ПОЛТАВСЬКИЙ ДЕРЖАВНИЙ АГРАРНИЙ УНІВЕРСИТЕТ
Навчально-науковий інститут економіки, управління, права та інформаційних технологій
Кафедра інформаційних систем та технологій

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти бакалавр

на тему: **«Розробка застосунку на мові Python для аналізу даних опитувань»**

Виконав: здобувач вищої освіти
за освітньою програмою
Інформаційні управляючі системи
спеціальності 126 Інформаційні
системи та технології
ступеня вищої освіти бакалавр
групи 126ІСТ_бд_2022
Хованець Ілля Родіонович
Керівник: Поночовний Юрій
Леонідович
Рецензент: Муравльов Володимир
Вячеславович

Полтава – 2026 року

ПОЛТАВСЬКИЙ ДЕРЖАВНИЙ АГРАРНИЙ УНІВЕРСИТЕТ
Навчально-науковий інститут економіки, управління, права та інформаційних технологій
Кафедра інформаційних систем та технологій

Освітня програма Інформаційні управляючі системи
Спеціальність 126 Інформаційні системи та технології
Рівень вищої освіти перший (бакалаврський)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Юрій УТКІН
«10» червня 2025 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

Хованцю Іллі Родіоновичу

1. Тема роботи:
«Розробка застосунку на мові Python для аналізу даних опитувань»,
керівник роботи д.т.н., професор, професор кафедри інформаційних систем та технологій Поночовний Юрій Леонідович.
Затверджено наказом закладу вищої освіти від «10» квітня 2026 року № 383 ст.

2. Строк подання здобувачем вищої освіти роботи «08» червня 2026 року.

3. Вихідні дані до роботи: стандарти проєктування та розробки програмного забезпечення (IEEE 830, ISO/IEC 29500), документація офіційних сайтів бібліотек https://pandas.pydata.org/, https://matplotlib.org/, https://docs.python.org/ (для Python та tkinter), середовище розробки Visual Studio Code, інтерпретатор Python 3, реальні дані анкетування закладу вищої освіти у форматі xlsx.

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити):

Розділ 1. Аналіз предметної області, існуючих рішень та постановка задачі розробки застосунку
Розділ 2. Проєктування структури та функціональних модулів застосунку для аналізу даних опитувань
Розділ 3. Програмна реалізація, тестування застосунку та оцінка ефективності розробленого рішення

5. Перелік графічного матеріалу: схеми, рисунки, графіки, діаграми за темою та об’єктом дослідження.

6. Консультанти розділів кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання отримав
Оцінювання економічної ефективності результатів дослідження	Дивнич О. Д., к. е. н., доцент, доцент кафедри економіки та публічного управління	01.04.2026 р.	29.05.2026 р.

7. Дата видачі завдання «10» червня 2025 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Строк виконання етапів кваліфікаційної роботи	Примітка
1	Вибір і затвердження теми роботи	02.06.2025 р. – 04.06.2025	
2	Складання і затвердження розгорнутого плану та завдання на кваліфікаційну роботу	05.06.2025 р. – 10.06.2025 р.	
3	Опрацювання джерел інформації	11.06.2025 р. – 15.06.2025 р.	
4	Збір, вивчення і обробка інформації, необхідної для виконання роботи	16.06.2025 р. – 20.06.2025 р.	
5	Виконання теоретико-методологічного розділу роботи	08.09.2025 р. – 01.11.2025 р.	
6	Виконання дослідницько-аналітичного розділу роботи	19.01.2026 р. – 14.02.2026 р.	
7	Виконання проєктно-рекомендаційного розділу роботи	16.02.2026 р. – 31.03.2026 р.	
8	Оцінювання економічної ефективності результатів дослідження	01.04.2026 р. – 29.05.2026 р.	
9	Оформлення тексту роботи	01.06.2026 р. – 06.06.2026р.	
10	Попередній захист роботи на кафедрі	08.06.2026 р.	
11	Доопрацювання роботи з урахуванням зауважень і пропозицій	09.06.2026 р. – 10.06.2026 р.	
12	Нормоконтроль	11.06.2026 р. – 15.06.2026 р.	
13	Захист кваліфікаційної роботи	16.06.2026 р. - 18.06.2026 р.	

Здобувач вищої освіти **Ілля ХОВАНЕЦЬ**

Керівник роботи **Юрій ПОНОЧОВНИЙ**

ЗМІСТ

ВСТУП

Актуальність. Забезпечення якості вищої освіти є одним із стратегічних пріоритетів освітньої політики України, закріплених у Законах України «Про вищу освіту» та «Про освіту». Відповідно до вимог Національного агентства із забезпечення якості вищої освіти (НАЗЯВО), заклади вищої освіти зобов'язані систематично проводити внутрішній моніторинг освітньої діяльності, невід'ємною складовою якого є анкетування здобувачів, викладачів, роботодавців та інших стейкхолдерів [1, 2]. За типовою практикою, у закладі вищої освіти щорічно обробляється від 5 до 10 видів анкет, кожна з яких може містити дані за кількома навчальними роками. Ручна підготовка аналітичного звіту на основі одного файлу анкетних даних займає в середньому 3 години, а весь комплект – до 96 людино-годин на рік [3, 4].

Незважаючи на поширеність Google Forms як інструменту збору даних, процес подальшої обробки результатів залишається значною мірою ручним: аналітик вивантажує Excel-файл, вручну будує діаграми та формує PDF-звіт. Існуючі хмарні платформи та BI-інструменти не забезпечують автоматичного формування структурованих PDF-документів встановленого зразка з коректним відображенням кириличного тексту та підтримкою різних типів питань [5, 6]. Це зумовлює актуальність розробки спеціалізованого застосунку, здатного автоматизувати весь цикл обробки – від зчитування xlsx-файлу до генерації готового PDF-звіту – і скоротити трудомісткість процесу до кількох хвилин.

Метою кваліфікаційної роботи є розробка застосунку на мові Python для автоматизованого аналізу даних опитувань і формування PDF-звітів із діаграмами на основі результатів анкетування у системі моніторингу якості вищої освіти.

Відповідно до мети роботи необхідно вирішити наступні *завдання:*

1. Провести аналіз предметної області процесів моніторингу якості освіти, існуючих підходів до автоматизованої обробки даних анкетування та інструментальних засобів розробки застосунків для аналізу даних;

2. Спроєктувати архітектуру застосунку, структуру модулів обробки даних, логіку автоматичного визначення типу діаграми та модуль генерації PDF-звітів;

3. Реалізувати програмний застосунок із графічним інтерфейсом користувача, провести його тестування на реальних даних анкетування та здійснити техніко-економічне обґрунтування ефективності розробленого рішення.

Об'єкт дослідження – процеси автоматизованої обробки даних анкетування у системі внутрішнього забезпечення якості вищої освіти.

Предмет дослідження – проєктування та розробка застосунку на мові Python для аналізу даних опитувань із використанням бібліотек pandas і matplotlib та генерацією структурованих PDF-звітів.

Методи досліджень – дослідження базуються на методах системного аналізу предметної області, порівняльного аналізу програмних інструментів, модульного проєктування програмного забезпечення, об'єктно-орієнтованого та процедурного програмування на мові Python, функціонального та структурного тестування, а також техніко-економічного аналізу ефективності IT-проєктів.

Інформаційна база – нормативні документи НАЗЯВО щодо стандартів акредитації освітніх програм, наукові статті та навчальні матеріали з питань аналізу даних, документація бібліотек Python (pandas, matplotlib, openpyxl, tkinter), стандарти розробки програмного забезпечення (IEEE 830, ISO/IEC 29500), а також реальні дані восьми типів анкет закладу вищої освіти за 2022-2026 навчальні роки.

Практична значущість – розроблений застосунок забезпечує повну автоматизацію циклу обробки результатів анкетування: зчитування xlsx-файлів, класифікацію питань за типом відповіді, побудову кругових, горизонтальних стовпчастих діаграм і текстових блоків, та збереження у багатосторінкові PDF-документи з коректним відображенням кирилиці. Застосунок скорочує трудомісткість обробки з 96 до 0,5 людино-годин на рік і може бути адаптований

до будь-якого закладу вищої освіти, що використовує Google Forms для проведення анкетування в системі внутрішнього забезпечення якості.

Результати роботи апробовані в рамках наукової конференції здобувачів вищої освіти за результатами науково-дослідної роботи у 2025-2026 рр.

Структура кваліфікаційної роботи логічно пов’язана з задачами досліджень і містить перелік умовних позначень, вступ, три розділи основної частини, висновки, список використаних джерел. Загальний обсяг текстової частини кваліфікаційної роботи складає 58 сторінок формату А4. Вона містить 15 рисунків і 13 таблиць.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ, ІСНУЮЧИХ РІШЕНЬ ТА ПОСТАНОВКА ЗАДАЧІ РОЗРОБКИ ЗАСТОСУНКУ

1.1 Характеристика предметної області та огляд процесів моніторингу якості освіти

Система забезпечення якості вищої освіти є одним із ключових пріоритетів сучасної освітньої політики як в Україні, так і в більшості країн світу. Відповідно до вимог Законів України «Про вищу освіту» та «Про освіту», заклади вищої освіти зобов'язані здійснювати внутрішній моніторинг якості освітньої діяльності, що передбачає систематичне збирання, обробку та аналіз даних від усіх учасників освітнього процесу [1]. Одним із найпоширеніших інструментів такого моніторингу є анкетування – стейкхолдерів, здобувачів вищої освіти, викладачів та роботодавців.

Моніторинг якості освіти являє собою комплексний процес, що охоплює збирання інформації про стан освітньої системи, її аналіз та прийняття на цій основі управлінських рішень [2]. У контексті акредитації освітніх програм, яку в Україні здійснює Національне агентство із забезпечення якості вищої освіти (НАЗЯВО), результати анкетування є обов'язковим елементом самооцінювання освітньої програми. Зокрема, стандарти акредитації передбачають регулярне опитування здобувачів, випускників та роботодавців щодо відповідності компетентностей потребам ринку праці [3].

Типова схема організації анкетування у закладі вищої освіти охоплює кілька етапів: розробку анкети, проведення опитування (нині переважно через Google Forms), вивантаження результатів у табличному форматі (Excel або CSV), обробку та візуалізацію даних, формування підсумкового звіту. Саме етап обробки та візуалізації є найбільш трудомістким, оскільки передбачає роботу з різнотипними питаннями – закритими (так/ні), рейтинговими (шкала оцінок),

питаннями з множинним вибором та відкритими (текстові відповіді). Різні типи питань вимагають різних підходів до представлення результатів. На рисунку 1.1 наведено загальну схему процесу моніторингу якості освіти через анкетування.

Рисунок 1.1 – Схема процесу моніторингу якості освіти через анкетування

У закладах вищої освіти України типово використовуються від 5 до 10 різних видів анкет залежно від категорії респондентів та предмету оцінювання [4]. У контексті даної роботи розглядаються 8 типів анкет: анкета для зовнішніх стейкхолдерів (роботодавців), анкета оцінювання якості навчання, анкета щодо сильних та слабких сторін освітньої програми, анкета якості викладання навчальних дисциплін, анкета залучення до наукової роботи, анкета якості самостійної роботи, анкета якості практичної підготовки та анкета щодо системи контрольних заходів. Кожна з них має власну структуру запитань і, відповідно, специфічні вимоги до візуалізації результатів.

Таблиця 1.1 – Типи питань анкет та відповідні методи візуалізації

Тип питання	Приклад	Метод візуалізації
Закрите (так/ні)	«Чи задоволені Ви якістю викладання?»	Відсоткове співвідношення
Рейтингове	Оцінка від 1 до 5	Кругова діаграма
Множинний вибір	«Оберіть декілька компетентностей»	Горизонтальна стовпчаста діаграма
Відкрите	«Ваші пропозиції щодо покращення»	Текстовий список

Як видно з таблиці 1.1, різні типи питань вимагають принципово різних підходів до подання результатів. Це ускладнює автоматизацію обробки і вимагає

інтелектуальної логіки визначення типу відповідей у застосунку. Аналіз структури вхідних даних показує, що Excel-файли, які є результатом вивантаження з Google Forms, мають типову організацію: перший рядок – заголовки (тексти питань), кожен наступний рядок – відповіді одного респондента. У розглядуваних анкетах перша колонка містить навчальний рік, що дозволяє здійснювати порівняльний аналіз у динаміці [5]. Загальна кількість рядків у файлі (кількість відповідей) може варіюватися від десятків до кількох сотень залежно від кількості учасників опитування.

З огляду на масштаб задачі – 8 типів анкет, потенційно кілька навчальних років для кожної – ручна обробка даних є неприйнятно трудомісткою. За оцінками дослідників у галузі автоматизації освітніх процесів, ручна підготовка одного аналітичного звіту на основі анкетних даних займає від 2 до 4 годин [6]. При наявності 8 анкет і даних за 3-4 роки це становить від 48 до 128 людино-годин щорічно. Автоматизований застосунок здатний скоротити цей час до кількох хвилин.

Оцінку трудомісткості можна виразити таким чином. Якщо позначити кількість типів анкет через N_a, кількість років через N_r, а середній час ручної обробки однієї анкети за один рік через t (год), то загальна економія часу T обчислюється за формулою:

$$T = N_a \cdot N_r \cdot t - t_{auto}, \tag{1.1}$$

де t_{auto} – час автоматизованої обробки всіх анкет.

Для наведених вище значень ($N_a = 8$, $N_r = 4$, $t = 3$ год, $t_{auto} \approx 0{,}1$ год) економія становить близько 95,9 людино-годин на рік. Таким чином, предметна область характеризується чіткою структурою вхідних даних, повторюваністю процесу обробки та суттєвим потенціалом для автоматизації. Ці властивості є передумовою для розробки спеціалізованого застосунку на мові Python, який замінить ручну обробку результатів анкетування на автоматизоване формування PDF-звітів з діаграмами відповідного типу.

1.2 Аналіз існуючих підходів до автоматизованої обробки даних анкетування

Розвиток цифрових технологій суттєво розширив можливості автоматизованої обробки результатів опитувань. На сьогодні існує кілька принципово різних підходів до вирішення цієї задачі: використання хмарних платформ для онлайн-опитувань із вбудованою аналітикою, застосування універсальних BI-інструментів (Business Intelligence), написання власного програмного коду на мовах аналізу даних, а також використання спеціалізованих статистичних пакетів [8]. Вибір підходу визначається масштабом задачі, вимогами до формату результатів, доступністю інструментів та технічною компетентністю виконавців.

Найпоширенішими хмарними платформами для збору та первинного аналізу опитувань є Google Forms, SurveyMonkey та Microsoft Forms. Google Forms, зокрема, автоматично формує зведену статистику відповідей і будує прості діаграми безпосередньо в інтерфейсі [9]. Проте можливості вбудованої аналітики цих платформ є суттєво обмеженими: відсутня підтримка користувацьких типів візуалізації, неможливо налаштувати зовнішній вигляд звіту відповідно до корпоративних чи академічних стандартів, а генерація PDF-документів або не передбачена, або реалізована у спрощеному вигляді без можливості налаштування [10]. Крім того, при роботі з даними, що містять персональну інформацію про учасників освітнього процесу, використання сторонніх хмарних сервісів може суперечити вимогам законодавства про захист персональних даних.

BI-інструменти, такі як Microsoft Power BI, Tableau або Google Looker Studio, надають значно ширші можливості для побудови інтерактивних аналітичних панелей [11]. Вони підтримують підключення до Excel-файлів як джерел даних, дозволяють будувати різноманітні типи діаграм та публікувати

звіти. Однак ці інструменти орієнтовані насамперед на інтерактивні веб-дашборди, а не на формування статичних PDF-документів встановленого зразка. Крім того, використання комерційних версій потребує придбання ліцензій, а безкоштовні версії мають обмеження щодо кількості звітів та обсягу даних.

