СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВОЛОДИМИРА ДАЛЯ

Навчально-науковий інститут (факультет)
Інформаційних технологій та електроніки
(повне найменування інституту, факультету)

Кафедра Інформаційних технологій та програмування
(повна назва кафедри)

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної випускної роботи

освітній ступінь бакалавр
спеціальність 121 «Інженерія програмного забезпечення»

на тему «Підсистема ведення документообігу підприємства»

Виконав: студент групи ІПЗ-20бд _____ А.О. Полінський
(підпис) (ініціали і прізвище)

Керівник _____ В.О. Лифар
(підпис) (ініціали і прізвище)

Завідувач кафедри _____ О.І. Захожай
(підпис) (ініціали і прізвище)

Рецензент _____ Д.В. Ратов
(підпис) (ініціали і прізвище)

Київ – 2024

СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВОЛОДИМИРА ДАЛЯ

Навчально-науковий інститут (факультет)
інформаційних технологій та електроніки
(повне найменування інституту, факультету) .
Кафедра інформаційних технологій та програмування
(повна назва кафедри)
Освітній ступінь _____бакалавр_____
(бакалавр, магістр)
Спеціальність 121 «Інженерія програмного забезпечення»
(шифр і назва спеціальності)

ЗАТВЕРДЖУЮ

Завідувач кафедри
_____О. І. Захожай
“____” _____2024 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ ВИПУСКНУ РОБОТУ СТУДЕНТУ

_____Полінський Артем Олександрович_____
(прізвище, ім’я, по батькові)
1. Тема роботи «Підсистема ведення документообігу підприємства»_____
_____
Керівник роботи Лифар Володимир Олексійович, доцент, д.т.н._____,
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджений наказом університету від “06” _травня_2024року №171/15.15-С__

2. Строк подання студентом роботи 12.06. 2024 р._____
3. Вихідні дані до роботи Об'єктом даної роботи є процес розробки програми електронного обігу документів підприємства_____
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) Розробка підсистеми електронного документообігу, автоматизація робочих процесів, покращення організації документообігу та реалізація проекту
5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслеників)
_____
_____
_____

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання: 30.03.2024

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання кваліфікаційної випускної роботи	Строк виконання етапів	Примітка
1	Одержання завдання на виконання роботи	30.03.24	
2	Укладання та погодження з керівником плану і етапів виконання роботи	05.04.24	
3	Узагальнення даних літературних джерел, укладання розділу «Аналіз предметної галузі»	11.04.24	
4	Аналіз шляхів виконання завдання. Вибір і погодження з керівником оптимального шляху	22.04.24	
5	Укладання та тестування програмного продукту	06.05.24	
6	Укладання, оформлення та погодження пояснювальної записки з керівником	03.05.24	
7	Здача готової пояснювальної записки на кафедру	12.06.24	

Студент _____ А.О. Полінський
(підпис) (ініціали і прізвище)

Керівник роботи _____ В.О. Лифар
(підпис) (ініціали і прізвище)

ЛИСТ ПОГОДЖЕННЯ І ОЦІНЮВАННЯ

дипломної роботи студента гр. ІПЗ-20бд Полінського А.О.

Науковий керівник

Доцент, к.т.н. _____ Лифар В.О.

Оцінка наукового керівника: _____

Рецензент Ратов Д.В., СНУ ім. В.Даля, доцент, к.т.н._____
(ПІБ, місто роботи, посада)

Оцінка рецензента: _____

Кінцева оцінка за результатами захисту:

_____

Голова ЕК _____ О.С.Меняйленко_____
підпис (ініціали і прізвище)

РЕФЕРАТ

Робота містить: 80 сторінок основного тексту, 23 сторінки додатків, 29 рисунків, 32 використаних джерел.

Полінський А. О. «Підсистема ведення документації підприємства».

Кваліфікаційна випукна робота бакалавра за спеціальністю: 121 «Інженерія програмного забезпечення». – СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ВОЛОДИМИРА ДАЛЯ. – Київ, 2024.

Метою випускної кваліфікаційної роботи є вивчення особливостей розробки підсистеми електронного документообігу, автоматизації робочих процесів, покращення організації документообігу. В роботі розроблена підсистема ведення електронного документообігу підприємства.

Інформаційна підсистема дозволяє ефективно та зручно передавати документи по підприємству, зменшує затрати на друкування документів, та підвищує продуктивність співробітників.

Вироблено опис процесу розробки і тестування підсистеми, реалізовано та описано користувальницький інтерфейс, зроблено знімки екранних форм програмного засобу. Продемонстровано результат виконаної роботи. Підсистема задовольняє всім вимогам, котрі пред'явлені в технічному завданні.

Зміст

ВСТУП

Тема електронного документообігу є дуже актуальною в наш час, оскільки більшість установ використовують паперові документи та процеси ручного обміну документами. Це часто призводить до низької ефективності та затримок в роботі, а також до збільшення витрат на обробку та зберігання документів. Метою нашої роботи є детальне дослідження принципів роботи підсистеми електронного документообігу, аналіз функцій її архітектури роботи, різні моделі під час розробки, а також економічний розрахунковий план виконання проекту.

Мета дослідження: Розробка підсистеми електронного документообігу на підприємстві полягає в створенні ефективного та автоматизованого інструменту для обробки, зберігання та управління документами. Головною метою є поліпшення процесів роботи з документами, зменшення часу і затрат, покращення доступності, надійності та безпеки документів.

Дослідження спрямоване на розробку підсистеми, яка забезпечить ефективну обробку вхідних і вихідних документів, автоматизацію робочих процесів, керування правами доступу, забезпечення конфіденційності та цілісності документів, а також зручний пошук та аналіз інформації.

Мета дослідження полягає в покращенні організації документообігу на підприємстві, спрощенні рутинних завдань, підвищенні продуктивності працівників, зменшенні ймовірності помилок та втрати документів, а також в ефективному управлінні та контролі за документами. Основним результатом дослідження є розробка функціональної та надійної підсистеми електронного документообігу, яка буде впроваджена на підприємстві для покращення роботи з документами та оптимізації бізнес-процесів.

Об’єкт дослідження: є розробка підсистеми електронного документообігу на підприємстві. Цей об'єкт включає в себе всі аспекти,

пов'язані з управлінням, обробкою, зберіганням і обміном документами на підприємстві.

Дослідження охоплює аналіз поточного стану документообігових процесів на підприємстві, виявлення проблем та недоліків, а також визначення вимог та потреб щодо покращення системи обробки. Об'єктом дослідження є також вивчення різних принципів, методів та інструментів, які можуть бути використані для реалізації підсистеми.

Під час дослідження будуть аналізовані різні аспекти, такі як структура документообігових процесів, вимоги до безпеки та конфіденційності документів, інтеграція з іншими системами, ефективність та продуктивність роботи з документами. Об'єктом дослідження є не лише сама підсистема електронного документообігу, але й його вплив на загальну організацію роботи підприємства.

Результатом дослідження буде розробка функціональної та ефективної підсистеми у вигляді веб - додатку електронного документообігу, яка відповідатиме вимогам і потребам підприємства і сприятиме поліпшенню управління документами та оптимізації бізнес-процесів.

Предмет дослідження: охоплює розгляд різних аспектів підсистеми електронного документообігу, включаючи його функціональні можливості, інтерфейси, архітектуру, методи аутентифікації та авторизації, засоби забезпечення безпеки та конфіденційності, а також інтеграцію з іншими системами на підприємстві.

Також розглянемо аналіз різних методологій та підходів до розробки підсистеми електронного документообігу застосовуючи декілька мов програмування, вивчення сучасних технологій, стандартів та протоколів, які можуть бути застосовані. Також будуть розглянуті питання щодо вибору оптимальних технологій, використання баз даних, розробки інтерфейсів користувача, тестування та забезпечення якості підсистеми.

Предмет дослідження дозволить глибше розуміти особливості розробки підсистеми електронного документообігу, його можливості та переваги, а також визначити оптимальні рішення для впровадження на підприємстві. Результатом дослідження буде розробка функціональної та ефективної підсистеми веб - додаток електронного документообігу, яка буде сприяти поліпшенню роботи з документами.

