Полтавський університет економіки і торгівлі
Навчально-науковий інститут денної освіти
Форма навчання денна
Кафедра комп'ютерних наук та інформаційних технологій

Допускається до захисту
Завідувач кафедри
_____Олена ОЛЬХОВСЬКА
(підпис)
«___»_____202_ р.

КВАЛІФІКАЦІЙНА РОБОТА

на тему

«СТВОРЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ВІЗУАЛІЗАЦІЇ НАУКОВОЇ ПРОДУКТИВНОСТІ НПП КАФЕДРИ ТА ЇЇ АНАЛІЗУ»

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

Виконавець роботи Нелюба Богдан Віталійович
_____«__»_____202_р.
(підпис)
Науковий керівник к.ф.-м.н.,доц., Парфьонова Т.О
_____«__»_____202_р.
(підпис)
Рецензент

ПОЛТАВА 2025

РЕФЕРАТ

Записка: 53с., 5 рис., 1 додаток, 11 джерел.

СТВОРЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ВІЗУАЛІЗАЦІЇ НАУКОВОЇ ДІЯЛЬНОСТІ ВИКЛАДАЧІВ ТА ЇЇ АНАЛІЗ

Об'єктом розробки є інформаційна система, що призначена для автоматизованого збору, зберігання та графічного представлення даних про наукову продуктивність науково-педагогічного складу кафедри.

Мета роботи полягає у розробці програмного забезпечення, що дозволить здійснювати ефективний аналіз та моніторинг наукової діяльності викладачів, а також надасть зручний інтерфейс для перегляду, фільтрації та візуального відображення показників наукової активності.

Методи дослідження включають в себе: аналіз потреб користувачів; проектування структури бази даних; розробку серверної частини з використанням Flask; застосування сучасних вебтехнологій (HTML, CSS, Python) для створення адаптивного та інтерактивного інтерфейсу, а також методи візуалізації даних для представлення результатів.

В процесі роботи було проведено аналіз ключових показників наукової продуктивності, зокрема кількості публікацій, індексації наукових робіт, кількості цитувань та інших релевантних даних. Досліджено специфіку організації інформації про наукові досягнення та потреби користувачів у швидкому та зручному доступі до цієї інформації.

Створено архітектуру програмного забезпечення, яка містить базу даних, серверну частину та клієнтський вебінтерфейс. Інтерфейс розроблено з урахуванням принципів UX/UI-дизайну та адаптованості, що забезпечує коректне відображення на різних пристроях, починаючи від персональних комп'ютерів і завершуючи мобільними телефонами.

Програмна реалізація виконана із використанням HTML, CSS та Python (Flask). Здійснено тестування функціональності на різних екранах та

платформах, що підтвердило стабільність роботи системи та зручність її використання для кінцевих користувачів.

Результатом розробки став інструмент, який значно полегшує процес моніторингу та аналізу наукової діяльності кафедри, підвищує прозорість та оперативність прийняття управлінських рішень, а також сприяє мотивації викладачів до активнішої наукової роботи.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ

Умовні позначення, символи, скорочення, терміни	Пояснення умовних пояснень, скорочень, символів
НПП	Науково-педагогічні працівники
БД	База Даних
ПЗ	Програмне забезпечення

ВСТУП

В умовах стрімкого розвитку цифрових технологій та підвищення ролі наукометричних показників у вищій освіті виникає потреба у розробці спеціалізованих програмних інструментів для об'єктивного аналізу та представлення наукової продуктивності науково-педагогічних працівників. Цифровізація освіти і науки вимагає інструментів, котрі дозволяють не лише зберігати інформацію про наукові публікації, а й ефективно її обробляти, аналізувати та наочно демонструвати результати діяльності співробітників.

Актуальність теми пояснюється необхідністю створення інтерактивної системи, котра б автоматизувала процес обліку наукових публікацій та давала змогу адміністративному персоналу кафедри, факультету або закладу вищої освіти швидко й зручно оцінювати наукову активність окремих викладачів та науковців за різні часові періоди. Використання таких систем сприяє прозорості оцінювання, формуванню обґрунтованої кадрової політики, а також дозволяє ефективно планувати подальший розвиток наукових напрямів.

Програмне забезпечення, розроблене в межах цієї кваліфікаційної роботи, дозволяє завантажувати публікації у форматі PDF, додавати до них відповідних авторів, фільтрувати наукову продуктивність за часовими інтервалами, генерувати звіти у візуалізованому вигляді (графіки, таблиці тощо) та проводити базовий аналіз публікаційної активності.

Кваліфікаційна робота підготовлена згідно з чинними методичними рекомендаціями, з урахуванням сучасних підходів до побудови інформаційних систем та веб-застосунків.

Метою цієї кваліфікаційної роботи є створення програмної реалізації інтерактивної системи для візуалізації та аналізу наукової продуктивності науково-педагогічних працівників кафедри.

Об’єктом дослідження є цифрові засоби для обліку, аналізу та візуалізації наукової інформації. Предметом дослідження є програмне забезпечення для автоматизованого збору, обробки та представлення даних про наукову продуктивність НПП кафедри.

У процесі виконання кваліфікаційної роботи були застосовані методи теоретичного аналізу та узагальнення літературних джерел, що стосуються наукометрії, цифрової обробки даних, систем візуалізації та веб-розробки.

Особливу увагу приділено вивченню сучасних технологій створення інтерактивних веб-застосунків, зокрема HTML, CSS, JavaScript та Python у поєднанні з фреймворком Flask. Також використовувалися методи проєктування програмних систем, реалізації інтерфейсів користувача, аналізу ефективності програмного продукту та його апробації в тестовому середовищі.

ПОСТАНОВКА ЗАДАЧІ

Мета цього кваліфікаційного дослідження – створення програмного забезпечення для візуалізації та аналізу наукової продуктивності науково-педагогічних працівників (НПП) кафедри. Основна ідея цього програмного продукту полягає у впровадженні зручного та інтерактивного інструменту, що дає змогу автоматизувати процес обліку наукових публікацій, аналізувати активність авторів у визначені проміжки часу та генерувати наочні звіти. Розробити систему, яка у доступному, гнучкому та функціональному форматі забезпечить ефективну роботу з науковими даними кафедри.

Програмне забезпечення повинно мати зрозумілий та структурований веб-інтерфейс, який дозволяє користувачам без спеціальних технічних знань легко завантажувати публікації, переглядати звіти та фільтрувати інформацію за потрібними параметрами. Інтерфейс адаптовано до роботи на персональних комп’ютерах, з інтуїтивною навігацією та мінімалістичним дизайном, що сприятиме позитивному сприйняттю і зручності використання.

Для забезпечення повного функціоналу системи реалізовано можливість завантаження наукових публікацій із прив’язкою до відповідних авторів кафедри, фільтрацію активності за вибраний часовий проміжок (наприклад, рік або семестр), автоматичну генерацію графіків і таблиць для візуалізації наукової активності, пошук за прізвищем, назвою публікації, датою тощо.

Важливим аспектом розробки є забезпечення комфортного користувацького досвіду: система має швидко обробляти дані, оперативно реагувати на дії користувача, працювати стабільно та підтримувати різні браузери. Пріоритетом є простота у використанні та логічна структура.

Ключове завдання користувача полягає у введенні публікацій, створенні звітів та аналізі отриманих даних, що дозволить кафедрі об'єктивно оцінювати наукову активність співробітників, планувати розвиток

дослідницьких напрямів та приймати управлінські рішення на основі достовірної інформації.

У результаті виконання даної кваліфікаційної роботи буде створено веб-орієнтовану програмну систему, яка відповідатиме всім заявленим вимогам та стане ефективним інструментом для ведення та аналізу наукової продуктивності науково-педагогічного складу кафедри. Впровадження такого програмного продукту сприятиме підвищенню рівня організації наукової діяльності, прозорості оцінювання та загальному розвитку академічного середовища.

2. ІНФОРМАЦІЙНИЙ ОГЛЯД

2.1 Аналіз існуючих розробок з аналогічною тематикою

В сучасних реаліях науково-освітнього прогресу, цифровізація управлінських процесів, моніторингу та аналізу наукової діяльності у вищих навчальних закладах стає надзвичайно актуальною. Ключовим компонентом у цьому контексті є розробка програмних продуктів для візуалізації та аналізу наукової продуктивності науково-педагогічних працівників (НПП). Завдяки цьому адміністрація кафедр, факультетів та університетів може швидко оцінювати здобутки наукової діяльності, відслідковувати зміни, визначати переваги та недоліки у науковій активності колективу.