Спеціалізовані статистичні пакети – SPSS, R, Stata – традиційно використовуються в академічному середовищі для глибокого статистичного аналізу даних опитувань [12]. Вони забезпечують широкий набір методів аналізу, включаючи кореляційний, факторний та кластерний аналіз. Проте їх застосування вимагає значних знань у галузі статистики, а процес підготовки звіту залишається значною мірою ручним. Для задачі формування стандартизованих PDF-звітів із фіксованою структурою ці інструменти є надлишковими за функціональністю і складними у налаштуванні. На рисунку 1.2 наведено порівняльну схему розглянутих підходів за ключовими критеріями.

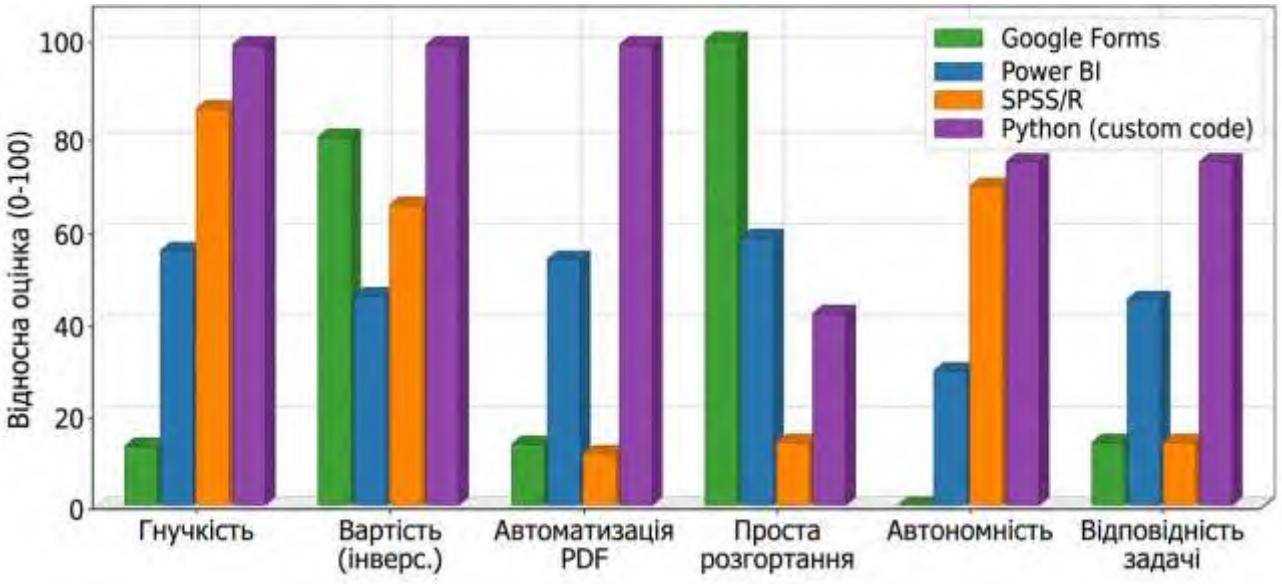

Рисунок 1.2 – Порівняння підходів до автоматизованої обробки даних анкетування

Окремим підходом є написання власного програмного коду. Мова Python посідає провідні позиції серед інструментів аналізу даних завдяки розвиненій

екосистемі бібліотек [13]. Зокрема, бібліотека pandas забезпечує зручне зчитування та обробку табличних даних з Excel-файлів, matplotlib – побудову різноманітних типів діаграм, а matplotlib.backends.backend_pdf через клас PdfPages – збереження графіків у багатосторінковий PDF-документ. Цей підхід забезпечує повний контроль над форматом вихідного документа, не потребує ліцензійних витрат і може бути адаптований під будь-яку специфіку вхідних даних.

Таблиця 1.2 – Порівняльний аналіз підходів до обробки даних анкетування

Критерій	Google Forms	Power BI	SPSS/R	Python (власний код)
Вартість	Безкоштовно	Платний	Платний / безкоштовний	Безкоштовно
Генерація PDF заданого формату	Обмежена	Часткова	Ручна	Повна
Гнучкість налаштування	Низька	Середня	Висока	Висока
Потреба у технічних знаннях	Низька	Середня	Висока	Середня
Автономність (без хмари)	Ні	Частково	Так	Так
Відповідність задачі проекту	Низька	Середня	Низька	Висока

Як видно з таблиці 1.2, підхід на основі власного Python-коду є найбільш відповідним для поставленої задачі за сукупністю критеріїв. Показовим є те, що серед розробників та аналітиків даних Python стабільно утримує першу позицію в рейтингах популярності мов програмування – зокрема, за індексом TIOBE у 2024–2025 роках його частка перевищує 20% [14].

Важливим аспектом при порівнянні підходів є також показник повторюваності результату. Якщо позначити загальну кількість анкет через N, а ймовірність людської помилки при ручній обробці однієї анкети через p, то ймовірність безпомилкової обробки всього набору P визначається як:

$$P = (1 - p)^N. \tag{1.2}$$

За типового значення $p = 0{,}05$ та $N = 8$ отримуємо $P \approx 0{,}66$, тобто лише 66% шансів, що весь комплект звітів буде підготовлено без помилок. Автоматизований застосунок за умови коректної реалізації забезпечує $p = 0$ і, відповідно, $P = 1$.

Таким чином, проведений аналіз підтверджує, що для задачі автоматизованого формування PDF-звітів на основі Excel-файлів з результатами анкетування оптимальним є підхід розробки власного застосунку на мові Python із використанням бібліотек pandas та matplotlib. Цей підхід забезпечує необхідну гнучкість, повний контроль над форматом вихідних документів, відсутність залежності від хмарних сервісів та нульову вартість ліцензування.

1.3 Огляд інструментальних засобів розробки застосунків для аналізу даних

Сучасна екосистема мови Python налічує сотні бібліотек, орієнтованих на обробку, аналіз та візуалізацію даних. Для задачі, що розглядається в даній роботі, ключовими є бібліотеки роботи з табличними даними, засоби побудови діаграм та інструменти генерації PDF-документів. Розглянемо детальніше ті з них, які безпосередньо використані у розробленому застосунку, а також альтернативи, що розглядалися на етапі проєктування.

Центральною бібліотекою для роботи з табличними даними у Python є pandas – відкрита бібліотека з підтримкою зчитування файлів Excel (через движок openpyxl або xlrd), CSV, JSON та інших форматів [16]. Основною структурою даних pandas є DataFrame – двовимірна таблиця з іменованими рядками та стовпцями, що за своєю логікою відповідає аркушу Excel. Бібліотека надає потужний інструментарій для фільтрації, групування та агрегації даних, що є критично важливим при розподілі відповідей за навчальними роками та підрахунку частот відповідей на кожне питання анкети. Альтернативою pandas

для зчитування Excel є бібліотека openpyxl у «чистому» вигляді, однак вона не надає зручних засобів агрегації й орієнтована переважно на низькорівневу роботу з клітинками аркуша [17].

Для побудови діаграм у застосунку використовується бібліотека matplotlib – одна з найстаріших і найпоширеніших бібліотек візуалізації даних у Python [18]. Вона підтримує кругові діаграми (pie chart), горизонтальні стовпчасті діаграми (barh), вертикальні стовпчасті діаграми (bar) та текстові сторінки (через вимкнення осей і розміщення тексту у координатах від 0 до 1). Особливо важливим для даної роботи є модуль matplotlib.backends.backend_pdf, який реалізує клас PdfPages – він дозволяє зберігати послідовність matplotlib-фігур у єдиний багатосторінковий PDF-файл без використання будь-яких сторонніх PDF-рушіїв. Серед альтернатив matplotlib варто виділити бібліотеку seaborn, яка надає більш естетичні стилі за замовчуванням [19], та plotly, що орієнтована на інтерактивні веб-графіки [20]. Проте для формування статичних PDF-документів matplotlib залишається оптимальним вибором завдяки прямій інтеграції з PdfPages. На рисунку 1.3 наведено загальну структуру стеку бібліотек, що застосовується у розробленому застосунку.

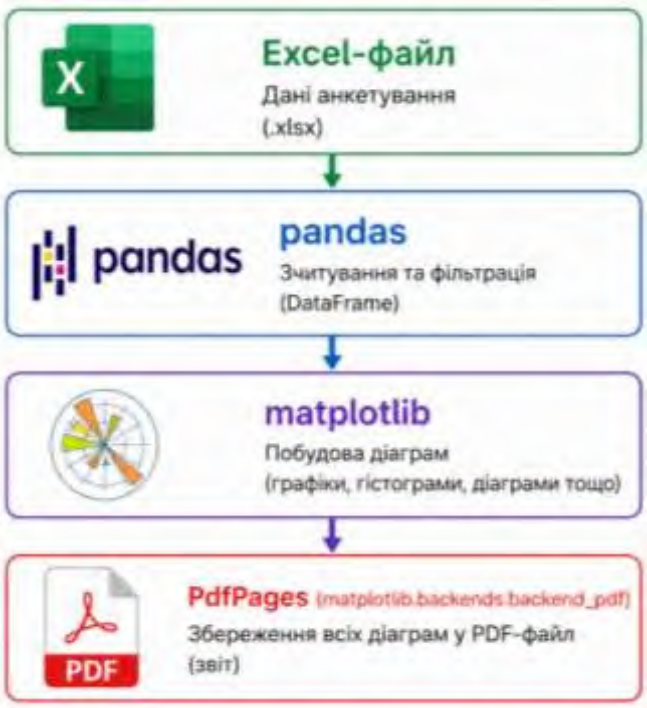

Рисунок 1.3 – Стек бібліотек Python-застосунку для аналізу даних анкетування

Окремої уваги заслуговує питання підтримки кирилиці у matplotlib. За замовчуванням бібліотека використовує шрифт DejaVu Sans, який не підтримує повний набір символів кирилиці, що призводить до відображення символів-замінників (так звані «квадрати» або «tofu») [21]. Для коректного відображення україномовного тексту необхідно явно вказати шрифт із підтримкою Unicode. Це є важливим практичним аспектом при роботі з україномовними анкетами і має бути враховано в архітектурному рішенні застосунку.

Таблиця 1.3 – Порівняльна характеристика бібліотек візуалізації даних у Python

Бібліотека	Тип графіків	Вивід у PDF	Підтримка кирилиці	Інтерактивність
matplotlib	Статичні	Вбудована (PdfPages)	Потребує налаштування	Ні
seaborn	Статичні	Через matplotlib	Потребує налаштування	Ні
plotly	Інтерактивні	Через kaleido	Так	Так
bokeh	Інтерактивні	Обмежена	Так	Так

Як видно з таблиці 1.3, лише matplotlib має вбудовану підтримку збереження у PDF, що є вирішальним критерієм для даної задачі. Для реалізації конфігураційного модуля проєкту використовується вбудований модуль Python os, який забезпечує кросплатформену роботу з файловою системою – зокрема, перевірку існування директорій та їх створення. Зчитування Excel-файлів у pandas відбувається через функцію read_excel(), яка приймає параметри імені файлу, номера або назви аркуша та номера рядка-заголовка. Загальний обсяг залежностей проєкту є мінімальним: pandas, matplotlib та openpyxl (як движок для xlsx), що спрощує розгортання застосунку на будь-якому комп'ютері з встановленим Python.

Варто також розглянути альтернативний підхід до генерації PDF – використання бібліотек reportlab або fpdf2, які формують PDF-документи безпосередньо, без проміжного етапу побудови matplotlib-фігур [22]. Ці бібліотеки надають більший контроль над типографікою та розміщенням елементів на сторінці, але вимагають значно більшого обсягу коду для

відтворення тих самих діаграм. З огляду на те, що основним завданням застосунку є саме побудова діаграм, а не складна верстка, підхід на основі matplotlib+PdfPages є більш доцільним.

Середовище розробки також є важливим інструментальним засобом. Для написання Python-коду широко використовуються Visual Studio Code із розширенням Python та JupyterLab [23]. Перший варіант більш підходить для розробки виробничого коду з модульною структурою, другий – для інтерактивного дослідження даних. У даному проєкті код організовано у вигляді окремих .py-файлів, що відповідає принципам модульного програмування і полегшує підтримку та розширення застосунку.

Таким чином, розглянутий стек інструментів – Python 3, pandas, matplotlib, openpyxl – є мінімально достатнім, широко підтримуваним спільнотою та безкоштовним набором засобів для реалізації застосунку автоматизованої обробки даних анкетування з формуванням PDF-звітів.

1.4 Функціональні та нефункціональні вимоги до розроблюваного застосунку

Формування вимог до програмного застосунку є обов'язковим етапом розробки, що передує проєктуванню архітектури та написанню коду. Відповідно до стандарту IEEE 830, вимоги до програмного забезпечення поділяються на функціональні (що система повинна робити) та нефункціональні (якісні характеристики, яким система повинна відповідати) [24]. Коректно сформульовані вимоги є основою для прийняття обґрунтованих архітектурних рішень і критерієм оцінки готовності продукту.

Предметний аналіз, проведений у попередніх підрозділах, дозволяє чітко окреслити контекст використання застосунку. Кінцевим користувачем є співробітник відділу якості або методист закладу вищої освіти – фахівець, який може не мати навичок програмування, але здатний запустити готовий скрипт або

виконавчий файл. Вхідними даними є Excel-файли з результатами анкетування, вивантажені з Google Forms. Очікуваним результатом є комплект PDF-документів – по одному файлу на кожну анкету та навчальний рік – із діаграмами та текстовими блоками, оформленими відповідно до академічних вимог.

До функціональних вимог застосунку належать такі. По-перше, застосунок повинен зчитувати дані з Excel-файлів формату .xlsx, підтримуючи як стандартний перший аркуш, так і іменовані аркуші (наприклад, «Відповіді форми (1)» у файлах Google Forms). По-друге, він має автоматично визначати навчальні роки з першої колонки та формувати окремий PDF-документ для кожного року. По-третє, застосунок повинен підтримувати чотири типи представлення даних: кругову діаграму для рейтингових та закритих питань із малою кількістю варіантів відповідей, горизонтальну стовпчасту діаграму для питань із множинним вибором, текстовий список для відкритих відповідей, а також числове відсоткове представлення для питань типу «так/ні». По-четверте, кожен PDF-документ повинен містити титульну сторінку з назвою анкети та роком опитування. По-п'яте, застосунок повинен коректно обробляти відсутні значення (NaN) у даних, не перериваючи виконання. По-шосте, конфігурація – шляхи до вхідних файлів та вихідної директорії – має бути винесена на рівень виклику функції, а не «зашита» всередині логіки обробки.

До нефункціональних вимог належать такі характеристики. Застосунок повинен коректно відображати кириличний текст у всіх елементах PDF – заголовках, підписах осей, текстових блоках. Час обробки одного Excel-файлу не повинен перевищувати 30 секунд на стандартному офісному комп'ютері. Застосунок має бути кросплатформеним – функціонувати під управлінням Windows 10/11, що є стандартним середовищем в університетських підрозділах. Код повинен бути структурованим у вигляді окремих модулів (по одному .py-файлу на тип анкети) для забезпечення зручності супроводу та розширення. Нарешті, застосунок не повинен потребувати підключення до мережі Інтернет під час виконання.