Методи дослідження: для проведення дослідження розробки підсистеми електронного документообігу будуть використані наступні методи:

1. Аналіз літературних джерел: прочитаємо наукові статті, книги, публікації та інші джерела, що стосуються теми електронного документообігу. Цей дозволить отримати теоретичні знання про основні принципи, методи та технології, пов'язані з розробкою підсистеми.
2. Анкетування та опитування: Цей метод включає створення анкет або проведення опитування серед співробітників підприємства. Це дозволяє зібрати думки, пропозиції та вимоги щодо покращення документообігових процесів, а також виявити проблеми, з якими стикаються користувачі.
3. Спостереження: проведемо спостереження за роботою з документами на підприємстві, вивчити поточні процеси та ідентифікувати недоліки та проблеми. Цей метод дозволяє отримати практичні дані про реальний стан документообігу і виявити можливі шляхи його покращення.
4. Прототипування та тестування: створимо прототип підсистеми у вигляді веб - додатку та проведемо його тестування. Цей метод дозволяє перевірити функціональність, ефективність та надійність розробленої системи перед її впровадженням.

5. Кількісний аналіз: Для отримання об'єктивних даних та оцінки ефективності підсистеми можуть бути використані методи кількісного аналізу. Наприклад, можна порівняти показники роботи з документами до та після впровадження підсистеми, провести аналіз часу, витрат та інших метрик.

Використання комбінації цих методів дослідження дозволить отримати комплексне розуміння проблеми, виявити потреби та вимоги користувачів, розробити ефективну підсистему електронного документообігу та оцінити її результативність.

АНАЛІЗ ТА ДОСЛІДЖЕННЯ ПРОБЛЕМИ

1.1 Актуальність теми

Актуальність теми розробки підсистеми електронного документообігу на підприємстві є дуже важливою для сучасного бізнес-середовища. Ось деякі з основних аргументів, що підкреслюють актуальність цієї теми:

1) Ефективність і також продуктивність, електронний документообіг дозволяє автоматизувати процеси обробки, передачі, зберігання документів, також зменшує ризики припущення помилок під час оформлення документів.

2) Електронний документообіг надає зручний інтерфейс для створення, редагування, обміну та звичайно пошуку документів. Користувачі можуть отримати доступ до своїх документів за будь-який період, що позитивно сприяє більш ефективній роботі та зручності взаємодії з документацією.

3) Збереження ресурсів, а саме: перехід до електронного документообігу дозволяє знизити використання паперу, чорнил та інших витратних матеріалів для друкування паперового формату. Це позитивно сприяє на екологічне середовище, що немало важливо у наш час.

4) Безпека обігу електронних документів дозволяє забезпечити конфіденційність важливої інформації та відокремити документи певного типу за правами доступу до них.

5) Масштабованість і розширюваність: це зручно, оскільки, коли потреби підприємства зростають, таку підсистему можна розширювати та адаптувати, а також додавати різні нові функції для конкретних потреб підприємства або при зміні його сфери діяльності.

6) Глобальний доступ: незалежно від того, де знаходяться співробітники підприємства, вони можуть зручно обробляти документи будь-де.

Ця тема має великий потенціал для дослідження та розробки інформаційних технологій, що сприяє вдосконаленню бізнес-процесів і підвищенню конкурентоспроможності компанії в сучасному світі.

1.2 Аналіз проблеми

За допомогою аналізу проблем можна з’ясувати основні проблеми які можуть виникати в роботі з документами як для підприємства так і для співробітників. До таких проблем можна віднести:

1) Зберігання та пошук документів. Традиційні методи розповсюдження паперових документів можуть створити проблеми зі зберіганням і пошуком документів. З багатьма паперовими документами важко працювати, а пошук потрібного може зайняти багато зусиль та часу.

2) Втрата та пошкодження документів. Паперові документи можливо втратити або пошкодити при неправильному зберіганні, пожежі чи інших негативних незалежних факторах. Як наслідок маємо неповну або недоступну інформацію, що в свою чергу може понести за собою збитки для підприємства.

3) Повільність процесу. При паперовому документообігу обіг документів може бути повільним та неефективним. Документація може затримуватися в обробці на різних етапах, особливо в ті часи коли інші співробітники чекають доступу до необхідної їм інформації, як результат - зменшується продуктивність робочих процесів.

4) Велика кількість ручних операцій. Традиційний документообіг потребує немалу кількість ручних операцій: друкування, підписання, доставка та архівування документів. Виникають зайві затрати часу та ресурсів, до того ж зростає ймовірність допущення помилки.

5) Низька прозорість та контроль. В традиційному документообігу може бути складно забезпечити прозорість та контроль над процесами обробки документів. Відстеження статусу документів, контроль за доступом до них та визначення відповідальних осіб можуть бути недостатніми або незручними, що може призвести до недосконалості управління документами.

6) Велика кількість розсипаних даних. У традиційному паперовому документообігу інформація може бути розсіяна по різних документах, що ускладнює її пошук та аналіз. Відсутність централізованого зберігання та структурування даних може призвести до втрати важливої інформації та неефективного використання ресурсів

7) Відсутність мобільності та гнучкості. Традиційне керування документами може обмежити вашу здатність працювати з документами віддалено або в дорозі. Відсутність доступу до документів із мобільних пристроїв і обмежена можливість редагувати та ділитися документами поза офісом можуть знизити продуктивність і сповільнити робочий процес.

8) Труднощі в архівуванні та знищенні документів. У традиційному управлінні документами архівування та знищення документів може бути проблематичним процесом. Зберігання великої кількості паперових документів вимагає значних просторових ресурсів і особливих умов зберігання. Крім того, забезпечення безпеки та конфіденційності під час знищення документів може бути складним.

Аналіз цих питань виявив необхідність впровадження власної підсистеми електронного документообігу. Вирішення цих проблем дає такі переваги:

1) Зручніший доступ та швидкий пошук документів. Електронний документообіг забезпечує централізоване зберігання документів та зручний доступ до них з будь-якого місця та в будь-який час. Ефективні інструменти пошуку документів дозволяють швидко знаходити необхідну інформацію.

2) Підвищення продуктивності та швидкості роботи. Електронний документообіг дозволяє автоматизувати багато рутинних процесів, зменшуючи необхідність в ручній обробці документів. Це прискорює робочі процеси та забезпечує більш ефективне використання часу співробітників.

3) Покращена контрольованість і прозорість. Електронний документообіг дозволяє визначати доступ до документів, відстежувати їх рух

і контролювати стан обробки. Це допомагає забезпечити більшу прозорість, підзвітність і контроль документів.

4) Забезпечення безпеки та конфіденційності. Обіг електронних документів дозволяє використовувати механізми шифрування, автентифікації та контролю доступу до документів, забезпечуючи високий рівень безпеки та конфіденційності інформації.

5) Зменшення витрат. Впровадження електронного документообігу дозволяє знизити витрати на паперову документацію, друкування, доставку та зберігання документів. Це призводить до економії фінансових ресурсів та зменшення негативного впливу.

6) Підвищення якості та точності. Електронний документообіг дозволяє запобігати помилкам, пов'язаним з ручним введенням даних, дублюванням документів та втратою інформації. Автоматизовані процеси обробки документів забезпечують високу точність та якість даних.

7) Покращення співпраці та комунікації. Електронний документообіг сприяє зручній співпраці між співробітниками та підрозділами підприємства. Засоби коментування, обговорення та підписування документів дозволяють ефективно обмінюватись ідеями, отримувати згоду та здійснювати колективну роботу над документами.

8) Забезпечення відповідності та контроль процесів. Електронний документообіг дозволяє організаціям відповідати вимогам до зберігання, обробки та передачі даних, включаючи законодавчі та нормативні вимоги. Це допомагає забезпечити корпоративну відповідність і контролювати процеси обробки документів.

Аналіз проблеми розкриває потенційні вигоди та переваги впровадження підсистеми електронного документообігу в організаціях та на підприємствах. Електронний документообіг може підвищити ефективність роботи, забезпечити точність і безпеку даних, сприяти кращій, зручнішій співпраці, та посилити контроль процесів. Впровадження таких підсистем є актуальним

і важливим кроком для підвищення ефективності та конкурентоспроможності сучасних підприємств.

Основуючись на контексті розробки підсистеми електронного документообігу, побудуємо можливе дерево проблем на рисунку 1.1, які можуть виникнути на підприємстві.