На сьогоднішній день існує ряд платформ, що частково забезпечують аналогічний функціонал. Проаналізуємо деякі з них:

1. Google Scholar / Профілі дослідників

Сервіс Google Scholar дозволяє створювати персональні профілі науковців, відслідковувати їх публікації, кількість цитувань, h-індекс та інші показники.

Однак, цей сервіс має низку обмежень у контексті кафедрального рівня аналізу, зокрема:

- відсутність інтеграції публікацій по кафедрі/підрозділу;
- відсутність можливості гнучкої візуалізації на основі заданих параметрів (період, тип публікацій, автори тощо);
- складність автоматичного агрегування даних по кількох профілях.

Таким чином, Google Scholar залишається інструментом індивідуального моніторингу, а не аналізу колективної наукової продуктивності.

2. ORCID / ResearcherID / Scopus Author Profiles

Ідентифікатори науковців типу ORCID, ResearcherID, а також авторські профілі у Scopus/Web of Science широко застосовуються для ідентифікації авторів та їхніх публікацій. Вони дозволяють:

- формувати авторські бібліографії;
- збирати статистику цитувань;
- аналізувати наукову активність.

Проте:

- ці сервіси не дозволяють агрегувати дані в розрізі структурного підрозділу (кафедри);
- аналітичні інструменти є обмеженими або доступні лише у платних версіях (Scopus Analytics, Web of Science Reports);
- немає гнучких візуальних звітів або інтерфейсів для зручного представлення даних на рівні підрозділу.

3. Університетські системи внутрішньої звітності (наприклад, E-Reports, Університетський рейтинг)

У багатьох ЗВО України впроваджено внутрішні інформаційні системи для обліку наукової діяльності: наприклад, платформи електронної звітності, електронні кабінети НПП, рейтингові системи.

Їхні особливості:

- фіксується кількість публікацій, грантів, конференцій, стажувань;
- ведеться рейтинговий облік;
- частково реалізована візуалізація (у вигляді таблиць або простих діаграм).

Недоліки таких систем:

- відсутність гнучкої фільтрації за періодами, категоріями чи типами діяльності;
- закритість архітектури та обмеженість налаштувань;
- складність інтеграції з зовнішніми базами (Google Scholar, Scopus, ORCID);
- часто реалізовані без акценту на зручну візуалізацію (переважає таблична форма).

4. Власні ініціативи (Excel-аналітика, ручні таблиці, Power BI)

У деяких підрозділах вишів використовуються самостійно розроблені системи обліку, які ґрунтуються на:

- Excel-таблицях з формулами;
- графіках у PowerPoint або Power BI;
- ручному зборі звітності.

Це дозволяє частково вирішити проблему аналізу, однак:

- потребує значних зусиль при оновленні даних;
- не є масштабованим рішенням;
- не забезпечує автоматизації та перевірки даних;
- не дозволяє зручно відображати інформацію через вебінтерфейс.

Висновок

Аналіз існуючих рішень дозволяє зробити висновок, що на сьогодні відсутнє універсальне програмне забезпечення, орієнтоване на зручну візуалізацію та аналіз наукової продуктивності НПП саме на рівні кафедри.

Тому актуальним є створення власного веб-застосунку, який би:

- дозволяв завантажувати або імпортувати дані про публікації, авторів та роки активності;
- формував графіки, діаграми та рейтинги;
- забезпечував фільтрацію по роках, авторам, типам публікацій;
- давав змогу оцінити динаміку розвитку наукової діяльності кафедри.

Розробка такого програмного продукту дозволить підвищити прозорість, аналітичність та зручність прийняття управлінських рішень у сфері наукової діяльності кафедри.

2.2 Позитивні аспекти та недоліки оглянутих робіт

У процесі дослідження та аналізу існуючих рішень, призначених для обліку, аналізу та візуалізації наукової продуктивності науково-педагогічних працівників, було виявлено низку спільних характеристик, які варто враховувати при проєктуванні власного програмного забезпечення.

Розглянуті системи включають міжнародні наукометричні бази (Google Scholar, Scopus, Web of Science), а також окремі реалізації внутрішніх університетських інформаційних систем.

Серед позитивних сторін таких рішень варто виділити, насамперед, інтеграцію з наукометричними базами, що забезпечує автоматизоване збирання інформації про публікації, цитування та інші показники. Це значно спрощує процес обліку наукової активності та зменшує обсяг рутинної роботи, пов’язаної з формуванням звітності. Доступ до базової аналітики в таких системах дозволяє НПП самостійно аналізувати власну динаміку публікацій, що є корисним у межах індивідуального професійного розвитку.

Окремі інструменти, зокрема Scopus або університетські інформаційні системи, містять елементи візуалізації статистичних даних — графіки, діаграми, кругові діаграми тощо. Це сприяє кращому сприйняттю інформації та дозволяє виявляти тренди й аномалії у динаміці наукової продуктивності. Крім того, доступність таких систем у вебсередовищі забезпечує зручність використання незалежно від місця розташування користувача або типу пристрою.

Попри наявні переваги, було виявлено і низку недоліків, які знижують ефективність застосування існуючих рішень у межах кафедри або іншого структурного підрозділу ЗВО. Насамперед, значна частина систем орієнтована на індивідуальні наукові профілі й не передбачає можливості колективного аналізу даних. Тобто відсутні функції для зведення статистики по кафедрі, факультету чи науковому відділу без ручного експорту й обробки інформації.

Також обмеженими є можливості візуалізації: у більшості випадків користувачі не можуть самостійно обирати період аналізу, типи наукових робіт (статті, тези, монографії тощо) або форму участі (перша/співавторство). Це ускладнює побудову гнучких звітів, які відповідають внутрішнім вимогам звітності закладу освіти.

Окремою проблемою є складність адаптації таких рішень до національного академічного контексту. Міжнародні системи здебільшого не враховують українські особливості обліку та оцінювання наукової продуктивності, а внутрішні системи в ЗВО нерідко мають застарілий інтерфейс, низьку інтерактивність і потребують значного доопрацювання.

Таким чином, проведений аналіз засвідчує доцільність створення спеціалізованого програмного забезпечення, яке б враховувало актуальні потреби кафедри в плані аналізу, візуалізації та обліку наукової активності НПП. Такий підхід дозволить не лише автоматизувати звітність, а й підвищити прозорість та ефективність управління науковою діяльністю кафедри.

3. ТЕОРЕТИЧНА ЧАСТИНА

3.1 Роль візуалізації в аналізі наукової діяльності у вищій освіті

У сучасній системі вищої освіти науково-дослідницька діяльність є одним із ключових напрямів роботи науково-педагогічного працівника. Вона не лише формує інтелектуальний потенціал вишу, але й суттєво впливає на його авторитет на національному та міжнародному рівні. Кількісний і якісний аналіз наукової продуктивності дозволяє оцінити внесок кожного працівника в розвиток науки, здійснити об’єктивне рейтингове оцінювання та визначити вектори подальшого розвитку кафедри.

Однак традиційні методи збирання та аналізу таких даних — таблиці Excel, текстові звіти чи підсумкові документи у форматі PDF — мають низку обмежень. Вони не забезпечують інтерактивності, не дозволяють швидко відслідковувати динаміку публікацій, участі у конференціях, проектній діяльності тощо. У зв’язку з цим особливої актуальності набуває візуалізація наукової активності за допомогою цифрових інструментів. Інформація, представлена у вигляді графіків, діаграм чи дашбордів, сприймається

набагато легше, дозволяє швидко ідентифікувати закономірності, порівнювати результати між співробітниками кафедри або за окремими роками, напрямами роботи, типами публікацій.

Крім того, візуалізація слугує не лише інструментом внутрішнього аналітичного моніторингу, а й засобом комунікації — як з керівництвом університету, так і з зовнішніми партнерами. Це відкриває нові можливості для публічного представлення наукових досягнень, формування позитивного іміджу кафедри та пошуку потенційних грантів чи колаборацій.