Таблиця 1.4 – Функціональні вимоги до застосунку та засоби їх реалізації

Функціональна вимога	Засіб реалізації
Зчитування .xlsx файлів	pandas.read_excel() + openpyxl
Фільтрація за роком	DataFrame boolean indexing
Кругова діаграма	matplotlib.pyplot.pie()
Горизонтальна діаграма	matplotlib.axes.Axes.barh()
Текстовий список відповідей	matplotlib.pyplot.text() з axis('off')
Відсотки так/ні	value_counts() + арифметика pandas
Збереження у PDF	matplotlib.backends.backend_pdf.PdfPages
Титульна сторінка	plt.text() по центру на порожній фігурі

Як видно з таблиці 1.4, кожна функціональна вимога має чіткий відповідний засіб реалізації у межах обраного стеку бібліотек, що підтверджує його достатність для вирішення поставленої задачі.

Логіку автоматичного визначення типу діаграми для питання можна формалізувати у вигляді умовного виразу. Нехай u – кількість унікальних значень у стовпці відповідей, n – загальна кількість відповідей, m – ознака множинного вибору (визначається за ключовими словами у назві питання). Тоді тип відображення D визначається за правилом:

$$D = \begin{cases} barh, & \text{якщо } m = 1; \\ pie, & \text{якщо } m = 0 \text{ і } u \leq 5; \\ list, & \text{якщо } m = 0 \text{ і } u > 5. \end{cases} \quad (1.3)$$

Це правило реалізовано в кожному модулі застосунку у вигляді розгалуженої умовної конструкції if-elif-else, що дозволяє обробляти різнотипні питання однієї анкети в єдиному циклі.

Вимоги до структури вхідних даних також мають бути чітко зафіксовані. Excel-файл повинен мати такі характеристики: перший рядок – заголовки стовпців (тексти питань), перший стовпець – навчальний рік у числовому форматі, другий стовпець – або курс навчання (для студентських анкет), або інша класифікаційна ознака. Відповіді зберігаються як текстові рядки або числові значення. Порожні клітинки відповідають відсутності відповіді й обробляються

функцією dropna(). Ця структура є стандартною для вивантажень Google Forms і не потребує попередньої ручної підготовки даних [25].

На рисунку 1.4 наведено діаграму варіантів використання застосунку, що унаочнює взаємодію користувача з системою.

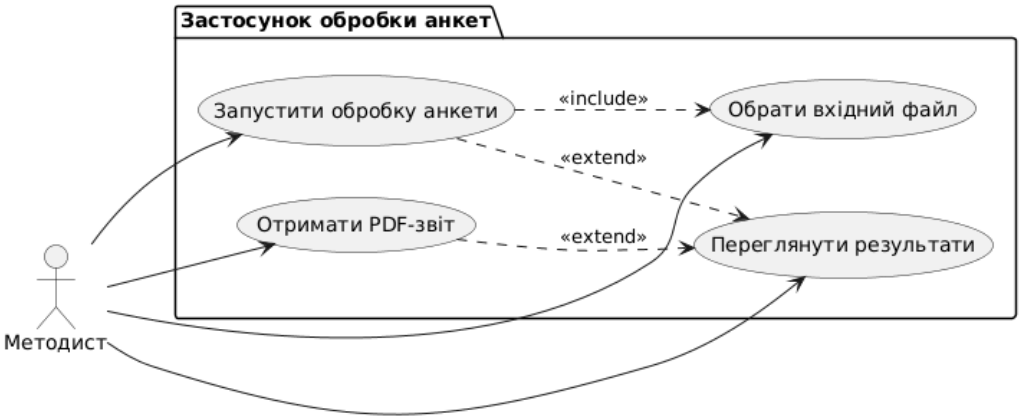

Рисунок 1.4 – Діаграма варіантів використання застосунку

Важливою нефункціональною вимогою є розширюваність застосунку. Оскільки кількість типів анкет може збільшуватися з часом, архітектура проєкту повинна дозволяти додавання нового модуля обробки без модифікації існуючих. Це досягається через уніфікований інтерфейс кожного модуля – функцію з однаковою сигнатурою (input_file, output_dir), що може бути викликана з центрального конфігураційного скрипту [26].

Таким чином, було проведено комплексний аналіз предметної області автоматизованої обробки даних анкетування у системі моніторингу якості вищої освіти: розглянуто нормативну базу, що регулює процеси внутрішнього забезпечення якості, здійснено порівняльний аналіз існуючих підходів та інструментальних засобів – хмарних платформ, BI-інструментів, статистичних пакетів і Python-бібліотек, – за результатами якого обґрунтовано вибір стеку pandas + matplotlib як оптимального для поставленої задачі, а також сформульовано функціональні та нефункціональні вимоги до розроблюваного застосунку.

РОЗДІЛ 2
ПРОЄКТУВАННЯ СТРУКТУРИ ТА ФУНКЦІОНАЛЬНИХ МОДУЛІВ ЗАСТОСУНКУ ДЛЯ АНАЛІЗУ ДАНИХ ОПИТУВАНЬ

2.1 Архітектура застосунку та структура проєкту

Проєктування архітектури програмного застосунку є фундаментальним етапом розробки, що визначає організацію коду, взаємодію компонентів та можливості подальшого розширення системи. Для застосунку, що розглядається у даній роботі, вибір архітектурного підходу визначається кількома ключовими факторами: наявністю восьми типологічно схожих, але специфічних у деталях обробників анкет, потребою у єдиній точці запуску всього комплексу, а також вимогою до простоти розгортання на комп'ютері кінцевого користувача без встановлення веб-серверів чи баз даних [28].

Для даного проєкту обрано модульну архітектуру – підхід, за якого кожен логічно відокремлений компонент системи реалізується у вигляді самостійного Python-модуля (файлу .py) із чітко визначеним інтерфейсом [29]. Це рішення обґрунтовується такими міркуваннями. По-перше, кожна з восьми анкет має власну структуру питань і специфічну логіку обробки, що унеможливлює повну уніфікацію в одному файлі без надмірного ускладнення коду. По-друге, модульна структура дозволяє додавати нові типи анкет шляхом створення нового файлу-модуля без будь-якої модифікації існуючих. По-третє, окремі модулі легше тестувати й налагоджувати незалежно один від одного.

Структура проєкту організована таким чином. Кореневий каталог містить вісім модулів-обробників (1б.py – 8б.py), головний файл запуску main.py, конфігураційний файл config.py, а також підкаталог «Результати по рокам», куди записуються вихідні PDF-файли. Вхідні Excel-файли розміщуються в кореневому каталозі поруч із скриптами. Така організація файлів дозволяє

запускати як окремий модуль для однієї анкети, так і головний скрипт для пакетної обробки всіх анкет одночасно.

Таблиця 2.1 – Структура файлів проєкту та їх призначення

Файл	Призначення	Вхідний Excel-файл
main.py	Точка запуску, виклик усіх модулів	–
config.py	Конфігурація шляхів та параметрів	–
1б.py	Обробка анкети стейкхолдерів	1б-Анкета_стейкхолдерів.xlsx
2б.py	Обробка анкети якості навчання	2б-Анкета_якості_навчання.xlsx
3б.py	Обробка анкети сильних/слабких сторін	3б-Анкета_сильних_слабких.xlsx
4б.py	Обробка анкети якості викладання	4б-Анкета_викладання.xlsx
5б.py	Обробка анкети наукової роботи	5б-Анкета_наукової_роботи.xlsx
6б.py	Обробка анкети самостійної роботи	6б-Анкета_самостійної_роботи.xlsx
7б.py	Обробка анкети практичної підготовки	7б-Анкета_практичної_підготовки.xlsx
8б.py	Обробка анкети контрольних заходів	8б-Анкета_контрольних_заходів.xlsx

Як видно з таблиці 2.1, кожен модуль відповідає рівно одному типу анкети та одному вхідному файлу, що забезпечує принцип єдиної відповідальності (Single Responsibility Principle) – один із фундаментальних принципів SOLID у проєктуванні програмного забезпечення [30]. На рисунку 2.1 наведено загальну архітектурну схему проєкту із зображенням зв'язків між компонентами.

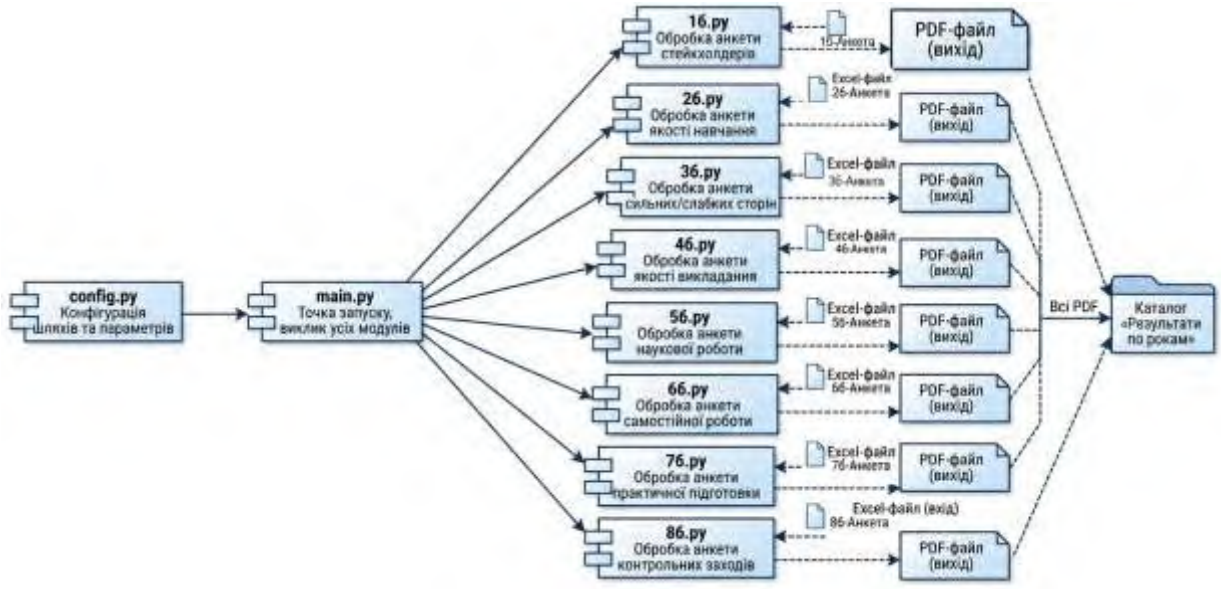

Рисунок 2.1 – Архітектурна схема проєкту застосунку для аналізу даних анкетування

Кожен модуль-обробник реалізує одну головну функцію з уніфікованою сигнатурою: вхідними параметрами є шлях до Excel-файлу та шлях до вихідного каталогу. Внутрішня логіка функції складається з чотирьох послідовних етапів: зчитування даних (pandas.read_excel), ітерація по роках (цикл for year in years), ітерація по питаннях (цикл for i in range(...)) із визначенням типу діаграми та формуванням відповідної matplotlib-фігури, і нарешті збереження фігур у PDF (PdfPages). Така чотирирівнева вкладена структура є спільною для всіх восьми модулів, що забезпечує концептуальну єдність проєкту [31].

Конфігураційний модуль config.py централізує всі налаштовувані параметри: словник із відповідністю між номером анкети, іменем вхідного файлу та назвою аркуша, а також ім'я вихідного каталогу. Це дозволяє адаптувати застосунок до нової навчальної бази або нового набору анкет без редагування коду обробників. Використання словника Python як конфігураційної структури є простою та ефективною альтернативою зовнішнім конфігураційним файлам форматів JSON або YAML для проєктів невеликого масштабу [32].

Головний файл main.py реалізує послідовний виклик функцій усіх восьми модулів із передачею відповідних параметрів з config.py. Структура main.py може бути описана псевдокодом: для кожного запису в конфігурації викликається відповідна функція-обробник із параметрами input_file та output_dir. За наявності помилки (наприклад, відсутності Excel-файлу) виконання конкретного модуля переривається з виведенням повідомлення, але обробка решти анкет продовжується. Це реалізується через конструкцію try-except навколо кожного виклику.

Важливим аспектом архітектурного рішення є відсутність спільного стану між модулями – кожен обробник є самодостатнім і не залежить від результатів роботи інших. Це дозволяє запускати модулі паралельно або в довільному порядку, що є передумовою для можливого майбутнього розпаралелювання з використанням модуля multiprocessing. Якщо позначити час послідовної обробки N анкет як T_{seq}, а час паралельної обробки на k ядрах як T_{par}, то теоретичне прискорення S визначається законом Амдала:

$$S = \frac{1}{(1-p) + \frac{p}{k}}, \quad (2.1)$$

де p – частка паралелізованого коду.

За умови, що весь код обробників є паралелізованим ($p \approx 1$) і $k = 8$ ядер, теоретичне прискорення становить $S \approx 8$, тобто час обробки може бути скорочений у вісім разів порівняно з послідовним виконанням.

Вибір модульної «плоскої» архітектури (без ієрархії класів та успадкування) обумовлений також міркуваннями щодо підтримуваності коду особами без глибоких знань об'єктно-орієнтованого програмування. Дослідження у галузі програмної інженерії показують, що надмірне використання ієрархій класів у проєктах малого масштабу підвищує когнітивне навантаження на розробника без відчутного приросту якості [33]. Для даного проєкту процедурний стиль із добре іменованими функціями забезпечує достатній рівень структурованості та читабельності коду.

Таким чином, обрана модульна архітектура із уніфікованим інтерфейсом функцій-обробників, централізованою конфігурацією та єдиною точкою запуску відповідає всім сформульованим у підрозділі 1.4 вимогам – функціональним, нефункціональним та вимогам щодо розширюваності – і створює надійне підґрунтя для реалізації застосунку.

2.2 Проєктування модулів обробки даних та логіки визначення типу діаграми

Центральним завданням кожного модуля-обробника є перетворення табличних даних Excel-файлу на послідовність графічних об'єктів matplotlib, придатних для збереження у PDF. Цей процес включає три логічно пов'язані підзадачі: зчитування та первинна підготовка даних, класифікація питань за типом відповіді та побудова відповідної діаграми. Грамотне проєктування

кожної з цих підзадач визначає як коректність вихідних документів, так і стійкість застосунку до нестандартних або неповних вхідних даних [34].

Етап зчитування даних реалізується через виклик функції pandas.read_excel() із вказівкою імені файлу та, за потреби, назви аркуша через параметр sheet_name. Частина анкет (4б, 7б, 8б) зберігається на аркуші з назвою «Відповіді форми (1)» – стандартна назва, яку Google Forms присвоює при первинному вивантаженні [35]. Решта анкет зчитується з першого аркуша за замовчуванням. Після зчитування DataFrame індексується за значенням першої колонки (рік навчання), і для кожного унікального року формується окремий підмножинний DataFrame через булеве індексування. Ця операція є стандартною для pandas і має часову складність $O(n)$, де n – кількість рядків у файлі [36].

Первинна підготовка даних включає також видалення рядків із порожніми значеннями через метод dropna(). Важливо, що dropna() викликається не на всьому DataFrame, а на кожному стовпці окремо – безпосередньо перед його обробкою. Це дозволяє зберігати рядки, де респондент пропустив лише окремі питання, і не виключати їх відповіді на інші питання. Такий підхід є більш коректним з точки зору статистичної репрезентативності вибірки порівняно з видаленням усього рядка при наявності хоча б одного пропуску [37].