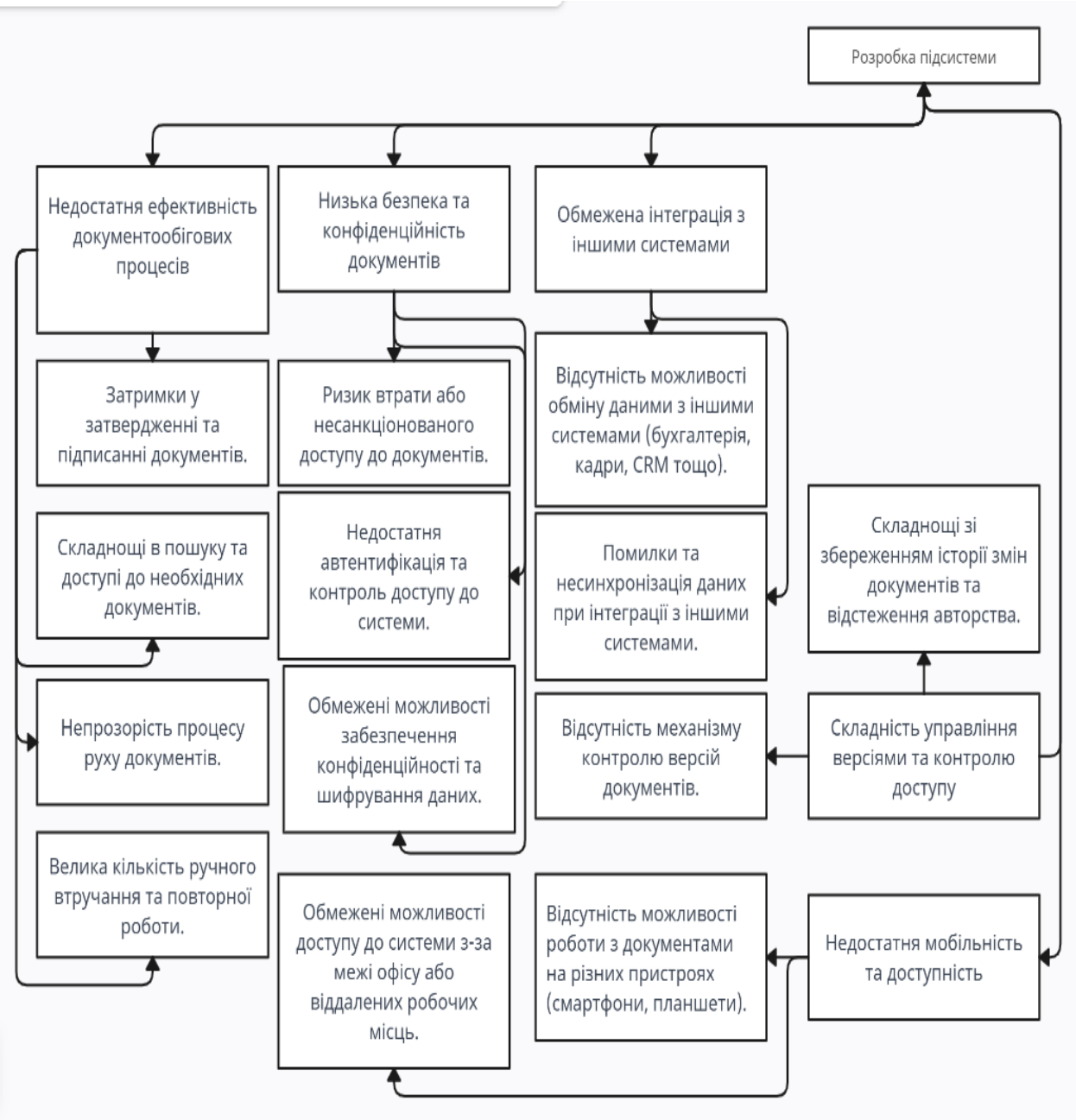

Рисунок 1.1 - дерево проблем

Після аналізу дерева проблем, видно деякі важливі аспекти:

1. Недостатня ефективність документообігових процесів впливає на кілька інших проблем, таких як затримки у затвердженні та підписанні

документів, складнощі в пошуку та доступі до необхідних документів, непрозорість процесу руху документів та велика кількість ручного втручання. Ці проблеми можуть мати негативний вплив на продуктивність та ефективність роботи на підприємстві.

2. Низька безпека та конфіденційність документів є ще однією значною проблемою. Ризик втрати або несанкціонованого доступу до документів може мати серйозні наслідки для підприємства. Для вирішення цієї проблеми необхідно впровадити механізми автентифікації, контролю доступу та шифрування даних.

3. Обмежена інтеграція з іншими системами також є проблемою, яка впливає на ефективність роботи. Відсутність можливості обміну даними з іншими системами може призводити до помилок та несинхронізації даних. Розробка механізмів інтеграції та взаємодії з іншими системами може полегшити роботу та покращити продуктивність.

4. Складність управління версіями та контролю доступу також потребує уваги. Відсутність механізму контролю версій документів та відстеження авторства може призводити до непослідовності та втрати історії змін. Розробка системи управління версіями та контролю доступу допоможе полегшити цей процес та забезпечити точність та надійність даних.

5. Недостатня мобільність та доступність є ще однією проблемою, яку варто вирішити. Обмежені можливості роботи з документами на різних пристроях та обмежений доступ до системи за межі підприємства обмежують гнучкість та продуктивність роботи. Розробка мобільних додатків та розширення доступу до системи можуть покращити мобільність та доступність для співробітників.

Ці проблеми вимагають комплексного підходу до вирішення. Розробка підсистеми електронного документообігу, яка враховує ці аспекти, може допомогти поліпшити ефективність, безпеку та доступність документообігових процесів на підприємстві.

1.3 Аналіз існуючих систем електронного документообігу

В першу чергу розуміємо, що системи електронного документообігу можуть поділятися на види (рисунок 1.2).

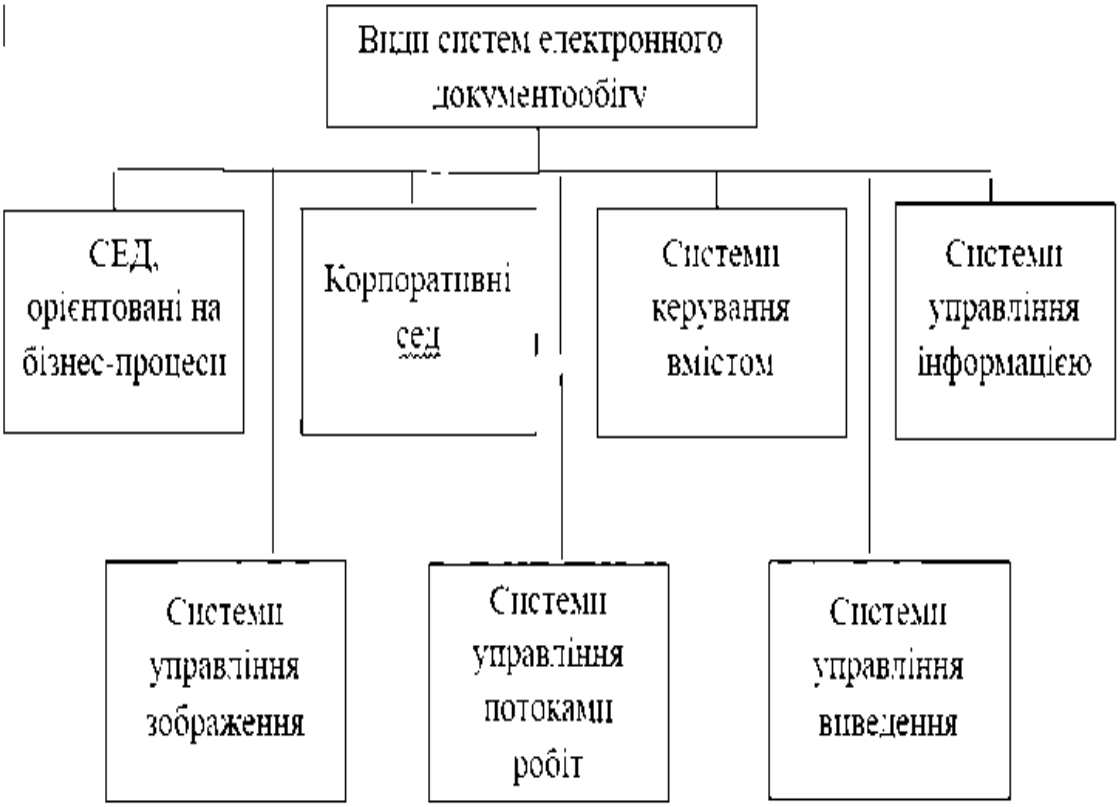

Рисунок 1.2 - Види електронного документообігу

Деякі з існуючих систем, які можна проаналізувати, включають:

1) Microsoft SharePoint: це програмне забезпечення для спільної роботи та керування документами, яке включає можливості керування електронними документами. Він має можливість створювати, редагувати та зберігати документи, включаючи контроль версій та доступ користувачів до документів.

