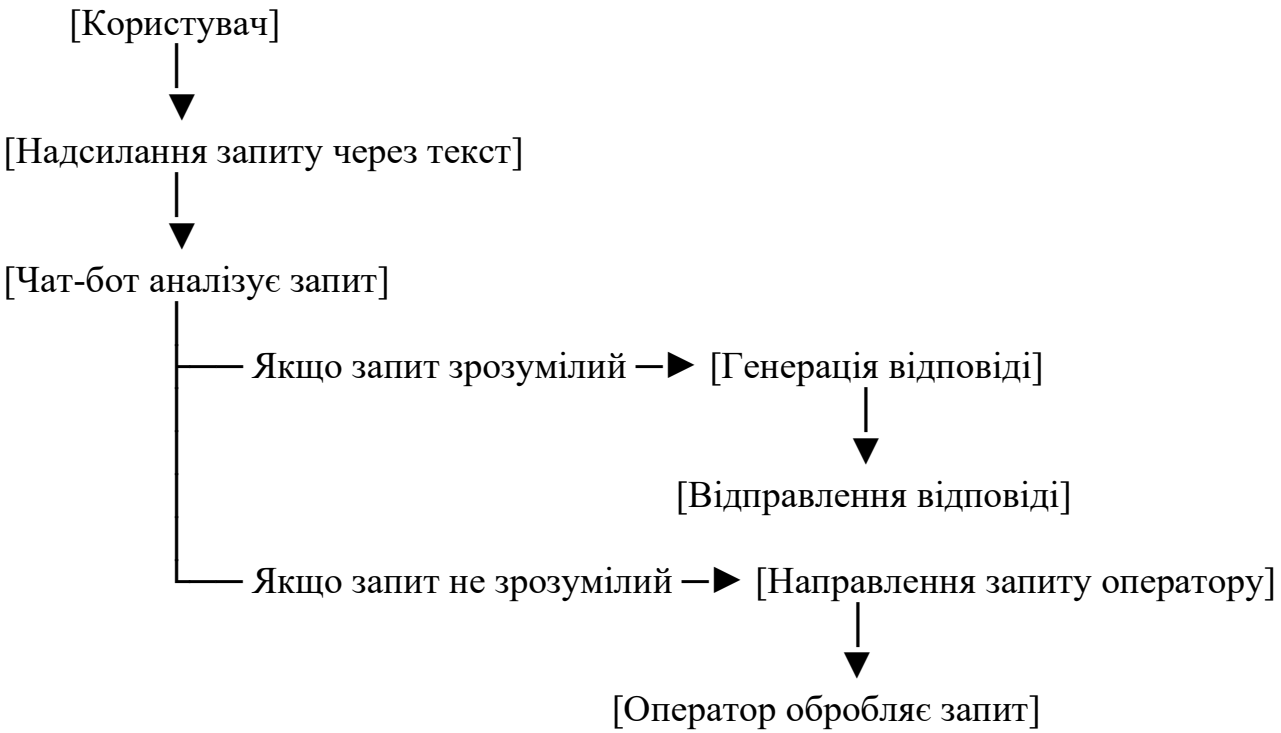


Рисунок 1.2. Діаграма взаємодії для віртуального асистента

Сценарій використання віртуального асистента може бути таким: компанія використовує віртуального асистента для автоматизації відповіді на запити клієнтів. При зверненні користувача за допомогою текстового

повідомлення бот аналізує ключові слова, порівнює їх із заздалегідь підготовленим сценарієм, а у випадку незрозумілого запиту – направляє повідомлення на обробку оператору.

Аналіз сучасних віртуальних асистентів із застосуванням класифікаційних підходів дозволяє побачити багатогранність їх застосування. Набір критеріїв – від функціонального призначення до алгоритму роботи – допомагає сформувати уявлення про гнучкість рішень, що використовуються для автоматизації взаємодії з користувачами. Отже, подальша розробка проекту має спиратися на інтеграцію передових технологій для вдосконалення безпеки, ефективності та зручності при формуванні текстових документів.

1.2 Аналітичний огляд і вибір платформи для віртуального асистента

Концепція месенджерів бере свій початок у часи поштових листів і телеграфних повідомлень, проте стрімкий розвиток Інтернету та мобільних технологій привів до появи сучасних платформ, що забезпечують миттєву комунікацію як між окремими користувачами, так і в групових чатах. Сьогодні месенджери стали невід'ємною частиною цифрового середовища, дозволяючи не лише обмінюватися повідомленнями, а й інтегрувати численні додаткові сервіси, серед яких – створення й управління віртуальними асистентами. За останніми даними, найбільш популярними платформами у 2025 році є WhatsApp, WeChat, Facebook Messenger, Telegram, Viber та Snapchat. Проте для розробки віртуального асистента для безпечного формування текстових документів важливими є не лише масштаби аудиторії, а й технічні можливості інтеграції та вимоги до інфраструктури. У табл. 1.2 наведено порівняльну інформацію, що ілюструє основні характеристики платформ з погляду розробки ботів.

Враховуючи вимоги проекту щодо безпеки, мінімізації інфраструктурних витрат і спрощення процесу розробки, Telegram виявився оптимальним вибором. Основні переваги, які стали вирішальними у прийнятті рішення:

- Спрощена архітектура. Режим long polling дозволяє працювати без необхідності розгортання спеціалізованого серверного додатку. Бот виступає

в ролі клієнта, що періодично надсилає запити до серверів Telegram для отримання нових повідомлень. Це значно зменшує вимоги до інфраструктури та сприяє прискоренню процесу розробки;

- Відкрита платформа з розвиненою документацією. API Telegram активно підтримується і постійно вдосконалюється. Це дозволяє легко інтегрувати додаткові функції, що є важливим для безпечного формування текстових документів та забезпечення контролю якості;
- Високий рівень безпеки. Враховуючи специфіку роботи з конфіденційними документами, можливості налаштування безпеки у Telegram є важливим чинником, що гарантує захист даних користувачів.

Таблиця 1.2. Основні характеристики платформ з погляду розробки ботів

Месенджер	**Активні користувачі**	**Особливості API для ботів**	**Коментар**
WhatsApp	понад 2 млрд	WhatsApp Business API орієнтований виключно на бізнес-рішення; використання API передбачає платну підписку	Висока аудиторія, але обмеження у використанні для некомерційних проектів і фінансове навантаження
Snapchat	значна аудиторія, але менше за лідерів	Офіційного API для створення ботів немає	Не оптимальний варіант для розробки асистента через відсутність відкритих інтерфейсів для ботів
Viber	стабільна популярність в регіоні	API базується на системі webhooks; вимагає розгортання постійно активного веб-сервісу для обробки вхідних повідомлень	Підвищена складність розгортання через необхідність налаштування серверної інфраструктури; мінімум зручності для швидкого прототипування
Telegram	близько 900 млн активних користувачів	Підтримує як webhooks, так і режим long polling; відкритий і добре документований API	Забезпечує спрощене налаштування (режим long polling дозволяє ботам працювати як клієнтам, що періодично опитують сервер), що значно знижує вимоги до інфраструктури; високий рівень безпеки та адаптивності

На рис.1.3 наведено діаграму, яка наочно демонструє принципову різницю між використанням webhook-функціоналу (як у Viber) та режимом long polling (як у Telegram).

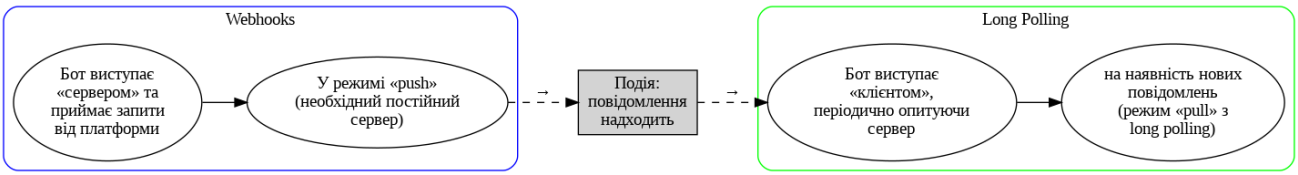


Рисунок 1.3. Відмінності між Webhooks та Long Polling

Як показує діаграма, використання long polling дозволяє значно спростити архітектуру системи, що є критично важливим для проекту з обмеженими ресурсами на розгортання.

На основі проведеного аналітичного огляду платформ та актуальних даних 2025 року було прийнято рішення використовувати Telegram як платформу для розробки віртуального асистента. Такий вибір забезпечує:

1. Спрощене розгортання завдяки режиму long polling;
2. Відкритість та гнучкість платформи, що дозволяє швидко адаптувати систему під специфічні вимоги проекту;
3. Високий рівень безпеки, що є пріоритетним для безпечного формування текстових документів.

Подальша розробка системи буде зосереджена на інтеграції функціональності Telegram API з урахуванням вимог до коректної обробки даних та оптимізації взаємодії з користувачем.

1.3 Огляд засобів формування текстових документів

У сучасному цифровому середовищі автоматизація бізнес-процесів стає запорукою оперативності, точності та безпеки. Одним із важливих напрямків є формування текстових документів із застосуванням автоматичного заповнення шаблонів. Сучасні платформи для цього надають бізнесу змогу швидко створювати, редагувати та зберігати документи без участі оператора, що значно знижує ризики людської помилки та прискорює робочий процес.

Серед популярних систем, що використовуються для автоматичного

формування документів і електронного підпису, можна виділити такі— DocuSign, PandaDoc та Formstack Documents. Розглянемо особливості кожного сервісу, їх переваги і недоліки, а також вплив на інтеграцію в систему обробки даних.

Огляд основних платформ:

1. DocuSign є однією з найбільш відомих платформ для електронного підпису та управління документами. Система дозволяє користувачам завантажувати шаблони, визначати ключові поля для заповнення й використовувати автоматичне заповнення на основі наданих даних. Основні переваги:

- Високий рівень безпеки. Забезпечення захисту конфіденційної інформації за допомогою сучасних криптографічних протоколів;
- Інтеграція з іншими системами. Можливість зв'язку з CRM, ERP та іншими бізнес-системами, що сприяє автоматизації робочих процесів;
- Зручність електронного підпису. Підтримка віддаленої роботи з документами, що дозволяє швидко оформлювати угоди;

Основні недоліки:

- Фінансові витрати. Використання платформи пов'язане з високими орієнтовними витратами, що може бути проблемою для бізнесів з обмеженим бюджетом;
- Обмежена функціональність. Система зосереджена переважно на електронному підписі, а можливості для розширеної автоматизації іноді можуть бути недостатніми;

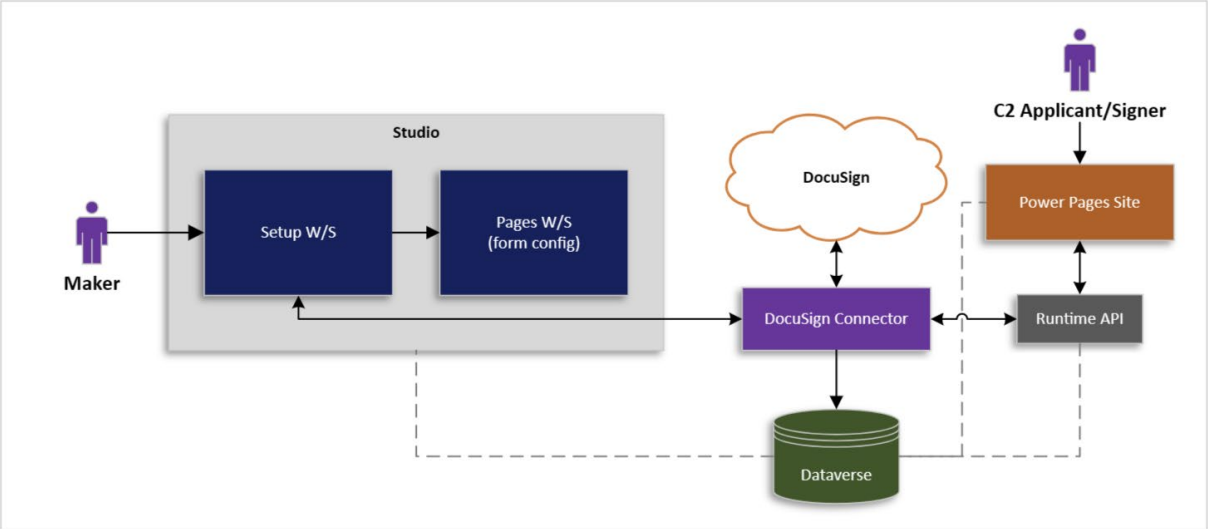


Рисунок 1.4. Формування текстових документів у DocuSign

2. PandaDoc пропонує інтегроване рішення для управління документами, яке дозволяє створювати, редагувати та автоматично заповнювати шаблони. Продукт розроблено для підвищення продуктивності бізнесу за рахунок швидкого формування документів і підтримки електронного підпису. Основні переваги:

- Швидкість автоматизації. Миттєве створення документів за допомогою заданих шаблонів;
- Підтримка електронного підпису. Надійне оформлення документів із забезпеченням правової сили електронного підпису;
- Інтеграція з CRM системами. Легка взаємодія з іншими бізнес-інструментами для синхронізації даних;

Недоліки:

- Необхідність навчання. Новим користувачам потрібно деякий час для освоєння інтерфейсу та функціональних можливостей;
- Витрати на ліцензії. Для повноцінного використання можуть знадобитися додаткові фінансові ресурси;

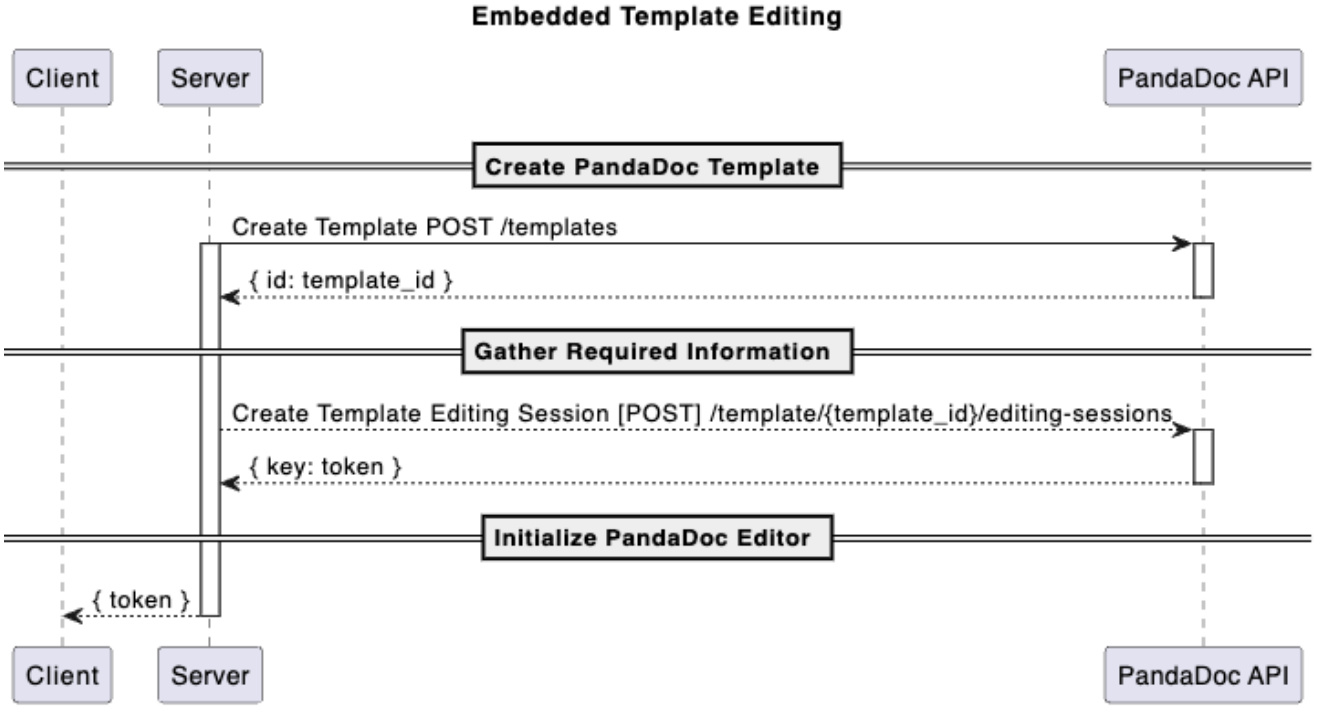


Рисунок 1.5. Формування текстових документів у PandaDoc

3. Formstack Documents спеціалізується на автоматизації створення та заповнення документів. Система дозволяє створювати шаблони з можливістю

налаштування умов заповнення, інтегрувати дані з різних джерел (CRM, бази даних, електронні таблиці) та адаптувати документи під конкретні потреби бізнесу.

Основні переваги:

- Гнучкість налаштування. Можливість створення індивідуальних шаблонів із використанням складних умов і логіки заповнення;
- Широка інтеграція з джерелами даних. Автоматичне отримання інформації з різних систем забезпечує зниження обсягів ручної роботи;
- Масштабованість. Здатність обробляти великі об’єми даних та забезпечувати масову генерацію документів;

Недоліки:

- Обмеження в візуалізації даних. Порівняно простий інтерфейс може не відповідати вимогам бізнесу з високими очікуваннями щодо візуального представлення;
- Вартість впровадження. Для великих організацій ціна ліцензій може бути значною.

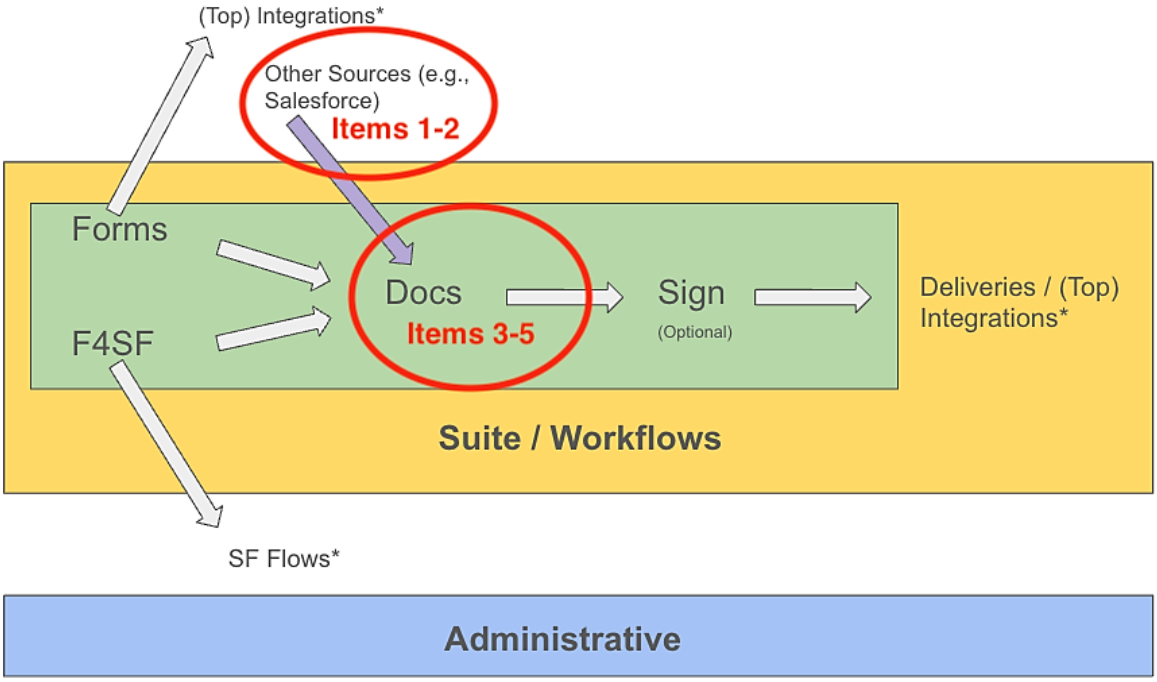


Рисунок 1.6. Формування текстових документів у Formstack Documents

У табл. 1.3 наведено порівняння основних характеристик розглянутих платформ. На рис. 1.7 наведено спрощену схему, що відображає типовий процес автоматичного формування документів із використанням подібних платформ.

Таблиця 1.3. Порівняння основних характеристик розглянутих платформ

Платформа	Основні переваги	Основні недоліки	Цільова аудиторія
DocuSign	- Високий рівень безпеки - Інтеграція з CRM, ERP - Зручність електронного підпису	- Висока вартість - Обмежена функціональність	Великі корпорації, організації з високими вимогами
PandaDoc	- Швидкість автоматизації - Підтримка електронного підпису - Інтеграція з CRM системами	- Витрати на ліцензії - Необхідність навчання користувачів	Малі та середні підприємства
Formstack Documents	- Гнучке налаштування шаблонів - Широка інтеграція з різними джерелами даних - Масштабованість	- Обмеження у візуалізації даних - Висока вартість впровадження	Організації, що потребують високої кастомізації

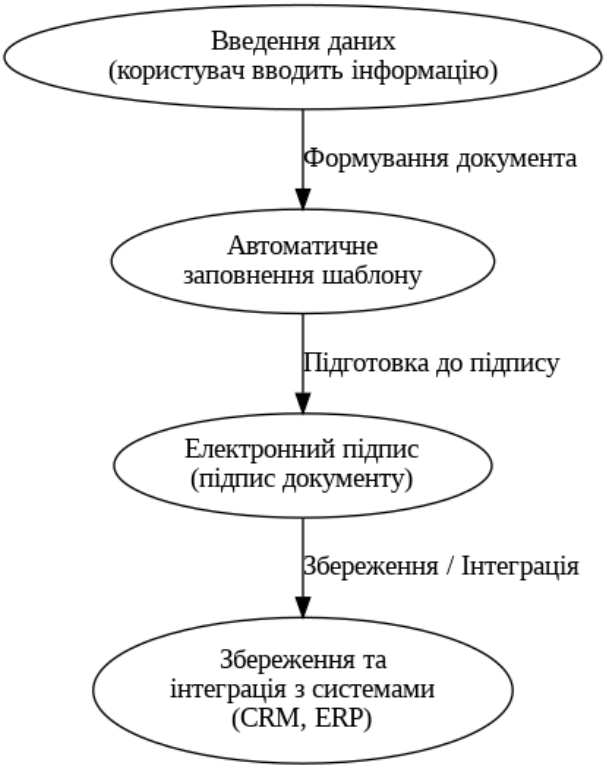


Рисунок 1.7. Діаграма процесу формування документів

Відповідно до рис. 1.7 реалізуються наступні етапи:

1. Введення даних. Користувач надсилає необхідну інформацію через інтерфейс (наприклад, чат-бот або веб-додаток);

2. Заповнення шаблону. Платформа автоматично інтегрує отримані дані у заздалегідь підготовлений шаблон документа;

3. Електронний підпис та інтеграція. Готовий документ може бути підписаний електронно та інтегрований із суміжними бізнес-системами.