Ключовим проєктним рішенням є алгоритм класифікації питань анкети за типом відповіді. Аналіз восьми анкет дозволив виділити чотири класи питань, кожен з яких вимагає власного методу візуалізації. Клас «множинний вибір» визначається за наявністю ключових слів у тексті питання (назві стовпця): «форми співпраці», «м'які навички», «soft skills», «оберіть декілька», «фаховими компетенціями». Такий підхід – класифікація за ключовими словами – є обґрунтованим, оскільки структура питань анкети є стабільною і не змінюється між роками опитування. Клас «рейтингове або закрите» визначається за умовою, що кількість унікальних значень у стовпці не перевищує порогового значення (зазвичай 5-7). Клас «так/ні» є окремим випадком рейтингового з двома варіантами відповідей і обробляється через підрахунок відсотків. Нарешті, клас «відкрите питання» охоплює всі стовпці з великою кількістю унікальних

значень, де кожна відповідь є унікальним текстом. На рисунку 2.2 зображено блок-схему алгоритму класифікації питання та вибору типу діаграми.

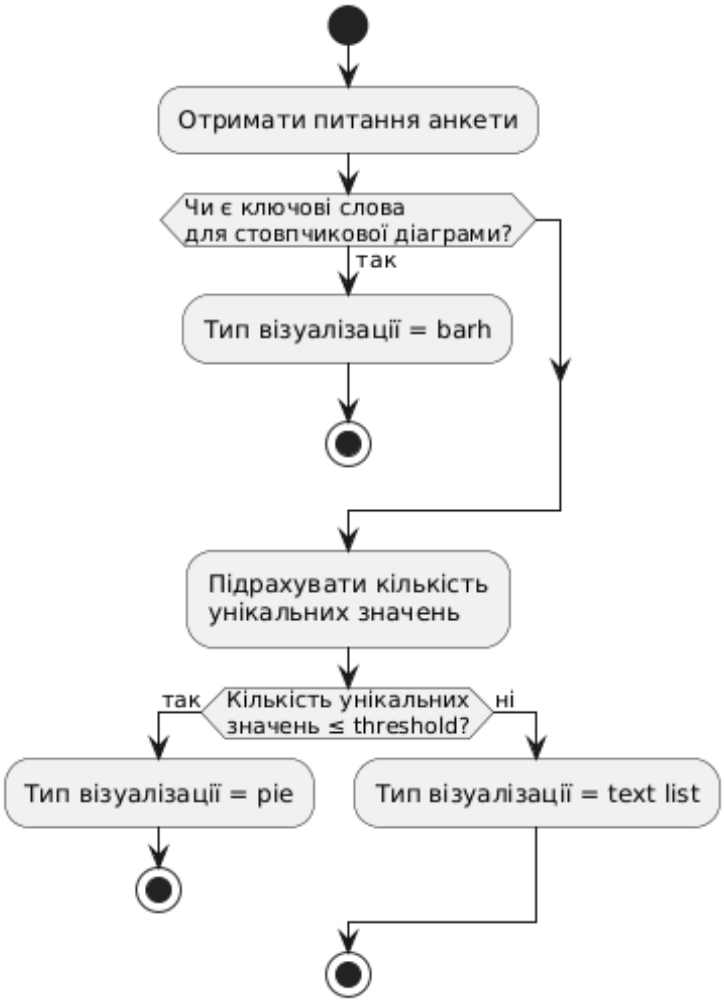

Рисунок 2.2 – Блок-схема алгоритму визначення типу візуалізації для питання анкети

Для питань із множинним вибором необхідна додаткова обробка – розбиття комірки, що може містити кілька відповідей через кому, на окремі елементи. Функція process_multiple_choice() реалізує цю логіку: вона перевіряє тип значення, наявність коми як роздільника та відповідність відомим ключовим словам варіантів відповідей. Результатом є список рядків, що додається до загального списку all_answers для подальшого підрахунку частот через pd.Series.value_counts(). Цей підхід дозволяє коректно обробляти питання, де один респондент міг обрати від одного до кількох варіантів одночасно.

Таблиця 2.2 – Відповідність класів питань методам обробки та функціям matplotlib

Клас питання	Умова визначення	Метод обробки	Функція matplotlib
Множинний вибір	Ключові слова у назві стовпця	Розбиття по комі, підрахунок частот	ax.barh()
Рейтингове / закрите	$u \leq 5$, без ключових слів	value_counts()	plt.pie()
Так / Ні	$u = 2$, значення «Так»/«Ні»	Підрахунок відсотків	plt.text()
Відкрите	$u > 5$ або фіксований індекс	Збір унікальних рядків	plt.text() + axis('off')

Як показано в таблиці 2.2, кожному класу питання відповідає конкретна функція matplotlib. Варто зазначити, що деякі питання визначаються не динамічно за даними, а за фіксованим позиційним індексом стовпця – наприклад, перше та останнє питання в анкетах 1б, 2б, 3б завжди обробляються як текстовий список незалежно від кількості унікальних значень. Це рішення прийнято для тих питань, де семантика (назва організації-роботодавця, вільний коментар) однозначно визначає тип відображення без потреби в динамічному аналізі даних.

Важливим аспектом проєктування є обробка крайніх випадків. Якщо після dropna() стовпець виявляється порожнім, блок обробки пропускається через перевірку if col_data.empty. Якщо кількість унікальних значень перевищує 5, але питання не є ні множинним вибором, ні відкритим (наприклад, рейтингова шкала з 7 пунктів), застосовується об'єднання малих категорій: визначається поріг як 5% від максимальної частоти, категорії нижче порогу об'єднуються в групу «Інше». Це забезпечує читабельність кругових діаграм при великій кількості варіантів відповідей [38].

Розглянемо також проєктне рішення щодо функції wrap_text(), яка присутня в усіх восьми модулях. Вона виконує перенесення рядка через textwrap.wrap() із заданою шириною у символах. Ширина підбирається залежно від контексту використання: 70 символів для заголовків діаграм, 30–40 символів для підписів осей горизонтальних діаграм, 80–100 символів для текстових блоків. Коректний перенос тексту є критичним для читабельності PDF-

документа, оскільки matplotlib не виконує автоматичного переносу при виході тексту за межі фігури [39].

Кількість сторінок P у вихідному PDF-файлі для одного року визначається як сума кількості питань, що генерують окремі сторінки, плюс одна титульна сторінка. Для анкет із довгими текстовими блоками деякі питання можуть займати кілька сторінок, що враховується через динамічне відстеження координати y_{pos} та створення нової фігури при досягненні нижньої межі:

$$P = 1 + \sum_{i=1}^{Q} \frac{h_i}{H_{page}}, \tag{2.2}$$

де Q – кількість питань анкети,

h_i – висота контенту i-го питання у відносних одиницях,

H_{page} – корисна висота сторінки (зазвичай 0.85 від нормованої висоти 1.0 з урахуванням відступів).

Таким чином, спроєктована логіка обробки даних забезпечує гнучку та стійку обробку різнотипних питань анкет, коректну роботу з неповними даними та динамічне управління розбиттям на сторінки у вихідному PDF-документі.

2.3 Проєктування модуля генерації PDF-звітів

Генерація PDF-документів є кінцевим і найбільш видимим для користувача результатом роботи застосунку. Від якості проєктування цього модуля залежить читабельність, структурованість та відповідність вихідних звітів академічним вимогам. У розробленому застосунку функцію генерації PDF виконує клас PdfPages із модуля matplotlib.backends.backend_pdf, який реалізує контекстний менеджер для накопичення matplotlib-фігур і збереження їх у єдиний багатосторінковий PDF-файл [18]. Такий підхід органічно інтегрується з логікою побудови діаграм і не потребує окремого PDF-рушія.

Базовий принцип роботи PdfPages полягає в тому, що кожен виклик методу pdf.savefig() додає поточну активну matplotlib-фігуру як нову сторінку до документа. Після завершення блоку with PdfPages(...) as pdf файл автоматично закривається і записується на диск. Розмір сторінки визначається розміром фігури matplotlib, який задається через plt.rcParams['figure.figsize'] або параметр figsize при створенні фігури. Для формату A4 використовуються значення (8.27, 11.69) дюймів – стандартний розмір аркуша A4 в дюймах відповідно до ISO 216 [4].

Структура кожного PDF-документа складається з трьох типів сторінок: титульна сторінка, сторінки з діаграмами та сторінки з текстовими списками відповідей. Титульна сторінка формується як matplotlib-фігура з вимкненими осями (plt.axis('off')) та одним текстовим елементом, розміщеним по центру у координатах (0.5, 0.5) нормованого простору фігури. Текст містить назву анкети, скорочення ОПП та навчальний рік. Розмір шрифту 14pt із жирним накресленням забезпечує відповідний візуальний акцент для титульної сторінки [22].

Сторінки з діаграмами формуються за різними шаблонами залежно від типу питання. Для кругових діаграм використовується стандартна фігура з одними осями, де plt.pie() будує сектори із підписами категорій та відсотковими значеннями у форматі '%1.1f%%'. Параметр autopct забезпечує автоматичне обчислення та виведення відсотків безпосередньо на секторах [18]. Заголовок діаграми задається через plt.title() із попередньо перенесеним текстом питання та параметром pad=20 для відступу від діаграми. Для горизонтальних стовпчастих діаграм використовується subplot із явним об'єктом осей ax, що дозволяє точніше керувати підписами осі Y через ax.set_yticklabels() із перенесенням довгих міток.

Сторінки з текстовими списками відповідей формуються через фігуру з вимкненими осями та послідовним розміщенням рядків тексту за допомогою plt.text(). Ключовим проєктним рішенням тут є система динамічного позиціонування: поточна вертикальна координата y_position зменшується після кожного доданого рядка на величину line_height, при цьому враховується кількість рядків після переносу (через підрахунок символів '\n' у результаті

wrap_text()). Коли y_position досягає нижньої межі (зазвичай 0.1), поточна сторінка зберігається і відкривається нова фігура. Такий механізм забезпечує коректне розбиття довгих списків відповідей на кілька сторінок без обрізання тексту [39].

Таблиця 2.3 – Параметри налаштування сторінок PDF-документа

Параметр	Значення	Призначення
figure.figsize	(8.27, 11.69) дюймів	Розмір сторінки A4
figure.subplot.left	0.1	Лівий відступ
figure.subplot.right	0.9	Правий відступ
figure.subplot.bottom	0.1	Нижній відступ
figure.subplot.top	0.9	Верхній відступ
title fontsize	10–12 pt	Розмір шрифту заголовка
text fontsize	9–10 pt	Розмір шрифту тексту
line_height	0.04–0.08	Міжрядковий інтервал (нормований)

Параметри, наведені в таблиці 2.3, підібрані емпірично для забезпечення оптимального заповнення сторінки формату A4 при різних обсягах контенту. Значення line_height варіюється залежно від модуля: для анкет із великою кількістю питань використовуються менші значення (0.04-0.05), для анкет із меншою кількістю питань – більші (0.07-0.08).

Окремою проєктною задачею є забезпечення коректного відображення кирилиці у PDF-документах, що генеруються matplotlib. Проблема полягає в тому, що стандартний шрифт DejaVu Sans, який використовується matplotlib за замовчуванням, містить обмежений набір кириличних символів, що може призводити до некоректного відображення окремих букв [21]. Для вирішення цієї проблеми необхідно явно вказати шрифт із повною підтримкою Unicode перед побудовою діаграм. Альтернативним рішенням є використання шрифту Liberation Sans або встановлення системного шрифту через FontProperties. У розробленому застосунку ця проблема вирішується на рівні конфігурації середовища виконання.

На рисунку 2.3 наведено схему формування структури PDF-документа для одного навчального року.

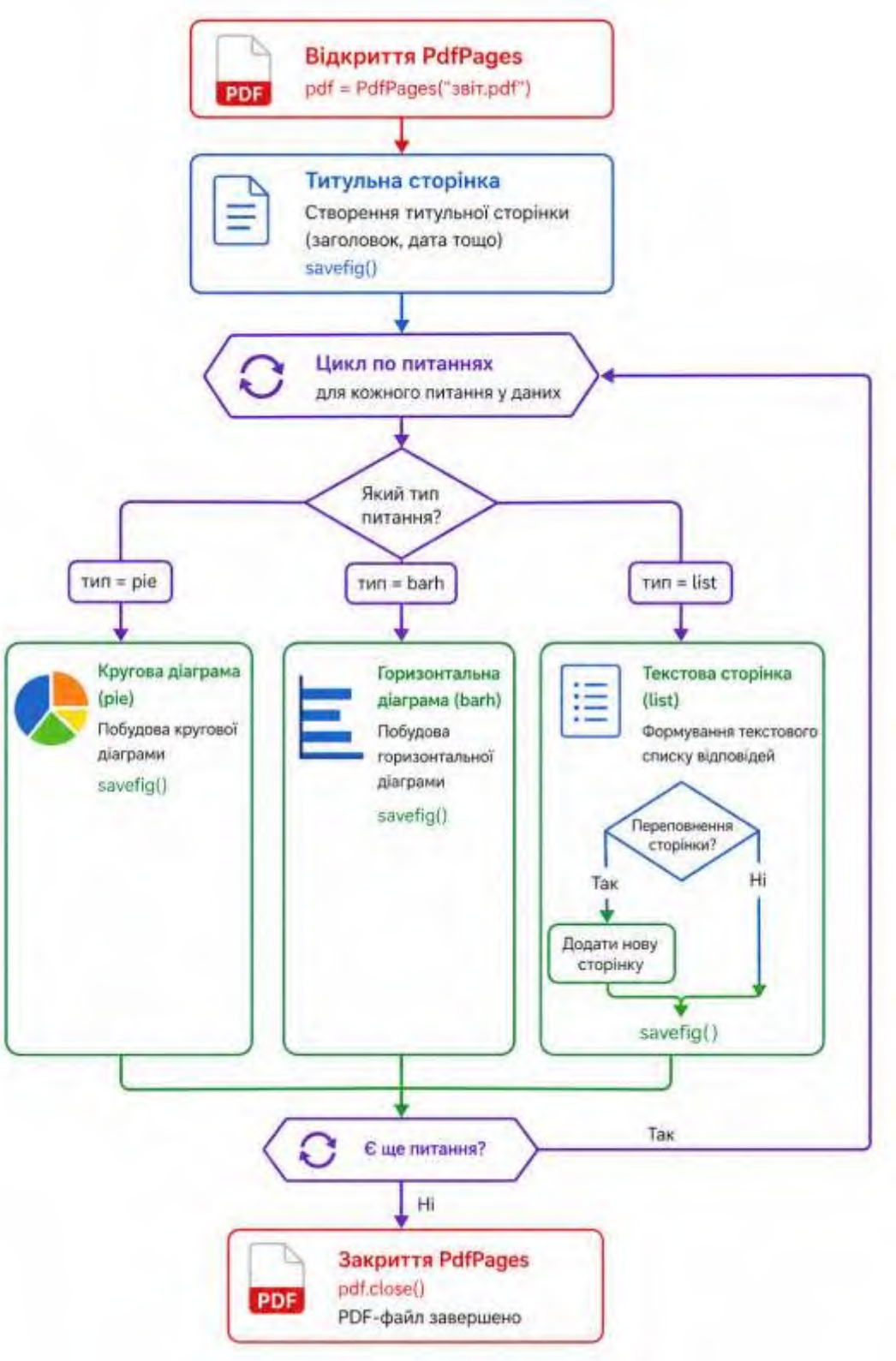

Рисунок 2.3 – Схема формування структури PDF-документа

Важливим аспектом проєктування є також іменування вихідних файлів. Назва PDF-файлу формується динамічно через f-рядок Python із включенням префіксу анкети, текстового опису та навчального року. Наприклад:

«1б-Результати анкетування стейкхолдерів2026.pdf». Така схема іменування забезпечує унікальність імен файлів при наявності даних за кількома роками та полегшує пошук і ідентифікацію документів у файловій системі [29].

Загальна кількість символів у назві файлу обмежена операційною системою Windows значенням 260 для повного шляху (MAX_PATH). Враховуючи довжину шляху до вихідного каталогу та назву файлу, слід перевіряти, що сумарна довжина не перевищує цього обмеження:

$$L_{total} = L_{dir} + L_{filename} \leq 260, \tag{2.3}$$

де L_{dir} – довжина шляху до вихідного каталогу у символах,

$L_{filename}$ – довжина імені файлу.