2) OpenKM: це відкрите програмне забезпечення для керування документами, яке включає функції керування електронними документами. OpenKM здатний створювати, редагувати та зберігати документи, а також контролювати версії, розповсюджувати документи та автоматизувати робочий процес.

3) Alfresco: це програмне забезпечення для керування документами та спільної роботи з відкритим вихідним кодом, яке включає можливості

керування електронними документами. Alfresco має можливість створювати, редагувати та зберігати документи, а також контролювати версії, контролювати доступ і співпрацювати з документами.

4) Диск Google: це хмарне сховище даних з електронним керуванням документами. Диск Google має можливість створювати, редагувати та зберігати документи, а також контролювати версії та співпрацювати з документами.

5) DocuSign: це програмне забезпечення для електронного підпису та керування документами з функціями керування електронними документами. DocuSign має можливість створювати та редагувати документи, керувати їх версіями та зберігати їх в електронному вигляді, а також має функції підпису та надсилання документів.

Простіше кажучи, ми розуміємо, які функції та можливості є найбільш важливими для користувачів, а також який підхід до документообігу є найбільш ефективним і зручним у використанні. Наприклад, можна визначити, що користувачі повинні мати можливість легко та зручно завантажувати документи, відстежувати їхній статус і надсилати їх на підписи або, можливо, автоматизувати робочі процеси за допомогою бізнес-правил та інтеграції з іншими системами.

1.4 Дерево цілей

Основні цілі розробки автоматизованої підсистеми електронного документообігу на підприємстві можна представити у вигляді дерева цілей з такою структурою (рисунок 1.3):

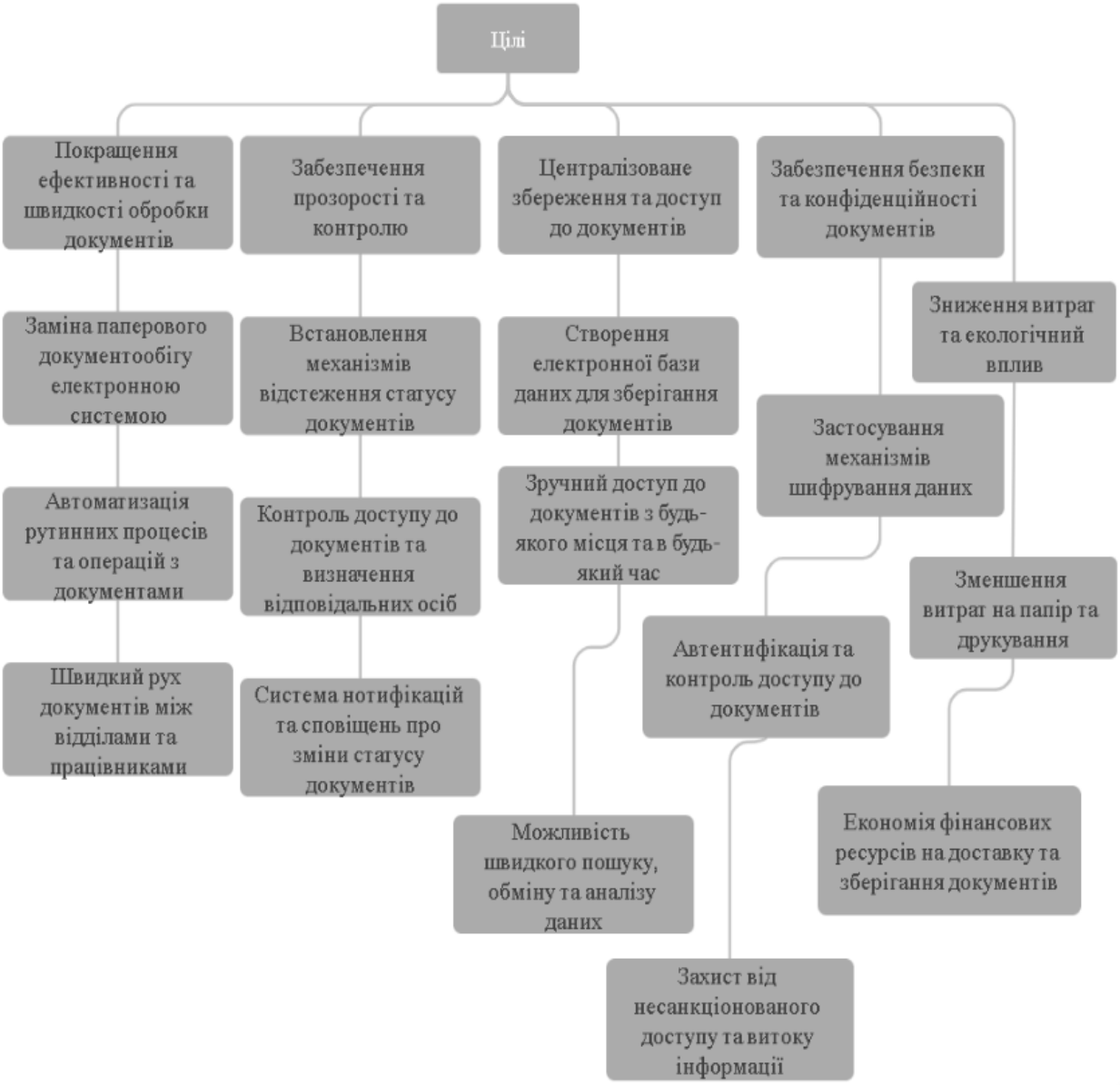

Рисунок 1.3 - дерево цілей

Ці цілі враховують потреби та вимоги підприємства та досягнення цих цілей принесе значні переваги підприємству.

1.5 Вибір підприємства для виконання роботи

В обраній темі дипломної роботи розробка підсистеми автоматичного електронного документообігу буде для вищого навчального закладу. Такий вибір зроблено з метою дослідження та впровадження сучасних технологій і практик у сфері електронного документообігу в освітнє середовище.

Основними причинами вибору вищого навчального закладу для розробки та впровадження підсистеми автоматизованого електронного документообігу є:

1) Розмір і складність документообігу. Вищі навчальні заклади мають розгалужену структуру та великий потік документів, які опрацьовуються щодня, тому для роботи буде взято одну кафедру навчального закладу. Для забезпечення оперативної передачі, зберігання та обробки інформації потрібна ефективна та надійна система документообігу.

2) Покращення організації та ефективності роботи. Розробка підсистеми автоматичного електронного документообігу дозволить вищим навчальним закладам оптимізувати робочий процес, скоротити час обробки документів та покращити комунікацію між співробітниками.

3) Навчання та розвиток студентів. Розвиток підсистеми електронного документообігу надасть можливість студентам набути практичних навичок реалізації проектів з використанням сучасних інформаційних технологій. Вони зможуть поглибити знання у сфері електронного документообігу та отримати цінний досвід співпраці з вищими навчальними закладами.

4) Проведення дослідницької роботи. Вищі навчальні заклади часто проводять дослідницьку роботу в галузі інформаційних технологій та управління документами. Розвиток автоматизованих підсистем електронного документообігу надасть можливості для поглиблення наукових знань, проведення досліджень і експериментів, впровадження нових розробок і вдосконалень у сфері документообігу.

5) Високі вимоги до конфіденційності та безпеки. Вищі навчальні заклади повинні обробляти важливу та конфіденційну інформацію, таку як персональні дані студентів і викладачів, наукові дослідження, інформація про фінансові новини тощо. Розробка підсистеми автоматизованого документообігу дозволить забезпечити високий рівень захисту даних і контролю доступу.

6) Подати приклад іншим установам. Вищі навчальні заклади можуть бути прикладом іншим установам у впровадженні електронного документообігу. Результати розробки та впровадження модуля можуть бути використані для створення стандартів і методик у сфері електронного документообігу в освітній галузі.

Вибір вищого навчального закладу для розробки автоматичної підсистеми електронного документообігу сприятиме покращенню організації роботи, розвитку студентів, проведенню наукових досліджень, показуватиме приклад іншим закладам.

1.6 Аналіз діяльності підприємства, як організаційно - технічної системи.