Попри існування широкого спектру комерційних рішень для автоматизації формування документів, розробка власної системи через чат-бот має свої явні переваги:

- Гнучкість функціоналу. Можливість створення рішення, яке точно адаптоване під специфічні потреби бізнесу;
- Повний контроль над даними. Забезпечення додаткових заходів безпеки та збереження конфіденційності;
- Інтеграція з іншими сервісами. Спрощення взаємодії з зовнішніми системами без надмірного навантаження на інфраструктуру.

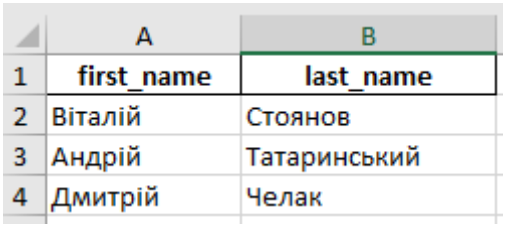
Аналіз сучасних засобів автоматичного заповнення шаблонів документів демонструє, що кожна з платформ має свої унікальні особливості. Для проекту «Розробка віртуального асистента для безпечного формування текстових документів» розробка власного рішення дозволяє врахувати специфічні вимоги, зберегти гнучкість функціоналу та забезпечити високий рівень безпеки обробки інформації. Сучасні засоби автоматизації документів не лише пришвидшують бізнес-процеси, але й знижують витрати, пов’язані з ручним введенням даних. У 2025 році аналітичні платформи та інструменти як DocuSign, PandaDoc та Formstack Documents активно використовуються в різних галузях, що підтверджує важливість впровадження подібних рішень у сучасних інформаційних системах.

1.4 Визначення форматів текстових документів для обробки віртуальним асистентом

Для забезпечення ефективної роботи віртуального асистента, який реалізує безпечне формування текстових документів, важливо правильно організувати

структуру даних та шаблонів. У рамках даного проекту бот працюватиме з двома групами файлів: перша група містить файли, де зберігаються інформація та дані, а друга – файли-шаблони, що задають структуроване оформлення кінцевих документів.

До файлів, які містять інформацію для підстановки в шаблони, належать CSV, XLSX, JSON і XML. Файли CSV (Comma-Separated Values) є текстовими, де дані розділені комами або іншими розділювачами. Такий формат вирізняється своєю простотою й широкою сумісністю з різними програмами, що робить його популярним для обміну табличними даними, однак відсутність єдиного стандарту щодо розділювачів чи наявності заголовків може ускладнювати автоматичне розпізнавання структури даних. У свою чергу, файли XLSX, що використовуються електронними таблицями (наприклад, Microsoft Excel), дозволяють зберігати дані у вигляді кількох аркушів із підтримкою форматування, формул і обчислень. Це забезпечує потужну функціональність для роботи з великою кількістю даних, проте парсинг таких файлів вимагає використання спеціалізованих бібліотек і може бути більш ресурсномістким.



	A	B
1	first_name	last_name
2	Віталій	Стоянов
3	Андрій	Татаринський
4	Дмитрій	Челак

Рисунок 1.8. Приклад CSV-файлу

Формат JSON (JavaScript Object Notation) дозволяє зберігати структуровані дані у вигляді вкладених об’єктів і масивів, що робить його надзвичайно зручним для передачі даних через веб-API. Проте, його простий синтаксис вимагає уважності до форматування, оскільки маленькі помилки (наприклад, пропущені коми чи лапки) можуть порушити процес парсингу. XML (eXtensible Markup Language) слугує для представлення даних у вигляді ієрархічної структури з використанням тегів і атрибутів. Хоча XML забезпечує високу гнучкість і можливість валідації за допомогою схем, його розміченість часто збільшує обсяг файлу і ускладнює обробку, що робить його менш ефективним у порівнянні з JSON

для простих завдань.

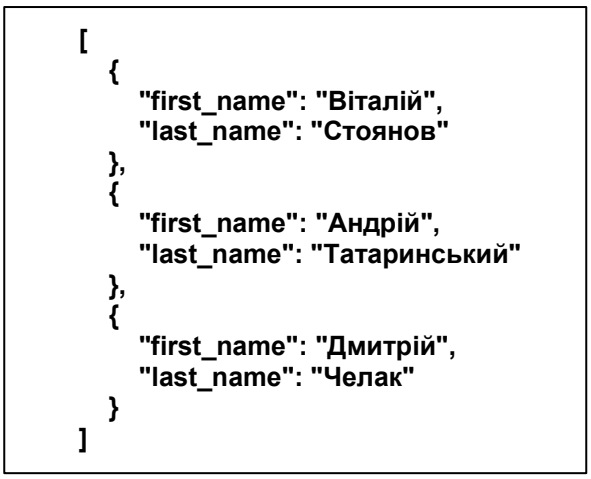


Рисунок 1.9. Приклад JSON-файлу

Таким чином, вибір формату для збереження даних визначається компромісом між простотою, масштабованістю та вимогами до обробки інформації. Кожен із зазначених форматів критично залежить від того, як дані організовано – для коректної роботи віртуальному асистенту дані повинні бути структуровані у вигляді масиву об’єктів або коректно оформленої таблиці.

Щодо шаблонів документів, що використовуються для формування кінцевих текстових документів, в проекті передбачено роботу з файлами типів TXT, MD, DOCX і PPTX. Файли TXT забезпечують просте збереження чистого тексту без форматування, що дає змогу швидко завантажувати і обробляти інформацію, однак вони не дозволяють зберігати стилізацію або структуроване оформлення. Формат MD (Markdown) є зручним завдяки простому синтаксису, який дозволяє створювати структурований текст із заголовками, списками, таблицями та посиланнями, що може бути легко конвертовано в HTML або інші формати. Однак для створення документів з професійним оформленням Markdown може виявитися обмеженим. Файли DOCX (Microsoft Word) підтримують широкий спектр форматувань, включаючи стилі, таблиці, зображення та складні макети, що робить їх ідеальним варіантом для генерації офіційних документів, але їхнє використання вимагає застосування спеціальних бібліотек для читання та парсингу, що збільшує загальну складність системи. Подібно, файли PPTX застосовуються для створення презентацій із слайдами, де поряд із текстом можуть бути включені графічні

елементи й анімаційні ефекти. Хоча цей формат є потужним інструментом для візуального представлення інформації, його застосування стосується, як правило, спеціалізованих сценаріїв, де обов’язково потрібна візуальна складова.

Кожен шаблон має містити спеціальні місця для підстановки даних – так звані маркери або placeholders, які допомагають віртуальному асистенту автоматично замінювати їх на відповідну інформацію під час формування фінального документа. Правильно підібрані формати дозволяють забезпечити сумісність системи з різними джерелами даних та знизити вимоги до парсингу і обробки, створюючи гнучке середовище для автоматизації великих обсягів документів.

Враховуючи специфіку завдання, обрані формати дозволяють поєднати простоту обробки із забезпеченням високої якості кінцевого продукту. Наприклад, при роботі з CSV чи XLSX-файлами важливо, щоб структура даних відповідала очікуванням – дані мають бути організовані як масив об’єктів або таблиця, де кожен рядок відповідає окремому запису. Аналогічно, шаблони у форматі DOCX або MD повинні містити чітко визначені місця для змінних даних, що дозволяє автоматично здійснювати їх заміну.

Таким чином, визначення форматів текстових документів для обробки віртуальним асистентом ґрунтується на ретельному аналізі властивостей кожного формату та його сумісності з поставленими завданнями. Використання CSV, XLSX, JSON і XML для збереження даних у комплексі з шаблонами у вигляді TXT, MD, DOCX і PPTX створює основу для ефективної, гнучкої та безпечної автоматизації формування документів, що відповідає вимогам сучасних бізнес-систем та забезпечує зручний обмін інформацією між різними джерелами даних.

1.5 Аналіз засобів безпеки для віртуального асистента

Сучасний віртуальний асистент – це комплексне рішення, що інтегрується з платформами месенджерів та має взаємодіяти з численними джерелами даних і шаблонами документів. В умовах, коли кожен месенджер має власні правила взаємодії та API, системі необхідно забезпечити не лише коректність функціонування, але й високий рівень безпеки, який гарантує захист даних

користувачів і зменшує ризик неправомірного доступу.

Одним із важливих аспектів є сумісність із API месенджерів. Протягом розробки необхідно ретельно вивчити документацію кожної платформи, адже специфічні обмеження, формат передачі даних та методи автентифікації можуть суттєво впливати на безпеку системи. Правильна реалізація автентифікації, використання шифрованих з'єднань (наприклад, HTTPS) та контроль доступу до API допомагають уникнути витоків даних і запобігти атакам, пов'язаним з несанкціонованим доступом.

Не менш важливою є обробка та безпечна робота з різними типами шаблонів і форматів файлів, що надходять до бота. Оскільки чат-бот має опрацьовувати файли різних форматів – текстові (.txt, .md), документи Office (.docx, .pptx) – для кожного з них потрібно реалізувати специфічний алгоритм валідації і парсингу. Перед тим, як виконати операції пошуку і заміни в шаблоні, система має перевірити, що файл відповідає очікуваним характеристикам, враховуючи як синтаксичні, так і структурні вимоги. Таке підходження забезпечує не лише якість заповнення даних, але й мінімізує можливість атаки шляхом завантаження неструктурованих або змінених файлів, що можуть містити шкідливий код.

При обробці даних, що надходять у форматах CSV, XLSX, JSON або XML, особлива увага приділяється процесу нормалізації. Оскільки клієнти можуть використовувати різні формати файлів для внесення даних, необхідно не лише розпізнати тип файлу на основі розширення чи заголовка, але й провести його валідацію для перевірки відповідності очікуваній структурі. Наприклад, якщо дані мають бути представлені як масив структурованих об'єктів, система перевірить, що кожен запис містить усі обов'язкові атрибути. Нормалізація даних дозволяє звести всі вхідні дані до єдиного формату, що спрощує подальшу їхню обробку і знижує ризик помилок, які можуть негативно вплинути на безпеку операцій.

Ще одним важливим механізмом безпеки є реалізація станової машини (state machine) для управління взаємодією з користувачем. Чат-бот, який підтримує декілька етапів збору і обробки даних, повинен чітко контролювати свої стани, щоб гарантувати правильну послідовність операцій.

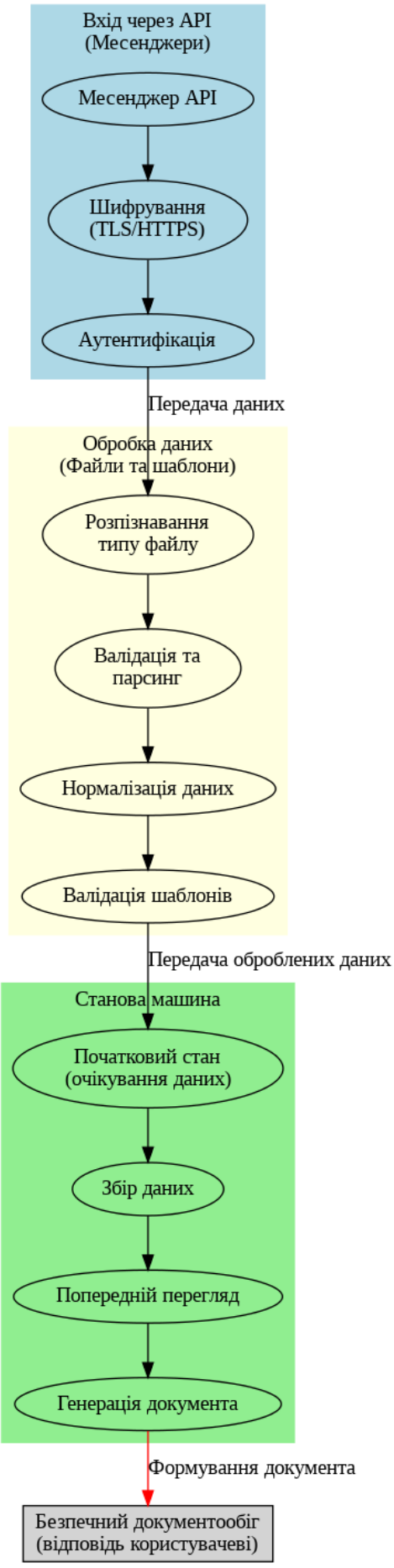


Рисунок 1.10. Модель захисту віртуального асистента

Схема на рис.1.10 демонструє основні блоки, що забезпечують безпечну роботу чат-бота, зокрема:

1. Безпечний вхід через API месенджерів (з використанням шифрування та автентифікації);

2. Модуль обробки даних і шаблонів (розпізнавання типу файлу, валідація, парсинг, нормалізація);

3. Станову машину для контролю послідовності дій користувача (від збору даних до генерації кінцевого документа);

4. Формування фінального захищеного документообігу й відправка результату користувачеві.

Відповідно до схеми на рис.1.10 реалізуються наступні етапи:

1. Вхід через API:

- Месенджер API: отримує запит від користувача через месенджер (наприклад, Telegram);
- Шифрування (TLS/HTTPS): забезпечує захищене з'єднання для передачі даних;
- Аутентифікація: перевіряє, що запит походить від авторизованого джерела;

2. Обробка даних та шаблонів:

- Розпізнавання типу файлу: визначає, з яким форматом має справу (CSV, XLSX, JSON, XML або формати шаблонів TXT, MD, DOCX, PPTX);
- Валідація та парсинг: перевіряє коректність файлу та перетворює його в структуровану інформацію;
- Нормалізація даних: приводить дані до єдиного стандарту для подальшої обробки;
- Валідація шаблонів: перевіряє, чи шаблони містять необхідні маркери для підстановки даних;

3. Станова машина:

- Від початкового стану (очікування даних) до збору інформації,

предпросмотру (перевірка) введених даних та фінальної генерації документа;

- Контролює послідовність дій і дозволяє гарантувати — дані вводяться правильно, а оформлення здійснюється згідно з заданою логікою;

4. Фінальний вузол (Безпечний документообіг): після успішного проходження всіх етапів формується кінцевий документ, який відправляється користувачеві, забезпечуючи високий рівень безпеки та відповідності заданим стандартам.

В цілому, забезпечення високої безпеки віртуального асистента базується на інтеграції ряду механізмів:

- Надійна взаємодія з API месенджерів за рахунок використання сучасних протоколів шифрування та автентифікації;
- Гнучке і розширюване рішення для обробки різних типів шаблонів і файлів даних, що включає попередню валідацію, парсинг і нормалізацію інформації;
- Контроль послідовності операцій за допомогою станової машини, що дозволяє коректно керувати процесом введення і обробки даних.

Комплексний підхід до забезпечення безпеки віртуального асистента є ключовим фактором для досягнення високої ефективності системи, зниження ризиків і гарантування високого рівня захисту як даних користувачів, так і інформації, що використовується для заповнення текстових документів. Ретельне вивчення документації, розробка модульних рішень і впровадження ефективних алгоритмів обробки даних забезпечать надійну основу для створення сучасного і безпечного чат-бота.

1.6 Аналіз програмних засобів розробки віртуального асистента

У сучасному середовищі розробки віртуальних асистентів важливим аспектом є вибір потужного інтегрованого середовища розробки (IDE) та оптимальної мови програмування. Ці рішення впливають не тільки на продуктивність коду, але й на ефективність взаємодії з API месенджерів, що слугують каналом для роботи чат-ботів.

1.6.1 Вибір фреймворку, API та технологій зберігання даних

Одним із найбільш популярних інструментів для веб-розробки є WebStorm – сучасне IDE від компанії JetBrains, що базується на платформі IntelliJ IDEA (рис.1.11). WebStorm призначений для розробки веб-додатків із застосуванням HTML, CSS, JavaScript та популярних фреймворків (Angular, React, Vue.js, Node.js). Потужний редактор коду з підсвічуванням синтаксису, автодоповнення, зручне згортання блоків коду, а також вбудований дебагер роблять цей інструмент надзвичайно зручним для розробників. Особливу увагу приділено інтеграції з системами контролю версій (Git, SVN, Mercurial), підтримці плагінів для розширення функціональності, а також можливості «Live Edit», коли зміни у коді одразу відображаються у браузері. Завдяки кросплатформеності WebStorm доступний на Windows, macOS та Linux, що дозволяє обирати улюблене середовище незалежно від операційної системи.

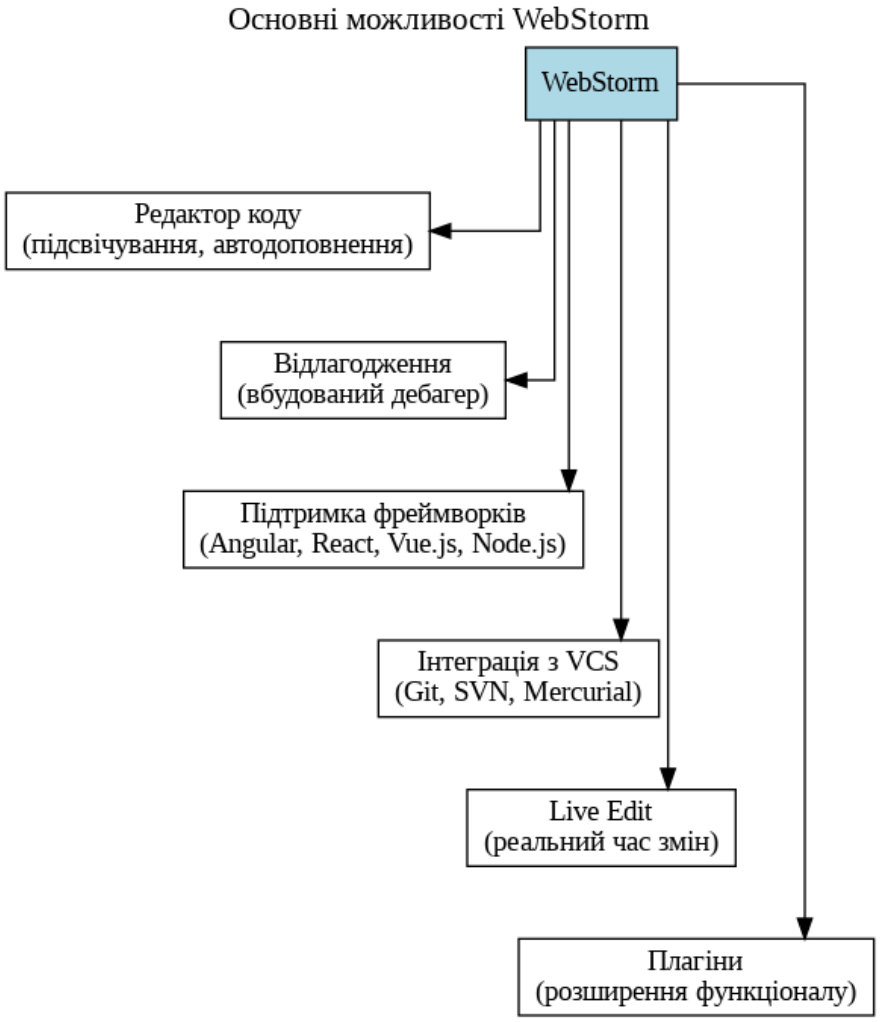


Рисунок 1.11. Схема основних можливостей WebStorm

Основну увагу для розробки віртуального асистента приділено вибору мови програмування. Серед можливих варіантів – Python, Java, PHP та Node.js.

Python має простий синтаксис, багатий набір бібліотек і фреймворків (Flask, Django), а також потужні засоби обробки тексту. Проте, через інтерпретовану природу, виконання коду може бути дещо повільнішим. Java відзначається стабільністю та масштабованістю, але вимагає більшого обсягу коду для досягнення аналогічного результату, що може сповільнити розробку. PHP широко застосовується у веб-розробці завдяки таким фреймворкам, як Laravel чи Symfony, проте його можливості щодо обробки тексту та шаблонів можуть бути обмеженими в порівнянні з іншими мовами.

Найбільш перспективним варіантом, враховуючи асинхронну природу сучасних API месенджерів, є Node.js. Завдяки високій швидкості, широкому вибору пакетів і здатності обробляти багатокористувацькі запити, Node.js дозволяє ефективно реалізувати подієву модель роботи чат-бота. Особливо цікавим є використання фреймворку NestJS, який забезпечує високу модульність, внутрішню шину повідомлень і дозволяє розбивати обробку кожного типу файлу на окремі юзкейси.