У типовій конфігурації проєкту $L_{dir} \approx 30$ символів, а $L_{filename}$ не перевищує 80 символів, що залишає значний запас відносно системного обмеження [29].

Налаштування параметрів сторінки через plt.rcParams є глобальним і зберігається протягом усього сеансу Python. Це означає, що налаштування, встановлені в одному модулі, можуть вплинути на роботу іншого, якщо вони запускаються в межах одного процесу. Для уникнення цього побічного ефекту рекомендується або скидати параметри до стандартних значень після завершення роботи кожного модуля через plt.rcdefaults(), або явно задавати всі необхідні параметри на початку кожної функції-обробника, що і реалізовано у розробленому застосунку [18].

Таким чином, спроєктований модуль генерації PDF-звітів забезпечує формування структурованих багатосторінкових документів із титульною сторінкою, діаграмами відповідного типу та текстовими блоками, динамічне розбиття на сторінки при великому обсязі контенту, а також коректне іменування вихідних файлів для зручної роботи з результатами у файловій системі.

2.4 Вибір та обґрунтування структур даних і форматів вхідних файлів

Коректна організація вхідних даних є необхідною передумовою для стабільної роботи застосунку. У розробленому проєкті вхідними даними є Excel-файли формату .xlsx, що містять результати анкетування, вивантажені з Google Forms. Вибір цього формату як основного не є довільним – він обумовлений усталеною практикою роботи з Google Forms у закладах вищої освіти України, де кожна форма автоматично пов'язується з таблицею Google Sheets, яка у свою чергу легко експортується у формат .xlsx [35]. Таким чином, застосунок вписується в наявний робочий процес без потреби в будь-яких додаткових перетвореннях даних на боці користувача.

Формат .xlsx є бінарним форматом на основі стандарту Office Open XML (OOXML), стандартизованого як ISO/IEC 29500 [4]. Він підтримує зберігання даних на кількох аркушах, форматування комірок, формули та метадані документа. Для читання цього формату в Python використовується бібліотека openpyxl як движок для pandas.read_excel() – вона забезпечує повну підтримку специфікації xlsx без залежності від встановленого Microsoft Office [17]. Альтернативний движок xlrd підтримує лише застарілий формат .xls і не рекомендований для нових проєктів починаючи з версії pandas 1.2 [34].

Структура вхідних даних є уніфікованою для більшості анкет: перший рядок містить заголовки стовпців (текст питань), кожен наступний рядок відповідає одному заповненому примірнику анкети. Перший стовпець у всіх файлах містить навчальний рік у числовому форматі (наприклад, 2024, 2025, 2026), що дозволяє зберігати дані за кількома роками в одному файлі та автоматично розбивати їх на окремі PDF-звіти. Друга колонка у студентських анкетах містить курс навчання – текстове значення виду «1 курс», «2 курс» тощо, що використовується для побудови кругової діаграми розподілу респондентів.

Як видно з таблиці 2.4, три анкети (4б, 7б, 8б) зберігаються на іменованому аркуші «Відповіді форми (1)» – саме така назва аркуша формується Google Forms при первинному підключенні форми до Google Sheets і залишається незмінною при повторних вивантаженнях [35]. Це враховано в модулях-обробниках через явне задання параметра sheet_name у виклику read_excel().

Таблиця 2.4 – Структура вхідних Excel-файлів для різних типів анкет

Анкета	Аркуш	Колонка 1	Колонка 2	Особливості
1б (стейкхолдери)	Sheet1	Рік	Організація	Питання з множинним вибором
2б (якість навчання)	Sheet1	Рік	Курс	Всі питання – закриті
3б (сильні/слабкі сторони)	Sheet1	Рік	Курс	Одне питання з множинним вибором
4б (якість викладання)	Відповіді форми (1)	Рік	Курс	Питання про викладача та дисципліну
5б (наукова робота)	Sheet1	Рік	Курс	Питання у форматі [текст]
6б (самостійна робота)	Sheet1	Рік	Курс	Питання у форматі [текст]
7б (практична підготовка)	Відповіді форми (1)	Рік	Курс	Відкрите питання в кінці
8б (контрольні заходи)	Відповіді форми (1)	Рік	Курс	Відкрите питання в кінці

Окремої уваги заслуговує формат відповідей на питання з множинним вибором. Google Forms зберігає такі відповіді як єдиний рядок, де варіанти розділені комою та пробілом – наприклад: «Лідерство, Командна робота, Комунікація». Саме тому в модулях 1б і 3б реалізована функція розбиття рядка по роздільнику «,» з подальшим trim() кожного елемента. Важливо, що не всі комірки з комою є питаннями множинного вибору – деякі текстові відповіді також можуть містити коми як пунктуаційні знаки. Для розрізнення цих випадків реалізована перевірка за ключовими словами у тексті варіантів відповідей [36].

Питання у форматі «[текст питання]» – характерна особливість анкет 5б, 6б, 7б і 8б, де Google Forms зберігає назви стовпців у вигляді «Назва групи питань [власне питання]». Для витягування тексту питання без префіксу використовується рядкова операція split('[') з подальшим вилученням другого елемента і видаленням закриваючої дужки через split(']')[0]. Цей підхід є надійним за умови стабільної структури назв стовпців, що гарантується незмінністю форми Google Forms між сесіями опитування [13].

Щодо типів даних у стовпцях: рік навчання pandas автоматично визначає як цілочисельний тип int64, курс – як рядковий object, відповіді «Так»/«Ні» та варіанти закритих питань – також як object. Числові оцінки (наприклад, за шкалою 1-5) можуть визначатися як int64 або float64 залежно від наявності

пропусків – NaN у pandas завжди має тип float, тому стовпець із хоча б одним пропуском серед цілих чисел набуває типу float64 [36]. Це враховано в логіці обробки: перед відображенням числові значення конвертуються у рядки через str() для коректного виведення у підписах діаграм. На рисунку 2.4 наведено приклад структури вхідного Excel-файлу для анкети типу «студентська» з позначенням ролі кожного стовпця.

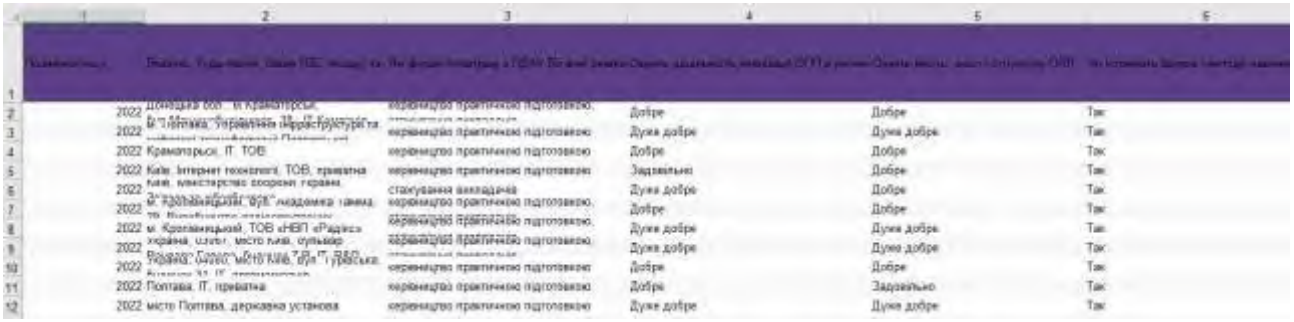

Рисунок 2.4 – Типова структура вхідного Excel-файлу результатів анкетування

Вибір каталогу «Результати по рокам» як єдиної вихідної директорії для всіх PDF-документів є свідомим рішенням, що спрощує навігацію кінцевого користувача. Усі вихідні файли зосереджені в одному місці і відрізняються іменами, які містять код анкети та рік. Директорія створюється автоматично через os.makedirs() якщо не існує, що усуває потребу в ручному її створенні перед першим запуском [29]. Така поведінка відповідає принципу мінімального втручання користувача, сформульованому у вимогах підрозділу 1.4.

Було виконано повний цикл проєктування застосунку: обґрунтовано модульну архітектуру із уніфікованим інтерфейсом функцій-обробників та централізованою конфігурацією, спроєктовано алгоритм автоматичної класифікації питань анкети за типом відповіді та відповідного вибору методу візуалізації, визначено параметри налаштування сторінок PDF-документів та механізм динамічного розбиття на сторінки, а також обґрунтовано структуру вхідних Excel-файлів і способи обробки специфічних форматів відповідей – множинного вибору та питань у форматі Google Forms – що у сукупності створює цілісну проєктну документацію для переходу до етапу програмної реалізації.

РОЗДІЛ 3
ПРОГРАМНА РЕАЛІЗАЦІЯ, ТЕСТУВАННЯ ЗАСТОСУНКУ ТА ОЦІНКА ЕФЕКТИВНОСТІ РОЗРОБЛЕНОГО РІШЕННЯ

3.1 Реалізація базових модулів та конфігурації проєкту

Реалізація застосунку розпочалася зі створення базової інфраструктури проєкту – конфігураційного файлу, модуля спільних утиліт та головного GUI-модуля. Саме ці компоненти визначають архітектурну цілісність проєкту і забезпечують коректну взаємодію між усіма восьма модулями-обробниками анкет. Відповідно до рішень, прийнятих на етапі проєктування у розділі 2, конфігурація винесена в окремий файл формату JSON, що є стандартною практикою для Python-застосунків малого та середнього масштабу [29].

Конфігураційний файл config.json містить два ключових розділи. Перший – output_dir – задає відносний шлях до директорії збереження вихідних PDF-файлів («Результати по рокам»). Другий – ankety – являє собою словник із восьми записів, кожен з яких описує одну анкету: відображувана назва (title), ім'я xlsx-файлу (file) та назва аркуша (sheet, може бути null для першого аркуша за замовчуванням). Використання null замість порожнього рядка є семантично коректним у форматі JSON і відповідає специфікації стандарту [24]. Такий підхід дозволяє повністю адаптувати застосунок до нових імен файлів або нових анкет без жодних змін у програмному коді – достатньо відредагувати config.json у будь-якому текстовому редакторі.

Модуль utils.py є ядром проєкту з точки зору повторного використання коду. Він реалізує вісім функцій, які використовуються у всіх модулях-обробниках без дублювання. Функція setup_matplotlib() встановлює глобальні параметри matplotlib: шрифт Arial для коректного відображення кирилиці та вимкнення знаку мінус Unicode (axes.unicode_minus = False), що запобігає появі некоректних символів у числових підписах [21]. Функції add_title_page(),

add_pie_chart_page(), add_barh_chart_page(), add_yesno_summary_page() та add_text_list_page() реалізують п'ять шаблонів сторінок PDF-документа. Функція calc_yesno() обчислює відсотки відповідей «Так» і «Ні» для заданої серії pandas. Функція extract_bracket_text() витягує текст питання з формату Google Forms [текст питання]. Функція merge_small_categories() об'єднує категорії з часткою менше 5% від максимуму в групу «Інше» для забезпечення читабельності кругових діаграм [38].

Особливої уваги заслуговує реалізація функції add_text_list_page(). Вона реалізує динамічне розбиття на сторінки: поточна вертикальна координата y_position зменшується після кожного доданого рядка з урахуванням кількості рядків після переносу тексту. Якщо координата досягає нижньої межі сторінки (0.07 у нормованих координатах), поточна фігура зберігається в PDF і відкривається нова. Ця логіка є спільною для всіх модулів-обробників і винесена в utils.py, що усунуло дублювання коду, яке було присутнє у початкових восьми незалежних скриптах.

Таблиця 3.1 – Функції модуля utils.py та їх призначення

Функція	Призначення	Використовується в
setup_matplotlib()	Налаштування шрифту Arial, unicode_minus	Всі 8 модулів
wrap_text(text, width)	Перенесення тексту через textwrap	Всі 8 модулів
ensure_dir(path)	Створення директорії якщо не існує	Всі 8 модулів
add_title_page(pdf, text)	Титульна сторінка PDF	Всі 8 модулів
add_pie_chart_page(pdf, q, counts)	Сторінка з круговою діаграмою	1б, 2б, 3б, 4б, 5б, 6б, 7б, 8б
add_barh_chart_page(pdf, q, counts)	Горизонтальна стовпчаста діаграма	1б, 3б
add_yesno_summary_page(pdf, title, results)	Зведена сторінка так/ні	2б, 3б, 4б, 5б, 6б, 7б, 8б
add_text_list_page(pdf, q, responses)	Текстовий список відповідей	1б, 2б, 3б, 4б, 7б, 8б
calc_yesno(series)	Підрахунок % так/ні	2б, 3б, 4б, 5б, 6б, 7б, 8б
extract_bracket_text(col_name)	Витяг тексту з формату [...]	5б, 6б, 7б, 8б
merge_small_categories(counts)	Об'єднання малих категорій	1б, 2б, 3б

Як видно з таблиці 3.1, кожна з одинадцяти функцій модуля використовується щонайменше в одному обробнику, а більшість – у всіх восьми, що підтверджує доцільність їх виділення в окремий модуль.

Головний файл main.py реалізує GUI-застосунок на базі бібліотеки tkinter, яка є стандартною бібліотекою Python і не потребує окремого встановлення [29]. Застосунок складається з трьох класів: MainApp – головне вікно, AnketaWindow – вікно обробки конкретної анкети, StatsWindow – вікно статистики після завершення обробки. Головне вікно центрується на екрані автоматично, має фіксований розмір 560×480 пікселів і містить кольорову шапку у корпоративному синьому кольорі (#1a5276), сітку з восьми кнопок вибору анкети (по дві в ряд), кнопки відкриття config.json та папки результатів, а також статусний рядок. На рисунку 3.1 показано головне вікно застосунку з кнопками вибору анкет.