Щоб зрозуміти роботу відділу, проаналізуємо його роботу як підрозділу. Кафедра є основним структурним підрозділом навчального закладу, оскільки об’єднує у своїй діяльності всі напрямки діяльності та розвитку закладу даного типу. Модель управління кафедрою, а точніше її діяльністю, складається з п'яти взаємопов'язаних елементів:

1) метою управління діяльністю кафедри з питань розвитку інноваційних процесів є сприяння впровадженню інновацій у підсистему загальної освіти через розвиток інноваційних процесів у системі професійного навчання педагогічного колективу та підвищення їх кваліфікації;

2) теоретичні основи та методика управління діяльністю кафедри з питань розвитку інноваційних процесів;

3) організаційно-педагогічні умови ефективного управління діяльністю кафедри з питань розвитку інноваційних процесів;

4) технологія, яка керує діяльністю кафедри з розвитку інноваційних процесів у взаємодоповнювальній системі професійної підготовки викладачів (набір дій, заходів і процесів, як організовано більшість умов),

щоб забезпечити досягнення бажаних результатів і включає наступні кроки: аналітико-пошуковий, організаційно-діяльнісний.

5) результатом управління діяльністю кафедри з розвитку інноваційних процесів є нова трансформація системи професійної освіти педагогів та системи загальної освіти.

Кафедра, як структурний підрозділ.

Ми розуміємо ефективність управління кафедрою як комплексну концепцію, що складається щонайменше з трьох компонентів. Під економічною ефективністю діяльності кафедрами розуміємо ступінь використання ресурсів і засобів, які є у розпорядженні завідувача кафедрою. Ефективність галузевої освітньої діяльності залежить від якості освіти за напрямом галузевої діяльності, яку можна визначити як здатність випускників відповідати вимогам національних освітніх стандартів та вимогам конкретних посад. Крім того, результативністю виховної діяльності характеризують розроблені та реалізовані у процесі практичної діяльності навчальні програми: навчально-методичної, наукової, організаційної та організаційно-виховної роботи. Соціальна ефективність досліджуваної галузі характеризується такими показниками:

• Згуртованість колективу та їх розвиток трудової активності

• Розвиток особистості кожного співробітника кафедри

• Соціально-психологічне середовище всередині факультету

• Соціальні відносини та їх координація

• Здатність завідуючого кафедрою утримувати перспективних викладачів та залучати їх до різного роду діяльності.

Сутність поняття «кафедра» та її сучасне розуміння, а також визначення основних напрямків діяльності і основних функцій кафедри (освітня, методична, науково-педагогічна, підготовка кадрів, професійний розвиток тощо). Поняття науково-педагогічної діяльності суб’єкта, під якою ми розуміємо діяльність викладачів і співробітників кафедри щодо забезпечення реалізації основних функцій і напрямків роботи кафедри (рисунок 1.4).

Функції кафедри вищого навчального закладу

Навчальна	Методична	Науково-дослідна	Виховна	Підготовка науково-педагогічних кадрів та підвищення кваліфікації фахівців
проведення всіх видів занять, передбачених навчальними планами; проведення заходів з організації самостійної роботи студентів розробка і використання нових технологій навчання створення умов для задоволення потреб студентів в підвищенні рівня професійних знань, умінь, навичок	розробка навчальних програм з комплексу дисциплін спеціальностей кафедри, розгляд їх на своїх засіданнях, затвердження документів в рамках своїх повноважень підготовка підручників, навчально-методичних посібників; розробка навчально-методичних рекомендацій і матеріалів, ведення науково-методичної роботи з актуальних проблем сучасної освіти	організація і проведення науково-дослідної роботи з актуальних проблем сучасної освіти і з упровадження результатів на практиці; вивчення ефективності своєї освітньої професійної діяльності, співпраця з іншими освітніми установами і органами управління освітою; участь у науково-дослідних програмах; модифікація існуючих та створенні нових освітніх моделей та технологій; підготовка наукових розробок на конференціях та семінарах	упровадження в освітній процес сучасних педагогічних технологій, що сприяють самореалізації особистісного потенціалу студентів; проведення заходів з організації виховної роботи серед студентів	формування наукового потенціалу шляхом залучення співробітників до навчання в аспірантурі та докторантурі; вивчення, узагальнення і розповсюдження досвіду роботи викладачів; моніторинг ефективності своєї освітньої професійної діяльності, співпраця з іншими освітніми установами і органами управління освітою; підготовка висококваліфікованих фахівців, що володіють глибокими теоретичними і необхідними практичними знаннями

Рисунок 1.4 - функції кафедри

При опису та розгляді функцій кафедри для більшого розуміння побудуємо схеми для кожної функції. Тому для усіх функцій кафедри схеми відображено на рисунках 1.5 – 1.9.