3.2 Сучасні технології аналізу та візуалізації даних

Упродовж останніх років технології аналізу даних розвинулись до рівня, який дозволяє інтегрувати складні аналітичні функції у відносно прості у користуванні системи. Основою таких систем є програмні платформи та мови програмування, що дозволяють працювати з великими обсягами структурованої або напівструктурованої інформації. Одним із популярних підходів є використання стеку Python (разом із Flask або Django для створення веб-застосунків) та візуалізаційних бібліотек, таких як Matplotlib, Plotly, Seaborn, або інтерактивних JavaScript-інструментів — наприклад, Chart.js чи D3.js.

Візуалізація — це не лише побудова діаграм. Це процес, що включає попередню обробку даних, фільтрацію, групування за категоріями, формування звітів та інтерактивних дашбордів. Для потреб кафедри важливо, щоб програма дозволяла працювати з такими параметрами, як:

- період публікацій (за роками)
- тип наукової роботи
- автор
- тематичні напрями діяльності

Інструменти мають забезпечити можливість вибору потрібного діапазону, порівняння показників, автоматичне підбиття підсумків. Також

важливою є реалізація можливості експорту звітів, збереження результатів аналізу у зручному форматі (PDF, Excel) та зручний механізм навігації по даних.

Окремо варто зазначити, що інтерфейс такого програмного забезпечення повинен бути максимально інтуїтивним, доступним для користувачів без технічної підготовки. Адже основні користувачі — це викладачі та адміністрація, які не обов’язково мають досвід роботи з IT-системами. Тому врахування принципів UX/UI дизайну є не менш важливим, ніж правильна архітектура бази даних.

3.3 Необхідність створення спеціалізованого програмного забезпечення для кафедр

Незважаючи на наявність великої кількості платформ для обліку наукової активності, таких як Google Scholar, Scopus, Web of Science, ORCID та інші, жодна з них не враховує специфіки внутрішньої організації кафедри. Наприклад, у більшості університетів існує потреба у підготовці звітів за конкретні періоди (семестри, навчальні роки), зі згрупованими даними по викладачах, напрямах діяльності або структурних підрозділах. У таких випадках загальнодоступні платформи або надто складні у налаштуванні, або не мають необхідного функціоналу.

Саме тому доцільним є створення спеціалізованого програмного продукту, адаптованого під потреби конкретної кафедри або факультету. Така система має враховувати структуру наукових напрямів, дозволяти зручно завантажувати інформацію про публікації, автоматично формувати аналітичні блоки, створювати рейтинги, динамічні діаграми, графіки. Додатково, з огляду на сучасні вимоги до безпеки даних, важливо передбачити механізми авторизації, контроль доступу до інформації, а також можливість резервного копіювання даних.

Розробка такого програмного забезпечення сприятиме підвищенню ефективності управління науковим потенціалом кафедри, зменшенню ручної роботи під час звітності, формуванню прозорої та зрозумілої системи оцінювання діяльності НПП. Крім того, це надасть змогу викладачам краще орієнтуватися у власній науковій активності, аналізувати динаміку та ефективність своєї роботи, виявляти слабкі місця й планувати подальші наукові кроки.

4. ПРАКТИЧНА ЧАСТИНА

4.1 Проєктування архітектури програмного забезпечення

У процесі реалізації програмного забезпечення для візуалізації наукової продуктивності науково-педагогічних працівників кафедри було обрано технології, які забезпечують простоту використання, стабільність роботи, зрозумілий інтерфейс і можливість швидкої інтеграції в освітній процес. Основу веб-додатку становить серверний фреймворк Flask, написаний мовою Python, який дозволяє ефективно організувати обробку HTTP-запитів, маршрутизацію та взаємодію з базою даних.

Для зберігання інформації про публікації, їх авторів, роки видання, типи праць тощо, використовується вбудована база даних SQLite, яка є легкою, надійною та не потребує окремого серверного середовища. Дані структуровано у вигляді взаємопов’язаних таблиць. Основними таблицями є таблиця публікацій та таблиця авторів. Така структура забезпечує логічну організацію інформації та зручність у подальшому опрацюванні даних.

Таблиця uploaded_file зберігає інформацію про завантажені файли наукових публікацій. Основні поля:

- id – унікальний ідентифікатор запису (ціле число, первинний ключ);
- filename – назва файлу, збереженого на сервері;
- filepath – шлях до файлу;
- upload_date – дата завантаження файлу;

- publisher – видавництво;

Ця таблиця забезпечує прив’язку кожного документа до відповідного викладача кафедри та дозволяє здійснювати фільтрацію й сортування.

Таблиця author містить інформацію про науково-педагогічних працівників:

- id – унікальний ідентифікатор автора;
- name – повне ім’я автора;

Кожен автор може мати кілька пов’язаних публікацій. Зв’язок між таблицями реалізується через author_id.

Інтерфейс користувача реалізовано за допомогою HTML-шаблонів і стилізовано з використанням CSS-фреймворку, що забезпечує сучасний, адаптивний і зручний вигляд вебсторінок.

Усі сторінки побудовані таким чином, щоб користувач міг швидко орієнтуватися в системі, обирати потрібну інформацію та виконувати необхідні дії без потреби у додаткових інструкціях.

4.2 Реалізація основного функціоналу

Функціональні можливості розробленої системи охоплюють усі етапи обліку та аналізу наукової діяльності викладачів. Зокрема, реалізовано механізм додавання нових записів до бази даних через веб-інтерфейс. Користувач має змогу ввести назву публікації, вказати рік видання, прикріпити PDF-файл публікації, а також обрати авторів із відповідного списку.

Завантаження файлів

```
<body>
  {% extends 'base.html' %}

  {% block title %}Список файлів{% endblock %}
```

```
   {% block content %}
<div class="container">
   <h1>Завантаження файла</h1>

   <form method="POST" enctype="multipart/form-data">

      <label for="file">Оберіть файл:</label>
      <input type="file" name="file" id="file" required>

      <label for="custom_filename">Введіть назву файлу:</label>
        <input type="text" name="custom_filename" placeholder="Введіть назву файла" id="custom_filename" required >

      <label for="publisher">Видавництво:</label>
<input type="text" name="publisher" id="publisher" value="{{ request.form.publisher or '' }}">

      <label>Оберіть авторів:</label>
      <div>
         {% for author in authors %}
         <label>
            <input type="checkbox" name="author_ids" value="{{ author.id }}">
            {{ author.name }}
         </label><br>
         {% endfor %}
      </div>

      <label for="upload_date">Введіть дату:</label>
      <input type="datetime-local" name="upload_date" id="upload_date">
```

```
        <button type="submit">Завантажити</button>
    </form>


</div>
{% endblock %}
</body>
```