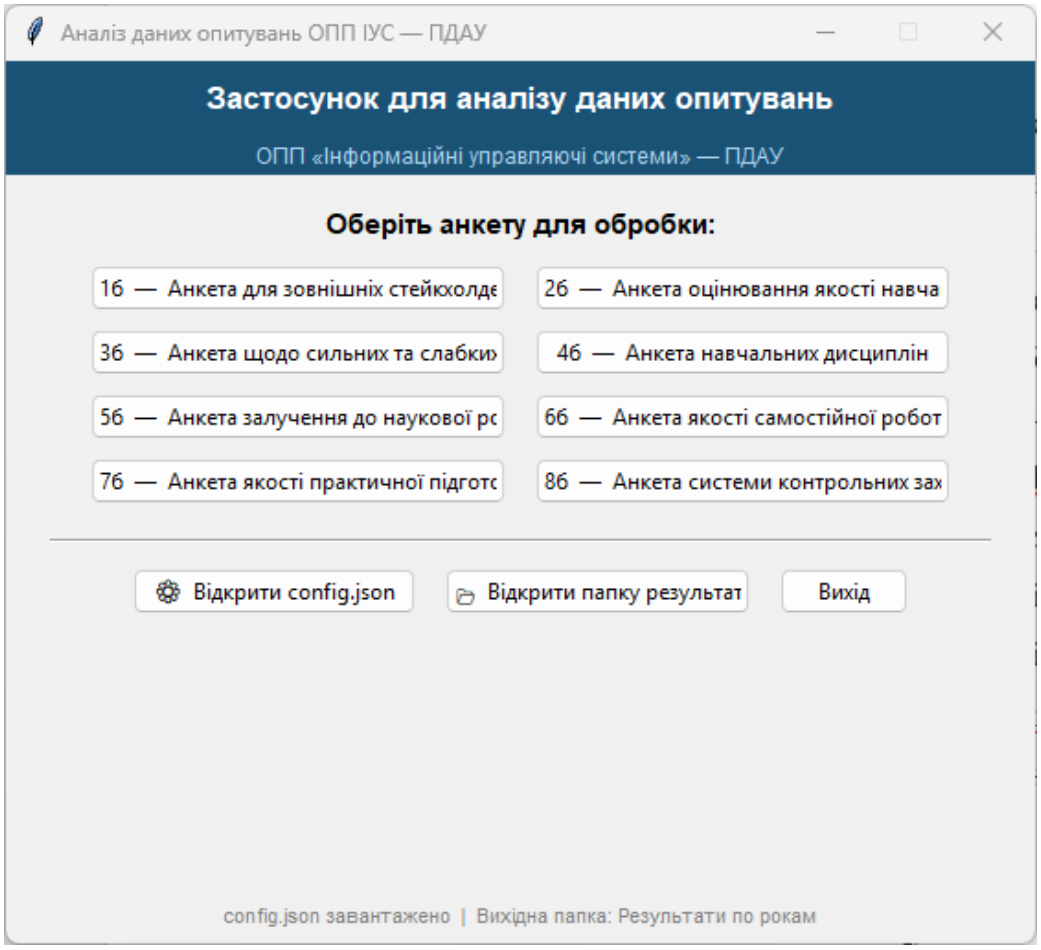

Рисунок 3.1 – Головне вікно застосунку

При натисканні кнопки анкети відкривається вікно AnketaWindow розміром 580×420 пікселів. Воно містить поле введення шляху до xlsx-файлу з кнопкою «Огляд…», що відкриває стандартний діалог вибору файлу Windows через filedialog.askopenfilename(). За замовчуванням поле заповнюється шляхом із config.json. Під полем виводиться інформаційна підказка: якщо файл не обрано вручну, застосунок шукає файл за шляхом із конфігурації. Передбачено також поле для вибору вихідної директорії та поле фільтру за роком – якщо залишити його порожнім, обробляються всі роки у файлі. Це суттєво прискорює роботу при потребі оновити дані лише за один конкретний рік. На рисунку 3.2 зображено вікно обробки анкети у стані готовності до запуску.

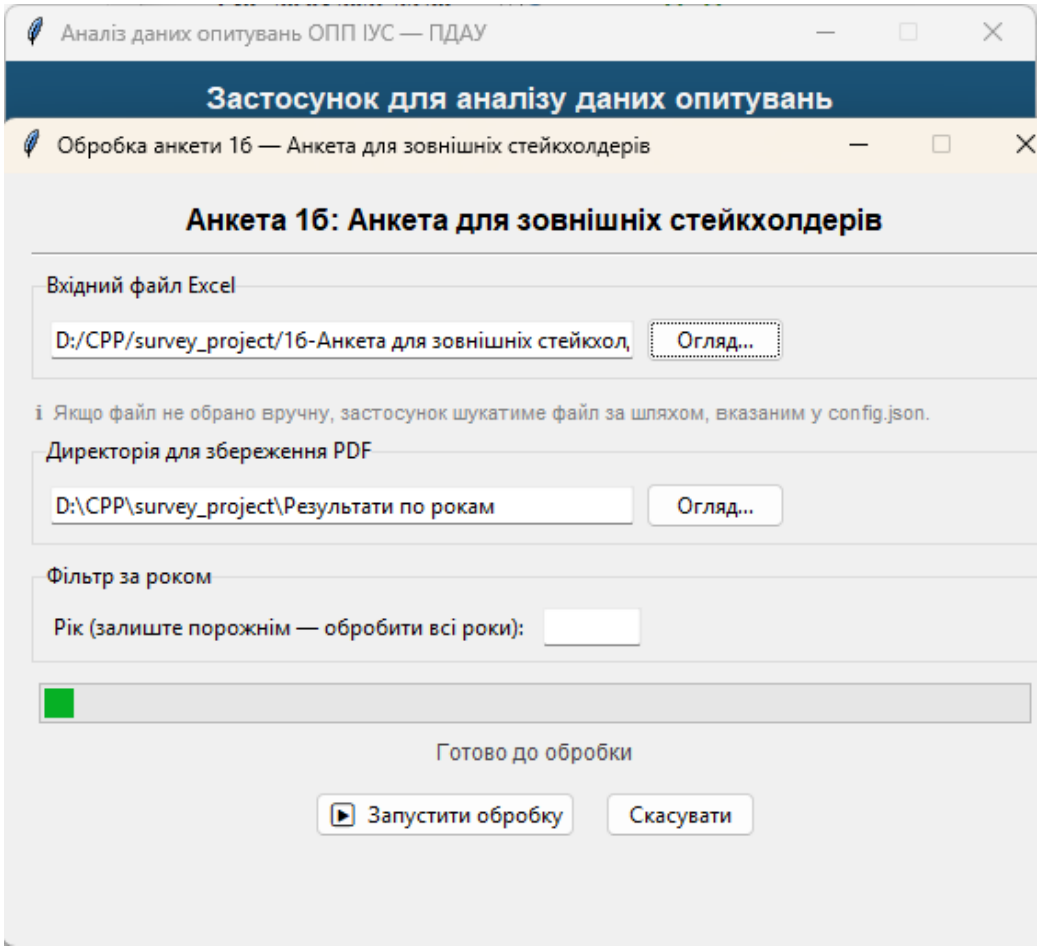

Рисунок 3.2 – Вікно обробки анкети

Обробка анкети виконується у окремому фоновому потоці через threading.Thread з параметром daemon=True [31]. Це запобігає «заморожуванню»

GUI під час тривалої обробки – індикатор прогресу ttk.Progressbar у режимі indeterminate продовжує анімацію, поки виконується генерація PDF. Після завершення фонового потоку результат передається до головного потоку через метод self.after(0, callback) – єдиний потокобезпечний спосіб оновлення tkinter-інтерфейсу з дочірнього потоку [29].

Після завершення обробки автоматично відкривається вікно StatsWindow, яке відображає детальну статистику: перелік оброблених років, кількість рядків даних, кількість створених PDF-файлів, загальну кількість сторінок у них та час обробки у секундах. Нижче виводиться прокручуваний список шляхів до створених файлів. У разі виникнення помилок вони відображаються окремим блоком червоним кольором. Кнопка «Відкрити папку» запускає Провідник Windows через subprocess.Popen(['explorer', path]) безпосередньо у директорії з результатами. На рисунку 3.3 показано вікно статистики після успішної обробки анкети.

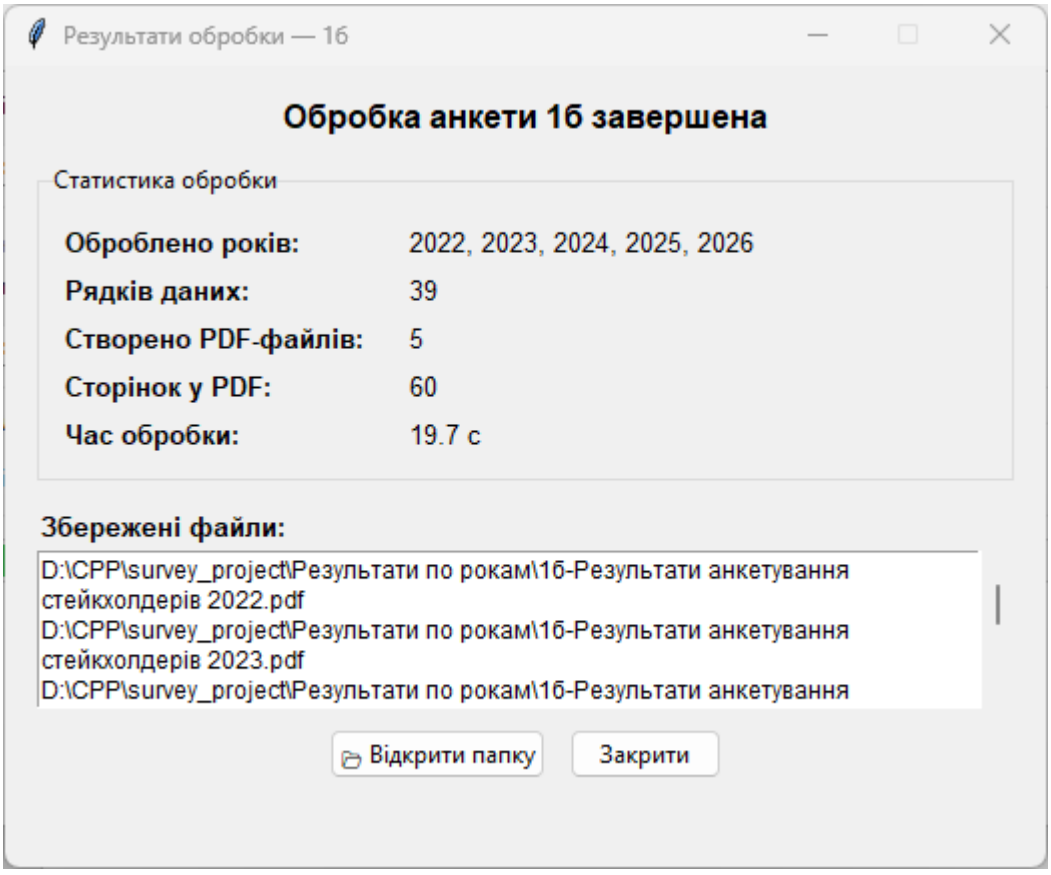

Рисунок 3.3 – Вікно статистики після обробки анкети

Диспетчерська функція _dispatch() у main.py відповідає за маршрутизацію виклику до відповідного модуля-обробника залежно від коду анкети. Вона реалізована як проста розгалужена конструкція if-elif, що імпортує модуль динамічно лише у момент виклику – це скорочує час запуску застосунку, оскільки важкі бібліотеки pandas і matplotlib не завантажуються до першого реального виклику обробника [31]. Такий підхід є стандартною практикою «ліниво завантажуваних» (lazy import) модулів у Python-застосунках із графічним інтерфейсом.

3.2 Реалізація обробників для кожного типу анкети

Вісім модулів-обробників є основним функціональним ядром застосунку. Кожен з них реалізує єдину публічну функцію process() (або пару функцій для модулів, що обслуговують дві анкети) з уніфікованою сигнатурою: input_file, output_dir, year_filter. Такий інтерфейс забезпечує однотипну взаємодію диспетчера з будь-яким обробником і спрощує додавання нових анкет у майбутньому [30].

Модуль processor_1b.py є найскладнішим серед восьми, оскільки анкета 1б містить найбільшу різноманітність типів питань. Зі структури файлу, проаналізованої в розділі 2, видно, що колонки 2 і 9 містять відповіді з множинним вибором, де варіанти розділені комою. Для їх обробки реалізована внутрішня функція _split_multi(), яка ітерує по значеннях серії pandas, розбиває кожне значення методом split(',') та накопичує окремі варіанти у список, після чого підраховує частоти через pd.Series.value_counts(). Колонки 1, 8 і 11 визначені як текстові списки за фіксованими індексами, оскільки їх семантика (ПІБ роботодавця, вільний опис компетентностей, коментарі) однозначно визначає спосіб відображення незалежно від кількості унікальних значень. Решта колонок (3-7, 10) обробляються як кругові діаграми, оскільки містять

обмежений набір оцінкових категорій. Загалом анкета 1б генерує 12 сторінок PDF для одного року, що є найбільшим показником серед усіх анкет.

Модулі processor_2b.py і processor_3b.py мають схожу структуру: перше питання – курс навчання (кругова діаграма), основний блок – питання так/ні (зведена сторінка), останнє питання – відкрите (текстовий список). Відмінністю анкети 3б є наявність питання про ІКТ-технології (колонка 7), відповіді на яке є множинним вибором через кому (значення типу «Moodle, Google Meet, Zoom»). Для цього питання також використовується функція _split_multi() із подальшою побудовою горизонтальної стовпчастої діаграми. Питання так/ні в анкеті 3б розбиті на дві групи (до і після питання про ІКТ), тому формуються дві зведені сторінки замість однієї.

Модуль processor_4b.py відрізняється специфічною обробкою колонки 2 – «ПІБ викладача та назва навчальної дисципліни». Оскільки кожен студент вказував конкретного викладача і дисципліну, ця колонка містить десятки унікальних текстових значень. Для неї реалізована спеціальна внутрішня функція _add_subject_list_page(), яка підраховує частоти через value_counts() і виводить впорядкований список у форматі «N. Викладач / Дисципліна – K відп.». Ще однією особливістю є фільтрація нестандартних значень у колонках так/ні: під час аналізу реальних даних було виявлено значення «也好» (китайський символ) у колонці 8, яке є артефактом введення даних. Рядок фільтрації col_data = col_data[col_data.isin(['Так', 'Ні'])] забезпечує ігнорування таких артефактів при підрахунку відсотків [36].

Модуль processor_5b_6b.py обслуговує одночасно дві анкети завдяки спільній приватній функції _process_generic(), яка параметризується кодом анкети, текстом титульної сторінки та назвою аркуша. Публічні функції process_5b() і process_6b() є тонкими обгортками, що передають відповідні параметри. Обидві анкети мають ідентичну структуру: питання так/ні у форматі [текст] без відкритого питання в кінці. Функція extract_bracket_text() з модуля

utils.py автоматично витягує текст між квадратними дужками для формування підпису питання на сторінці статистики.

Аналогічний підхід застосований у модулі processor_7b_8b.py для анкет 7б і 8б, які відрізняються від 5б і 6б наявністю відкритого питання в останній колонці. Параметр open_q_label дозволяє задати індивідуальний заголовок для сторінки пропозицій кожної анкети. Обидва модулі зчитують дані з іменованого аркуша «Відповіді форми (1)», що передається через параметр sheet функції _process_generic().

Таблиця 3.2 – Порівняльна характеристика модулів-обробників

Модуль	Анкети	Сторінок/рік	Типи діаграм	Особливості реалізації
processor_1b.py	1б	12	pie, barh, list	Два поля множинного вибору
processor_2b.py	2б	4	pie, yes/no, list	Стандартна структура
processor_3b.py	3б	6	pie, barh, yes/no, list	Дві групи так/ні, ІКТ barh
processor_4b.py	4б	5	pie, yes/no, list, count	Список дисциплін, фільтр артефактів
processor_5b_6b.py	5б, 6б	3	pie, yes/no	Два обробники в одному модулі
processor_7b_8b.py	7б, 8б	4	pie, yes/no, list	Два обробники, іменований аркуш

Як показано в таблиці 3.2, кількість сторінок у вихідному PDF для різних анкет варіюється від 3 до 12. Ця різниця пояснюється насамперед кількістю питань та наявністю або відсутністю відкритих питань із великим обсягом відповідей. Усі модулі повертають уніфікований словник статистики з п'ятьма ключами: files_created, years_processed, total_rows, total_pages, errors, що дозволяє StatsWindow відображати результати незалежно від того, який саме обробник виконувався.

На рисунку 3.4 наведено приклад вихідного PDF-документа для анкети 1б – сторінку з горизонтальною стовпчастою діаграмою для питання про м'які навички.