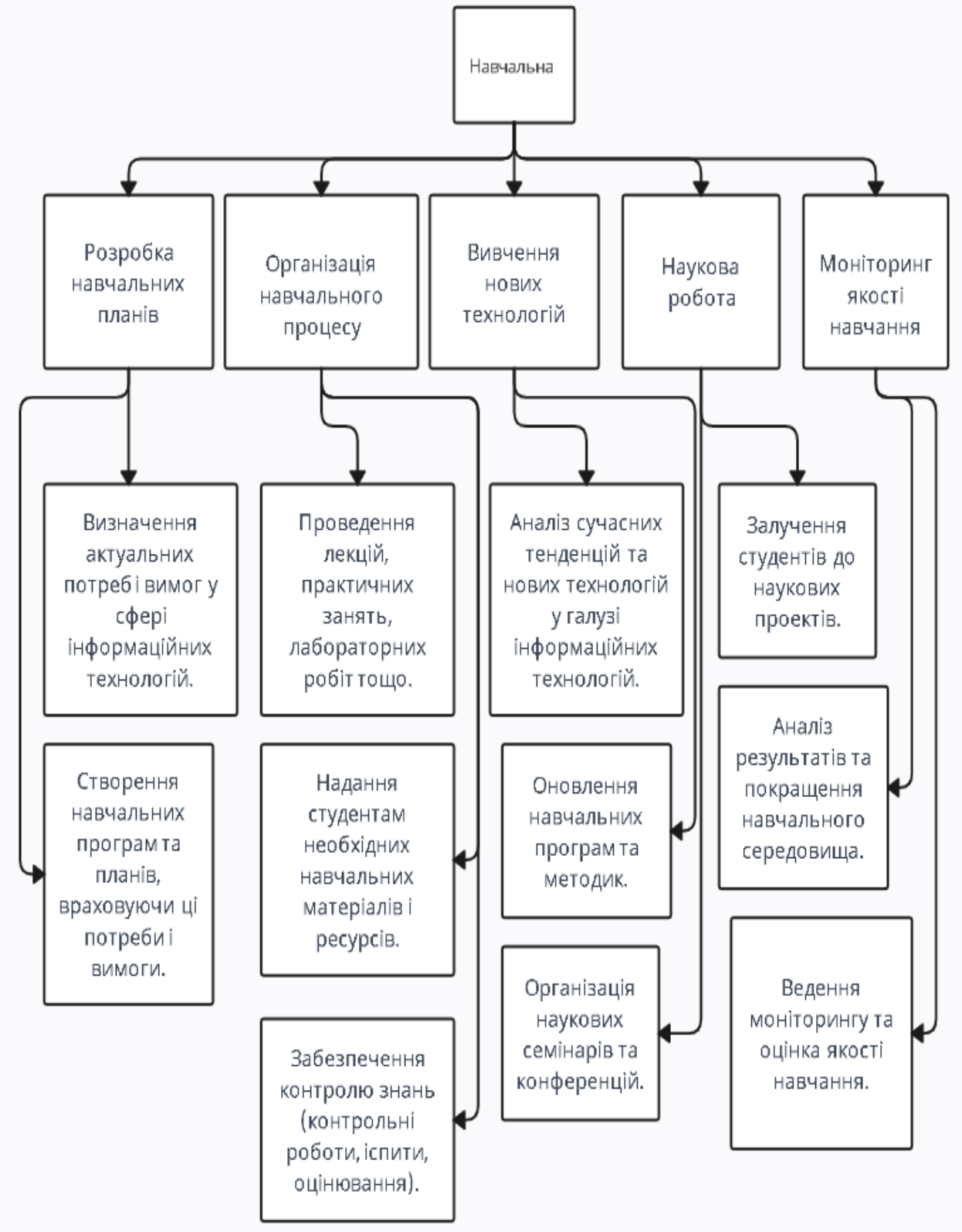

Рисунок 1.5 - Навчальна функція кафедри

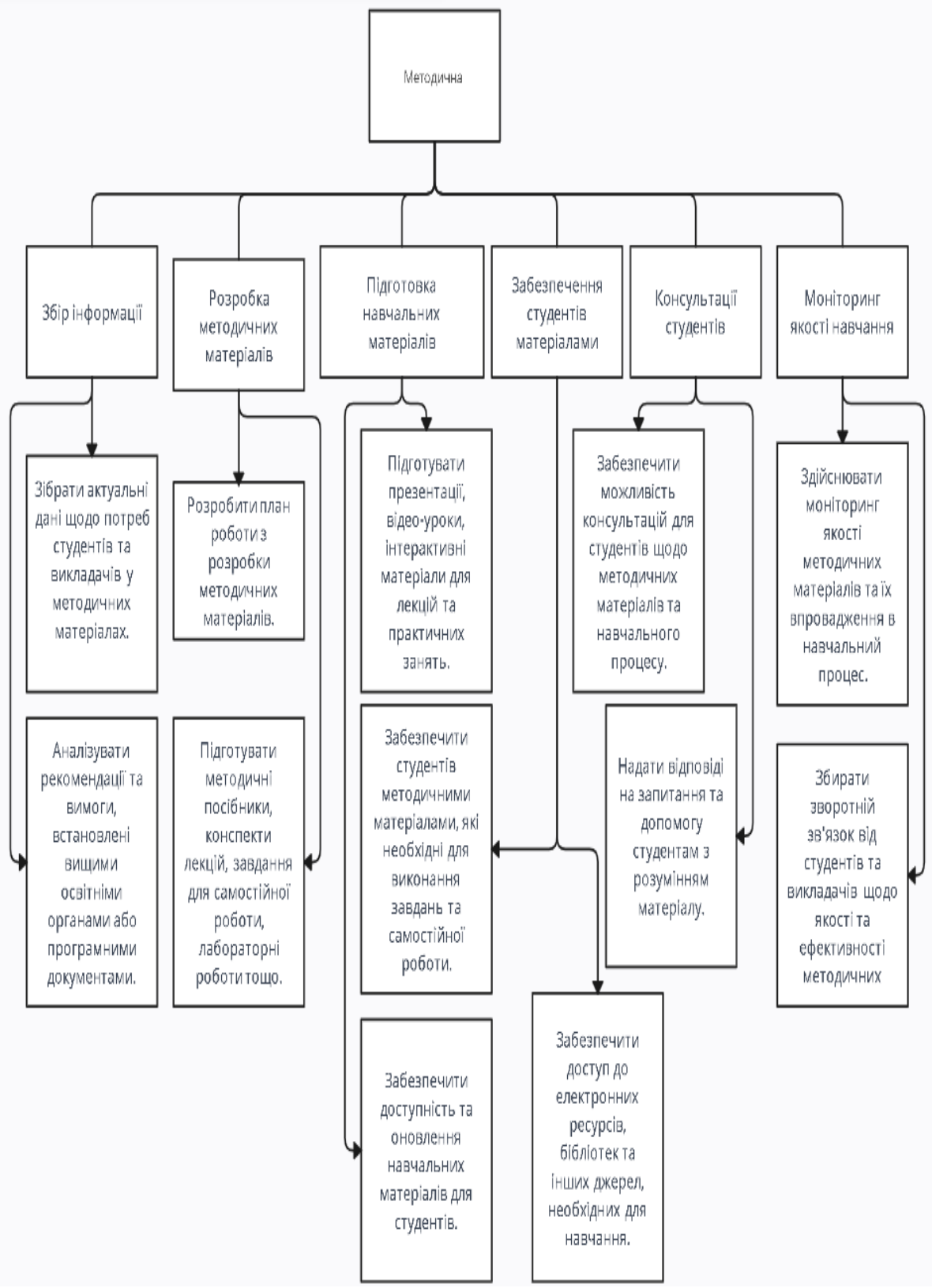

Рисунок 1.6 - Методична функція кафедри

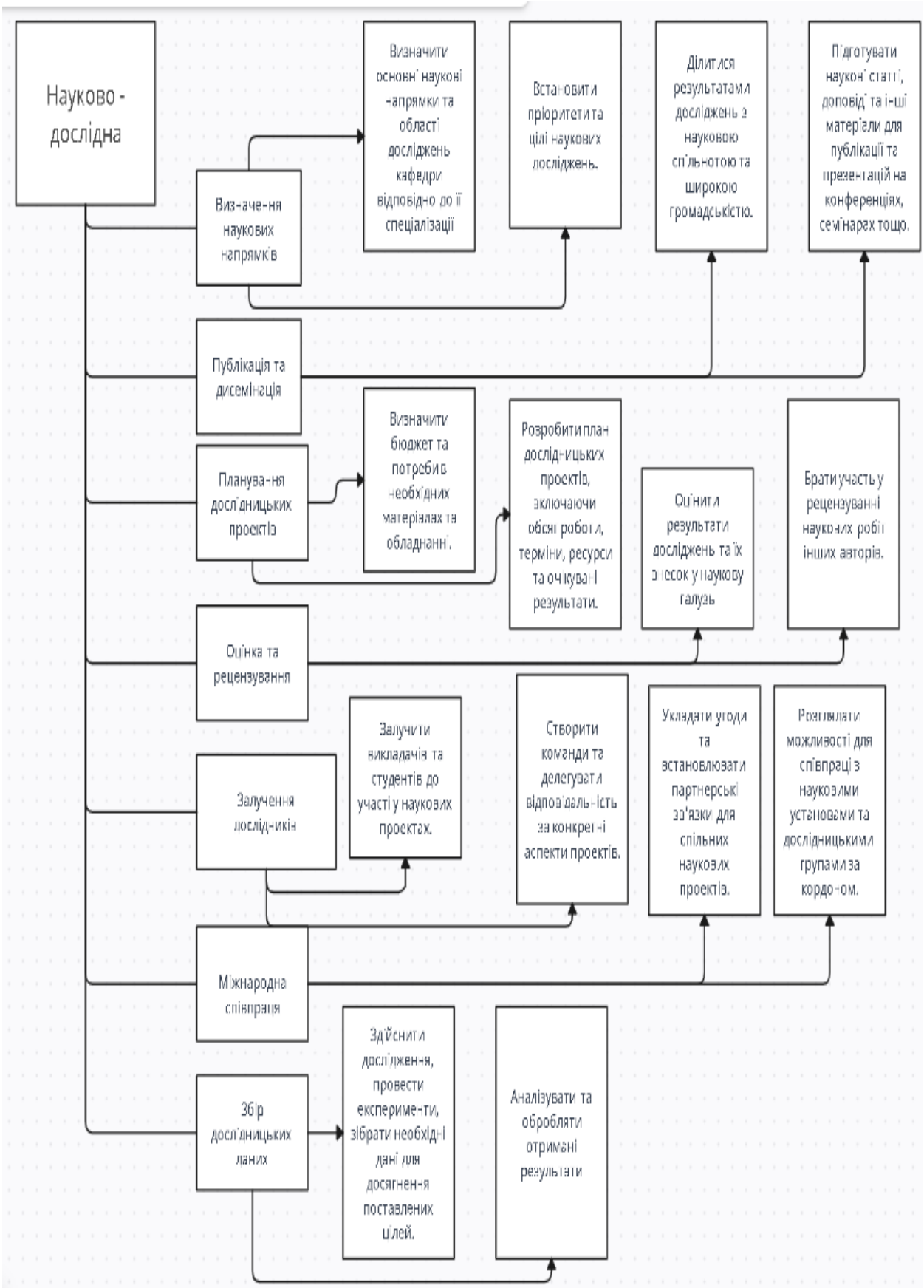

Рисунок 1.7 – Наукова - дослідна функція кафедри

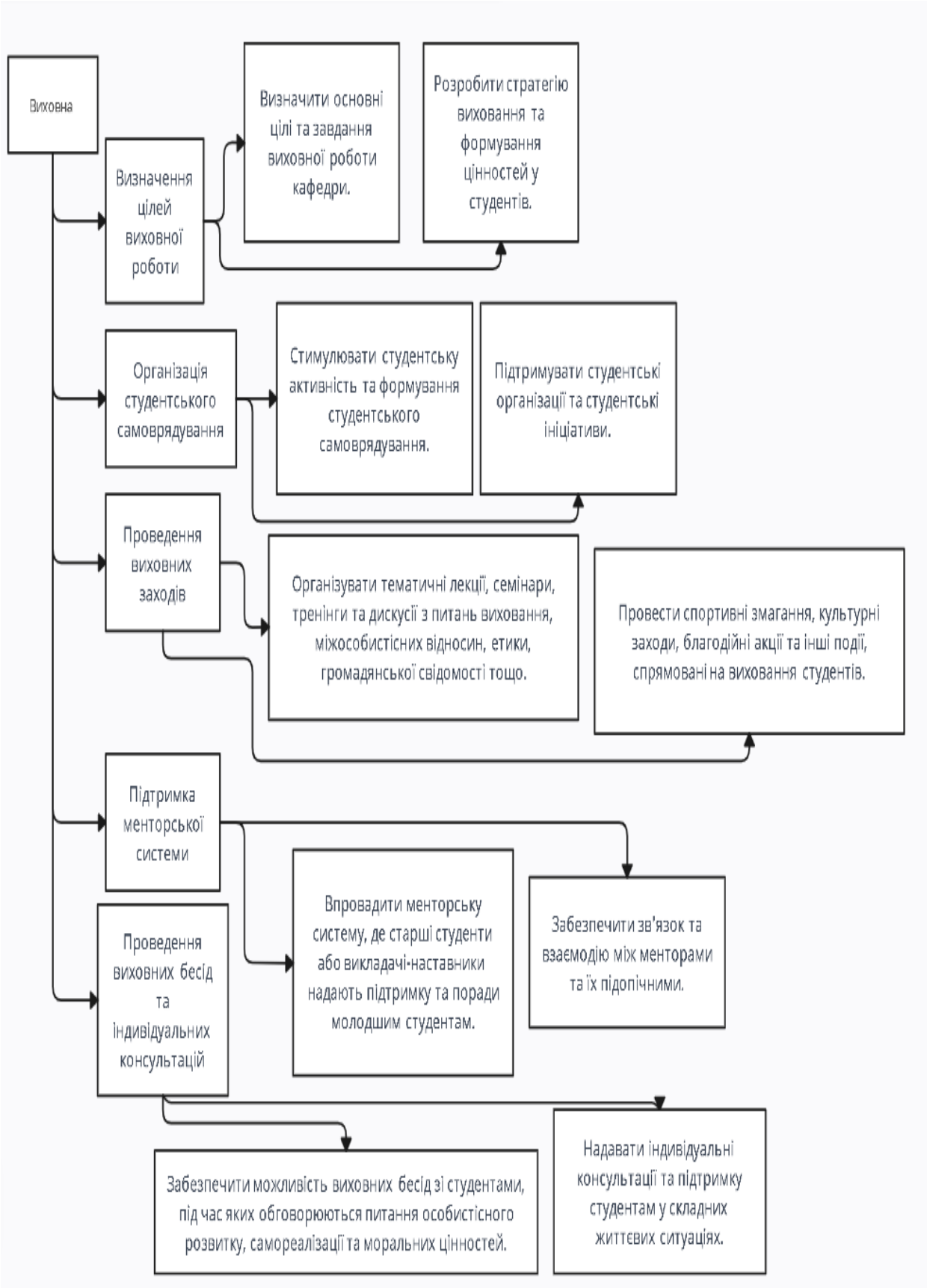

Рисунок 1.8 - Виховна функція кафедри

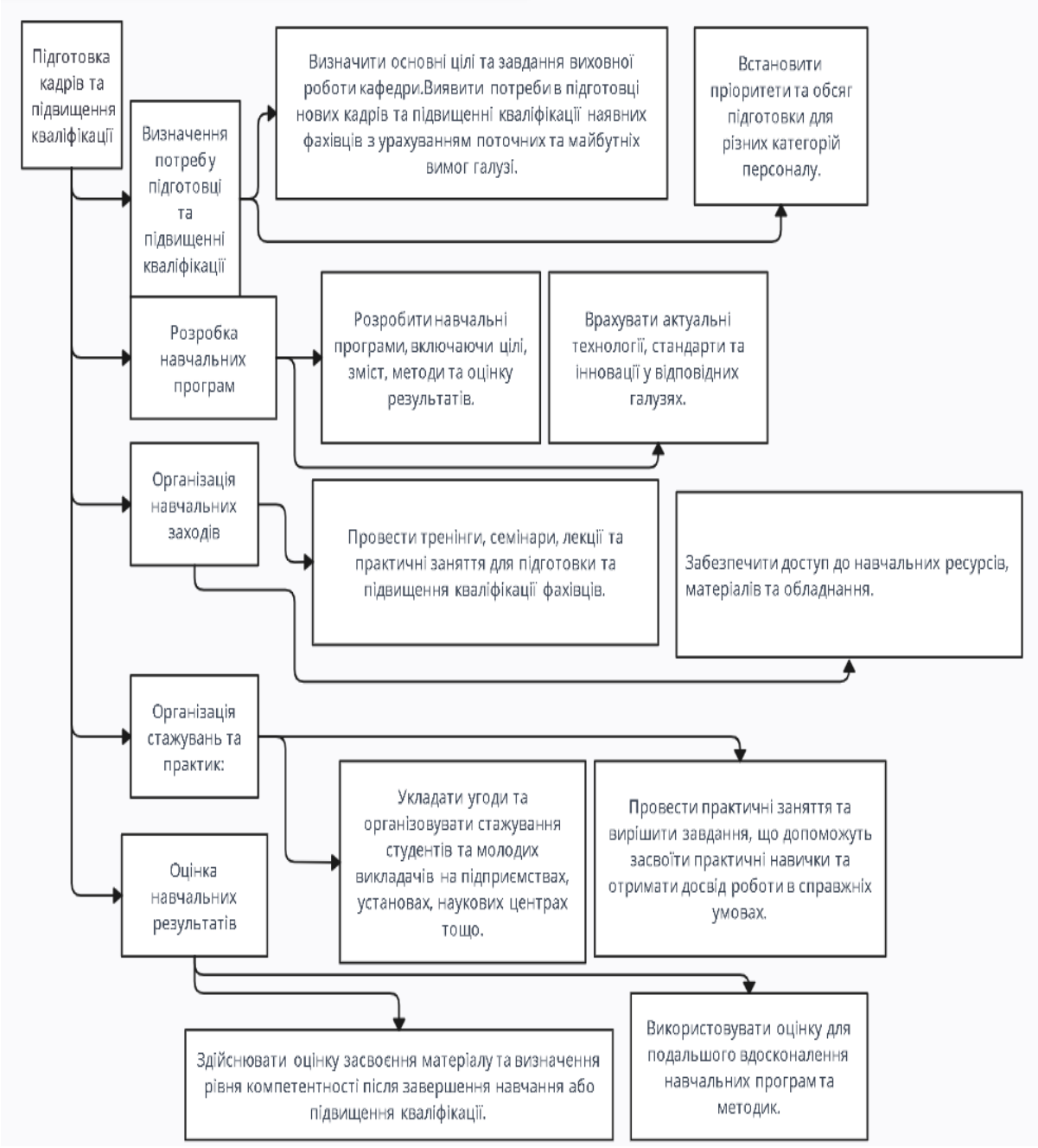

Рисунок 1.9 – Підготовка кадрів та підвищення кваліфікації фахівців

Після того як ми розробили схеми роботи для різних функцій кафедри, таких як навчальна, методична, науково-дослідна, виховна, підготовка кадрів та підвищення кваліфікації фахівців, маємо розуміння, що застосування даних схем може мати декілька переваг, а саме:

1. Структурованість: Алгоритми надають структурований підхід до виконання різних функцій кафедри. Вони розбивають процес на окремі

кроки, що сприяє більш організованому та систематичному підходу до роботи.

2. Оптимізація часу: Алгоритми допомагають ефективно розподілити час та ресурси для виконання різних завдань. Вони дозволяють уникнути зайвих перекладань та втрати часу, оскільки кожен крок має чітко визначену послідовність дій.

3. Консистентність: Алгоритми допомагають забезпечити консистентність виконання функцій кафедри. Вони стандартизують процеси та дозволяють досягати бажаних результатів згідно з встановленими стандартами та процедурами.

4. Моніторинг та оцінка: Алгоритми дозволяють здійснювати моніторинг та оцінку виконання функцій кафедри. Кожен крок алгоритму може бути відстежуваний та оцінений для визначення ефективності та внесення необхідних змін для поліпшення роботи кафедри.