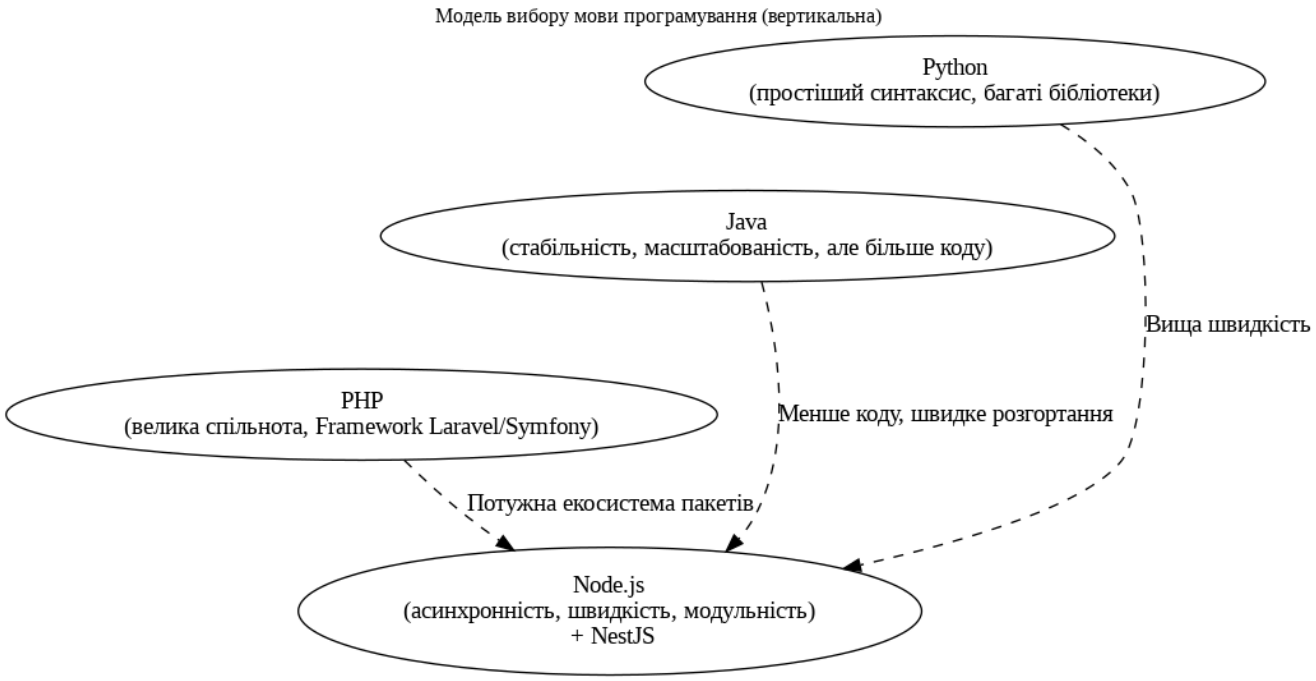


Рисунок 1.8. Модель вибору мови програмування
для розробки віртуального асистента

Аналіз засобів розробки показує, що WebStorm є потужною платформою для веб-розробки завдяки своєму інтуїтивному редактору коду, вбудованому дебагеру та гнучким можливостям налаштування робочого середовища. З-поміж мов програмування оптимальним рішенням для розробки віртуального асистента є Node.js із застосуванням фреймворку NestJS завдяки його асинхронній, подієвій моделі та здатності ефективно працювати з API месенджерів.

Вибір технологій охоплює кілька важливих компонентів:

1. Інтеграція з месенджерами та обробка повідомлень. За основу ми беремо взаємодію з Telegram API через long polling, що значно спрощує логіку комунікації і мінімізує потребу у постійно активних серверних ресурсах. Таким чином, бот працює, як клієнт, що періодично опитує сервер, що дозволяє ефективно контролювати вхідні дані при зниженні ризиків помилок;

2. Відокремлення обробки даних і шаблонів. Система розділена на декілька незалежних модулів: один відповідальний за аналіз запитів і парсинг вхідних повідомлень, інший – за роботу з базами даних, третій виконуватиме логіку підбору та заповнення шаблонів документів. Така модульність дозволяє легко додавати нові функціональні можливості і змінювати окремі частини без впливу на інші;

3. Вибір баз даних. Для зберігання структурованих даних (наприклад, інформації про сесії, користувачів, шаблони документів та файли даних) ми проаналізували переваги декількох СКБД:

- PostgreSQL вирізняється потужною реалізацією реляційної моделі, підтримкою багатьох типів даних, розширених можливостей індексації та високою безпекою. Вона є оптимальним рішенням для забезпечення цілісності й складних зв’язків між даними;
- MySQL та MSSQL також демонструють високу ефективність, але у нашому проекті уподіблено схильність до використання PostgreSQL завдяки її відкритості та гнучкості;
- MongoDB надає значну гнучкість завдяки нереляційній моделі даних, проте для завдань, де важлива сувора цілісність зв’язків між записами,

вона не є першочерговим вибором;

- Redis використовується як кеш і сховище для оперативного зберігання станів (наприклад, для стейт-машини), забезпечуючи надзвичайну швидкість доступу завдяки зберіганню даних безпосередньо в оперативній пам’яті;

4. Використання сучасних мов програмування. Для реалізації бекенду нашого асистента оптимальним є застосування Node.js у поєднанні з популярним фреймворком NestJS. Така конфігурація дозволяє впровадити асинхронну та подієву модель обробки запитів. Додатково використання TypeScript дозволяє застосовувати статичну типізацію, що сприяє виявленню помилок на етапі компіляції та покращенню читабельності коду. Ці рішення підвищують стабільність і як якість, так і швидкість розробки, що є критичним для проектів з високими вимогами до продуктивності.

Для наочного представлення архітектурного підходу було створено схему, що демонструє взаємодію ключових модулів та технологічних рішень (рис.1.9).

Рисунок 1.9. Інтегрована архітектура віртуального асистента

Центральними елементами є Telegram API, який надсилає запити від користувачів, та модуль бізнес-логіки (BOT), який реалізується в середовищі Node.js + NestJS з використанням TypeScript;

- Обробка запитів здійснюється окремим компонентом, який взаємодіє з базою даних PostgreSQL для збереження та валідності даних, а також з Redis для кешування та побудови стану взаємодії;
- Модуль шаблонів здійснює автоматичне заповнення збережених даних у підготовлені шаблони документів, після чого генерується остаточний документ, що відправляється назад користувачу;
- Завдяки інтеграції сучасних інструментів розробки та баз даних забезпечується висока продуктивність, безпека та зручність оперативного управління даними.

Вибір технологій, таких як PostgreSQL для забезпечення структурованого зберігання даних і Redis для високошвидкісного кешування, у поєднанні з асинхронною платформою на базі Node.js + NestJS та використанням TypeScript дозволяє створити сучасне, масштабоване і гнучке рішення. Такий підхід забезпечує стабільну роботу віртуального асистента навіть у разі високої навантаженості і змін у вимогах бізнесу, а також сприяє легкості подальшого розширення функціоналу системи.

1.6.2 Вибір необхідних бібліотек

При розробці віртуального асистента, що автоматично заповнює шаблони документів через Telegram, критично важливо правильно підібрати засоби програмування, які дозволять побудувати систему з високою масштабованістю, модульністю та безпекою. У цьому контексті окрему увагу приділено вибору бібліотек, які не лише забезпечують основу для реалізації серверної логіки, а й сприяють оптимальній інтеграції з зовнішніми сервісами й зручному тестуванню.

NestJS є сучасним фреймворком для створення серверних додатків на базі Node.js із використанням TypeScript. Завдяки своїй архітектурі, натхненній Angular, він забезпечує справжню модульність, що дозволяє розділити функціональність системи на окремі, незалежні блоки. Також сильне місце

фреймворку – зручна маршрутизація, використання експресивних декораторів і middleware, що дозволяють легко додавати додаткову логіку до обробки запитів. Вбудована система Dependency Injection (DI) забезпечує ізоляцію компонентів, що значно полегшує як розширення системи, так і її тестування. Крім того, підтримка патерну CQRS дозволяє чітко розділити операції, що змінюють стан системи, від читальних запитів, створюючи можливості для майбутньої інтеграції з додатковими інтерфейсами, наприклад, для іншого месенджера або мобільного додатка.

Для роботи з Telegram API ми обрали бібліотеку Telegraf, яка забезпечує інтуїтивно зрозумілий інтерфейс для розробки чат-ботів на JavaScript і TypeScript. Telegraf дозволяє легко обробляти вхідні повідомлення, команди, події та навіть роботу з медіа, підтримуючи використання middleware для додаткової логіки обробки. Інтеграція Telegraf із NestJS дозволяє створити єдиний робочий простір для побудови серверної частини та спрощує адаптацію до додаткових функціональних вимог.

Для взаємодії з реляційними базами даних було обрано Prisma — сучасний ORM, що забезпечує зручний API із підтримкою статичної типізації. Prisma спрощує роботу зі створенням, модифікацією та відкочуванням міграцій, а також генерує оптимізований код для виконання SQL-запитів. Такий підхід гарантує, що дані зберігатимуться структуровано, забезпечуючи високий рівень цілісності та продуктивності. Можливість роботи з різними СКБД (PostgreSQL, MySQL, SQLite) дає змогу легко адаптувати систему під конкретні вимоги проекту.

Контроль якості програмного забезпечення є невід'ємною частиною розробки, тому для написання тестів обрано бібліотеку Jest. Завдяки її простому синтаксису, автоматичному моніторингу змін та можливості паралельного запуску тестів, Jest дозволяє швидко виявляти помилки та забезпечує високу стабільність коду. Додаткові інструменти для генерації мок-об'єктів і створення звітів по покриттю тестами роблять її незамінною при реалізації автоматизованого тестування.

Сучасні засоби, зокрема NestJS, Telegraf, Prisma і Jest, дозволяють створити

гнучке, масштабоване та безпечне рішення. Використання цих бібліотек забезпечує розбиття функціональності на незалежні модулі, полегшує управління залежностями і сприяє легкості тестування, що в цілому мінімізує ризики помилок і дозволяє зосередитися на реалізації бізнес-логіки додатку. Таким чином, комплекс обраних технологій створює надійну основу для подальшої розробки та інтеграції віртуального асистента з іншими сервісами та інтерфейсами.

1.7 Вибір і розробка архітектури програмного забезпечення віртуального асистента

Архітектура програмного забезпечення є критично важливим аспектом будь-якого проекту, оскільки вона визначає структуру, організацію та взаємодію компонентів системи. Для розробки віртуального асистента, який реалізує безпечне формування текстових документів через месенджер Telegram, було обрано підхід, що поєднує традиційну клієнт-серверну модель із принципами сучасних архітектурних патернів, таких як гексагональна архітектура та Domain-Driven Design.

Розробка архітектури програмного забезпечення віртуального асистента спирається на наступні ключові принципи:

- Розширюваність та модульність. Система розбита на незалежні модулі, кожен з яких відповідає за конкретну функціональність (обробка запитів, валідація даних, генерація документів тощо). Це дозволяє легко додавати нові можливості й адаптувати систему до зростаючих вимог;
- Ізоляція бізнес-логіки. Використання гексагональної (порт-адаптер) архітектури дозволяє ізолювати ядро додатку (бізнес-логіку) від зовнішніх залежностей (API месенджерів, файлових систем, сервісів зберігання даних);
- Завдяки цьому можна впроваджувати зміни в технологічні аспекти (наприклад, переходити від одного месенджера до іншого) без втрати цілісності логіки системи;
- Застосування принципів SOLID та GRASP. Вони гарантують, що кожен клас чи модуль має чітку відповідальність, легко тестується та може бути розширений. Принцип однієї відповідальності (SRP) означає, що модулі, що

обробляють запити або генерують документи, повинні вирішувати лише визначену задачу. Принцип інверсії залежності (DIP) допомагає ізолювати ядро бізнес-логіки від конкретних реалізацій зовнішніх компонентів;

- Domain-Driven Design (DDD). Цей підхід дозволяє точно змоделювати бізнес-домен, розбити систему на контексти із чіткими межами та забезпечити тісну співпрацю між розробниками й бізнес-експертами.

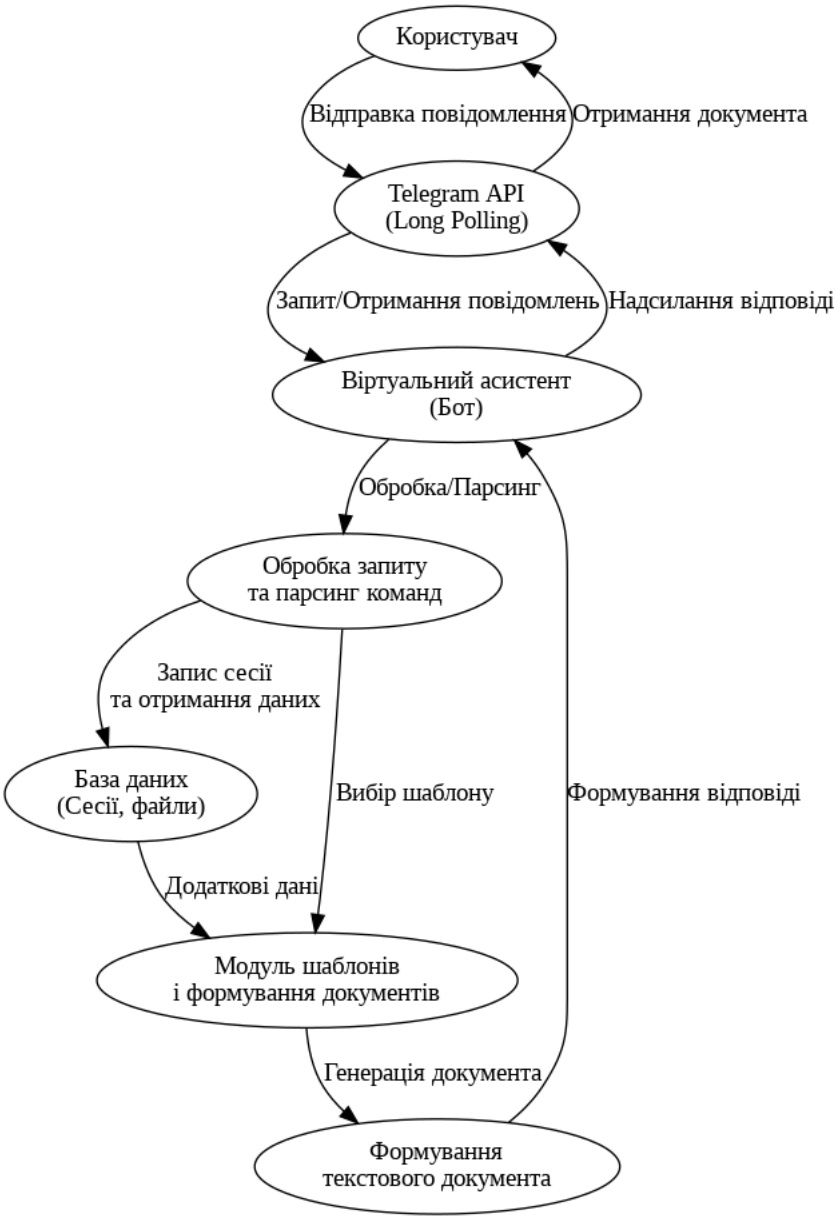


Рисунок 1.9. Гексагональна архітектура віртуального асистента

Для віртуального асистента було розроблено архітектурну модель, що включає такі основні компоненти:

- Користувач. Початковий вузол, який надсилає запит або повідомлення через месенджер;

- Telegram API. Забезпечує з’єднання з Telegram серверами у режимі long polling. Бот періодично опитує сервер для отримання нових повідомлень;
- Віртуальний асистент (Бот). Ядро системи, яке приймає повідомлення від Telegram API та обробляє їх;
- Обробка запиту і парсинг. Модуль, що аналізує повідомлення, визначає команду (наприклад, збереження файлу або заповнення шаблону) та розподіляє завдання;
- База даних. Зберігає сесії користувачів, файли з даними (CSV, XLSX, JSON, XML) і шаблони документів;
- Модуль шаблонів даних. Вибирає відповідний шаблон і підставляє до нього дані;
- Формування документа. Останній етап, де формується готовий документ, що відправляється назад користувачу через Telegram API.

Використання long polling у взаємодії з Telegram API дозволяє значно спростити архітектуру – бот виступає як клієнт, що час від часу опитує сервер, що зменшує вимоги до постійно активної серверної інфраструктури, підвищуючи таким чином безпеку системи. На рис.1.9 наведено діаграму, яка узагальнює описані компоненти та їх взаємозв'язки:

1. Користувач (U) надсилає запит або повідомлення через месенджер, яке надходить на сервер Telegram за допомогою функції long polling (T);
2. Telegram API (T) передає запит до ядра системи – Віртуального асистента (B), яке обробляє повідомлення і визначає подальшу дію;
3. Модуль обробки запиту (A) аналізує повідомлення, здійснює парсинг і визначає потрібний тип операції (наприклад, збереження файлів або заповнення шаблону);
4. Отримані дані передаються для валідації та збереження у Базі даних (D), а також для вибору відповідного шаблону в Модулі шаблонів (S);
5. Формування документа (R) – останній етап, де на основі шаблону та перевірених даних формується остаточний документ, який відправляється назад користувачу.

Використання гексагональної архітектури в поєднанні з принципами Domain-Driven Design дозволяє створити гнучке й розширюване рішення для розробки віртуального асистента. Інтеграція з різними API месенджерів, обробка файлів даних і шаблонів, а також застосування чітко структурованої послідовності обробки запитів гарантує, що система буде ефективно й безпечно функціонувати протягом усього життєвого циклу проекту.

1.8 Оптимальний вибір платформи хостингу для віртуального асистента

Під час розробки систем, що забезпечують безперебійний доступ до послуг та гарантують високу масштабованість, правильний вибір хостингової платформи є вирішальним моментом. Для даного проекту, який реалізує функціонал Telegram-бота для заповнення шаблонів документів, були проаналізовані кілька варіантів, серед яких — AWS, Google Cloud, Heroku, Digital Ocean та Railway. Кожен із цих провайдерів має свої переваги, але остаточний вибір залежить від конкретних вимог щодо продуктивності, безпеки та швидкості розгортання.

AWS (Amazon Web Services) надає величезний набір сервісів, що дозволяють розгортати майже будь-які типи додатків зі складною логікою. Потужність, високий рівень безпеки та можливість горизонтального масштабування роблять AWS привабливим для проектів з високими вимогами до стабільності. Однак, використання AWS потребує значних знань системного адміністрування та ретельної оптимізації витрат, оскільки комплексний функціонал може обернутися високою вартістю експлуатації.

Google Cloud Platform майже не поступається AWS за масштабованістю та набором сервісів, а також забезпечує відмінну інтеграцію з іншими продуктами Google. Проте для деяких користувачів його може бути складніше адаптувати через специфіку налаштувань і менш широку підтримку в деяких регіонах.

Heroku пропонує дуже зручний і швидкий спосіб розгортання веб-додатків із спрощеним робочим процесом. Завдяки підтримці розгортання через Git і серії плагінів, Heroku дозволяє швидко запустити прототип, що є важливим для проектів з обмеженими ресурсами на старті. Однак система має певні обмеження у

масштабуванні, що може стати проблемою на етапі переходу до продуктивної експлуатації при великій кількості користувачів.

Digital Ocean вирізняється простотою використання та доступною ціновою політикою. Її інтерфейс зручний для розробників, що віддають перевагу швидкому розгортанню і прозорості тарифів. Проте, Digital Ocean може не забезпечувати такого ж рівня розширеної функціональності, як AWS чи Google Cloud, що може обмежувати можливості при складнішому функціоналі.

Railway — відносно нова платформа, яка орієнтована на швидкість розгортання та зручне управління проектами через Git. Railway спрощує налаштування і автоматичне масштабування, що робить її привабливим вибором для невеликих та швидкозростаючих проектів. Однак, у порівнянні з усталеними провайдерами, Railway може мати обмеження у функціональних можливостях або регіональній доступності деяких сервісів.

Вибір платформи значною мірою залежить від пріоритетних критеріїв: якщо основним завданням є максимальна масштабованість та розширена безпека, AWS чи Google Cloud можуть стати оптимальним вирішенням. Для ранніх етапів розробки та швидкого прототипування за рахунок мінімальних налаштувань хорошим вибором може бути Heroku або Digital Ocean. Для нашого конкретного проекту, де важлива простота розгортання, зручність управління та можливість майбутнього розширення функціоналу (зокрема, з урахуванням потенційного переходу до додаткових інтерфейсів, як-от Viber або мобільний додаток), Railway виглядає дуже приваблюючим варіантом. Проте, з огляду на очікуване зростання користувальницької бази, система може впослід мігрувати до більш потужного рішення, такого як AWS або Google Cloud, що підтримує високий рівень ресурсів та розширені можливості управління.

На рис.1.10 наведена схема, що ілюструє критерії вибору хостингової платформи для проекту розробки віртуального асистента. Вибір хостингової платформи безпосередньо впливає на продуктивність, надійність та масштабованість проекту. Аналіз різних варіантів дозволяє підібрати рішення, яке відповідає поточним потребам розробки та має потенціал для розширення. Для

проекту віртуального асистента для безпечного формування текстових документів, де важлива швидкість розгортання та легкість управління на ранніх стадіях, Railway демонструє значні переваги. Проте, при зростанні навантаження можливе переходження до більш розвинених платформ, таких як AWS або Google Cloud, що гарантує безперебійну роботу системи на довгострокову перспективу.