Ось такий має візуальний вигляд

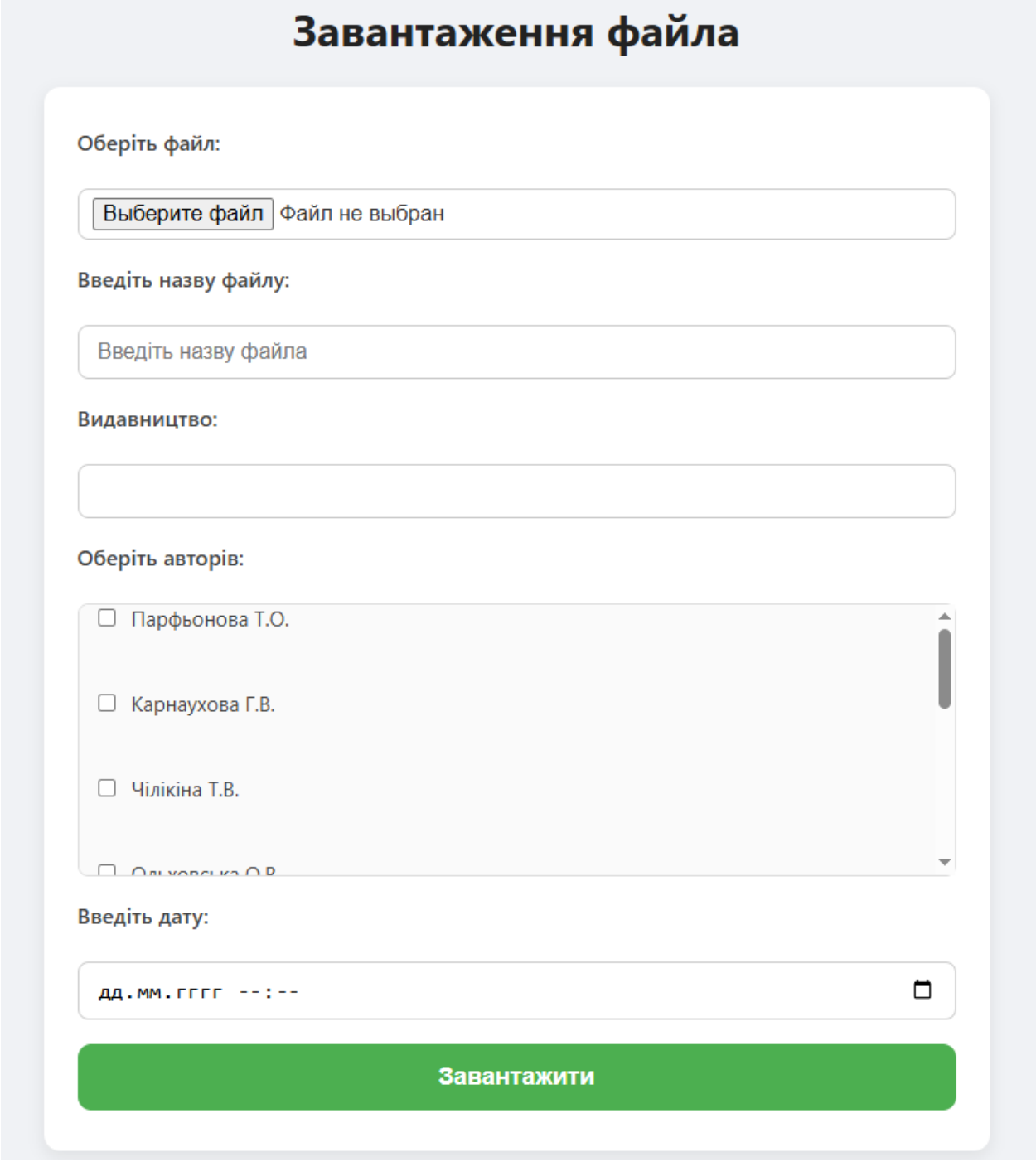

Рис. 4.1 – Сторінка завантаження файлів

На цій сторінці ми можемо обрати наш файл який потрібно завантажити, та зі списку обрати автора.

Особливу увагу приділено можливості перегляду та аналізу вже наявної інформації. Система дозволяє переглядати усі додані публікації у вигляді впорядкованої таблиці з фільтрацією за роками, типами праць, прізвищами авторів. Також передбачено сортування за різними критеріями (за алфавітом, датою, типом тощо). Усі записи містять детальну інформацію про публікацію та посилання для завантаження певного файлу.

Список файлів

```
{% extends 'base.html' %}

{% block title %}Список файлів{% endblock %}

{% block content %}
<h1>Список файлів</h1>

<form method="GET" action="{{ url_for('list_files') }}">
   <label for="search">Пошук по назві:</label>
            <input type="text" id="search" name="search" value="{{ request.args.get('search', '') }}" placeholder="Введіть ім'я файлу">

   <label>Оберіть авторів:</label>
    <div class="authors-checkboxes" style="max-height: 150px; overflow-y: auto; border: 1px solid #ddd; padding: 10px; border-radius: 6px; margin-bottom: 15px;">
       {% set selected_authors = request.args.getlist('author_ids') %}
       {% for author in authors %}
```

```
            <label style="display: block; margin-bottom: 5px; cursor: pointer;">
                <input type="checkbox" name="author_ids" value="{{ author.id }}"
                    {% if author.id|string in selected_authors %}checked{% endif %}>
                {{ author.name }}
            </label>
        {% endfor %}
    </div>

    <label>Фільтр по даті:</label>
    <div style="display: flex; gap: 10px; margin-bottom: 15px;">
        З:
                        <input type="date" name="date_from" value="{{
request.args.get('date_from', '') }}" placeholder="Дата з">
        По:
          <input type="date" name="date_to" value="{{ request.args.get('date_to', '')
}}" placeholder="Дата до">
    </div>

    <button type="submit">Застосувати</button>
          <a href="{{ url_for('list_files') }}" class="delete-button">Скинути
фільтри</a>
</form>

<h2>Файли</h2>
<ul class="file-list">
    {% if files %}
        {% for file in files %}
            <li class="file-item">
                <div class="file-name">{{ file.filename }}</div>
                <div class="file-meta">
```

```
            {% if file.publisher %}
               <p><strong>Видавництво:</strong> <br>{{ file.publisher }}</p>
            {% endif %}

            <p><strong>Автори:</strong>
               {% for author in file.authors %}
                  <br> {{ author.name }}{% if not loop.last %}, {% endif %}
               {% endfor %}
            </p>

            {% if file.upload_date %}
                                                <p><strong>Дата:</strong>    <br>{{
file.upload_date.strftime('%d.%m.%Y') }}</p>
            {% else %}
               <p><strong>Дата:</strong> Немає дати</p>
            {% endif %}
         </div>

         <div class="file-actions">
                  <a href="{{ url_for('download_file', filename=file.filename) }}"
class="download-link">Завантажити</a>
                     <form action="{{ url_for('delete_file', file_id=file.id) }}"
method="POST" class="delete-form">
                      <button type="submit" onclick="return confirm('Видалити цей
файл?')" class="delete-button">Видалити</button>
            </form>
         </div>
      </li>
   {% endfor %}
{% else %}
```

```
        <p>Немає файлів.</p>
    {% endif %}
</ul>
{% endblock %}
```

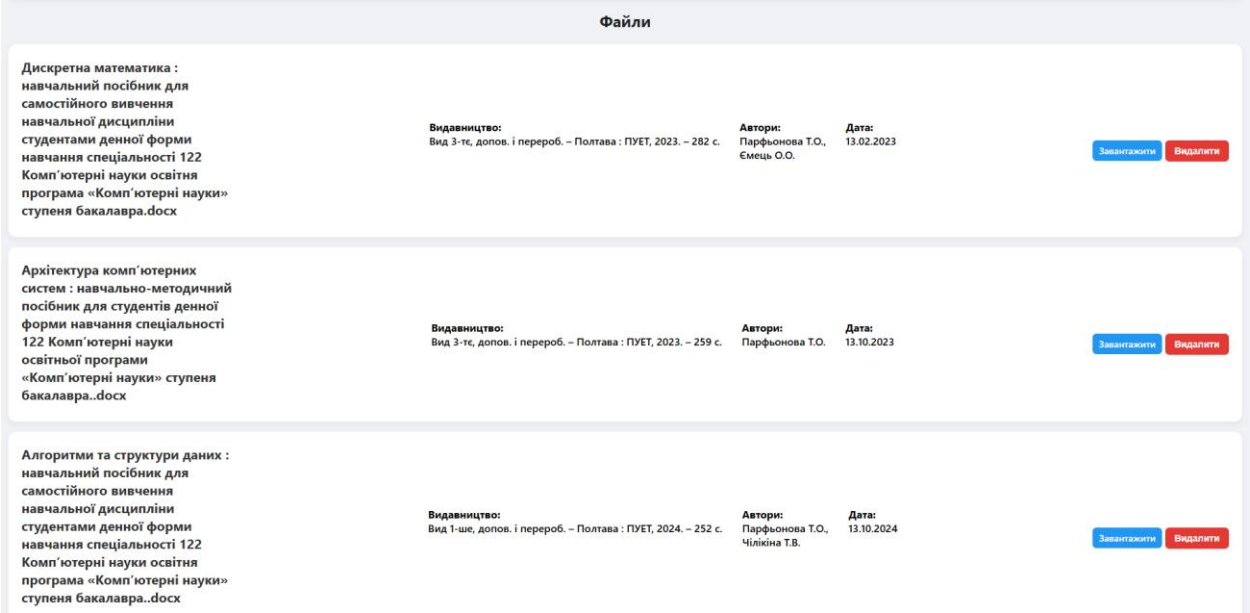

Рис. 4.2 – Список файлів

На цій сторінці ми бачимо всі файли які були завантаженні в БД. Також до кожного файлу є прив'язка до свого автора. Для кожного файлу є можлівість видалити його з БД або заванжати собі на ПК. З верху ми бачимо блок фільтрації, є пошук по назві, фільтрації по авторам та даті.