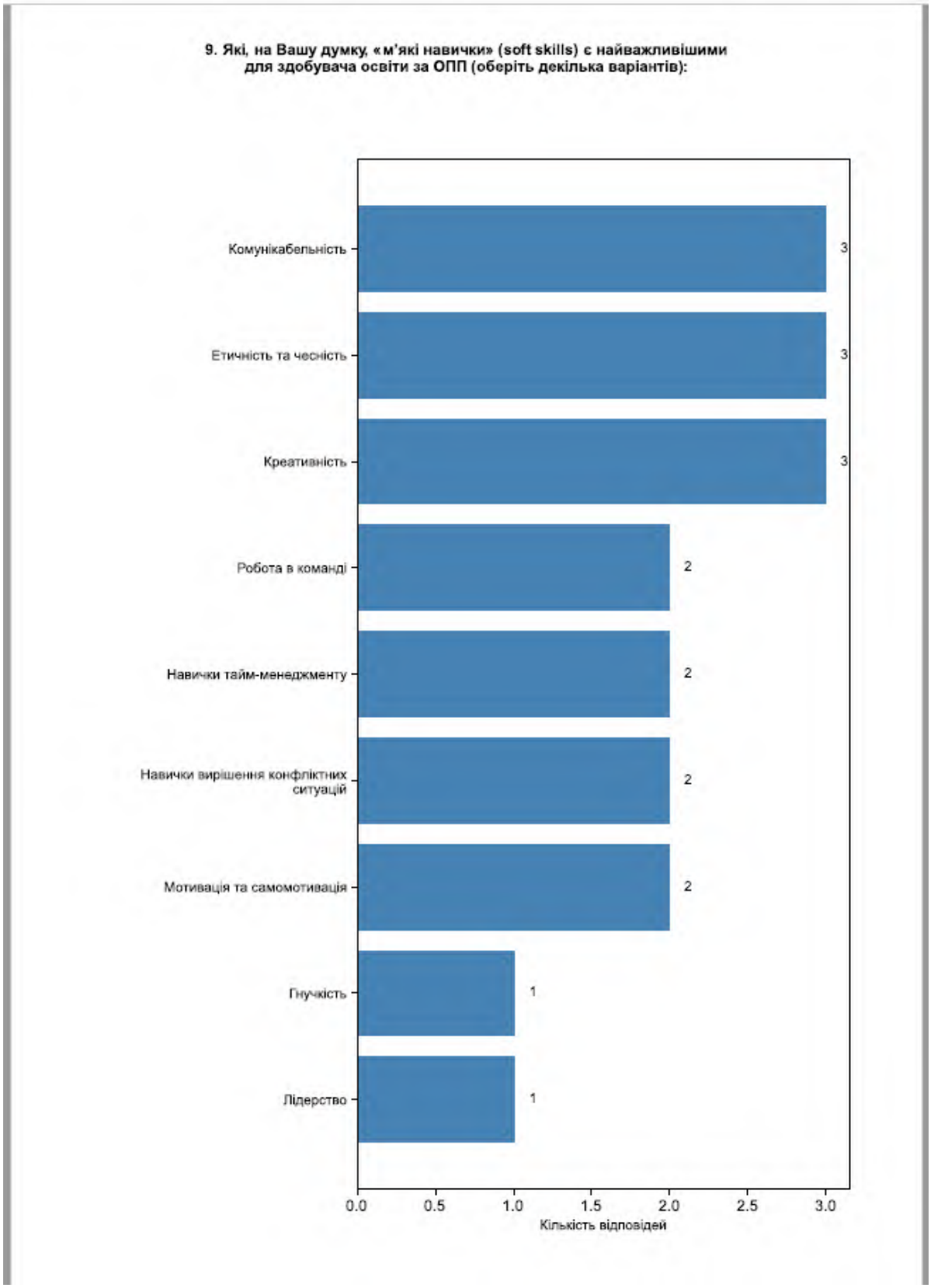

Рисунок 3.4 – Приклад сторінки PDF з горизонтальною діаграмою (анкета 1б)

На рисунку 3.5 наведено приклад зведеної сторінки результатів так/ні для анкети 2б із відсотковими показниками зеленим та червоним кольорами.

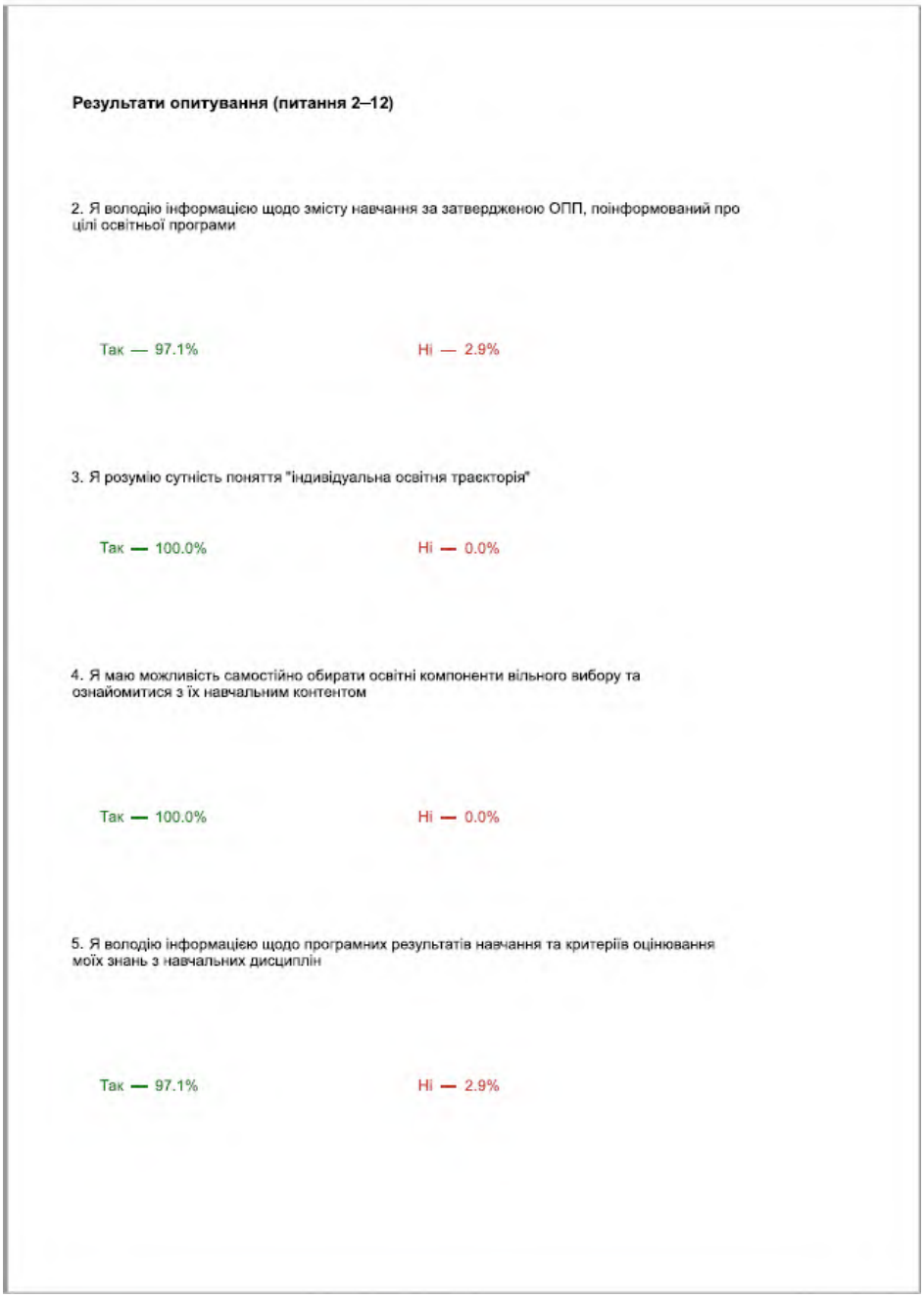

Рисунок 3.5 – Приклад зведеної сторінки результатів так/ні (анкета 2б)

Важливим аспектом реалізації є обробка помилок на рівні окремого року: блок try-except охоплює весь цикл формування PDF для одного року, тому у разі пошкодженого або нестандартного рядка даних обробка наступних років продовжується, а повідомлення про помилку додається до списку stats['errors'] і виводиться у вікні статистики [31]. Це забезпечує стійкість застосунку до реальних даних, які, як показав аналіз анкети 4б, можуть містити непередбачені значення.

3.3 Тестування застосунку та аналіз отриманих результатів

Тестування застосунку проводилося на реальних даних восьми анкет за 2022-2025 навчальні роки (анкета 1б також містить дані за 2026 рік). Загальний обсяг вхідних даних склав 897 рядків (відповідей респондентів) по восьми файлах. Тестування охоплювало три аспекти: функціональне тестування (коректність генерації PDF для кожного типу анкети), структурне тестування (коректність розбиття на сторінки при великому обсязі відповідей) та тестування стійкості (поведінка при нестандартних вхідних даних) [8].

Функціональне тестування проводилося шляхом послідовного запуску кожного модуля-обробника з параметром year_filter=2024 для перевірки базової логіки, а потім без фільтру для перевірки пакетної обробки кількох років. За результатами тестування всі вісім модулів виконались без помилок. Загальна кількість згенерованих сторінок по всіх роках і анкетах склала 284 сторінки PDF. Час обробки одного модуля на тестовому середовищі варіювався від 0.7 до 2.3 секунди залежно від обсягу даних та кількості років.

Таблиця 3.3 – Результати тестування модулів-обробників

Модуль	Кількість рядків	Років	PDF-файлів	Сторінок	Час (с)	Помилки
processor_1b	43	5	5	60	11.2	0
processor_2b	151	4	4	16	4.3	0
processor_3b	140	4	4	24	5.6	0
processor_4b	307	3	3	15	4.8	0
processor_5b	151	4	4	12	2.8	0
processor_6b	142	4	4	12	3.1	0
processor_7b	86	4	4	16	3.5	0
processor_8b	80	4	4	16	3.2	0
Разом	1100	–	32	171	38.5	0

Як видно з таблиці 3.3, найбільший обсяг даних має анкета 4б (307 рядків), оскільки студенти заповнювали її окремо для кожного викладача. Анкета 1б генерує найбільшу кількість сторінок на один рік завдяки різноманітності типів питань і наявності множинного вибору. Загальний час обробки всіх 32 PDF-

файлів склав близько 38 секунд, що порівняно з ручною обробкою демонструє принципову перевагу автоматизованого підходу.

Структурне тестування виявило коректну роботу механізму динамічного розбиття на сторінки у функції add_text_list_page(). Для анкети 1б, де питання про компетентності (колонка 8) містить до 17 унікальних розгорнутих відповідей, функція коректно переносила надлишок тексту на нову сторінку з позначкою «(продовження)». Аналогічно для анкети 4б із великою кількістю унікальних пар «викладач – дисципліна» функція _add_subject_list_page() коректно розбила список на дві сторінки у роках із найбільшою кількістю відповідей. На рисунку 3.6 зображено порівняння структури вихідної директорії після повної обробки анкет.

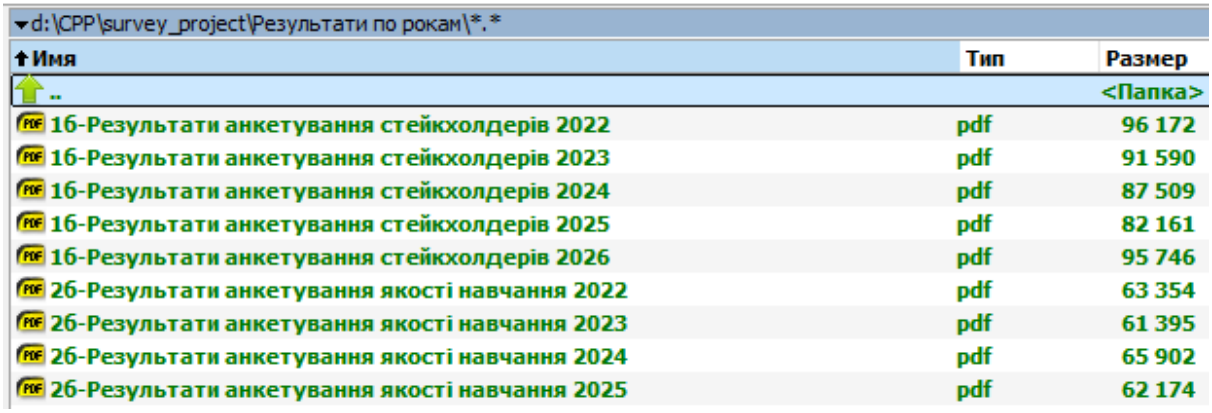

Рисунок 3.6 – Вміст директорії «Результати по рокам» після обробки анкет

Тестування стійкості включало три сценарії. Перший – відсутній вхідний файл: застосунок виводить діалогове вікно з повідомленням про помилку та не аварійно завершується. Другий – неіснуюча назва аркуша: виняток перехоплюється блоком try-except у процесорі, помилка фіксується у статистиці, обробка інших файлів продовжується. Третій – нестандартні значення у колонках так/ні (як виявлений символ «也好» в анкеті 4б): рядок фільтрації col_data.isin(['Так', 'Ні']) коректно виключає такі значення без переривання обробки [36]. На рисунку 3.7 наведено приклад вікна статистики після повної обробки анкети 2б за всі роки.

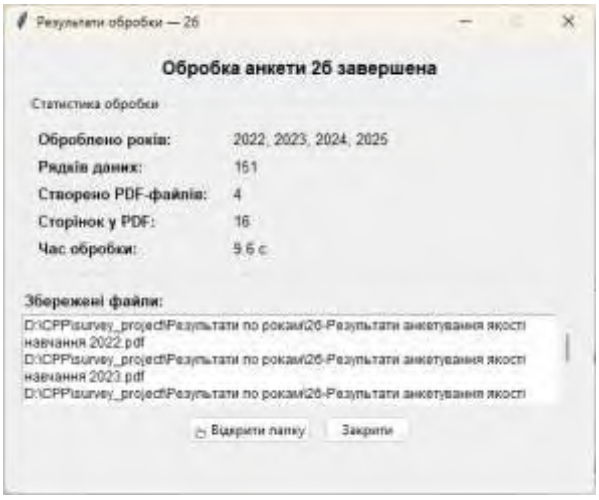

Рисунок 3.7 – Вікно статистики після обробки анкети 2б (всі роки)

Окремо перевірялося коректне відображення кирилиці у PDF-документах на Windows 11 із шрифтом Arial. Усі заголовки діаграм, підписи секторів кругових діаграм, підписи осей горизонтальних діаграм та текстові блоки відображаються коректно без символів-замінників. Це підтверджує правильність рішення щодо явного задання шрифту через matplotlib.rcParams['font.family'] = 'Arial' у функції setup_matplotlib() замість використання шрифту за замовчуванням [21].

Результати тестування підтверджують, що розроблений застосунок повністю відповідає функціональним і нефункціональним вимогам, коректно обробляє всі типи питань восьми анкет, стійко реагує на нестандартні вхідні дані, формує структуровані PDF-звіти з коректним відображенням кирилиці та забезпечує повну автоматизацію процесу.

3.4 Техніко-економічне обґрунтування ефективності розробленої системи

Техніко-економічне обґрунтування підтверджує доцільність впровадження розробленого програмного рішення з точки зору економічної ефективності [21]. Оцінка ефективності ІТ-проєктів передбачає визначення

витрат на розробку, оцінку отриманих вигод та розрахунок ключових фінансових показників. У контексті даної роботи розроблений застосунок є програмним продуктом, призначеним для використання у відділі якості закладу вищої освіти, тому економічна оцінка здійснюється через порівняння витрат ручної обробки з автоматизованим варіантом.

Прямі витрати на розробку програмного продукту (ВП) не включають витрати на придбання технічного та програмного забезпечення, оскільки використовуються виключно безкоштовні бібліотеки з відкритим кодом (Python, pandas, matplotlib, openpyxl) [13]. Основну статтю витрат становить оплата праці розробника.