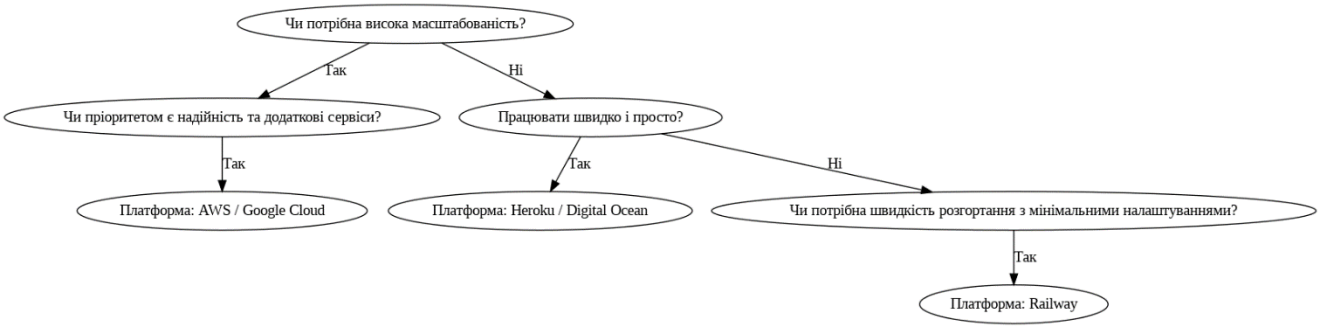


Рисунок 1.10. Схема вибору платформи хостингу проекту віртуального асистенту

1.9 Розробка та реалізація застосунку віртуального асистента

Цей розділ описує комплекс заходів із створення та впровадження застосунку, який дозволить автоматизувати процес заповнення текстових шаблонів через месенджер. Логіка системи побудована на сучасних технологічних рішеннях, що гарантують швидкий обмін інформацією, масштабованість, безпеку та ефективну обробку запитів користувачів. Розробка застосунку віртуального асистента базується на модульному підході, який ізолює бізнес-логіку від прикладних залежностей та забезпечує можливість легкої адаптації системи до змін у вимогах.

Розроблюваний застосунок взаємодіє з користувачем через Telegram API, використовуючи long polling, що дозволяє регулярно отримувати повідомлення без потреби в постійно активній серверній інфраструктурі. Далі вхідний запит проходить через модуль обробки повідомлень, який аналізує отриману інформацію, виконує парсинг і визначає потрібні операції.

На основі отриманих даних здійснюється взаємодія з базою даних, де зберігаються сесії користувачів, шаблони документів та інші необхідні дані, а також відбувається вибір відповідного шаблону для заповнення. Остаточний етап – це генерація текстового документу, який відправляється назад користувачу через Telegram API.

1.9.1 Розробка бази даних користувачів, шаблонів і файлів

Для забезпечення ефективної роботи віртуального асистента, який автоматично заповнює шаблони документів, критично важливою складовою є правильно спроектована база даних. Вона дозволяє централізовано зберігати та управляти інформацією про користувачів, їх шаблони та дані, що надходять для заповнення. У розробленій системі визначено три основні сутності:

- Користувачі (User);
- Шаблони (Template);
- Дані (Data).

Цей принцип забезпечує зв'язок між користувачем та його ресурсами, а також сприяє впорядкованому й ефективному збереженню інформації.

В таблиці User (табл.1.4) зберігаються основні дані про користувачів, що взаємодіють із чатом асистента. Ця таблиця містить такі поля, як унікальний ідентифікатор користувача (UUID), ідентифікатор чату, username та особисті дані користувача. Таблиця гарантує, що кожен користувач має унікальні дані, що дозволяє легко відстежувати історію взаємодії.

Таблиця 1.4. Організація таблиці User (користувачі)

Назва поля	Тип	Опис	Індекс
ID користувача	uuid	Унікальний ідентифікатор (PK)	PK
chatId	text	Ідентифікатор чату в Telegram	
Username	text	Username користувача в Telegram	
firstName	text	Ім'я користувача в Telegram	
lastName	text	Прізвище користувача в Telegram	

Таблиця Template (табл.1.5) призначена для зберігання інформації про шаблони документів, які асоціюються з конкретними користувачами. Для кожного шаблону зберігаються його унікальний ідентифікатор, ідентифікатор файлу (як він представлений у сервісі Telegram), назва шаблону та ідентифікатор користувача, що є зовнішнім ключем, який зв'язує шаблон з відповідним користувачем.

Таблиця Data (табл.1.6) містить записи з даними, які використовуються для заповнення шаблонів. Подібно до шаблонів, кожен запис зберігає інформацію про унікальний ідентифікатор, ID файлу, назву самого файлу та посилання на

користувача через зовнішній ключ. Це дозволяє легко встановлювати зв’язок між отриманими даними та користувачами, що їх надсилають.

Таблиця 1.5. Організація таблиці Template (шаблони)

Назва поля	Тип	Опис	Індекс
fileId	text	ID файла в Telegram	–
Name	text	Назва файлу/шаблону	–
userId	uuid	Ідентифікатор користувача (FK)	FK

Таблиця 1.6. Організація таблиці Data (дані)

Назва поля	Тип	Опис	Індекс
fileId	text	ID файла в Telegram	–
name	text	Назва файлу	–
userId	uuid	Ідентифікатор користувача (FK)	FK

Для наочного представлення зв’язків між основними сутностями було створено ER-діаграму сутностей і зв’язків, яка демонструє, як таблиці пов’язані між собою (рис.1.11).

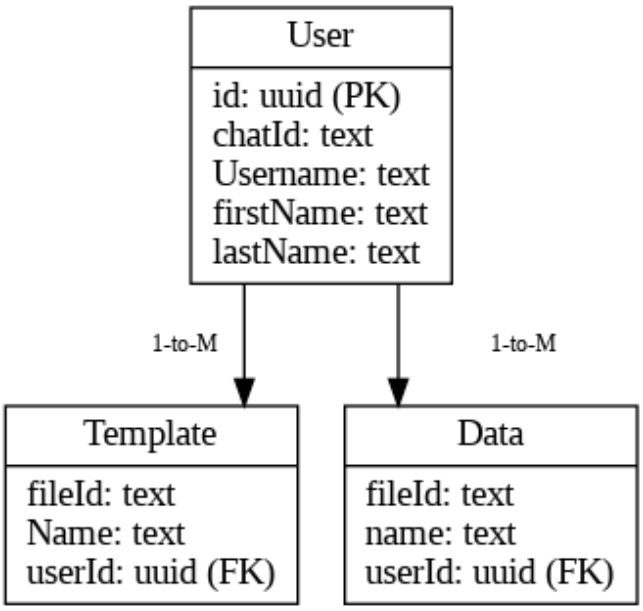


Рисунок 1.11. ER-діаграма сутностей і зв’язків у БД віртуального асистента

Розробка бази даних для віртуального асистента передбачає наступні ключові етапи:

1. Визначення сутностей. На основі функціональних вимог були визначені три основні сутності: користувачі, шаблони і дані. Це дозволило сформувати базову структуру для зберігання інформації;

2. Проектування таблиць і встановлення зв’язків. Для кожної сутності створено окрему таблицю з відповідними полями та ключами. Зовнішні ключі забезпечують логічні зв’язки між таблицями, що сприяє цілісності даних;
3. Розробка ER-діаграми. За допомогою діаграми сутностей і зв’язків були наочно представлені зв’язки між таблицями. Це дозволяє забезпечити зрозумілість структури даних для подальшої розробки і тестування системи;
4. Оптимізація і тестування. Розроблену структуру бази даних перевіряють на відповідність вимогам про швидкість та ефективність обробки даних. Застосовуються засоби моніторингу продуктивності, щоб забезпечити надійність і масштабованість системи.

1.9.2 Розробка діаграми станів віртуального асистента

Цей підрозділ присвячено створенню діаграми переходів станів, яка відображає логіку керування процесами обробки запитів користувачів через чат-бота для формування текстових документів. Основною метою є візуалізація послідовності змін станів під час взаємодії користувача з системою, а також визначення способів відновлення операцій при можливих збоях.

Оскільки в процесі функціонування застосунку важливо забезпечити збереження поточного контексту взаємодії, було прийнято рішення використовувати зовнішнє сховище для зберігання інформації про стан користувача. Для цього вибрано Redis — високошвидкісну розподілену систему зберігання даних типу "ключ-значення", що дозволяє гарантувати збереження інформації навіть у разі аварійного завершення роботи застосунку.

Для побудови діаграми визначено ключові стани, які характеризують життєвий цикл взаємодії:

- Standby (Спокій):Базовий стан, коли система знаходиться в режимі очікування нових запитів від користувача;
- Data_Input (Очікування даних): Стан, у якому бот отримує файли з даними, які будуть використані для заповнення шаблонів документів;
- Template_Input (Очікування шаблону): Стан, коли система чекає на файл-шаблон, який визначає формат та структуру майбутнього документа;

- File_Saving (Обробка збереження): Стан, що активується, коли отримані дані та шаблон обробляються для збереження файлу або формування кінцевого документу;
- File_Deleting (Обробка видалення): Стан, коли система виконує видалення файлу згідно з командою користувача.

Переходи між цими станами організовують ланцюжок, згідно з яким після отримання файлу з даними наступним кроком є очікування шаблону. Далі, в залежності від отриманої команди, здійснюється або обробка збереження файлу, або видалення ресурсу. При виникненні таймауту або скасування операції стан повертається до Standby, що гарантує відсутність «завислих» проміжних станів.

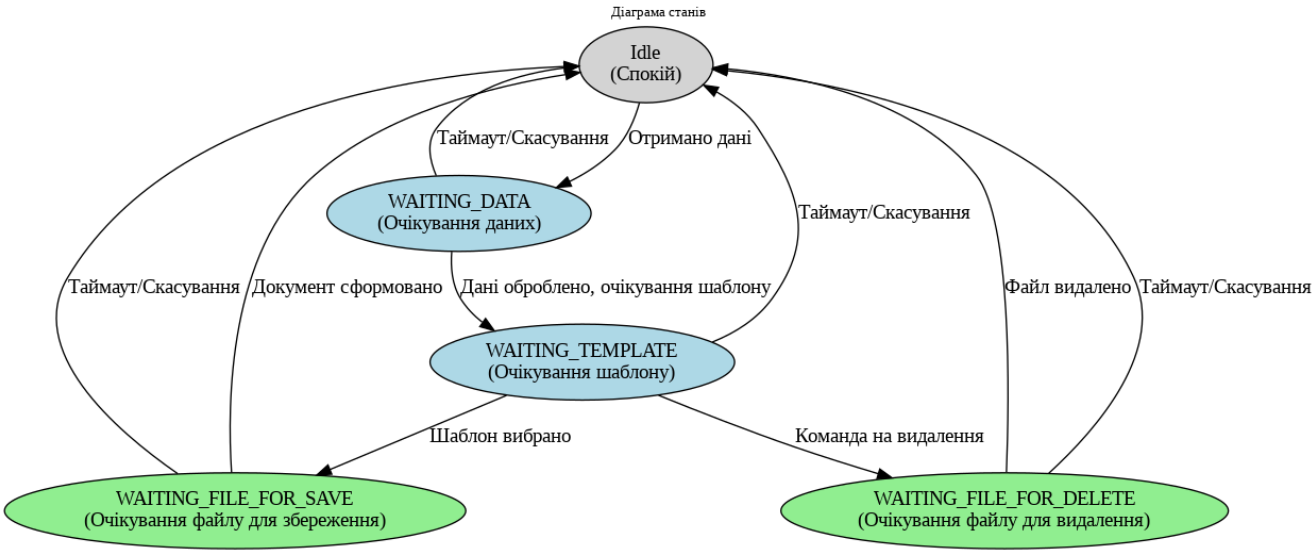


Рисунок 1.12. Діаграма станів віртуального асистента

Діаграму станів віртуального асистента наведено на рис.1.12:

1. Standby (Спокій): Система перебуває у цьому стані, коли не відбувається активна взаємодія з користувачем. При отриманні файлів з даними відбувається перехід до стану Data_Input;
2. Data_Input (Очікування даних): Цей стан відображає етап прийому файлів, які містять дані для подальшого заповнення шаблонів. Після успішної обробки запиту здійснюється перехід до Template_Input або повернення на Standby при скасуванні;
3. Template_Input (Очікування шаблону): У цьому стані очікується файл-шаблон. Від вибору шаблону залежить наступний перехід: у разі вибору —

виконання збереження (File_Saving), або, якщо отримано команду видалення — перехід у стан File_Deleting;

4. File_Saving (Обробка збереження): На цьому етапі проводиться фінальна обробка отриманих даних і шаблону для формування документу. Після завершення операції відбувається повернення до стану Standby;
5. File_Deleting (Обробка видалення): У даному стані система виконує операцію видалення файлу. Після успішного видалення ресурс повертається у базовий стан Standby.

Застосування зовнішнього сховища Redis для збереження поточних станів забезпечує, що інформація про стан конкретного користувача не втрачатиметься після збоїв або перезапуску застосунку, що критично для систем із великою кількістю запитів.

Введення діаграми переходів станів із використанням інструментів візуалізації дозволяє чітко описати логіку виконання послідовних операцій у системі. Це сприяє підвищенню стабільності, надійності та масштабованості процесів обробки запитів, що є ключовим для роботи віртуального асистента.

1.9.3 Забезпечення безпеки при формуванні текстових документів

При розробці застосунку, що генерує текстові документи на основі даних користувачів, критично важливо впровадити механізми захисту для гарантування цілісності, конфіденційності та аутентичності кінцевих документів. Одним із ефективних підходів до забезпечення безпеки є впровадження цифрового підписування сформованого документа з використанням алгоритму HMAC (Hash-based Message Authentication Code). Цей метод дозволяє виявити будь-які несанкціоновані зміни в структурі чи вмісті документа. Основні етапи реалізації механізму захисту (рис.1.13):

1. Валідація вхідних даних та санітизація. До формування документа проводиться ретельна перевірка і очищення даних, отриманих від користувача, для усунення можливості SQL-ін’єкцій, XSS-атак та інших потенційних загроз;
2. Формування документа. На основі перевірених даних і шаблону формується

кінцевий текстовий документ. Загальний формат і структура документа встановлюються заздалегідь, що дозволяє ізолювати критичну логіку від зовнішніх впливів;

3. Обчислення цифрового підпису (HMAC). Після формування документа проводиться обчислення контрольного підпису за допомогою криптографічної функції HMAC із застосуванням секретного ключа. Результуючий хеш додається до документа як цифровий підпис або зберігається окремо;
4. Перевірка та валідація. При отриманні документа для подальшої обробки або передачі кінцевому користувачу розраховується контрольне значення HMAC і порівнюється з отриманим підписом. Якщо значення збігаються, документ вважається незмінним.

Алгоритм захисту документу можна окреслити наступним чином:

1. Отримання та санітизація даних із джерела;
2. Формування текстового документу на основі шаблону з включенням отриманих даних.
3. Обчислення HMAC-контрольної суми сформованого документу за допомогою секретного ключа;
4. Попереднє збереження або приєднання цифрового підпису до документу;
5. При подальшій обробці перевірка цифрового підпису для гарантування цілісності документа;
6. У разі невідповідності – відхилення подальшої обробки та генерація сповіщення про спробу несанкціонованої модифікації.

Наведено код функцій на TypeScript для обчислення та перевірки цифрового підпису документа з використанням модуля crypto:

```
import * as crypto from 'crypto';
// Секретний ключ для обчислення цифрового підпису.
// Ключ має зберігатися конфіденційно (наприклад, у змінних середовища).
const SECRET_KEY: string =
process.env.DOC_SECRET_KEY || 'super_secret_key';
/**
 * Функція для генерації HMAC-підпису документу.
 * @param documentText Вміст сформованого документа.
```

```
 * @returns HMAC-підпис у вигляді рядка в шістнадцятковому форматі.
 */
function generateDocumentHMAC(documentText: string): string {
  const hmac = crypto.createHmac('sha256', SECRET_KEY);
  hmac.update(documentText);
  return hmac.digest('hex');
}
/**
 * Функція для перевірки цифрового підпису документу.
 * @param documentText Вміст сформованого документа.
 * @param signature Отриманий підпис, що додається до документу.
 * @returns true, якщо підпис відповідає, або false у протилежному випадку.
 */
function verifyDocumentHMAC(documentText: string, signature: string): boolean
{
  const expectedSignature = generateDocumentHMAC(documentText);
  return expectedSignature === signature;
}
// Демонстрація роботи функцій:
const sampleDocument: string =
"Згенерований текстовий документ з конфіденційною інформацією.";
const documentSignature: string = generateDocumentHMAC(sampleDocument);
console.log("Цифровий підпис документа:", documentSignature);
const isValid: boolean =
verifyDocumentHMAC(sampleDocument, documentSignature);
console.log("Документ не було змінено:", isValid);
```

Додаткові заходи безпеки:

- Шифрування даних під час передачі та збереження. Для забезпечення конфіденційності текстових документів може бути використане симетричне або асиметричне шифрування під час збереження документів або передачі даних по мережі;
- Контроль доступу і аудиту. Реалізація механізмів автентифікації та авторизації дозволяє обмежити доступ до функцій формування документів. Журналювання всіх дій із документами допомагає відслідковувати спроби несанкціонованого доступу або модифікації;
- Санітизація вихідних даних. Перед збереженням у базі даних або розповсюдженням крайній текстовий документ проходить процес санітизації для запобігання виводу шкідливого коду.

Рисунок 1.13. Блок-схема алгоритму забезпечення безпеки при формуванні текстових документів

Використання цифрового підпису на основі HMAC разом із засобами шифрування і контрольними механізмами дозволяє ефективно забезпечувати безпеку при формуванні текстових документів. Реалізація зазначеного механізму сприяє виявленню спроб несанкціонованої модифікації документів і гарантує збереження критично важливої інформації, що є запорукою стабільного та безпечного функціонування застосунку.

1.9.4 Реалізація команд віртуального асистента для формування текстових документів

Цей підрозділ описує впровадження команд, які відповідальні за отримання запитів від користувача і формування текстових документів за допомогою чат-бота. Команди організовуються таким чином, щоб забезпечити чітку обробку сесій, управління файлами з даними та шаблонами, а також можливість модифікації або видалення збережених ресурсів. Основні команди охоплюють ініціалізацію сесії (/start, /help), збереження файлів (/save_file), видалення файлів (/delete_data, /delete_template), автоматичне заповнення шаблонів (/fill_template), а також допоміжні команди для повторного використання раніше збережених шаблонів та даних (/use_template, /use_data). Основні функції команд:

1. Команди ініціалізації (/start, /help): При отриманні цих команд бот надсилає до кінцевого користувача відповідне привітальне повідомлення та справджує список доступних команд (рис.1.14). З одночасною ініціалізацією сесії інформація зберігається в базі даних. У випадку повторного введення команди забезпечується перевірка сесії, що дозволяє уникнути дублювання або несумісності даних;
2. Команда збереження файлів (/save_file): Після отримання команди /save_file бот встановлює стан у зовнішньому сховищі (Redis) як «очікування файлу для збереження» (рис.1.15). При надходженні файлу з месенджера, його унікальний ідентифікатор та ім'я парсяться, після чого відповідна інформація зберігається у базі даних, а стан сесії скидається. Код на TypeScript є таким:

```
if (command === '/save_file') {
   setState(userId, 'WAITING_FILE_SAVE');
   sendMessage(userId, 'Будь ласка, надішліть файл для збереження.');
```

```
}
onFileReceived(file) {
    if (getState(userId) === 'WAITING_FILE_SAVE') {
        const fileName = extractFileName(file);
        saveFileToDB(userId, file.file_id, fileName);
        clearState(userId);
        sendMessage(userId, 'Файл успішно збережено.');
    }
}
```

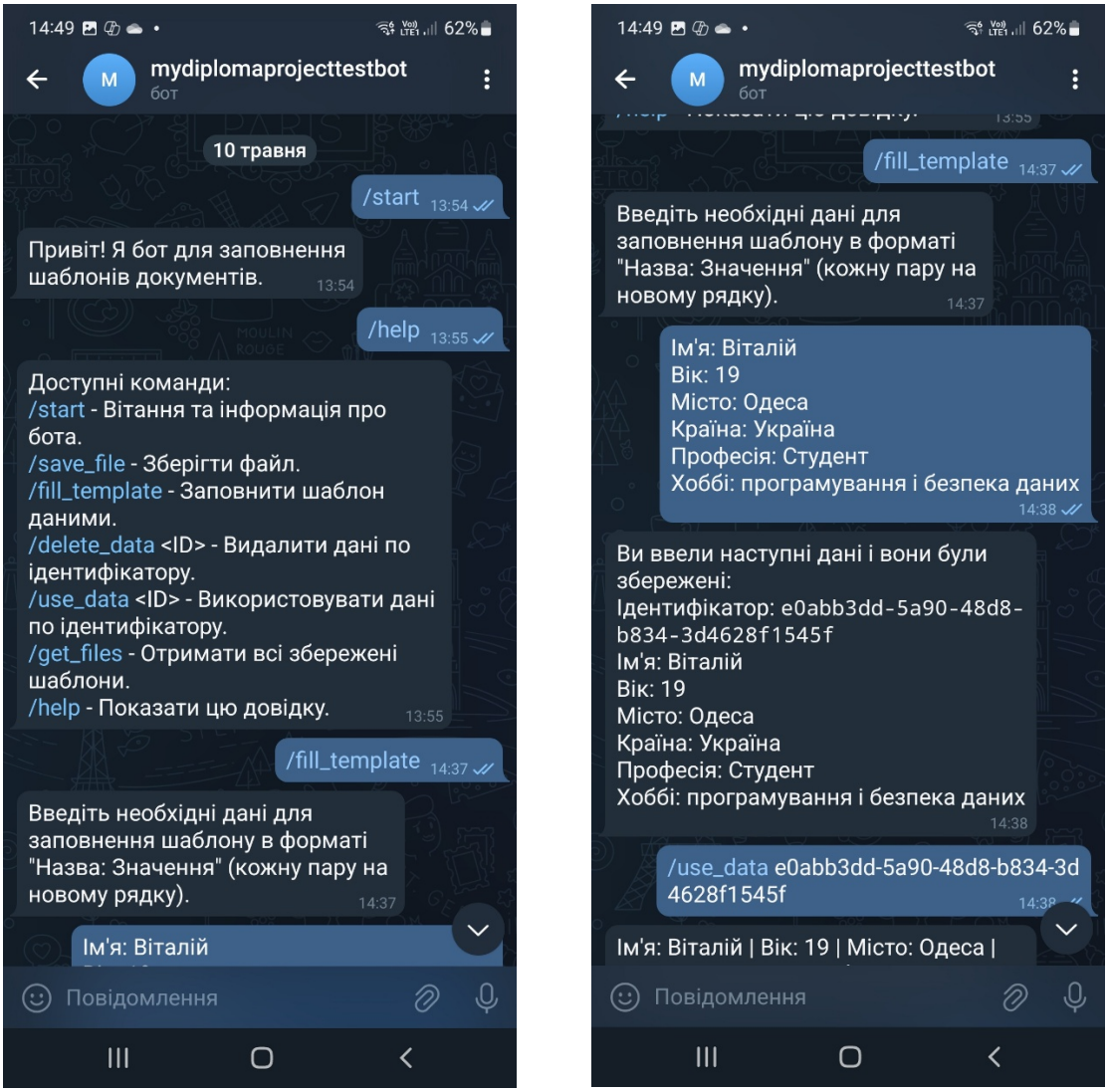


Рисунок 1.14. Використання команд /start, /help, /fill_template, /use_data віртуального асистента

3. Команди видалення (/delete_data, /delete_template): Для видалення зайвих файлів реалізовано команди, які запускають процес видалення записів з бази даних за заданим унікальним ідентифікатором файлу (рис.1.15). Процедура включає перевірку прав доступу та підтвердження операції, після чого відповідні дані видаляються. Код на TypeScript є таким:

```
if (command === '/delete_data') {
   const fileId = extractFileIdFromCommand(message);
   deleteDataRecord(userId, fileId);
   sendMessage(userId, 'Запис з даними видалено.');
}
if (command === '/delete_template') {
   const fileId = extractFileIdFromCommand(message);
   deleteTemplateRecord(userId, fileId);
   sendMessage(userId, 'Шаблон видалено.');
}
```

4. Команда заповнення шаблону (/fill_template): Основний процес роботи бота полягає у формуванні текстового документа на основі користувацьких даних та обраного шаблону (рис.1.14). При активації команди /fill_template запускається послідовність дій:
   - Введення даних користувачем (чи отримання збережених даних);
   - Вибір шаблону з доступних варіантів або підтвердження завантаженого файлу-шаблону (рис.1.16);
   - Перевірка коректності введених даних за умовами шаблону.
   - Генерація фінального документа, який формується згідно заданої структури і повертається користувачу для перегляду або подальшого використання (рис.1.17).