Сторінці авторів також приділимо увагу

Автори

```
<body>
    {% extends 'base.html' %}

    {% block title %}Список файлів{% endblock %}

    {% block content %}
```

```
<div class="container">
    <h1>Додати нового автора</h1>

    <form method="POST">
        <label for="name">Ім'я автора:</label>
        <input type="text" id="name" name="name" required>
        <button type="submit">Додати</button>
    </form>

    <h2>Список авторів</h2>

    <div class="authors-list">
        {% for author in authors %}
        <div class="author-item">
            <span class="author-name">{{ author.name }}</span>
                <form action="{{ url_for('delete_author', author_id=author.id) }}" method="POST" class="delete-form">
                    <button type="submit" onclick="return confirm('Видалити цього автора?')" class="delete-button">Видалити</button>
            </form>
        </div>
        {% else %}
            <p>Немає авторів.</p>
        {% endfor %}
    </div>


</div>
{% endblock %}
</body>
```

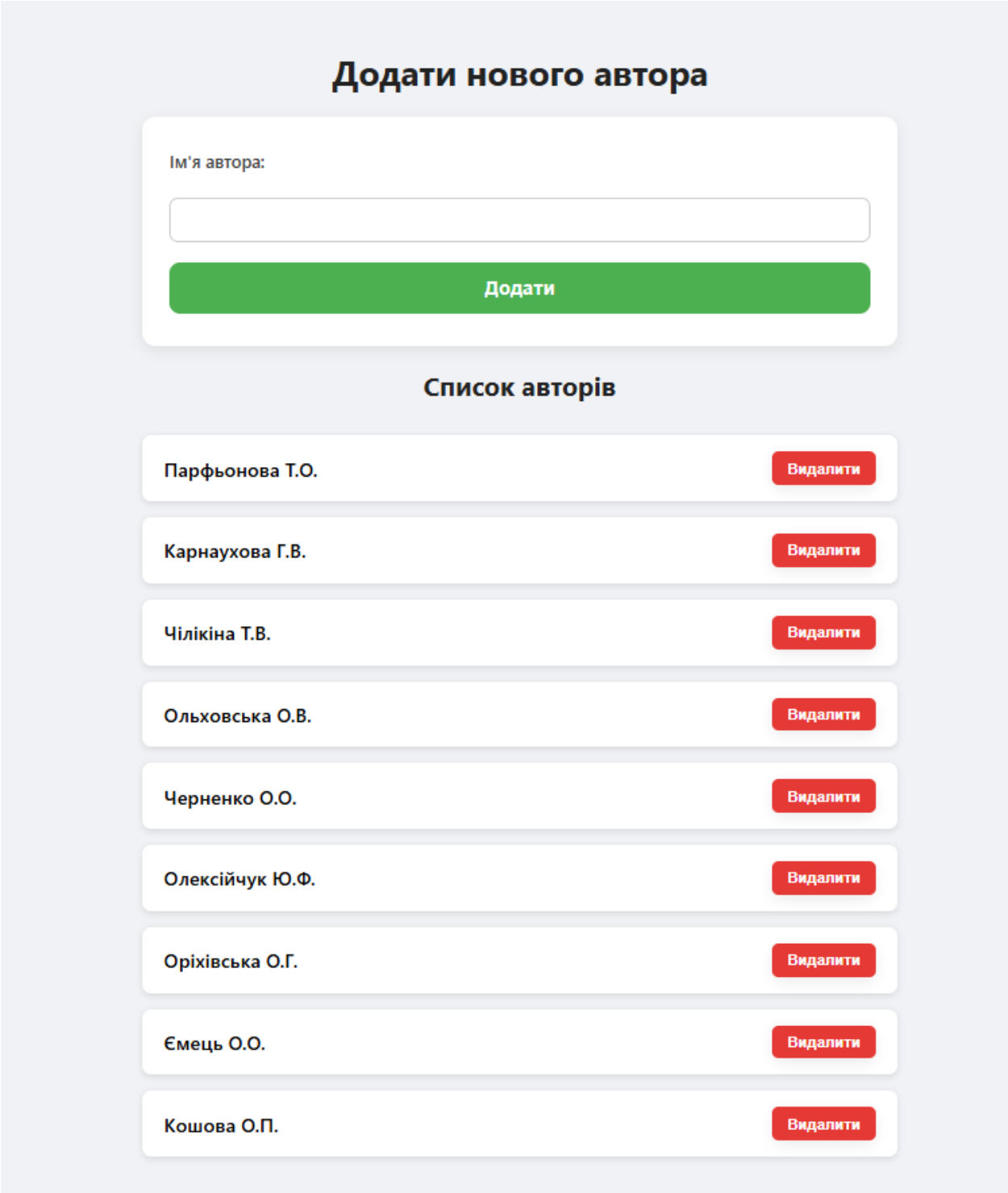

Рис. 4.3 – Сторінка авторів

На сторінці з авторами, ми маємо можливість перегляду вже існуючих авторів, також додати в систему нового автора та видали його з БД.

Для візуалізації статистичних даних використано бібліотеку Chart.js, яка забезпечує побудову інтерактивних діаграм і графіків без необхідності складної конфігурації. На основі даних із бази формується кілька типів графіків: кругова діаграма розподілу публікацій за типами, стовпчиковий графік кількості публікацій за роками, лінійна діаграма наукової активності певного автора тощо. Ці графіки автоматично оновлюються при зміні вмісту бази даних або застосуванні фільтрів. Завдяки цьому користувач може в реальному часі оцінити динаміку розвитку наукової діяльності, виявити періоди підвищеної активності, зробити висновки щодо загальної ефективності публікаційної роботи кафедри.

```
<div class="container">
  <h1>Аналітика активності</h1>

  <form method="POST">
    <label for="start_date">Дата початку:</label>
        <input type="date" id="start_date" name="start_date" value="{{ start_date.strftime('%Y-%m-%d') if start_date else '' }}">

    <label for="end_date">Дата закінчення:</label>
        <input type="date" id="end_date" name="end_date" value="{{ end_date.strftime('%Y-%m-%d') if end_date else '' }}">

    <button type="submit">Показати статистику</button>
  </form>

  <h2>Загальна кількість файлів: {{ total_files }}</h2>

  <div style="text-align: center;">
  <h2>Графік загальної кількості</h2>
```

```
    <img src="{{ graph_url1 }}" alt="Графік загальної кількості"
        style="display: block; margin: 0 auto; max-width: 100%; height: auto;">

    <h2>Графік по авторам</h2>
    <img src="{{ graph_url2 }}" alt="Графік по авторам"
        style="display: block; margin: 0 auto; max-width: 100%; height: auto;">
</div>
</div>
```