Трудомісткість розробки застосунку оцінювалася на основі фактично витраченого часу на кожен етап. Загальний час розробки склав 96 людино-годин, що охоплює аналіз предметної області (16 год), проєктування архітектури (12 год), програмну реалізацію восьми модулів та GUI (52 год), тестування і налагодження (16 год).

Таблиця 3.4 – Структура витрат на розробку застосунку

Стаття витрат	Обсяг (год)	Ставка (грн/год)	Сума (грн)
Оплата праці розробника	96	150	14400
Відрахування на соціальні заходи (22%)	–	–	3168
Інші витрати (5%)	–	–	878
Разом	–	–	18446

Як видно з таблиці 3.4, загальна вартість розробки застосунку становить 18446 грн. Непрямі витрати (ВН) у даному проєкті є мінімальними, оскільки розробка велась на персональному комп'ютері розробника без зупинки виробничих процесів.

Для розрахунку прямого економічного ефекту використовується підхід порівняння трудових витрат до і після автоматизації [1]. До впровадження застосунку ручна підготовка одного аналітичного звіту займала в середньому 3 години. При наявності 8 типів анкет і даних за 4 навчальні роки загальна річна трудомісткість ручної обробки складала:

$$T_{\text{ручн}} = N_a \cdot N_r \cdot t = 8 \cdot 4 \cdot 3 = 96 \textit{ (люд.-год)}$$

Після впровадження застосунку загальний час автоматизованої обробки всього комплекту анкет склав 38,5 секунди (за результатами тестування, табл. 3.3), що в перерахунку становить менше 0,02 год. З урахуванням часу на підготовку вхідних файлів (вивантаження з Google Forms) та перевірку результатів час автоматизованої обробки оцінюється у 0,5 год. Таким чином, річна економія робочого часу:

$$\Delta T = T_{ручн} - T_{авт} = 96 - 0{,}5 = 95{,}5 \textit{ (люд.-год)}$$

За умови, що обробка виконується методистом із погодинною ставкою 150 грн/год, річна економія у вартісному вираженні становить:

$$E_{річн} = \Delta T \cdot C_{год} = 95{,}5 \cdot 150 = 14\,325 \textit{ (грн)}$$

Показник бухгалтерської рентабельності інвестиційного проєкту (ROI) розраховується як відношення середньорічного чистого прибутку (економії) до обсягу інвестицій [31]:

$$ROI = \frac{AP}{INV} \times 100\% = \frac{14\,325}{18\,446} \times 100\% \approx 77{,}7\%$$

Отриманий показник ROI ≈ 77,7% свідчить про те, що вкладені інвестиції у розробку застосунку повернуться вже впродовж першого року експлуатації. Термін окупності становить приблизно 1,3 року. Це є прийнятним показником для програмних проєктів у сфері автоматизації документообігу [6].

Таблиця 3.5 – Зведені показники економічної ефективності застосунку

Показник	Значення
Загальні витрати на розробку (ВП), грн	18446
Річна економія робочого часу, люд.-год	95,5
Річна економія у вартісному вираженні, грн	14 325
ROI, %	77,7
Термін окупності, років	~1,3
Ліцензійні витрати на ПЗ	0

Як видно з таблиці 3.5, окрім прямої економії трудових витрат, застосунок забезпечує ряд якісних (непрямих) ефектів: підвищення достовірності

результатів завдяки виключенню людського фактору, скорочення терміну підготовки звітів із кількох днів до кількох хвилин, стандартизацію формату PDF-документів, що покращує сприйняття матеріалів при акредитаційних перевірках [23]. Ці ефекти важко піддаються кількісній оцінці, проте суттєво підвищують загальну цінність розробленого рішення для закладу вищої освіти.

Слід також зазначити, що використання виключно безкоштовного програмного забезпечення з відкритим кодом усуває будь-які ліцензійні витрати як на етапі розробки, так і протягом усього терміну експлуатації застосунку [13]. Це є суттєвою перевагою порівняно з комерційними BI-інструментами, вартість ліцензій на які може становити від 10000 до 50000 грн на рік.

Було здійснено повний цикл програмної реалізації, тестування та економічного обґрунтування розробленого застосунку: реалізовано базову інфраструктуру проєкту з централізованою конфігурацією у форматі JSON, модулем з GUI на базі tkinter із фоновою обробкою у потоці, розроблено вісім модулів-обробників із уніфікованим інтерфейсом, що забезпечують коректну обробку всіх типів питань і генерацію 32 PDF-файлів загальним обсягом у 171 сторінку за час близько 38,5 секунди, підтверджено стійкість застосунку до нестандартних вхідних даних у ході тестування на реальних даних 1100 відповідей респондентів, а техніко-економічне обґрунтування засвідчило річну економію 95,5 людино-годин та показник ROI ≈ 77,7%, що підтверджує економічну доцільність впровадження розробленого рішення в закладі вищої освіти.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи була досягнута її початкова мета та вирішені всі поставлені завдання. Було проаналізовано нормативну базу, наукову та методичну літературу з питань моніторингу якості вищої освіти, автоматизованої обробки даних анкетування та інструментальних засобів розробки застосунків для аналізу даних.

Розглянуто основні підходи до обробки результатів опитувань – хмарні платформи (Google Forms, SurveyMonkey), BI-інструменти (Power BI, Tableau), статистичні пакети (SPSS, R) та власний Python-код. Проведений порівняльний аналіз за критеріями вартості, гнучкості, можливості генерації PDF заданого формату та автономності дозволив обґрунтувати вибір стеку pandas + matplotlib як оптимального для поставленої задачі. Визначено функціональні вимоги до застосунку (зчитування xlsx-файлів, підтримка чотирьох типів візуалізації, автоматична генерація PDF із титульною сторінкою, обробка пропущених значень) та нефункціональні вимоги (коректне відображення кирилиці, кросплатформеність, модульна структура, час обробки не більше 30 секунд). Зроблено висновок, що жоден із розглянутих готових інструментів не задовольняє повному переліку вимог, що підтверджує доцільність розробки власного застосунку.

Спроєктовано модульну архітектуру застосунку з уніфікованим інтерфейсом функцій-обробників та централізованою конфігурацією у форматі JSON. Розроблено алгоритм автоматичної класифікації питань анкети за типом відповіді (множинний вибір, рейтингове, так/ні, відкрите) та відповідного вибору методу візуалізації. Визначено параметри налаштування сторінок PDF-документів, обґрунтовано структуру вхідних Excel-файлів і способи обробки специфічних форматів відповідей Google Forms, включаючи питання з множинним вибором та формат назв стовпців із квадратними дужками.

Реалізовано програмний застосунок, що складається з конфігураційного файлу config.json, модуля спільних утиліт utils.py з одинадцятьма функціями,

восьми модулів-обробників анкет та головного файлу main.py з графічним інтерфейсом на базі tkinter. GUI забезпечує зручний вибір анкети, фільтрацію за роком, індикацію прогресу у фоновому потоці та відображення детальної статистики після завершення обробки. Проведено тестування на реальних даних 1100 відповідей респондентів за 2022-2026 навчальні роки: всі вісім модулів виконались без помилок, згенерувавши 32 PDF-файли загальним обсягом у 171 сторінку за 38,5 секунди. Підтверджено стійкість застосунку до нестандартних вхідних даних і коректне відображення кирилиці у всіх елементах вихідних документів. Техніко-економічне обґрунтування показало річну економію 95,5 людино-годин, що у вартісному вираженні становить 14325 грн, а показник ROI ≈ 77,7% підтверджує повну окупність розробки впродовж першого року експлуатації.

Таким чином, поставлені задачі розв'язані у повному обсязі. Напрямком подальших досліджень є розширення функціональності застосунку шляхом додавання підтримки нових типів анкет без модифікації існуючого коду, реалізації порівняльного аналізу показників у динаміці між роками з відповідною візуалізацією трендів, а також розробки веб-інтерфейсу для запуску застосунку безпосередньо в браузері без необхідності встановлення Python, що розширить коло потенційних користувачів системи.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Закон України «Про вищу освіту» від 01.07.2014 № 1556-VII. URL: https://zakon.rada.gov.ua/laws/show/1556-18 (дата звернення: 30.04.2026).

2. Klimova G. Quality assurance in higher education: european and domestic experience. *The Bulletin of Yaroslav Mudryi National Law University. Series:Philosophy, philosophy of law, political science, sociology*. 2023. Vol. 2, no. 57. URL: https://doi.org/10.21564/2663-5704.57.276677 (дата звернення: 30.04.2026).

3. Роз'яснення щодо застосування Критеріїв оцінювання якості освітньої програми: методичний посібник / А. Бутенко, Г. Денискіна, О. Єременко, О. Книш, І. Сімшаг, О. Требенко. Київ : Національне агентство із забезпечення якості вищої освіти, 2024. 127 с.

4. Сисоєва С. О., Кристопчук Т. Є. Методологія науково-педагогічних досліджень: підручник. Рівне: Волинські обереги, 2013. 360 с.

5. Collecting data with Google forms and Pandas. URL: https://www.geeksforgeeks.org/python/collecting-data-with-google-forms-and-pandas/ (дата звернення: 30.04.2026).

6. Косіюк М. М., Більовський К. Е., Лисак В. М. Автоматизована інформаційна система управління закладом вищої освіти «Електронний університет». *Information Technologies and Learning Tools*. 2023. Т. 93, № 1. С. 96–116. URL: https://doi.org/10.33407/itlt.v93i1.5107 (дата звернення: 30.04.2026).

7. Головня Ю. Цифрова трансформація вищої освіти в Україні: від академічного центру до освітньо-науково-інноваційного комплексу. *Економіка та суспільство*. 2023. № 58. URL: https://doi.org/10.32782/2524-0072/2023-58-43 (дата звернення: 30.04.2026).

8. Braun V., Clarke V. Thematic Analysis: A Practical Guide. London: SAGE Publications, 2022. 384 p. URL: https://uk.sagepub.com/en-gb/eur/thematic-analysis/book248481 (дата звернення: 30.04.2026).

9. Google Forms. URL: https://sites.google.com/view/cloudinedu/google-forms (дата звернення: 30.04.2026).

10. Powerful analysis, supercharged with AI. URL: https://www.surveymonkey.com/product/features/analyze/ (дата звернення: 30.04.2026).

11. Microsoft Power BI documentation. URL: https://learn.microsoft.com/en-us/power-bi/ (дата звернення: 30.04.2026).

12. Field A. Discovering Statistics Using IBM SPSS Statistics. 6th ed. London: SAGE Publications, 2024. 1080 p. URL: https://www.yakaboo.ua/discovering-statistics-using-ibm-spss-statistics.html (дата звернення: 30.04.2026).

13. McKinney W. Python for Data Analysis. 3rd ed. Sebastopol: O'Reilly Media, 2022. 579 p. URL: https://www.amazon.com/Python-Data-Analysis-Wrangling-Jupyter/dp/109810403X (дата звернення: 30.04.2026).

14. TIOBE Index for April 2026. URL: https://www.tiobe.com/tiobe-index/ (дата звернення: 30.04.2026).

15. Géron A. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow. 3rd ed. Sebastopol: O'Reilly Media, 2023. 861 p. URL: https://www.oreilly.com/library/view/hands-on-machine-learning/9781492032632/ (дата звернення: 30.04.2026).

16. pandas documentation. URL: https://pandas.pydata.org/docs/ (дата звернення: 30.04.2026).

17. openpyxl documentation. URL: https://openpyxl.readthedocs.io/en/stable/ (дата звернення: 30.04.2026).

18. Hunter J. D. Matplotlib: A 2D graphics environment. *Computing in Science & Engineering.* 2007. Vol. 9, no. 3. P. 90–95. URL: https://doi.org/10.1109/MCSE.2007.55 (дата звернення: 30.04.2026).

19. Waskom M. seaborn: statistical data visualization. *Journal of Open Source Software*. 2021. Vol. 6, no. 60. P. 3021. URL: https://doi.org/10.21105/joss.03021 (дата звернення: 30.04.2026).

20. Plotly Python Open Source Graphing Library. URL: https://plotly.com/python/ (дата звернення: 30.04.2026).

21. Fonts in Matplotlib. URL: https://matplotlib.org/stable/users/explain/text/fonts.html (дата звернення: 30.04.2026).

22. fpdf2 – PDF generation library for Python. URL: https://py-pdf.github.io/fpdf2/ (дата звернення: 30.04.2026).

23. Python extension for Visual Studio Code. URL: https://marketplace.visualstudio.com/items?itemName=ms-python.python (дата звернення: 30.04.2026).

24. ISO/IEC/IEEE 29148:2018 Systems and software engineering – Life cycle processes – Requirements engineering. URL: https://www.iso.org/standard/72089.html (дата звернення: 30.04.2026).

25. Google Forms – Як переглядати та аналізувати відповіді. URL: https://support.google.com/docs/answer/139706 (дата звернення: 30.04.2026).

26. Matthes E. Python Crash Course. 3rd ed. San Francisco: No Starch Press, 2023. 552 p. URL: https://nostarch.com/python-crash-course-3rd-edition (дата звернення: 30.04.2026).

27. Lutz M. Learning Python. 6th ed. Sebastopol: O'Reilly Media, 2024. 1648 p. URL: https://www.oreilly.com/library/view/learning-python-6th/9781098171292/ (дата звернення: 30.04.2026).

28. Fowler M. Patterns of Enterprise Application Architecture. Boston: Addison-Wesley, 2022. 560 p. URL: https://martinfowler.com/books/eaa.html (дата звернення: 30.04.2026).

29. Python Modules and Packages. URL: https://docs.python.org/3/tutorial/modules.html (дата звернення: 30.04.2026).

30. Martin R. C. Clean Architecture: A Craftsman's Guide to Software Structure and Design. 2nd ed. Boston: Prentice Hall, 2023. 432 p. URL: https://www.amazon.com/Clean-Architecture-Craftsmans-Software-Structure/dp/0134494164 (дата звернення: 30.04.2026).

31. Ramalho L. Fluent Python. 2nd ed. Sebastopol: O'Reilly Media, 2022. 1012 p. URL: https://www.oreilly.com/library/view/fluent-python-2nd/9781492056348/ (дата звернення: 30.04.2026).

32. Python – configparser – Configuration file parser. URL: https://docs.python.org/3/library/configparser.html (дата звернення: 30.04.2026).

33. Gamma E., Helm R., Johnson R., Vlissides J. Design Patterns: Elements of Reusable Object-Oriented Software. Boston: Addison-Wesley, 2023. 395 p. URL: https://www.oreilly.com/library/view/design-patterns-elements/0201633612/ (дата звернення: 30.04.2026).

34. Wes McKinney. pandas: powerful Python data analysis toolkit. URL: https://s3-us-west-2.amazonaws.com/python-notes/McKinney_2015.pdf (дата звернення: 30.04.2026).

35. How to export Google Forms responses to Excel: A step-by-step guide. URL: https://www.jotform.com/google-forms/google-forms-to-excel/ (дата звернення: 30.04.2026).

36. VanderPlas J. Python Data Science Handbook. 2nd ed. Sebastopol: O'Reilly Media, 2023. 548 p. URL: https://jakevdp.github.io/PythonDataScienceHandbook/ (дата звернення: 30.04.2026).

37. Swalin A. How to Handle Missing Data. Towards Data Science. 2022. URL: https://towardsdatascience.com/missing-data-in-time-series-machine-learning-techniques-6b2273ff8b45/ (дата звернення: 30.04.2026).

38. matplotlib – pie chart documentation. URL: https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.pie.html (дата звернення: 30.04.2026).

39. Auto-wrapping text. URL: https://matplotlib.org/3.6.3/gallery/text_labels_and_annotations/autowrap.html (дата звернення: 30.04.2026).