   Код на TypeScript є таким:

```
if (command === '/fill_template') {
   setState(userId, 'WAITING_TEMPLATE_FILL');
   sendMessage(userId, 'Введіть дані для заповнення шаблону або
використовуйте команду /use_data.');
}
onDataInput(data) {
   if (getState(userId) === 'WAITING_TEMPLATE_FILL') {
      if (validateDataAgainstTemplate(data)) {
         const document = fillTemplate(data, selectedTemplate);
         sendDocumentToUser(userId, document);
         clearState(userId);
      } else {
         sendMessage(userId, 'Невірний формат даних. Спробуйте знову.');
      }
   }
}
```

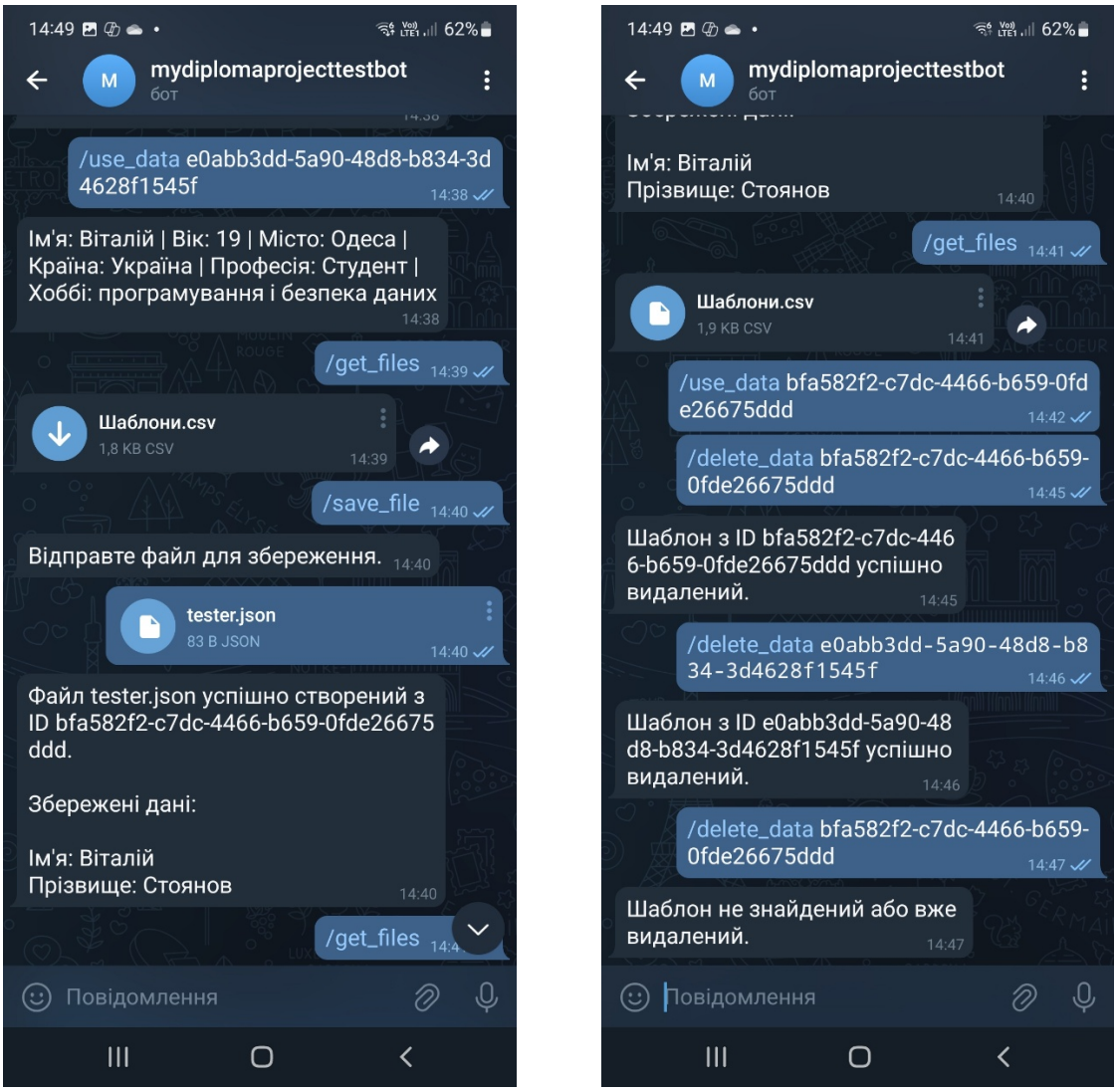


Рисунок 1.15. Використання команд /save_file, /get_files, /use_data, /delete_data віртуального асистента

5. Допоміжні команди (/get_files, /use_data): Ці команди спрямовані на полегшення процесу заповнення шаблону за рахунок повторного використання раніше збережених файлів з даними або шаблонів (рис.1.15). Вони активуються у контексті процесу /fill_template і дозволяють отримати список існуючих ресурсів, замість того щоб заставляти користувача завантажувати файли знову.

   Код на TypeScript є таким:

```
if (command === '/get_files') {
    const templates = getUserTemplates(userId);
    sendMessage(userId, `Доступні шаблони: ${templates.join(', ')}`);
}
if (command === '/use_data') {
    const dataFiles = getUserDataFiles(userId);
```

```
    sendMessage(userId, `Доступні записи даних: ${dataFiles.join(', ')}`);
}
```

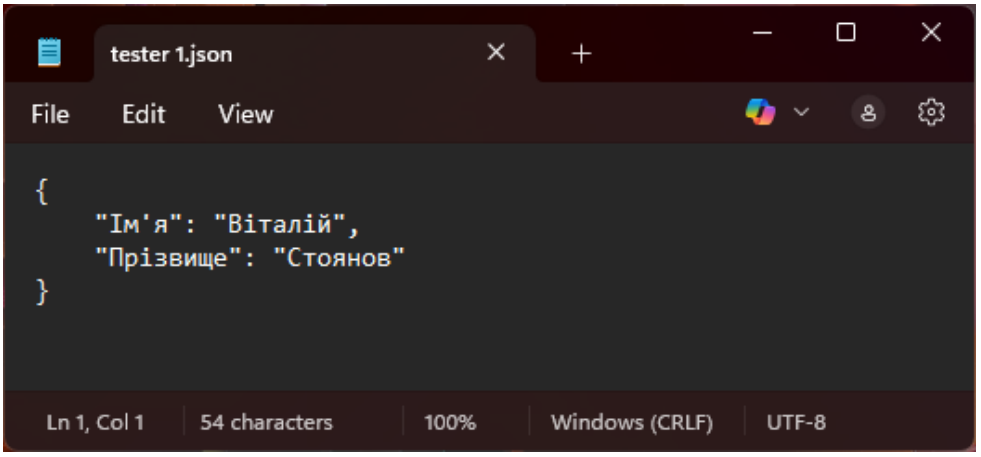


Рисунок 1.16. Вміст завантаженого json-файлу

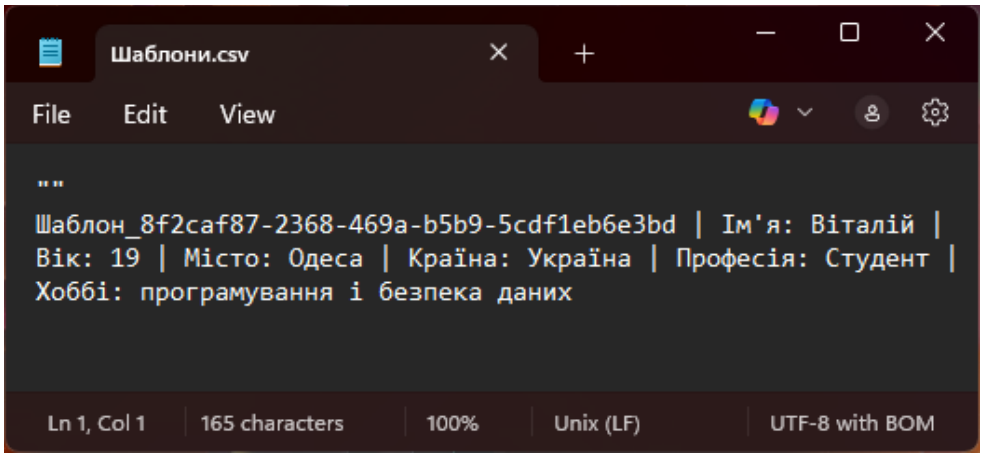


Рисунок 1.17. Вміст сформованого csv-файлу
за допомогою віртуального асистента

Реалізація команд здійснюється із застосуванням обраного фреймворку серверного застосунку (наприклад, NestJS), який забезпечує обробку HTTP-запитів і взаємодію з зовнішніми сервісами (Telegram API, Redis, база даних). Кожна команда обробляється окремим контролером або сервісом, що дозволяє чітко ізолювати логіку бізнес-процесів та забезпечити масштабованість і надійність системи. Захист стейт-машини, а також можливість повторного використання файлів, сприяють оптимізації роботи чат-бота у великих виробничих середовищах. Таким чином, впровадження команд із чітким управлінням станами та визначенням бізнес-процесів забезпечує швидку, надійну та безпечну роботу системи формування текстових документів, що відповідає сучасним вимогам до інтерактивних чат-ботів.

Фрагменти коду мовою Typescript програмної реалізації віртуального асистента для безпечного формування текстових документів демонструють основні аспекти реалізації Telegram-бота: прийом запитів через контролери, обробку команд сервісами, управління сесіями за допомогою Redis, роботу з базою даних для збереження файлів і шаблонів, а також інтеграцію з Telegram API для надсилання повідомлень, заповнення шаблонів (Додаток А). TelegramBotController приймає HTTP-запити від Telegram (наприклад, через webhook) і направляє обробку в сервіс. Сервіс TelegramService реалізовує логіку обробки основних команд. Тут наведено приклади для команд старту, збереження файлів, заповнення шаблону та використання збережених ресурсів. Сервіс SessionService призначений для управління сеансами користувачів із використанням Redis. Код демонструє, як встановлювати, отримувати та очищувати стан користувача. Сервіс FileService призначений для збереження, видалення та отримання інформації про файли, отримані від користувача. Сервіс TemplateService призначений для управління шаблонами, їх отримання та заповнення. Універсальна функція Telegram Utility призначена для відправлення повідомлень через Telegram API. Цей код використовує бібліотеку axios. Метод fillTemplate приймає шаблон і дані користувача, а потім замінює всі плейсхолдери ({name}, {age} тощо) відповідними значеннями з об'єкта data.

2 ЕКОНОМІЧНИЙ РОЗДІЛ

2.1 Резюме

Результатом даної роботи є створення чат-бота у Telegram, який забезпечуватиме інтерактивну взаємодію з користувачем для заповнення шаблонів документів, а також реалізацію механізмів безпечного збереження, обробки і маніпулювання файлами, що містять як дані, так і самі шаблони. Проект передбачає підтримку роботи з файлами різних форматів — зокрема, .csv, .xlsx, .json та .xml, що дозволить забезпечити гнучкість у роботі з даними та адаптацію до різних типів вихідних документів. Розробка віртуального асистента для безпечного формування текстових документів є важливим кроком у напрямку підвищення якості взаємодії користувача з цифровими сервісами.

Ефективність програмного забезпечення залежить не лише від його якості, а й від результативності процесу його створення. Якість ПЗ оцінюється за кількома критеріями: з погляду користувача, раціонального використання ресурсів і відповідності заданим вимогам. З позиції користувача також враховуються витрати на розробку, зокрема трудові та фінансові. У цьому розділі представлено розрахунок вартості створеного програмного продукту.

2.2 Визначення трудомісткості розробки програмного забезпечення

Тривалість створення програмного продукту визначається його масштабом, рівнем трудомісткості, кваліфікацією виконавців та встановленими ринковими термінами. Використовуючи метод структурної аналогії та аналіз відповідних каталогів аналогічного ПЗ, можна встановити обсяг програмного засобу, що виражається у тисячах умовних машинних команд для аналога.

Таблиця 2.1. Каталог аналогів

Найменування ПП	**Обсяг функції ПП – V_o, усл. машинних командах.**
1. ПП автоматизації засобів по каталогу	680 – 7000
2. ПП автоматизованих розрахунків	1300 – 8600
3. ПП оптимізації розрахунків	1300 – 4200

У таблиці 2.1 представлені аналоги програмного забезпечення, функції яких, у більшому або меншому ступені, виконує розроблений програмний продукт. Для нашого варіанта виділено сірим кольором.

Вибравши аналог ПЗ, що містить Vo в умовних машинних командах, трудомісткості визначати на основі табл.2.2

Таблиця.2.2. Норма часу

Обсяг ПЗ, тис.умов.машинних команд	**Норма часу, люд/год**
1.00	229
2.00	244
3.00	262

На підставі отриманого значення, по довіднику, визначається укрупнена норма часу на розробку аналога програмного забезпечення (коректується поправочним коефіцієнтом враховуючої умови розробки ПП, тобто в умовах комп'ютера, Кк=0,7÷0,8): Тар = 229 х 0,8 = 183,2 (люд/годин).

Трудомісткість програмного продукту визначається окремо для кожного етапу розробки, виходячи з показників трудомісткості відповідного аналога. При цьому враховують рівень складності розробки, ступінь інноваційності та частку використання стандартних модулів. Розрахунки проводяться згідно з наступними формулами:

$$T_{ТЗ} = T^a p \times L_1 \times K_H \quad (2.1)$$

$$T_{ТП} = T^a p \times L_2 \times K_H \quad (2.2)$$

$$T_{РП} = T^a p \times L_3 \times K_H \times K_T \quad (2.3)$$

Для розрахунку необхідні наступні коефіцієнти:

Li – питома вага і-го етапу розробки (див. табл. 2.3.);

Кн – поправочний коефіцієнт, що враховує ступінь новизни (див. табл. 2.4.);

Кт – поправочний коефіцієнт, що враховує ступінь використання в розробці типових програм (див. табл. 2.5)

Таблиця 2.3. Значення питомих коефіцієнтів трудомісткості стадії в загальній трудомісткості розробки ПП

Код стадії	**Ступінь новизни**		
	А	Б	В
ТЗ (L_1)	0,15	0,12	0,12
ТП (L_2)	0,16	0,15	0,11
РП (L_3)	0,55	0,58	0,61

Для нашого варіанта виділено сірим кольором.

Таблиця 2.4. Значення поправочного коефіцієнта, що враховує ступінь новизни

Код ступеня новизни	**Ступінь новизни**	**Значення $К_н$**
А	Принципово нові ПП	1,75 – 1,2
Б	ПП – розвиток визначеного параметричного ряду	1,0 – 0,8
В	ПП маючий аналог	0,7

Для нашого варіанта виділено сірим кольором.

Таблиця 2.5. Значення коефіцієнта ступеня використання в розробці типових програм

Ступінь охоплення реалізованих функцій розроблювального ПП типовими програмами, %	**Значення $К_т$**
60 і вище	0,6
40-60	0,7
20-40	0,8
До 20	0,9

Для нашого варіанта виділено сірим кольором.

Тепер розраховуємо трудомісткість по кожному етапу окремо:

Трудомісткість технічного завдання

$$T_{тз}=T^a * L_1 * К_н = 183,2 * 0,12 * 0,7 = 15,38 \text{ (люд/годин)} \quad (2.4)$$

Трудомісткість розробки технічного проекту

$$T_{тп}=T^a * L_2 * К_н = 183,2 * 0,11 * 0,7 = 14,11 \text{ (люд/годин)} \quad (2.5)$$

Трудомісткість розробки робочого проекту

$$T_{рп}=T^a * L_3 * К_н * К_т = 183,2 * 0,61 * 0,7 * 0,6 = 46,94 \text{ (люд/годин)} \quad (2.6)$$

Для подальших розрахунків визначили кількість папера, витраченого на кожен етап: технічне завдання $N_{тз}$= 1 (стр), розробка ТП $N_{тп}$=26 (стр), розробка робочого проекту $N_{рп}$=8 (стр), пояснювальна записка відповідно $N_{пз}$=23 (стр) Розрахунок зведений у таблицю 2.6.

Таблиця 2.6. Розрахунок трудомісткості ПП

Найменування етапів	**Розрахунок, годин**		
1.ТЗ	$T_{Ртз}$=15,38	$T_{кк}=0,7*N_{тз}$= 0,7*1=0,7	$T_{нк}=0,15*N_{тз}=0,15*1=0,15$
2.Розробка ТП	$T_{Ртп}$=14,11	$T_{кк}=0,7*N_{тп}=0,7*26=18,2$	$T_{нк}=0,15*N_{тп}=0,15*26=3,9$
3.Розробка РП	$T_{Ррп}$= 46,94	$T_{кк}=0,7*N_{рп}=0,7*8=5,6$	$T_{нк}=0,15*N_{рп}=0,15*8=1,2$
4.Розробка ПЗ	$T_{пз}=1,5*N_{пз}$= 1,5*23 =34,5	$T_{кк}=0,7*N_{тз}=0,7*23=16,1$	$T_{нк}=0,15*N_{пз}=0,15*23=3.45$
Усього, в т.ч.:	160,23		
- на розробку	$\Sigma T_р$=110,93		
- контроль керівника		$\Sigma T_{кк}$=40,6	
- нормоконтроль			$\Sigma T_{нк}$=8,7

2.3 Розрахунок ціни програмного продукту

Для оцінки вартості програмного продукту розглядаються основна заробітна плата виконавців, матеріальні витрати та загальні витрати на розробку ПЗ. Детальний розрахунок основної заробітної плати наведено у таблиці 2.7. Згідно зі статтею 8 «Закону про Державний бюджет України на 2025» з 1 січня 2025 року встановлено мінімальну місячну заробітну плату у розмірі 8000 гривень, а також мінімальну погодинну тарифну ставку – 48,00 грн.

Таблиця 2.7. Розрахунок основної заробітної плати виконавців

Найменування робіт	**Трудомісткість робіт, години**	**Погодинна тарифна ставка, грн.**	**Розрахунок, грн.**
1.Розробка ПП	110,93	48,00	5324,64
2.Контроль керівника	40,6	115,00	4669,00
3.Нормоконтроль	8,7	120,00	1044,00
Усього	-	-	ΣЗо= 11037,64

Зробимо розрахунок матеріальних витрат на розробку ПП. Розрахунок зведемо в таблицю 2.8.

Таблиця 2.8. Розрахунок матеріальних витрат на розробку ПЗ

Найменування матеріальних витрат	**Тип, модель**	**Кількість**	**Ціна одиниці, грн.**	**Вартість, грн.**
Папір	Лист А4	58	5.0	290,00
Разом	-	-	-	$В_{м\,і}$=290,00
Транспортно – заготівельні Витрати (10%)				$Втр\_з = 0{,}1 \times Вм1$ = 0,1*280=28,0
Усього				$В_м=В_{мі}+В_{тр\_з=}$ 308,00

На підставі отриманих даних по окремих статтях витрат складена калькуляція планової собівартості в цілому ПП за формою, приведеною в таблиці 2.9.

Таблиця 2.9. Розрахунок статей витрат планової собівартості

Стаття витрат	**Значення, грн.**	**Формула розрахунку**
1. Матеріали	319,00	$В_м$ (див. табл. 2.8.)
2. Основна заробітна плата	11037,64	$З_о$ (див. табл. 2.7.)
3.Додаткова заробітна плата	1103,76	$Зд = 0{,}1 \times Зо$ = 11037,64*0,1
4.Відрахування до єдиного фонду соціального внеску	2671,11	$Вє.с.в. = 0{,}22 \times (Зо + Зд)$=0,22* (11037,64+1103,76)
5. Накладні витрати	4415,06	$Внак. = 0{,}4 \times Зо$ =0,4 * 11037,64
6. Повна собівартість	19546,57	$С_{пов}$= $В_м$ +$З_о$ + $З_д$ +Вє.с.в. + Внак.= 319,00+11037,64+1103,76+2671,11 +4415,06

Розмір прибутку, що включається в ціну, визначаємо по наступній формулі:

$$П=(С_{пов}*Р)/100=(19546,57*10)/100=1954,66 \text{ грн} \qquad (2.7)$$

Де р – плановий рівень рентабельності (10-15%).