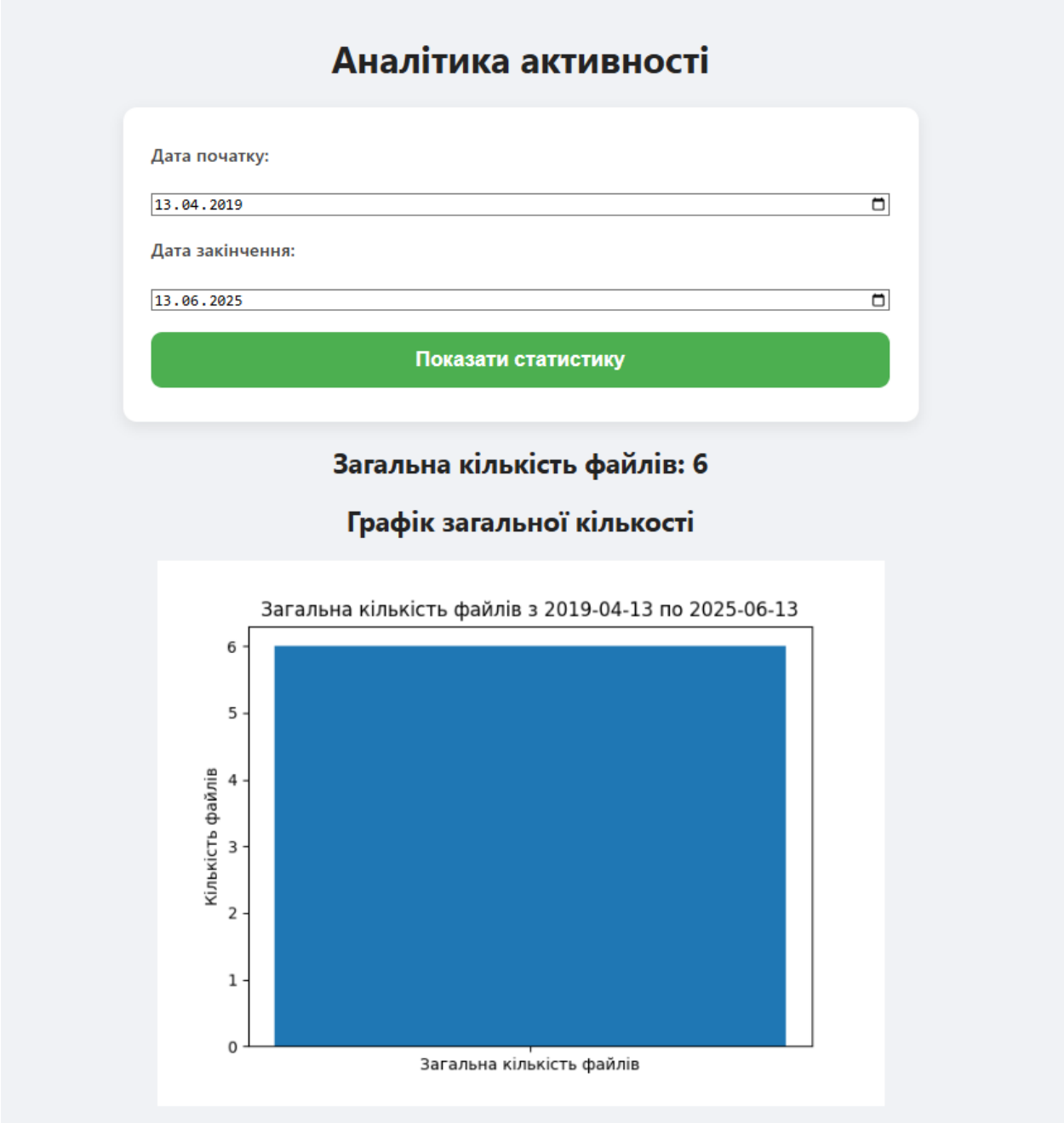

Рис. 4.4 – Сторінка активності

На сторінці активності в нас є графіки з загальною кількістю файлів та графік за кількістю файлів за окремим автором.

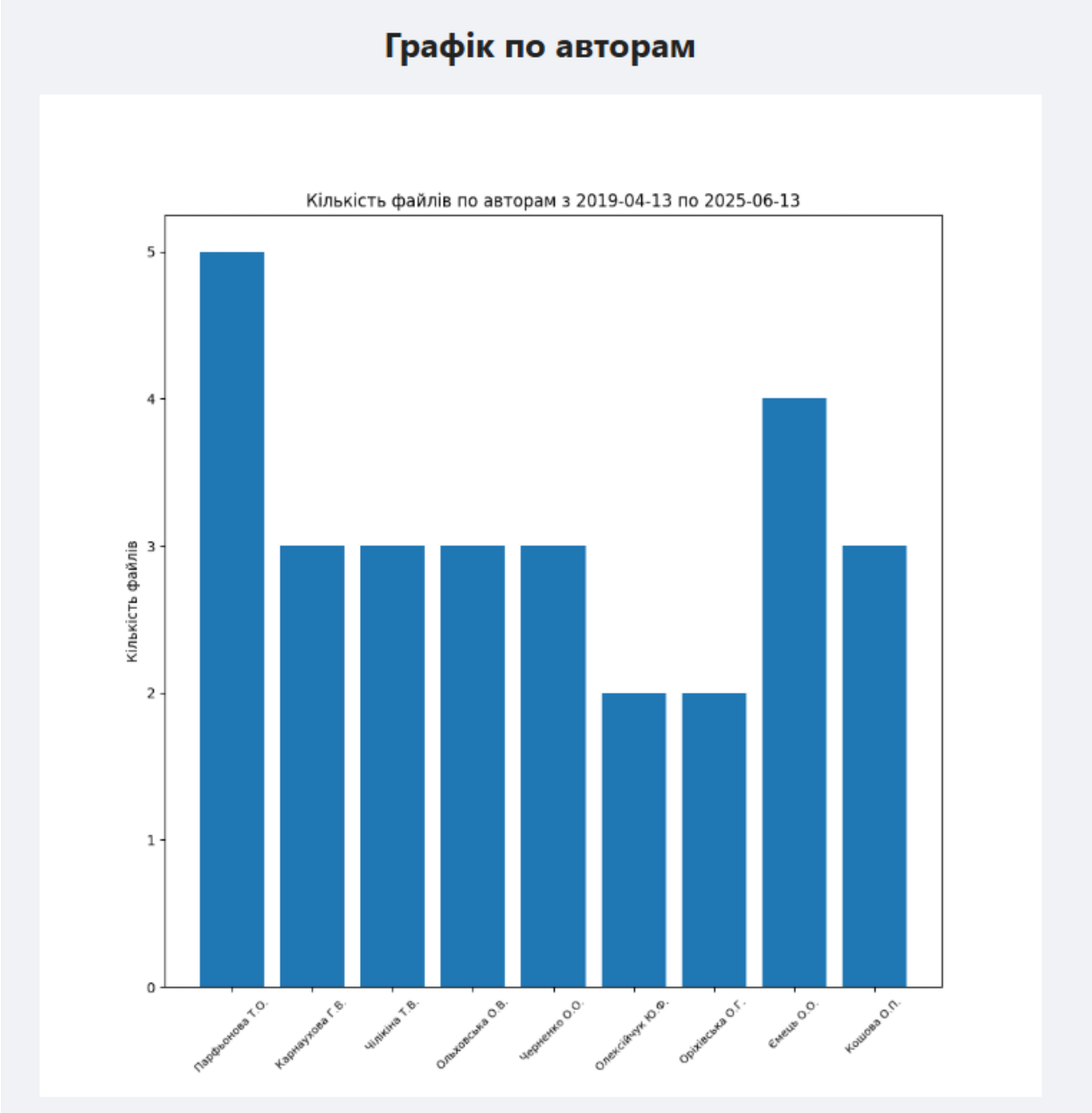

Рис. 4.5 – Графік по авторам

Окремим функціональним блоком є панель керування авторами. Кожен викладач фігурує в системі як окремий об’єкт з унікальним ідентифікатором. До нього прив’язуються всі публікації, у яких він зазначений як автор. Це дозволяє оперативно отримувати узагальнену статистику для кожного НПП: загальну кількість робіт, розподіл за роками, а також швидкий доступ до списку конкретних публікацій.

4.3 Аналіз розробленого програмного забезпечення для візуалізації наукової продуктивності НПП кафедри

У процесі створення та впровадження програмного забезпечення для візуалізації наукової продуктивності науково-педагогічних працівників кафедри було проаналізовано низку важливих аспектів, які впливають на ефективність та зручність його використання. Розроблена система поєднує широкий функціонал із простотою користування, що забезпечує централізований збір, збереження та обробку даних про наукові публікації, їх авторів, тематику і часові проміжки публікацій. Завдяки цьому користувачі отримують змогу швидко та зручно отримувати актуальну інформацію про наукову діяльність як окремих співробітників, так і кафедри в цілому.

Основною перевагою програмного продукту є наявність інтерактивних візуалізацій, які дозволяють не лише переглядати числові показники, а й виявляти закономірності та тенденції у динаміці наукової активності. За допомогою графіків, діаграм і таблиць можна чітко відстежувати зміни у кількості публікацій, співпрацю між авторами, а також аналізувати вплив певних подій або заходів на продуктивність кафедри. Інтерфейс системи розроблено з урахуванням потреб різних категорій користувачів — від викладачів, які хочуть отримати швидкий доступ до власних результатів, до керівників, що потребують комплексного моніторингу і планування наукової діяльності.

Крім того, важливою характеристикою системи є її гнучкість і масштабованість. Вона може бути легко інтегрована з існуючими базами даних та інструментами кафедри, а також адаптована під змінні вимоги в процесі подальшої експлуатації. Це дозволяє забезпечити безперервний розвиток і вдосконалення програмного забезпечення з урахуванням нових функцій, що виникатимуть у процесі роботи. Також значущим є те, що

система працює ефективно на різних платформах і не потребує високих апаратних ресурсів, що розширює коло потенційних користувачів і спрощує її впровадження у різних навчальних закладах.

Таким чином, розроблене програмне забезпечення є потужним інструментом для організації, аналізу та візуалізації наукової діяльності кафедри, що значно підвищує рівень керованості і прозорості в цій сфері. Воно сприяє ефективнішому використанню наявних ресурсів, мотивує науковців до більш активної роботи та полегшує процес прийняття управлінських рішень. Результати апробації підтверджують, що впровадження таких цифрових рішень відповідає сучасним тенденціям розвитку освіти і науки, а також відкриває нові можливості для покращення якості наукової роботи.

ВИСНОВОК

У ході виконання роботи було розроблено програмне забезпечення для візуалізації наукової продуктивності науково-педагогічних працівників кафедри та проведено його аналіз. Результати свідчать, що створена система успішно вирішує ключові завдання, пов’язані з централізацією, систематизацією та наочним відображенням інформації про наукові публікації та активність співробітників. Завдяки інтерактивним інструментам візуалізації, користувачі отримують можливість швидко оцінити динаміку наукової роботи, виявити тенденції, а також порівнювати результати окремих дослідників та кафедри загалом.

Впровадження такого програмного продукту сприяє підвищенню ефективності внутрішньої організації наукової діяльності, полегшує процес підготовки звітів та планування науково-дослідних заходів. Система відзначається простотою у використанні, що робить її доступною для широкого кола користувачів із різним рівнем технічної підготовки. Крім того, гнучкість і масштабованість програмного забезпечення дозволяють адаптувати його до потреб інших кафедр або навчальних закладів, що відкриває перспективи для подальшого розвитку та масштабування.

Таким чином, створена система є ефективним інструментом підтримки наукової діяльності, що відповідає сучасним вимогам цифрової трансформації освіти та науки. Вона сприяє підвищенню прозорості та якості наукової роботи, стимулює активність викладачів і допомагає приймати обґрунтовані управлінські рішення. В подальшому планується розширення функціоналу, впровадження додаткових аналітичних можливостей та інтеграція з іншими інформаційними системами для більш комплексного аналізу наукової продуктивності.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Бондаренко А. В. Основи баз даних: навчальний посібник. — Київ: Каравела, 2020. — 208 с.
2. Шень В. А. Веб-програмування: HTML, CSS, JavaScript. — Харків: Фактор, 2021. — 320 с.
3. Шилдт Г. Python. Повний курс. — К.: Діалектика, 2022. — 672 с.
4. Grinberg M. *Flask Web Development: Developing Web Applications with Python*. — 2nd ed. — O’Reilly Media, 2018. — 300 p.
5. Flask Documentation
6. SQLAlchemy Documentation
7. MDN Web Docs: HTML Reference
8. MDN Web Docs: CSS Reference
9. Власюк В. І. Програмна інженерія. — Львів: ЛНУ ім. І. Франка, 2019.
10. Python Software Foundation. Python 3.11 Documentation
11. Документація SQLite

ДОДАТОК А. Код програми

```
import os
from flask import Flask, request, render_template, redirect, url_for
from flask_sqlalchemy import SQLAlchemy
from werkzeug.utils import secure_filename
from datetime import datetime, timedelta
from flask import send_from_directory
from flask_migrate import Migrate
import matplotlib.pyplot as plt
import io
import base64

app = Flask(__name__)
app.config['UPLOAD_FOLDER'] = 'uploads'
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///database.db'
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False

db = SQLAlchemy(app)
migrate = Migrate(app, db)

class Author(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(100), nullable=False)

class UploadedFile(db.Model):
    id = db.Column(db.Integer, primary_key=True)

    filename = db.Column(db.String(255), nullable=False)
```

```
    filepath = db.Column(db.String(255), nullable=False)
    author_id = db.Column(db.Integer, db.ForeignKey('author.id'), nullable=False)
    author = db.relationship('Author', backref=db.backref('files', lazy=True))
    upload_date = db.Column(db.DateTime, default=datetime.utcnow)

@app.route('/')
def index():
    return redirect(url_for('upload_file'))

@app.route('/add_author', methods=['GET', 'POST'])
def add_author():
    if request.method == 'POST':
        name = request.form['name']
        new_author = Author(name=name)
        db.session.add(new_author)
        db.session.commit()
        return redirect(url_for('add_author'))

    authors = Author.query.all()
    return render_template('add_author.html', authors=authors)

@app.route('/upload', methods=['GET', 'POST'])
def upload_file():
    if request.method == 'POST':
        author_id = request.form['author_id']
        file = request.files['file']
        if file and author_id:
            filename = secure_filename(file.filename)
            filepath = os.path.join(app.config['UPLOAD_FOLDER'], filename)
            file.save(filepath)
```

```
                new_file = UploadedFile(filename=filename, filepath=filepath, author_id=author_id)
            db.session.add(new_file)
            db.session.commit()
            return redirect(url_for('upload_file'))

    authors = Author.query.all()
    return render_template('upload.html', authors=authors)

@app.route('/files', methods=['GET', 'POST'])
def list_files():
    author_id = request.args.get('author_id')
    sort_order = request.args.get('sort', 'asc')
    search_query = request.args.get('search', '').strip()

    query = UploadedFile.query

    if author_id:
        query = query.filter(UploadedFile.author_id == int(author_id))

    if search_query:
        query = query.filter(UploadedFile.filename.ilike(f'%{search_query}%'))

    if sort_order == 'desc':
        query = query.order_by(UploadedFile.upload_date.desc())
    else:
        query = query.order_by(UploadedFile.upload_date.asc())

    files = query.all()
```

```
    authors = Author.query.all()
        return render_template('list_files.html', files=files, authors=authors, sort_order=sort_order, search_query=search_query)

@app.route('/delete_file/<int:file_id>', methods=['POST'])
def delete_file(file_id):
    file = UploadedFile.query.get_or_404(file_id)

    if os.path.exists(file.filepath):
        os.remove(file.filepath)

    db.session.delete(file)
    db.session.commit()
    return redirect(url_for('list_files'))

@app.route('/delete_author/<int:author_id>', methods=['POST'])
def delete_author(author_id):
    author = Author.query.get_or_404(author_id)


    for file in author.files:
        if os.path.exists(file.filepath):
            os.remove(file.filepath)
        db.session.delete(file)


    db.session.delete(author)
    db.session.commit()
```

```
    return redirect(url_for('add_author'))

@app.route('/activity', methods=['GET', 'POST'])
def author_activity():

    start_date = request.form.get('start_date')
    end_date = request.form.get('end_date')

    if start_date:
        start_date = datetime.strptime(start_date, '%Y-%m-%d')
    if end_date:
        end_date = datetime.strptime(end_date, '%Y-%m-%d')
    else:
        end_date = datetime.utcnow()

    query = db.session.query(db.func.count(UploadedFile.id))

    if start_date:
        query = query.filter(UploadedFile.upload_date >= start_date)
    query = query.filter(UploadedFile.upload_date <= end_date)

    total_files = query.scalar()

    authors = Author.query.all()
    author_file_counts = {}
    for author in authors:
        count = db.session.query(db.func.count(UploadedFile.id)) \
            .filter(UploadedFile.author_id == author.id) \
            .filter(UploadedFile.upload_date <= end_date)
```

```
        if start_date:
            count = count.filter(UploadedFile.upload_date >= start_date)

        author_file_counts[author.name] = count.scalar()


    fig1, ax1 = plt.subplots()
    ax1.bar(['Загальна кількість файлів'], [total_files])
    ax1.set_ylabel('Кількість файлів')
    ax1.set_title(f'Загальна кількість файлів з {start_date.strftime("%Y-%m-%d") if start_date else "не вказано"} по {end_date.strftime("%Y-%m-%d")}')

    fig2, ax2 = plt.subplots()
    authors_names = list(author_file_counts.keys())
    author_counts = list(author_file_counts.values())
    ax2.bar(authors_names, author_counts)
    ax2.set_ylabel('Кількість файлів')
    ax2.set_title(f'Кількість файлів по авторам з {start_date.strftime("%Y-%m-%d") if start_date else "не вказано"} по {end_date.strftime("%Y-%m-%d")}')

    img1 = io.BytesIO()
    fig1.savefig(img1, format='png')
    img1.seek(0)
    graph_url1 = base64.b64encode(img1.getvalue()).decode('utf-8')

    img2 = io.BytesIO()
    fig2.savefig(img2, format='png')
    img2.seek(0)
    graph_url2 = base64.b64encode(img2.getvalue()).decode('utf-8')
```

```
        return render_template('activity.html', graph_url1=graph_url1, graph_url2=graph_url2, total_files=total_files, start_date=start_date, end_date=end_date)



@app.route('/download/<filename>')
def download_file(filename):
    return send_from_directory(app.config['UPLOAD_FOLDER'], filename, as_attachment=True)



with app.app_context():
    db.create_all()

if __name__ == '__main__':
    os.makedirs(app.config['UPLOAD_FOLDER'], exist_ok=True)
    app.run(debug=True)
```