Оптова ціна (кошторисна вартість) визначається по формулі:

$$Ц_о =С_{пов}+П = 19546,57+1954,66 = 21501,23 \text{ грн;} \qquad (2.8)$$

Виходячи з отриманих даних, ціна реалізації розробленого програмного забезпечення становитиме:

$$Ц_р=Ц_о+ПДВ = 21501,23 +21501,23 *0.2 = 25801,48 \text{ грн;} \qquad (2.9)$$

3 РОЗДІЛ ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ

Сучасні комп’ютерні технології відкривають широкі можливості для автоматизації рутинних процесів, оптимізації обробки даних та швидкого доступу до інформаційних ресурсів. Вони сприяють підвищенню продуктивності праці та ефективності управління, проте їхнє активне використання також створює певні виклики, зокрема збільшене навантаження на здоров’я користувачів через тривалий час роботи з комп’ютером.

У рамках даного дипломного проєкту розглядається оозробка віртуального асистента для безпечного формування текстових документів. Розроблюваний віртуальний асистент є аналітичним інструментом, заснованим на сучасних методах оцінки робочих середовищ. Він інтегрується в систему користувача, включаючи комп’ютерне обладнання та спеціалізоване програмне забезпечення, призначене для аналізу життєздатності, надійності та безпеки інформаційних систем. Особлива увага приділяється дотриманню стандартів безпеки праці, що гарантує комфортну та безпечну роботу користувача, навіть у умовах підвищеного навантаження. Асистент допомагає оптимізувати процес формування текстових документів, забезпечуючи їхню структурованість, точність і відповідність необхідним вимогам.

Таким чином, створення такої моделі дозволить не лише автоматизувати процес аналізу комп’ютерних систем і мереж, а й врахувати комплекс реальних факторів, що впливають на їхню стабільність та ефективність у практичному застосуванні.

3.1 Аналіз небезпечних і шкідливих факторів, що впливають на користувача ПК

До основних критеріїв забезпечення гігієни робочого середовища належать інтенсивність освітлення, температура повітря, вологість, рівень шумового забруднення, ступінь вібраційного впливу, токсичність, загазованість, а також обмеження загальної м'язової активності.

3.2 Гігієнічні вимоги до виробничого середовища

Державні санітарні норми, зокрема ДСАНПіН 3.3.2.007-98 «Гігієнічні вимоги до організації роботи з візуальними дисплейними терміналами електронно-обчислювальних машин», спрямовані на запобігання негативного впливу шкідливих чинників, що супроводжують роботу з візуальними дисплейними терміналами, на здоров’я працівників.

3.2.1 Вимоги до приміщення

Розміщення робочих місць із використанням ВДТ, ЕОМ і ПЕОМ заборонено у підвальних приміщеннях та на цокольних поверхах. Для кімнат, призначених для роботи з візуальними дисплейними терміналами, рекомендується орієнтувати вікна у напрямку півночі або північного сходу. На вікнах повинні бути встановлені регульовані жалюзі або штори, що дозволяють їх повністю закривати для забезпечення оптимальних умов освітлення.

Планувальні рішення будівель і приміщень, де розташовано відеодисплейні термінали, мають відповідати вимогам ДСАНПіН 3.3.2.007-98. Для робочого місця програміста передбачено мінімальну площу не менше 6 кв. м та об’єм приміщення не менше 20 куб. м. Крім того, стіни приміщень повинні бути пофарбовані матовою фарбою, а в приміщеннях з ВДТ обов’язково мають бути передбачені зони для відпочинку та психологічного розвантаження.

3.2.2 Освітлення та шум

Для забезпечення належного освітлення приміщення, де працює програміст, застосовується комбінована система, що поєднує природне освітлення із додатковим штучним світлом. Загальне оздоблення простору виконується за допомогою газорозрядних ламп типу ЛД. Згідно з встановленими нормами, для робочого місця, на якому здійснюються високоточні операції (де мінімальний розмір об'єкта розрізнення становить 0,3–0,5 мм), необхідна освітленість рівномірно має досягати 300 лк. В цілому, ці вимоги щодо освітлення забезпечені.

У робочих приміщеннях основним джерелом шумового навантаження є звуки, що генеруються ПЕОМ. Крім того, значну частину шуму створюють

джерела електромагнітного походження – це коливання компонентів електромеханічних пристроїв під впливом змінних магнітних полів. До того ж, в приміщеннях виникає структурний шум, який випромінюють поверхні конструктивних елементів (стіни, перекриття, перегородки) у звуковому спектрі частот. Для зниження або усунення негативного впливу шуму доцільно ізолювати робочі зони, розташовуючи їх у частинах будівлі, що знаходяться в глибині та ведуть своїми вікнами у двір – таким чином мінімізується вплив міського шуму. Крім цього, необхідно регулярно перевіряти герметичність корпусів комп'ютерної техніки та своєчасно здійснювати заміну вентиляторів охолодження.

3.3 Вимоги до організації робочого місця працівника

Конструкція робочого місця користувача комп'ютера, з урахуванням розташування сидіння, засобів керування та засобу відображення інформації, розроблена згідно з антропометричними, фізіологічними та психологічними вимогами, а також відповідно до специфіки виконуваної роботи. Робоче меблеве обладнання повинно бути оснащене можливістю індивідуального регулювання, що дозволить адаптувати його під зріст кожного користувача й підтримувати оптимальну, зручну поставу. Робочий стіл рекомендовано обробляти матовим покриттям, що сприяє зменшенню небажаних відблисків. > > Розміщення дисплея організовано таким чином, щоб його верхня межа відповідала рівню очей, а відстань до екрану становила приблизно 70 см – що повністю входить у допустимий інтервал від 60 до 90 см. Частота мерехтіння екрану fмер дорівнює 100 Гц, що значно перевищує мінімальне рекомендоване значення у 70 Гц. Крім цього, робоче місце розташовано перпендикулярно до віконних прорізів, що дозволяє уникнути прямого та відбитого світлового мерехтіння від вікон та джерел штучного освітлення.

3.4 Мікроклімат

Показники мікроклімату, складу іонів у повітрі, а також рівень шкідливих речовин у робочих зонах, де використовуються ПК, мають відповідати вимогам ДСН 3.3.6.042-99 «Санітарні норми мікроклімату виробничих приміщень».

Для підтримки нормативних значень мікроклімату та забезпечення

оптимального співвідношення позитивних і негативних іонів слід передбачити установку пристроїв зволоження, штучної іонізації або кондиціювання повітря. Крім того, рівні інфрачервоного випромінювання не повинні перевищувати встановлених нормативних меж згідно з ГОСТ 12.1.005. Також вміст озону в робочій зоні не має перевищувати 0,1 мг/м³, оксидів азоту – 5 мг/м³, а концентрація пилу повинна залишатися в межах 4 мг/м³.

3.5 Електробезпека

Приміщення, де використовуються імпульсні джерела живлення згідно з ОНТП24-86 і ПУЕ-87, віднесено до категорії об'єктів, де ризик ураження персоналу електричним струмом не є підвищеним. Це пояснюється тим, що відносна вологість повітря не перевищує 75%, температура залишається нижчою за 35°С, а хімічно агресивні середовища відсутні. Електроживлення обладнання організовано від двофазної мережі з заземленою нейтраллю, при напрузі 220 В і частоті 50 Гц, із застосуванням автоматичних пристроїв токового захисту.

В приміщенні обов’язково має бути встановлена схема заземлення. Ураження електричним струмом може виникнути у випадках: 1) при контакті з відкритими струмоведучими елементами; 2) при торканні неструмоведучих частин обладнання, які, через порушення ізоляції або інші причини, опинилися під напругою.

Відповідно до вимог ГОСТ-12.2.007.0-75 устаткування (за винятком ЕОМ II класу) відноситься до I класу та оснащене робочою ізоляцією згідно з ГОСТ 12.1.009-76. Підключення обладнання здійснено згідно з нормативами ПБЕ та ПУЕ, тому додаткових заходів щодо електробезпеки не вимагається.

3.6 Пожежна безпека

Робоче приміщення, що відповідає вимогам ПБЕ та ОНТП 24–86 у сфері вибухово-пожежної безпеки, класифікується як об’єкт категорії «В».

Основними потенційними причинами виникнення пожежі в такому приміщенні є:

1. Коротке замикання електропроводки;

2. Використання побутових електрорадіоприладів;

3. Недотримання встановлених норм протипожежного захисту.

Відповідно до ПУЕ, для зниження ризику виникнення пожежі необхідно забезпечити комплекс заходів, зокрема: ретельну ізоляцію всіх струмоведучих проводів, що підключені до робочих місць, регулярний огляд та перевірку стану їх ізоляції, а також суворе дотримання норм безпечної експлуатації обладнання.

Для гасіння пожеж на робочому місці користувача ПК застосовують як вуглекислотні, так і порошкові вогнегасники.

– Вуглекислотні вогнегасники випускаються у варіанті ручних пристроїв (наприклад, ВВК-5);

– Порошкові вогнегасники представлені моделями ВП-2, ВП-5, ВП-10 та іншими

Рисунок 3.1. Засоби пожежогасіння

З метою своєчасного оповіщення, на дільниці необхідно встановити протипожежну сигналізацію. Проходи та запасні виходи повинні бути вільними. Пожежний щит повинен розміщуватись в доступному місці та містити первинні засоби пожежогасіння (вогнегасник, лопату, відро, простирадло, ящик з піском)

ВИСНОВКИ

В ході дипломного проектування було опрацьовано та застосовано широкий спектр сучасних технологій та інструментів задля реалізації функціональності автоматизованої роботи з шаблонами та їх заповненням з використанням наданих даних. Програмна реалізація базується на мові програмування Node.js, що забезпечує високу продуктивність і ефективну обробку запитів. Завдяки використанню фреймворку NestJS вдалося організувати зручний API для взаємодії з додатком, що дозволило чітко структурувати бізнес-логіку та забезпечити масштабованість системи. Для зберігання структурованих даних застосовано СУБД PostgreSQL, яка гарантує надійність збереження і швидкий доступ до інформації, а для кешування інформації про користувачів і оптимізації процесу управління станом — Redis, що забезпечує стабільність роботи навіть у разі підвищеного навантаження.

Окрім цього, інтеграція спеціалізованих бібліотек для роботи з різними типами файлів (CSV, JSON) дозволила ефективно обробляти дані та генерувати результати у потрібному форматі. Основним бізнес-процесом додатку є заповнення шаблонів, що забезпечує зручний і швидкий спосіб створення документів, повідомлень або інших вихідних форматів за заданими параметрами. Процес включає вибір шаблону, введення та валідацію даних, а також генерацію фінального документу, що відповідає встановленій структурі.

Для забезпечення безпеки формованих текстових документів впроваджено комплекс заходів, орієнтований на гарантування цілісності, автентичності та захисту від несанкціонованих модифікацій. Зокрема, застосовується механізм цифрового підписування на основі HMAC із використанням секретного ключа, що дозволяє перевіряти, що згенерований документ не був змінений під час передачі або зберігання. Крім того, здійснюється ретельна санітизація та валідація вхідних даних для запобігання потенційним атакам, таким як SQL-ін’єкції або XSS, що додатково підвищує рівень захисту системи. Ці заходи роблять можливим безпечне зберігання, передачу та подальшу обробку документів, що є важливим для підтримання високих стандартів конфіденційності та надійності роботи додатку.

ПЕРЕЛІК ВИКОРИСТАНИХ ІНФОРМАЦІЙНИХ ДЖЕРЕЛ

1. Петров, І.О. Основи програмування на мові JavaScript. – Київ: Видавництво «Наука», 2018. – 320 с.
2. Іваненко, В.П. Node.js: Розробка високопродуктивних мережевих додатків. – Харків: ХНУ ім. В.Н. Каразіна, 2019. – 256 с.
3. Сидоренко, О.М. Веб-додатки на Node.js: методичний посібник. – Львів: Львівський національний університет, 2020. – 210 с.
4. Коваль, С.В. Розробка REST API з використанням NestJS. – Київ: VavaPress, 2021. – 195 с.
5. Мельник, П.О. PostgreSQL: Основи роботи з реляційними базами даних. – Київ: Бізнес-Прес, 2018. – 180 с.
6. Гуменюк, Л.А. Redis у сучасних веб-додатках. – Одеса: Одеса-Інформ, 2019. – 160 с.
7. Корнієнко, Ю.Ф. Телеграм-боти: практичний посібник з розробки. – Київ: Ліра-К, 2021. – 230 с.
8. Руденко, О.О. Основи криптографії та захисту даних. – Львів: Світ Ідей, 2017. – 280 с.
9. Бойко, Т.М. Алгоритми цифрового підпису. – Київ: Київський науковий центр, 2020. – 150 с.
10. Семендя, Г.І. Основи інформаційної безпеки. – Харків: ХНУ, 2020. – 300 с.
11. Паращук, В.Л. Безпека веб-додатків: сучасні підходи та технології. – Київ: Інфра-М, 2018. – 210 с.
12. Офіційна документація Node.js [Електронний ресурс]. – Режим доступу: https://nodejs.org/en/docs/, дата звернення: 01.04.2025.
13. Офіційна документація PostgreSQL [Електронний ресурс]. – Режим доступу: https://www.postgresql.org/docs/, дата звернення: 02.04.2025.
14. Офіційна документація Redis [Електронний ресурс]. – Режим доступу: https://redis.io/documentation, дата звернення: 02.04.2025.
15. Офіційна документація Telegram API [Електронний ресурс]. – Режим доступу: https://core.telegram.org/bots/api, дата звернення: 02.04.2025.

Додаток А. Фрагменти ключового коду telegram-боту (Node.js і NestJS) мовою Typescript

TelegramBotController

```
import { Controller, Post, Body } from '@nestjs/common';
import { TelegramService } from './telegram.service';

@Controller('telegram')
export class TelegramBotController {
  constructor(private readonly telegramService: TelegramService) {}

  @Post('webhook')
  async handleWebhook(@Body() update: any): Promise<any> {
    const message = update.message;
    if (message) {
      const chatId = message.chat.id;
      const text = message.text || '';

      if (text.startsWith('/start') || text.startsWith('/help')) {
        return this.telegramService.handleStart(chatId, message);
      } else if (text.startsWith('/save_file')) {
        return this.telegramService.handleSaveFile(chatId, message);
      } else if (text.startsWith('/delete_data')) {
        return this.telegramService.handleDeleteData(chatId, message);
      } else if (text.startsWith('/fill_template')) {
        return this.telegramService.handleFillTemplate(chatId, message);
      } else if (text.startsWith('/use_template')) {
        return this.telegramService.handleUseTemplate(chatId, message);
      } else if (text.startsWith('/use_data')) {
        return this.telegramService.handleUseData(chatId, message);
      }
    }
    return { status: 'no action' };
  }
}
```

TelegramService

```
import { Injectable } from '@nestjs/common';
import { sendMessageToTelegram } from './telegram.util';
import { SessionService } from './session.service';
import { FileService } from './file.service';
import { TemplateService } from './template.service';

@Injectable()
export class TelegramService {
  constructor(
    private readonly sessionService: SessionService,
    private readonly fileService: FileService,
    private readonly templateService: TemplateService,
  ) {}

  async handleStart(chatId: number, message: any): Promise<any> {
    // Створення або оновлення сесії користувача
    await this.sessionService.createOrUpdateSession(chatId, message.from);
    const reply = `Вітаємо! Доступні команди: /start, /help, /save_file,
/delete_data, /fill_template, /use_template, /use_data.`;
    await sendMessageToTelegram(chatId, reply);
    return { status: 'ok' };
  }

  async handleSaveFile(chatId: number, message: any): Promise<any> {
```

```
    // Встановлення стану "очікування файлу для збереження" у Redis
    await this.sessionService.setState(chatId, 'WAITING_FILE_SAVE');
    await sendMessageToTelegram(chatId, 'Будь ласка, надішліть файл для
збереження.');
    return { status: 'waiting file' };
  }

  async handleDeleteData(chatId: number, message: any): Promise<any> {
    const fileId = this.extractFileId(message.text);
    await this.fileService.deleteDataRecord(chatId, fileId);
    await sendMessageToTelegram(chatId, 'Запис з даними видалено.');
    return { status: 'data deleted' };
  }

  async handleFillTemplate(chatId: number, message: any): Promise<any> {
    await this.sessionService.setState(chatId, 'WAITING_TEMPLATE_FILL');
    await sendMessageToTelegram(chatId, 'Введіть дані для заповнення шаблону
або використовуйте команду /use_data.');
    return { status: 'waiting template data' };
  }

  async handleUseTemplate(chatId: number, message: any): Promise<any> {
    const templates = await this.templateService.getUserTemplates(chatId);
    const reply = templates.length ? `Доступні шаблони: ${templates.join(',
')}` : 'Шаблони відсутні.';
    await sendMessageToTelegram(chatId, reply);
    return { status: 'templates sent' };
  }

  async handleUseData(chatId: number, message: any): Promise<any> {
    const dataFiles = await this.fileService.getUserDataFiles(chatId);
    const reply = dataFiles.length ? `Доступні записи даних:
${dataFiles.join(', ')}` : 'Дані відсутні.';
    await sendMessageToTelegram(chatId, reply);
    return { status: 'data files sent' };
  }

  private extractFileId(text: string): string {
    // команда має вигляд: "/delete_data <file_id>"
    const parts = text.split(' ');
    return parts[1] || '';
  }
}
```

SessionService

```
import { Injectable } from '@nestjs/common';
import * as Redis from 'ioredis';

@Injectable()
export class SessionService {
  private readonly redisClient: Redis.Redis;

  constructor() {
    this.redisClient = new Redis({
      host: process.env.REDIS_HOST || 'localhost',
      port: Number(process.env.REDIS_PORT) || 6379,
    });
  }

  async createOrUpdateSession(chatId: number, userData: any): Promise<void> {
    // Реалізація реєстрації сесії; збереження даних у базі (симульовано)
    console.log(`Оновлено сесію для ${chatId}: ${JSON.stringify(userData)}`);
  }
```

```
  async setState(chatId: number, state: string): Promise<void> {
    await this.redisClient.set(`session:${chatId}:state`, state);
  }

  async getState(chatId: number): Promise<string> {
    const state = await this.redisClient.get(`session:${chatId}:state`);
    return state || '';
  }

  async clearState(chatId: number): Promise<void> {
    await this.redisClient.del(`session:${chatId}:state`);
  }
}
```

FileService

```
import { Injectable } from '@nestjs/common';

@Injectable()
export class FileService {
  async saveFileRecord(chatId: number, fileId: string, fileName: string):
Promise<void> {
    // Збереження інформації про файл в базі даних (симуляція)
    console.log(`Файл збережено для чату ${chatId}: ID=${fileId},
Назва=${fileName}`);
  }

  async deleteDataRecord(chatId: number, fileId: string): Promise<void> {
    // Видалення запису з даними з бази (симуляція)
    console.log(`Видалено запис даних для чату ${chatId}: fileId=${fileId}`);
  }

  async getUserDataFiles(chatId: number): Promise<string[]> {
    // Отримання списку файлів користувача
    return ['file1', 'file2'];
  }
}
```

TemplateService

```
import { Injectable } from '@nestjs/common';

@Injectable()
export class TemplateService {
  async getUserTemplates(chatId: number): Promise<string[]> {
    // Отримання списку шаблонів користувача (симуляція)
    return ['template1', 'template2'];
  }

  fillTemplate(template: string, data: any): string {
    // Просте заповнення шаблону; в реальній реалізації застосовуються
шаблонізатори
    return `Документ, заповнений даними: ${JSON.stringify(data)} із шаблону:
${template}`;
  }
}
```

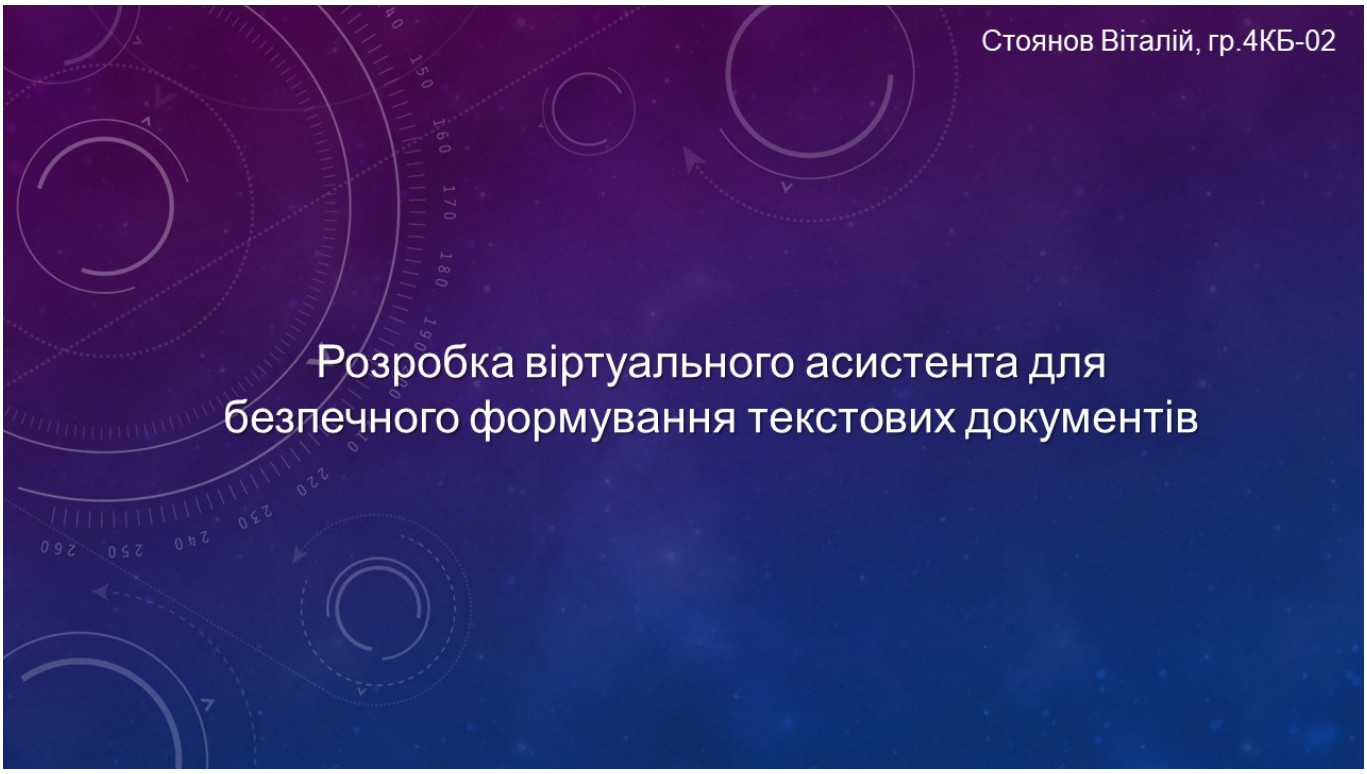
Telegram Utility

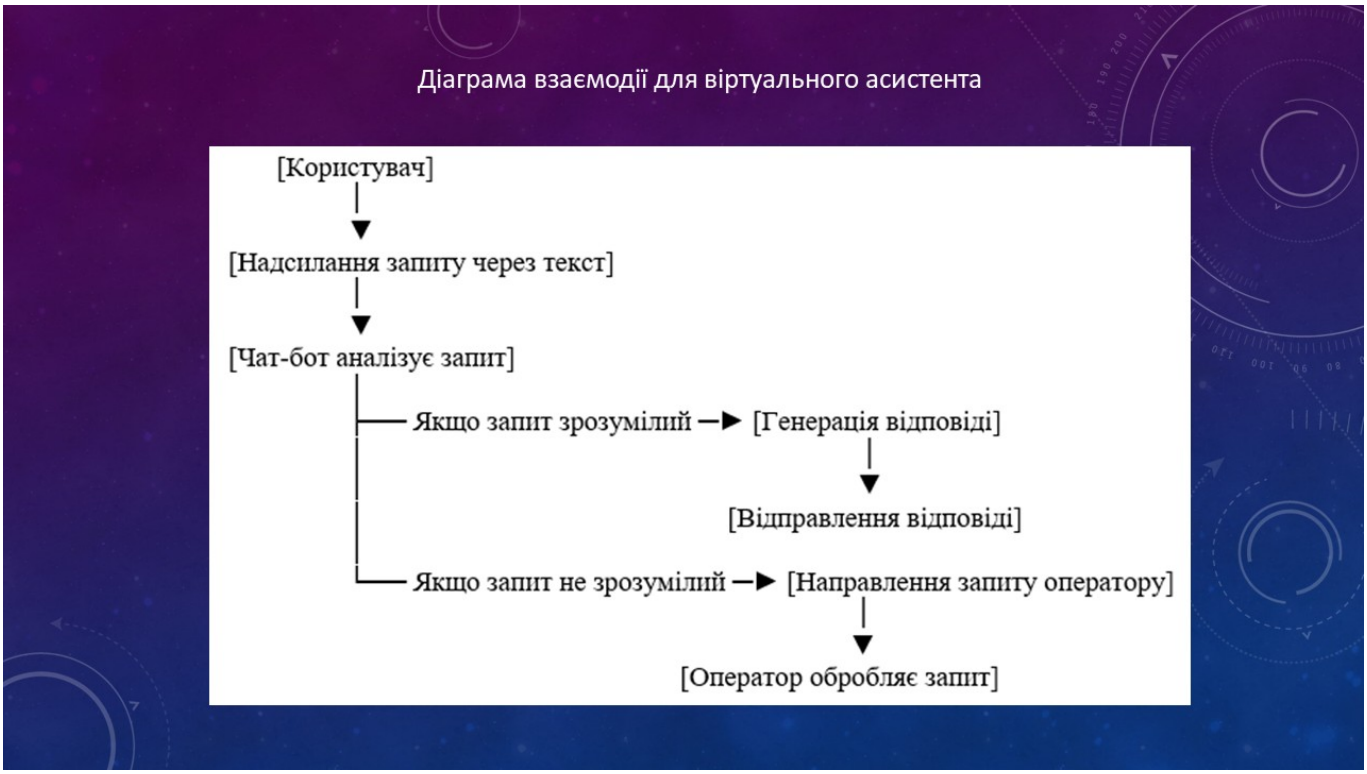
```
import axios from 'axios';

export async function sendMessageToTelegram(chatId: number, text: string):
Promise<void> {
  const botToken = process.env.TELEGRAM_BOT_TOKEN;
  const url = `https://api.telegram.org/bot${botToken}/sendMessage`;
  await axios.post(url, {
    chat_id: chatId,
    text: text,
  });
}
```

Процес заповнення текстового шаблону на основі даних користувача:

```
import { Injectable } from '@nestjs/common';

@Injectable()
export class DocumentService {

  /**
   * Заповнення шаблону даними користувача
   * @param template - Текстовий шаблон з плейсхолдерами
   * @param data - Об'єкт з даними користувача
   * @returns Заповнений шаблон
   */
  fillTemplate(template: string, data: Record<string, string>): string {
    let filledTemplate = template;

    // Замінюємо плейсхолдери у шаблоні відповідними значеннями користувача
    Object.keys(data).forEach((key) => {
      const placeholder = `{{${key}}}`;
      filledTemplate = filledTemplate.replace(new RegExp(placeholder, 'g'),
data[key]);
    });

    return filledTemplate;
  }
}

const documentService = new DocumentService();
const templateText = "Ім'я: {{name}}, Вік: {{age}}, Місто: {{city}}, Країна:
{{country}}.";
const userData = {
  name: "Віталій",
  age: "19",
  city: "Одеса",
  country: "Україна",
};

const filledDocument = documentService.fillTemplate(templateText, userData);
console.log(filledDocument);
// Виведення: "Ім'я: Віталій, Вік: 19, Місто: Одеса, Країна: Україна."
```

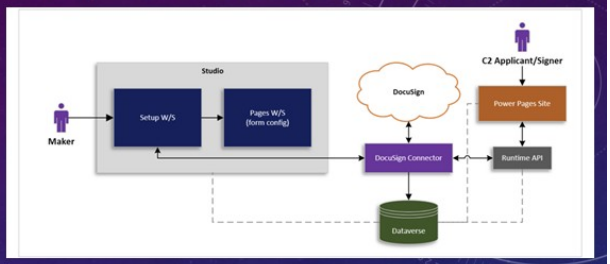
ДОДАТОК Б. Слайди мультимедійної презентації



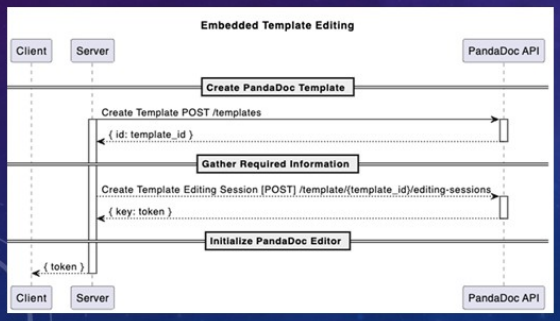


Відмінності між Webhooks та Long Polling

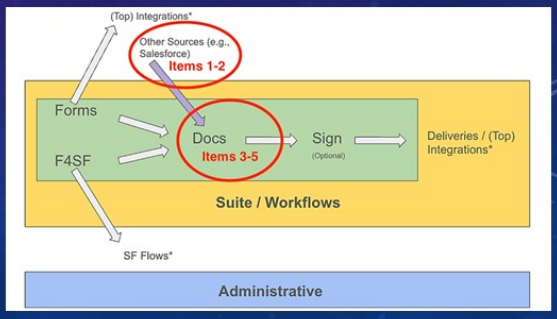
Формування текстових документів у DocuSign



Формування текстових документів у PandaDoc



Формування текстових документів у Formstack Documents



Діаграма процесу формування документів

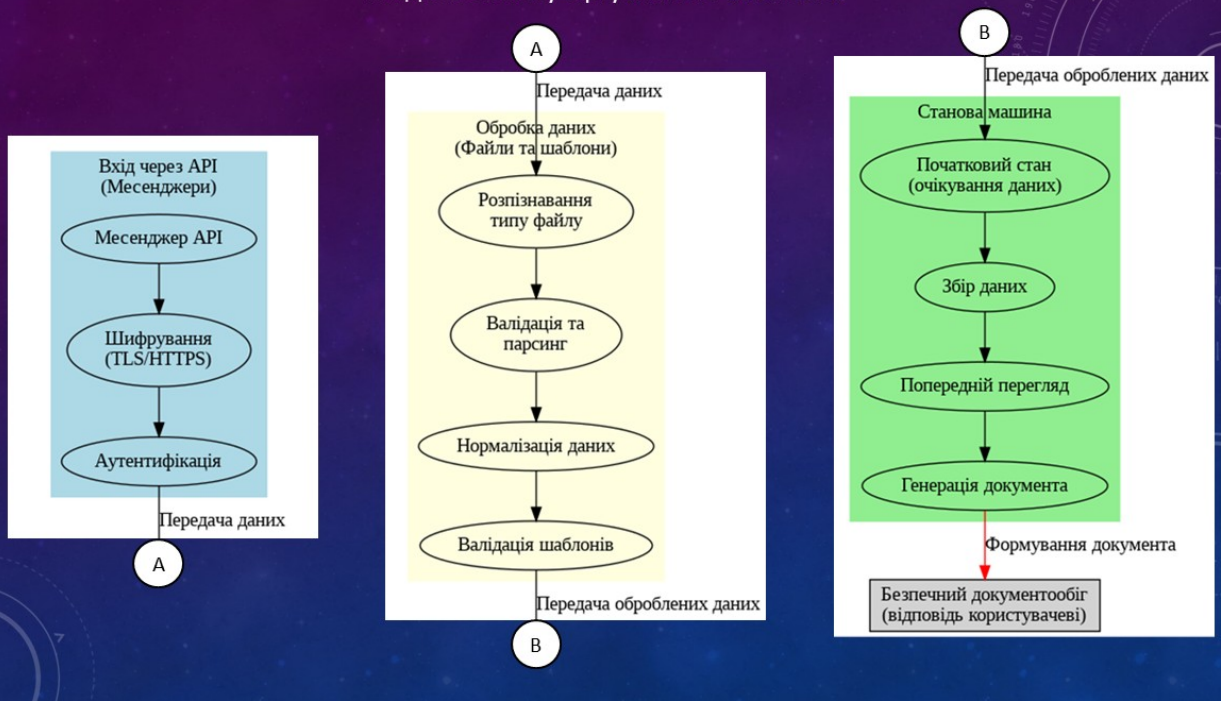
Приклад CSV файлу

	A	B
1	first_name	last_name
2	Віталій	Стоянов
3	Андрій	Татаринський
4	Дмитрій	Челак

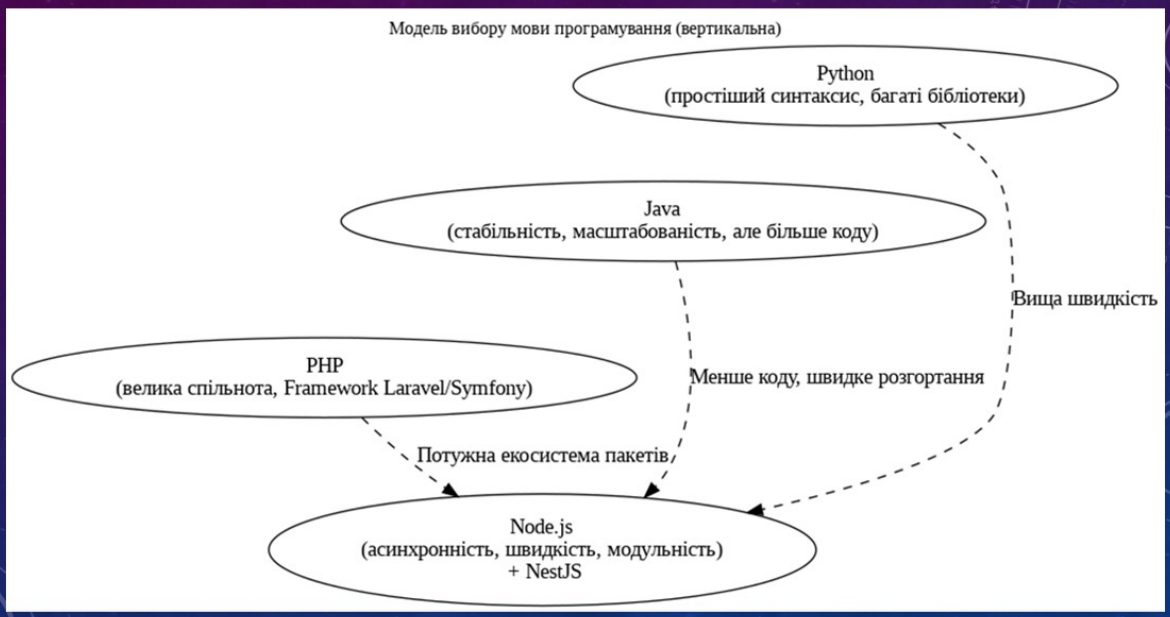
Приклад JSON файлу

```
[
  {
    "first_name": "Віталій",
    "last_name": "Стоянов"
  },
  {
    "first_name": "Андрій",
    "last_name": "Татаринський"
  },
  {
    "first_name": "Дмитрій",
    "last_name": "Челак"
  }
]
```

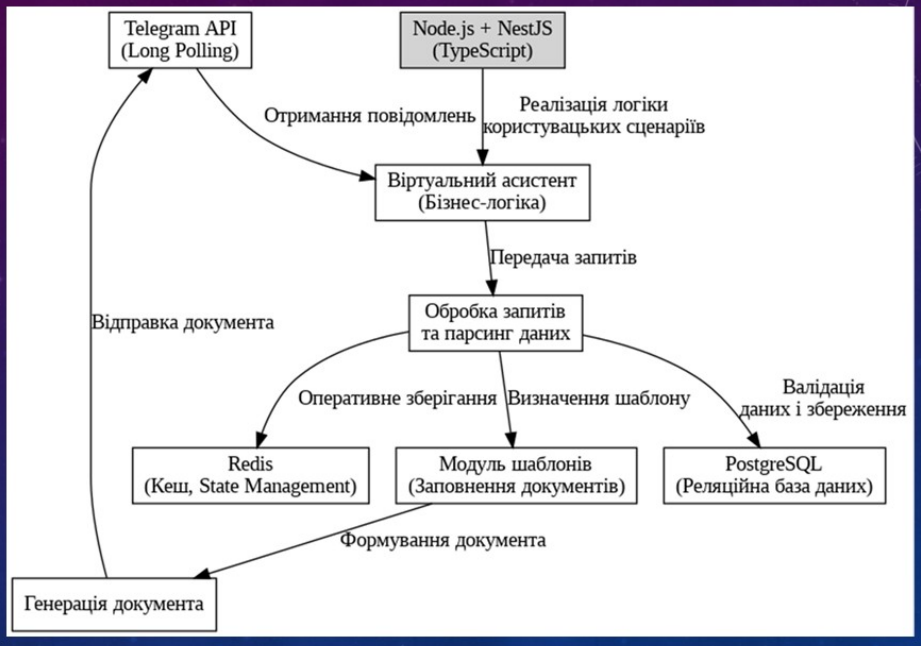
Модель захисту віртуального асистента



Модель вибору мови програмування для розробки віртуального асистента



Інтегрована архітектура віртуального асистента



Гексагональна архітектура віртуального асистента

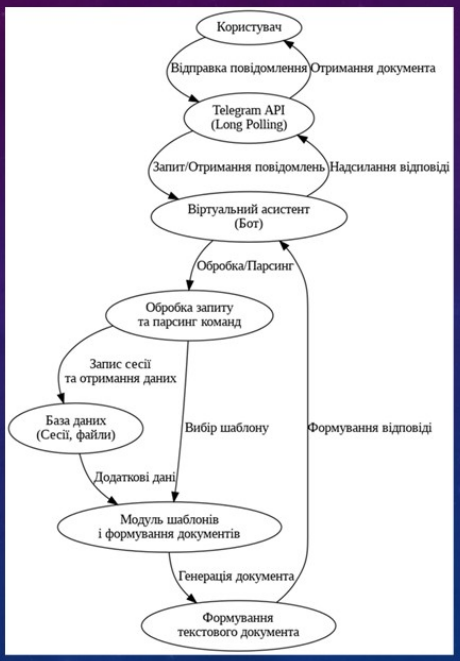
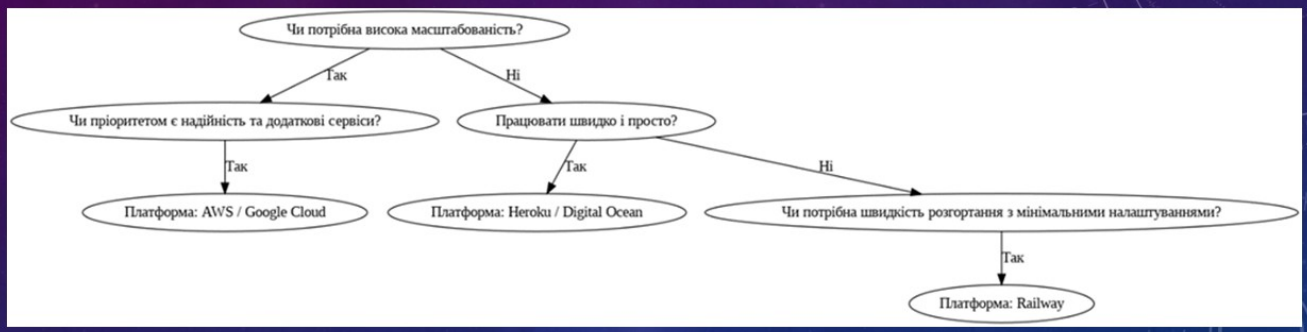


Схема вибору платформи хостингу проекту віртуального асистенту



Організація таблиці User (користувачі)

Назва поля	Тип	Опис	Індекс
ID користувача	uuid	Унікальний ідентифікатор (PK)	PK
chatId	text	Ідентифікатор чату в Telegram	
Username	text	Username користувача в Telegram	
firstName	text	Ім’я користувача в Telegram	
lastName	text	Прізвище користувача в Telegram	

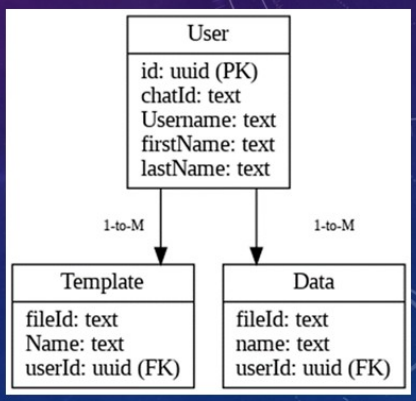
Організація таблиці Template (шаблони)

Назва поля	Тип	Опис	Індекс
fileId	text	ID файла в Telegram	–
Name	text	Назва файлу/шаблону	–
userId	uuid	Ідентифікатор користувача (FK)	FK

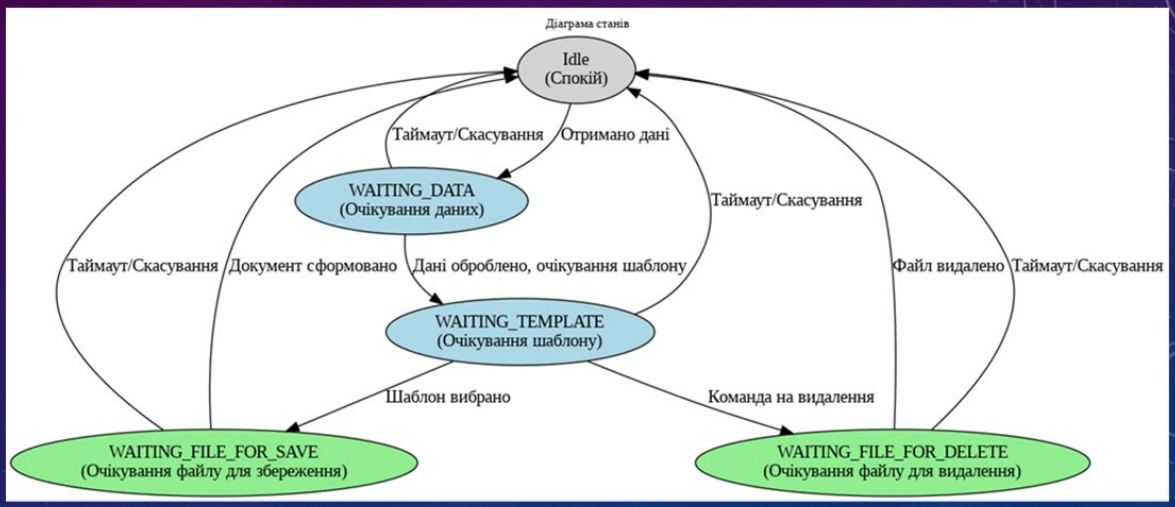
Організація таблиці Data (дані)

Назва поля	Тип	Опис	Індекс
fileId	text	ID файла в Telegram	–
Name	text	Назва файлу/шаблону	–
userId	uuid	Ідентифікатор користувача (FK)	FK