ДОДАТОК Б. CSS стилі

```
body {
    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
    background: #f0f2f5;
    margin: 0;
    padding: 20px;
    color: #333;
}

h1, h2 {
    text-align: center;
    color: #222;
}

.container {
    max-width: 700px;
    margin: 0 auto;
}

form {
    background: #fff;
    padding: 30px 25px;
    border-radius: 12px;
    box-shadow: 0 4px 12px rgba(0,0,0,0.08);
    display: flex;
    flex-direction: column;
    gap: 18px;
}
```

```
form label {
    font-weight: 600;
    margin-bottom: 6px;
    color: #444;
    font-size: 16px;
}

form input[type="file"],
form input[type="text"],
form input[type="datetime-local"] {
    padding: 10px 14px;
    border-radius: 8px;
    border: 1.5px solid #ccc;
    font-size: 16px;
    transition: border-color 0.3s ease;
    width: 100%;
    box-sizing: border-box;
}

form input[type="file"] {
    padding: 6px 10px;
}

form input[type="text"]:focus,
form input[type="datetime-local"]:focus,
form input[type="file"]:focus {
    border-color: #4CAF50;
    outline: none;
}
```

```
form > div {
    display: flex;
    flex-direction: column;
    gap: 8px;
    max-height: 200px;
    overflow-y: auto;
    padding-left: 10px;
    border: 1px solid #ddd;
    border-radius: 8px;
    background: #fafafa;
}

form > div label {
    font-weight: normal;
    font-size: 15px;
    cursor: pointer;
    user-select: none;
}

form input[type="checkbox"] {
    margin-right: 8px;
    cursor: pointer;
}

form button {
    background-color: #4CAF50;
    color: white;
    padding: 14px 22px;
    border: none;
    border-radius: 10px;
```

```
    font-weight: 700;
    font-size: 18px;
    cursor: pointer;
    transition: background-color 0.3s ease;
}

form button:hover {
    background-color: #45a049;
}

.file-list {
    list-style: none;
    padding: 0;
}

.file-name {
    max-width: 320px;
    font-weight: bold;
    font-size: 20px;
    margin-bottom: 8px;
    white-space: normal;
    word-wrap: break-word;
    overflow-wrap: break-word;
    display: block;
}

.file-item {
    background: #fff;
    margin-bottom: 15px;
    padding: 20px;
```

```
    border-radius: 12px;
    box-shadow: 0 4px 10px rgba(0,0,0,0.05);
    display: flex;
    justify-content: space-between;
    align-items: center;
    flex-wrap: nowrap;
    gap: 20px;
}

.file-meta {
    display: flex;
    gap: 30px;
    font-size: 1em;
    font-weight: 500;
    color: #000000;
}

.file-meta span {
    white-space: nowrap;
}

.file-actions {
    gap: 10px;
    flex-shrink: 0;
    justify-content: flex-end;
    align-items: center;
}

.download-link {
    background-color: #2196F3;
```

```
    color: white;
    text-decoration: none;
    padding: 6px 10px;
    font-size: 14px;
    border-radius: 6px;
    font-weight: 600;
    text-align: center;
    transition: background-color 0.3s, transform 0.2s;
}

.download-link:hover {
    background-color: #1976D2;
    transform: translateY(-1px);
}

.delete-button {
    background-color: #e53935;
    border: none;
    color: white;
    cursor: pointer;
    padding: 6px 10px;
    font-size: 14px;
    border-radius: 6px;
    font-weight: 600;
    text-align: center;
    transition: background-color 0.3s, transform 0.2s;
}

.delete-button:hover {
    background-color: #c62828;
```

```
    transform: translateY(-1px);
}

a {
    display: inline-block;
    margin-top: 30px;
    background-color: #4CAF50;
    color: white;
    padding: 10px 16px;
    border-radius: 8px;
    text-decoration: none;
    font-weight: bold;
    transition: background-color 0.3s;
}

a:hover {
    background-color: #45a049;
}

.delete-form {
    margin: 0;
    padding: 0;
    display: inline;
}

.authors-list {
    margin-top: 30px;
    display: flex;
    flex-direction: column;
    gap: 15px;
```

```
}

.author-item {
    background: white;
    padding: 15px 20px;
    border-radius: 8px;
    box-shadow: 0 2px 6px rgba(0, 0, 0, 0.1);
    display: flex;
    justify-content: space-between;
    align-items: center;
}

.author-name {
    font-size: 18px;
    font-weight: 500;
    color: #000000;
}

.delete-form button {
    color: white;
    border: none;
    padding: 8px 14px;
    border-radius: 6px;
    cursor: pointer;
    transition: background-color 0.3s;
}

.file-meta-container {
    display: flex;
    flex-direction: column;
```

```
    gap: 20px;
    margin-top: 20px;
}

.file-meta-block {
    background: #fff;
    border-radius: 12px;
    box-shadow: 0 4px 12px rgba(0,0,0,0.05);
    padding: 15px 20px;
    width: 200px;
    min-height: 120px;
    display: flex;
    flex-direction: column;
    justify-content: center;
    text-align: center;
    color: #333;
    font-weight: 600;
    font-size: 16px;
}

.file-meta-block strong {
    font-size: 18px;
    margin-bottom: 10px;
}

@media (max-width: 480px) {
    body {
        padding: 10px;
    }
```

```
.container {
    max-width: 100%;
    padding: 0 10px;
}

h1, h2 {
    font-size: 20px;
}

form {
    padding: 15px 10px;
    gap: 12px;
}

form input[type="file"],
form input[type="text"],
form input[type="datetime-local"] {
    font-size: 14px;
    padding: 8px 10px;
}

form label {
    font-size: 14px;
}

form > div {
    padding: 8px;
    max-height: 150px;
}
```

```
.file-item {
    flex-direction: column;
    align-items: flex-start;
    gap: 10px;
    padding: 15px;
}

.file-name {
    font-size: 16px;
}

.file-meta {
    flex-direction: column;
    gap: 6px;
    font-size: 14px;
}

.file-actions {
    width: 100%;
    display: flex;
    justify-content: flex-start;
    gap: 10px;
    flex-wrap: wrap;
}

.download-link,
.delete-button {
    width: 100%;
    font-size: 14px;
    padding: 8px;
```

```
    text-align: center;
}

.authors-list {
    gap: 10px;
}

.author-item {
    flex-direction: column;
    align-items: flex-start;
    gap: 10px;
    font-size: 14px;
}

.author-name {
    font-size: 16px;
}

.file-meta-container {
    flex-direction: column;
    gap: 10px;
}

.file-meta-block {
    width: 100%;
    font-size: 14px;
}

.file-meta-block strong {
    font-size: 16px;
```

```
    }

    a {
        width: 100%;
        text-align: center;
        font-size: 14px;
        padding: 8px 12px;
    }
}
```