ER-діаграма сутностей і зв’язків у БД віртуального асистента

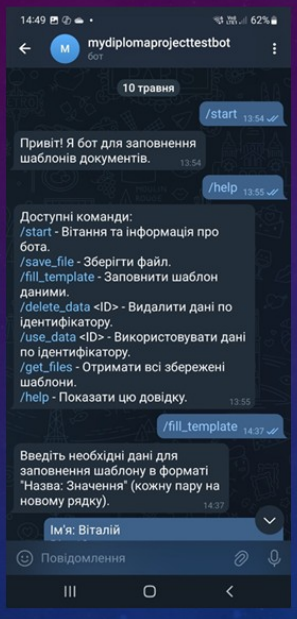


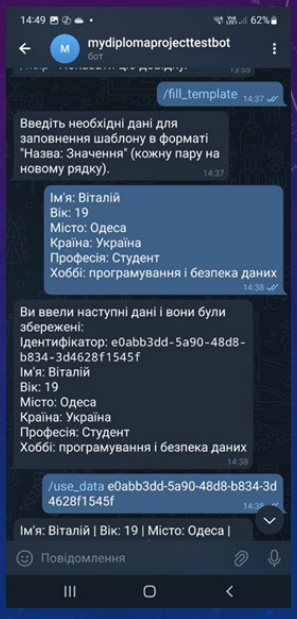
Діаграма станів віртуального асистента



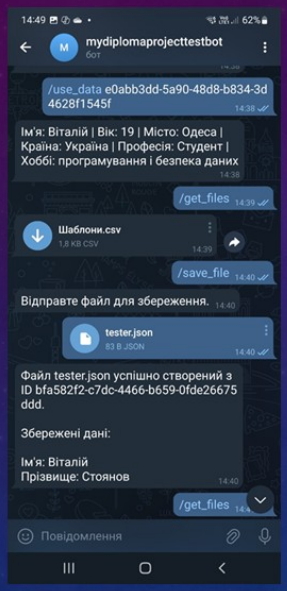
Блок-схема алгоритму забезпечення безпеки при формуванні текстових документів

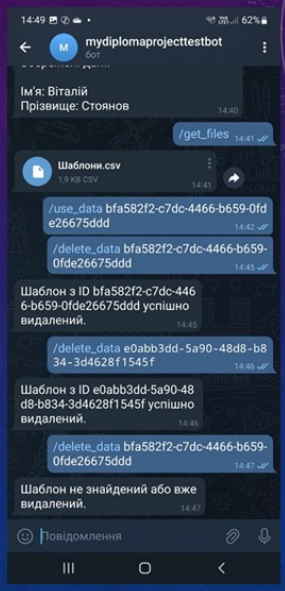
Використання команд /start, /help, /fill_template, /use_data віртуального асистента



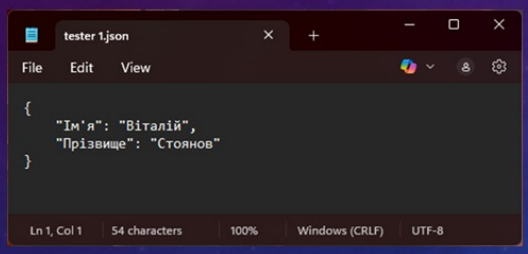


Використання команд /save_file, /get_files, /use_data, /delete_data віртуального асистента

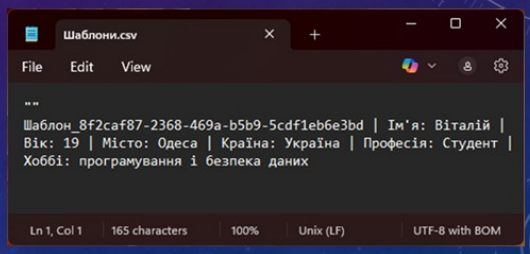




Вміст завантаженого json-файлу



Вміст сформованого csv-файлу за допомогою віртуального асистента



ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

РЕЦЕНЗІЯ

на дипломний проект здобувача (здобувачки) освіти

відділення комп'ютерних систем

Стоянова Віталія Степановича

(прізвище, ім`я та по батькові)

Спеціальність *123 «Комп'ютерна інженерія»*

Освітньо-професійна програма *«Безпека комп'ютерних систем і мереж»*

Керівник дипломного проекту (роботи) *Кривченко Анастасія Анатоліївна*

(прізвище, ім`я та по батькові)

Тема дипломного проекту (роботи) *Розробка віртуального асистента для безпечного формування текстових документів*

Обсяг розрахунково-пояснювальної записки *75* сторінок

Обсяг графічної (презентаційної) частини *16* аркушів (слайдів)

ХАРАКТЕРИСТИКА ДИПЛОМНОГО ПРОЕКТУ (РОБОТИ)

а) заключення про ступінь відповідності виконаного дипломного проекту завданню

Представлений дипломний проект відповідає затвердженій темі та виконаний відповідно технічному завданню. Дипломний проект присвячений створенню чат-ботів, розроблено застосунок віртуального асистенту у вигляді telegram-боту для безпечного формування текстових документів і складається з пояснювальної записки та мультимедійної презентації.

б) характеристика виконання кожного розділу дипломного проекту

Пояснювальна записка складається з основного розділу (Аналіз та класифікація віртуальних асистентів; Огляд засобів формування текстових документів; Визначення форматів текстових документів для обробки; Аналіз засобів безпеки; Аналіз програмних засобів розробки; Вибір і розробка архітектури ПЗ; Розробка та реалізація застосунку), економічного розділу, розділу охорони праці та додатків. Перелічені розділи поетапно охоплюють розробку, виконані докладно та обґрунтовано.

в) оцінка якості виконання пояснювальної записки та графічної частини дипломного проекту

Графічна частина складається з 16 слайдів мультимедійної презентації, виконаної у програмному продукті MS PowerPoint, які містять структурні та функціональні схеми, діаграми, блок-схеми алгоритмів, передбачені технічним завданням. Пояснювальна записка виконана акуратно та у відповідності до норм. Якість виконання пояснювальної записки відмінна, розробку виконано у повному обсязі.

г) перелік позитивних якостей дипломного проекту *Заповнення шаблонів документів у форматах .csv та .json на основі введених користувачем даних реалізовано у зручній та наочній формі; Реалізовано індексацію кожного файлу унікальним id для ведення бази даних файлів; Опис етапів розробки у пояснювальній записці виконано докладно та аргументовано*

д) основні недоліки дипломного проекту *У економічному розділі та розділі охорони праці присутні деякі відрізнення форматування тексту і таблиць відносно основного розділу; Бажано було передбачити перемикання інтерфейсу віртуального асистенту на англійську мову*

Оцінка розрахункової частини *Відмінно*

Оцінка графічної частини *Відмінно*

Загальна оцінка *Відмінно*

Прізвище, ім'я, по батькові рецензента *к.т.н. Рудніченко Микола Дмитрович*

Місце роботи і посада рецензента *Національний університет «Одеська політехніка», доцент кафедри інформаційних технологій*

Підпис

20 червня 2025 р.

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

ВІДГУК

керівника на дипломний проект здобувача (здобувачки) освіти

відділення комп'ютерних систем

Стоянова Віталія Степановича

(прізвище, ім`я та по батькові)

Спеціальність: *123 «Комп'ютерна інженерія»*

Освітня програма: *«Безпека комп'ютерних систем і мереж»*

Тема дипломного проекту: *Розробка віртуального асистента для безпечного формування текстових документів*

ХАРАКТЕРИСТИКА ДИПЛОМНОГО ПРОЕКТУ

а) обсяг і якість виконання проекту (графічного матеріалу і розрахунково-пояснювальної записки) *Дипломний проект виконано відповідно технічному завданню. Пояснювальна записка до дипломного проекту містить 75 сторінок. У пояснювальній записці виконано огляд та аналіз методів створення чат-ботів, розроблено застосунок віртуального асистенту у вигляді telegram-боту для безпечного формування текстових документів. Графічна частина складається з 16 слайдів, оформлених у вигляді презентації, передбачених технічним завданням. Якість виконання пояснювальної записки та слайдів добра.*

б) самостійність роботи над проектом: *Протягом виконання дипломного проекту здобувач освіти Стоянов Віталій поступово та послідовно виконував всі етапи, проявив ініціативу в створенні загальної концепції та реалізації роботи. Всі роботи здобувач освіти виконував самостійно, з оглядом на рекомендації керівника.*

в) теоретична підготовка випускника (випускниці): *Здобувач освіти Стоянов Віталій під час роботи над дипломним проектом вивчив достатньо багато літературних та інтернет-джерел за даною тематикою.*

Вважаю, що теоретична підготовка дипломника добра і він готовий до захисту проекту.

г) вміння розв'язувати виробничі та конструкторські питання *Під час виконання дипломного проекту здобувач освіти Стоянов Віталій показав вміння організовано працювати над поставленим завданням, застосовувати знання у сфері безпеки комп'ютерних систем і мереж, програмування, використовуючи сучасні комп'ютерні програмні засоби розробки, такі як Node.js, NestJS PostgreSQL, Redis, HMAC*

Оцінка розрахункової частини *Відмінно*

Оцінка графічної частини *Відмінно*

Загальна оцінка *Відмінно*

Прізвище, ім'я, по батькові керівника дипломного проекту *Кривченко Анастасія Анатоліївна*

Місце роботи і посада керівника дипломного проекту *ВСП «Одеський технічний фаховий коледж ОНТУ», викладач спецдисциплін циклової комісії комп'ютерних технологій та програмної інженерії, голова обласної методичної комісії викладачів комп'ютерної інженерії*

Підпис

«16» 06 2025 р.

ДОЗВІЛ
НА РОЗМІЩЕННЯ
ВИПУСКНОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ
(ДИПЛОМНОГО ПРОЕКТУ)
В ЕЛЕКТРОННОМУ РЕПОЗИТАРІЇ ВСП «ОТФК ОНТУ»

Ми, що нижче підписалися,

Стоянов В.С.,
здобувач освіти гр. 4КБ-02, та

Кривченко А.А.,
керівник дипломного проекту,

не заперечуємо щодо розміщення електронного варіанту пояснювальної записки до дипломного проекту фахового молодшого бакалавра на тему:

«Розробка віртуального асистента для безпечного формування текстових документів» (автор роботи – Стоянов В.С., керівник роботи – Кривченко А.А.)

виконаного у ВСП «Одеський технічний фаховий коледж Одеського національного технологічного університету» в 2025 році, у повному обсязі в електронному репозитарії ВСП «ОТФК ОНТУ» для вільного доступу через мережу Інтернет.

Несемо відповідальність за ідентичність електронного та друкованого варіантів випускної кваліфікаційної роботи і даємо згоду на обробку персональних даних.

Виконавець _____ / Стоянов В.С. /

Керівник _____ / Кривченко А.А. /

«16» червня 2025 р.

Д О В І Д К А

циклової комісії КТ та ПІ
про допуск до захисту дипломного проєкту
здобувача (здобувачки) освіти IV курсу
відділення комп'ютерних систем групи 4КБ-02

Стоянова Віталія Степановича

на тему *Розробка віртуального асистента для безпечного формування текстових документів*

Висновок відповідальної особи за проведення нормоконтролю:

пояснювальна записка до дипломного проєкту виконана з несуттєвими порушеннями ДСТУ та оформлена відповідно до вимог Положення про дипломне проєктування

(підпис)	*16.06.2025* (дата)	*Петрашова В.І.* (П.І.Б.)

Висновок відповідальної особи за перевірку роботи на наявність академічного плагіату *згідно звіту про перевірку від 05.06.2025 р. значення коефіцієнту подібності в роботі становить 15,19%, коефіцієнт цитування – 1,07%.*

(підпис)	*16.06.2025* (дата)	*Краснокутська К.Г.* (П.І.Б.)

Попередня експертиза (малий захист) дипломного проєкту

здобувача (здобувачки) освіти *Стоянова В.С.* (П.І.Б.)

проведена « *16* » *червня* 2025 р.

Висновки *Пояснювальна записка до дипломного проєкту виконана у повному обсязі. Випускна кваліфікаційна робота (дипломний проєкт) відповідає вимогам Положення про дипломне проєктування та рекомендована до захисту.*

Голова ЦК КТ та ПІ (підпис) *Кривченко Ю.В.* (П.І.Б.)

StrikePlagiarism.com

Дата звіту 6/5/2025
Дата редагування 6/5/2025

Документ прийнятий

Звіт подібності

метадані

Назва організації
Odesa Technical Professional College of Odesa National University of Technology
Заголовок
Розробка віртуального асистента для безпечного формування текстових документів

Автор
Стоянов Віталій Степанович
Науковий керівник / Експерт
Кривченко Анастасія Анатоліївна

підрозділ
Відокремлений структурний підрозділ "Одеський технічний фаховий коледж Одеського національного технологічного університету"

Обсяг знайдених подібностей

Коефіцієнт подібності визначає, який відсоток тексту по відношенню до загального обсягу тексту було знайдено в різних джерелах. Зверніть увагу, що високі значення коефіцієнта не автоматично означають плагіат. Звіт має аналізувати компетентна / уповноважена особа.

КП 1

КЦ

25
Довжина фрази для коефіцієнта подібності 2

13605
Кількість слів

114493
Кількість символів

Тривога

У цьому розділі ви знайдете інформацію щодо текстових спотворень. Ці спотворення в тексті можуть говорити про МОЖЛИВІ маніпуляції в тексті. Спотворення в тексті можуть мати навмисний характер, але частіше характер технічних помилок при конвертації документа та його збереженні, тому ми рекомендуємо вам підходити до аналізу цього модуля відповідально. У разі виникнення запитань, просимо звертатися до нашої служби підтримки.

Заміна букв		21
Інтервали		0
Мікропробіли		0
Білі знаки		0
Парафрази (SmartMarks)	a	73

Подібності за списком джерел

Нижче наведений список джерел. В цьому списку є джерела із різних баз даних. Колір тексту означає в якому джерелі він був знайдений. Ці джерела і значення Коефіцієнту Подібності не відображають прямого плагіату. Необхідно відкрити кожне джерело і проаналізувати зміст і правильність оформлення джерела.

10 найдовших фраз — Колір тексту

ПОРЯДКОВИЙ НОМЕР	НАЗВА ТА АДРЕСА ДЖЕРЕЛА URL (НАЗВА БАЗИ)	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	https://card-file.ontu.edu.ua/server/api/core/bitstreams/ead3fa83-2e3d-4cd7-bfbd-1d5ed04c1ce4/content	103 0.76 %
2	https://card-file.ontu.edu.ua/bitstreams/53ed22ad-8700-4162-b97a-082a1ad472d6/download	66 0.49 %
3	https://card-file.ontu.edu.ua/server/api/core/bitstreams/ead3fa83-2e3d-4cd7-bfbd-1d5ed04c1ce4/content	58 0.43 %
4	https://card-file.ontu.edu.ua/server/api/core/bitstreams/ead3fa83-2e3d-4cd7-bfbd-1d5ed04c1ce4/content	57 0.42 %
5	https://card-file.ontu.edu.ua/bitstreams/21173711-5b67-4b87-b17f-6302c25e7a31/download	51 0.37 %

6	https://card-file.ontu.edu.ua/server/api/core/bitstreams/44c16132-5f53-48e2-b6c0-61e9a2f0fd75/content	49 0.36 %
7	https://card-file.ontu.edu.ua/bitstreams/5240e379-7721-49f0-8ee8-27140b0b473a/download	47 0.35 %
8	https://card-file.ontu.edu.ua/server/api/core/bitstreams/ead3fa83-2e3d-4cd7-bfbd-1d5ed04c1ce4/content	43 0.32 %
9	https://card-file.ontu.edu.ua/bitstreams/53ed22ad-8700-4162-b97a-082a1ad472d6/download	32 0.24 %
10	http://cpsm.kpi.ua/stud/bak/DP_BAK_KARAULOVA_LU.pdf	32 0.24 %

з домашньої бази даних (0.29 %)

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	Створення web-застосунку цифрового помічника з використанням Open AI [redacted] 5/28/2025 Odesa Technical Professional College of Odesa National University of Technology (Відокремлений структурний підрозділ "Одеський технічний фаховий коледж Одеського національного технологічного університету")	33 (3) 0.24 %
2	Аналіз ефективності алгоритмів стискування відео-даних для стрімінгових сервісів [redacted] 6/3/2025 Odesa Technical Professional College of Odesa National University of Technology (Відокремлений структурний підрозділ "Одеський технічний фаховий коледж Одеського національного технологічного університету")	6 (1) 0.04 %

з програми обміну базами даних (0.96 %)

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	Телеграм бот для заповнення шаблонів [redacted] 3/16/2025 **National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute)**	**104 (10) 0.76 %**
2	ФКПІ_ІПЗ_2023_121_Різник_М.Ю [redacted] 7/11/2024 Ukrainian national aviation university (Ukrainian national aviation university)	11 (1) 0.08 %
3	Масалов_дипломна_робота [redacted] 6/3/2025 National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (IATE, К-ра цифрових технології в енергетиці)	8 (1) 0.06 %
4	Диплом Струк для проверкии.pdf [redacted] 12/22/2020 Odessa National Polytechnic University (УНІ, каф. підйомно-транспортного та робототехнічного обладнання)	7 (1) 0.05 %

з Інтернету (13.94 %)

ПОРЯДКОВИЙ НОМЕР	ДЖЕРЕЛО URL	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	https://card-file.ontu.edu.ua/bitstreams/1dff552d-7200-49b8-ae1d-ba76a1335685/download	709 (61) 5.21 %
2	**https://card-file.ontu.edu.ua/server/api/core/bitstreams/ead3fa83-2e3d-4cd7-bfbd-1d5ed04c1ce4/content**	**342 (10) 2.51 %**
3	**https://card-file.ontu.edu.ua/bitstreams/53ed22ad-8700-4162-b97a-082a1ad472d6/download**	**162 (8) 1.19 %**
4	**https://card-file.ontu.edu.ua/server/api/core/bitstreams/44c16132-5f53-48e2-b6c0-61e9a2f0fd75/content**	**105 (8) 0.77 %**

5	https://card-file.ontu.edu.ua/bitstreams/21173711-5b67-4b87-b17f-6302c25e7a31/download	67 (2) 0.49 %
6	https://card-file.ontu.edu.ua/bitstreams/035f6436-20b4-4ee6-8e99-bede670e308b/download	65 (4) 0.48 %
7	https://card-file.ontu.edu.ua/bitstreams/29489599-0581-4ce6-8890-c3b13d9f2e0e/download	55 (3) 0.40 %
8	https://card-file.ontu.edu.ua/bitstreams/5240e379-7721-49f0-8ee8-27140b0b473a/download	47 (1) 0.35 %
9	http://cpsm.kpi.ua/stud/bak/DP_BAK_KARAULOVA_LU.pdf	40 (2) 0.29 %
10	https://card-file.ontu.edu.ua/server/api/core/bitstreams/995bdcec-4e4d-4321-8070-4d6badcb8e49/content	36 (3) 0.26 %
11	https://card-file.ontu.edu.ua/bitstreams/549ee9fe-7574-4ae5-b500-9fe2711f33e6/download	34 (2) 0.25 %
12	https://card-file.ontu.edu.ua/server/api/core/bitstreams/4bb7255e-46d4-4349-9726-9698476da02d/content	27 (4) 0.20 %
13	https://card-file.ontu.edu.ua/bitstreams/c58b0ff5-46e0-49f8-8cbe-65c32256665d/download	26 (2) 0.19 %
14	https://card-file.ontu.edu.ua/server/api/core/bitstreams/a141b658-5fa7-4f90-b0bd-7f0ccaed21e5/content	25 (2) 0.18 %
15	https://card-file.ontu.edu.ua/bitstreams/f292747f-f875-4858-906b-e14629f6ec57/download	20 (1) 0.15 %
16	https://card-file.ontu.edu.ua/bitstreams/e4afae26-0a7e-4a4d-afc2-94341838de2a/download	18 (1) 0.13 %
17	https://card-file.ontu.edu.ua/bitstreams/62baa43e-b968-4993-bb54-8cf8761a89b2/download	17 (1) 0.12 %
18	https://card-file.ontu.edu.ua/bitstreams/12d5c0ab-e979-48f2-a8ec-d5fc31f71fd5/download	16 (1) 0.12 %
19	https://card-file.ontu.edu.ua/bitstreams/bbed74c8-2ea7-44c5-8d00-0fe3fd9790ee/download	16 (2) 0.12 %
20	https://www.yumpu.com/xx/document/view/36610022/-/219	16 (1) 0.12 %
21	https://card-file.ontu.edu.ua/server/api/core/bitstreams/6eb6bf1c-5813-45e6-93c5-25539b4709d3/content	16 (2) 0.12 %
22	https://card-file.ontu.edu.ua/bitstreams/82a6d375-2b69-4233-b80f-fbfd149b7747/download	12 (1) 0.09 %
23	https://card-file.ontu.edu.ua/bitstreams/34a6756b-592f-4b77-a805-183aa03a6a26/download	9 (1) 0.07 %
24	https://card-file.ontu.edu.ua/bitstreams/bbaf3f38-16a8-4070-bead-5562769b7c71/download	7 (1) 0.05 %
25	https://card-file.ontu.edu.ua/bitstreams/6cf43324-8f08-4031-ba42-f80b18efbbc8/download	5 (1) 0.04 %
26	https://ohoronapraci.com.ua/articles/666201-vydy-medychnykh-ohlyadiv-pratsivnykiv	5 (1) 0.04 %

Список прийнятих фрагментів (немає прийнятих фрагментів)

ПОРЯДКОВИЙ НОМЕР	ЗМІСТ	КІЛЬКІСТЬ ОДНАКОВИХ СЛІВ (ФРАГМЕНТІВ)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Спеціальність: 123 «Комп'ютерна інженерія»
Освітньо-професійна програма: «Безпека комп'ютерних систем і мереж» Група: 4КБ-02

Дипломний проєкт
здобувача **освіти денної форми навчання КБ.** 02.20.000.ДП

СТОЯНОВА
ВІТАЛІЯ СТЕПАНОВИЧА

м. Одеса