1.7 Аналіз законодавчих та нормативно - правових актів

Є два Закони України, які являються підґрунтям для використання електронного документообігу на законодавчому рівні. Ці закони були прийняті ще у 2003 році: «Про електронний документ та електронний документообіг» і «Про електронний цифровий підпис». В цих законах висвітлюються такі поняття як: електронний документ, цифровий електронний підпис, сертифікати ключів електронного підпису, загальні засади функціонування центрів сертифікації ключів, принципи організації електронного документообігу

Електронний документ - данний документ, що містить інформацію у вигляді електронних даних та містить обов’язкові реквізити документа. Це визначення поняття висвітлено в статті 5 Закону України «Про електронні документи та електронний документообіг».

Оригінали електронних документів – це електронні примірники документів, що містять обов'язкові реквізити, у тому числі електронні підписи автора, прирівняним до власноручного підпису відповідно до Закону України «Про електронний цифровий підпис». У разі надсилання електронного документа кільком адресатам кожна його копія вважається оригіналом електронного документа. Оригінал електронного документа повинен підтверджувати його цілісність та справжність у встановленому законом порядку.

Слід зазначити, що паперові документи та електронні документи схожі за інформацією та реквізитами документа та мають однакову юридичну силу. Необхідні реквізити електронного документа в електронному документі є необхідні дані, без яких це неможливо є основою для бухгалтерського обліку і не має юридичної сили.

Що стосується юридичної сили електронного документа, то його не можна заперечувати або не враховувати лише тому, що він має електронну форму. Також слід зазначити, що електронний документ не може бути використаний як оригінал, якщо він:

1) Є свідоцтвом про право на спадщину;
2) Документ, який може бути створений лише в оригіналі, лише в одному примірнику, згідно з чинним законодавством. Виняток: якщо наявне централізоване сховище електронних оригіналів;
3) В інших випадках, передбачених законом.

З правової точки зору забезпечення дотримання нормативно-правових актів у сфері застосування електронного документообігу та електронного цифрового підпису регулюється також іншими документами:

1) Закони України від 22.05.03 № 852-IV «Про електронний цифровий підпис»; Постанови Кабінету Міністрів України:

2) «Про затвердження Порядку засвідчення наявності електронного документа (електронних даних) на певний момент часу» від 26 травня 2004 р. № 680;

3) «Про затвердження Положення про центральний засвідчувальний орган» від 28 жовтня 2004 р. № 1451;

4) «Про затвердження Порядку застосування електронного цифрового підпису органами державної влади, органами місцевого самоврядування, підприємствами, установами та організаціями державної форми власності» від 28 жовтня 2004 р. № 1452;

5) «Про затвердження Типового порядку здійснення електронного документообігу в органах виконавчої влади» від 28 жовтня 2004 р. № 1453;

6) «Про затвердження Порядку обов’язкової передачі документованої інформації» від 28 жовтня 2004 р. № 1454.

ПРИНЦИП РОБОТИ

2.1 Архітектура електронного обігу документів підприємства

Розглянемо документообіг двох варіантів, ручного та електронного. Для більшого розуміння можна розглянути на рисунку 2.1 та табличний варіант переваг і недоліків обох варіантів в таблиці 2.1:

Рисунок 2.1 документообіг

Таблиця 2.1

Переваги автоматизованої технології роботи з документами

Робота з паперовими документами (традиційна технологія)	**Робота з електронними документами (автоматизована технологія)**
Близько 15% усіх документів безповоротно втрачаються	Зростання продуктивності праці працівників складає 25 - 50%
До 30% робочого часу працівників витрачається на пошук необхідних матеріалів	Час обробки одного документа зменшується більше ніж на 75%
Усього 8% від затрат часу в роботі з документом витрачається на роботу над його змістом	Вивільняється до 65% часу на роботу саме над змістом документа, а не на рутинні операції введення, сортування, розмноження, маршрутизації документу
Для кожного документа створюється в середньому 19 його копій	Час створення одного документу скорочується на 20-30% (завдяки швидкості пошуку і наявності прототипів)
Прямий потік документів, скорочення зворотних переміщень документів в організації	Оптимізація ділових і управлінських процесів, маршрутизація документів в організації на основі корпоративних інформаційних технологій, можливість одночасної роботи над електронним документом декількох учасників документообігу
Зосередження більшої частини формальних і технічних операцій з документами в діловодній службі, а змістовних – в інших підрозділах організації	Співробітники організації (включаючи фахівців і керівників) стають безпосередніми учасниками електронного документообігу в рамках ділових і управлінських процесів

Аналізуючи дану таблицю, можна зробити висновок, що основними перевагами при використанні електронного документообігу є підвищення продуктивності праці співробітників, зменшення часу створення та обробки документів, скорочення витрат на паперові документи.

Також наведемо приклад руху документів по співробітникам підприємства та їх ролі в програмному забезпеченню на рисунку 2.2.

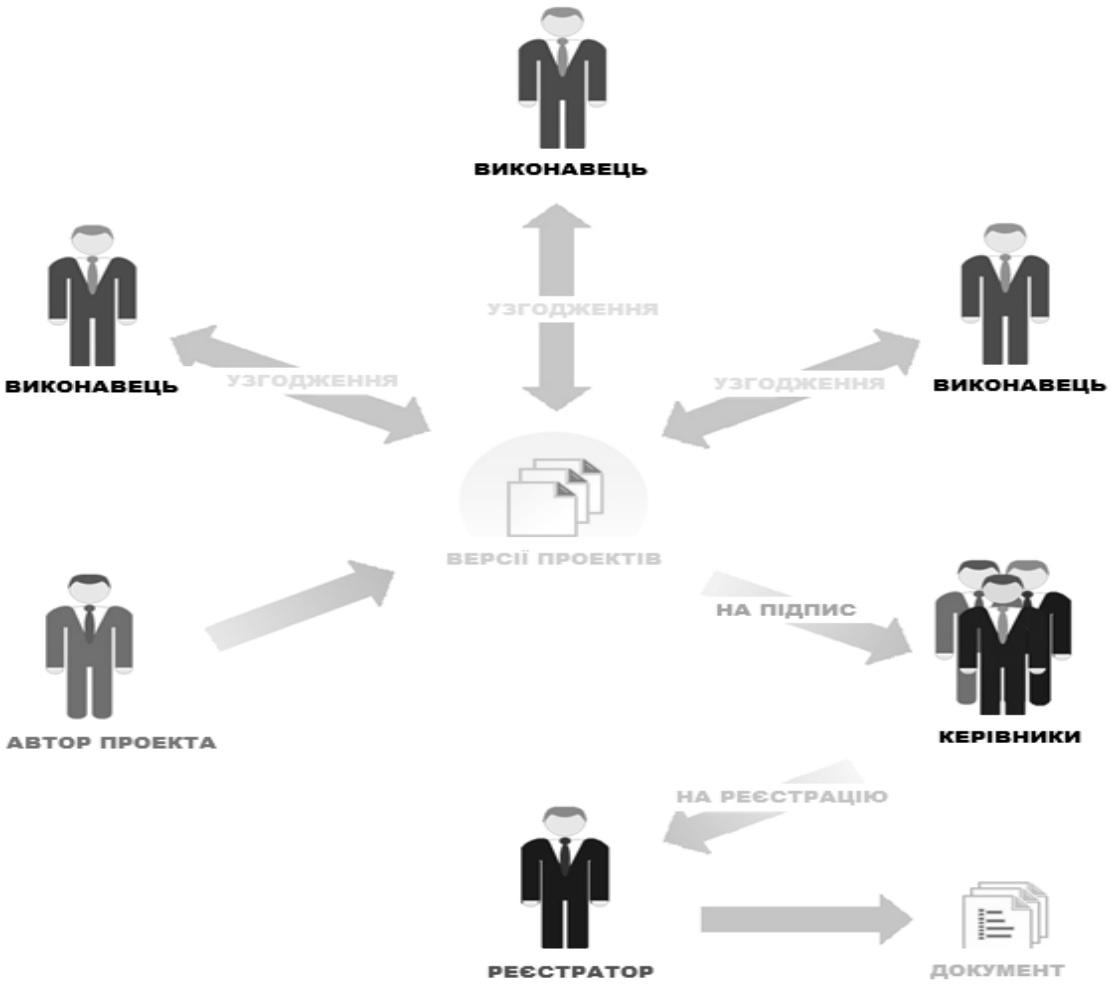

Рисунок 2.2 - приклад руху документів

Також побудуємо та відобразимо UseCase діаграму яка також є важливим інструментом для візуалізації функціональності системи та взаємодії користувачів з нею. В рамках нашого проекту вона буде виглядати наступним чином, рисунок 2.3:

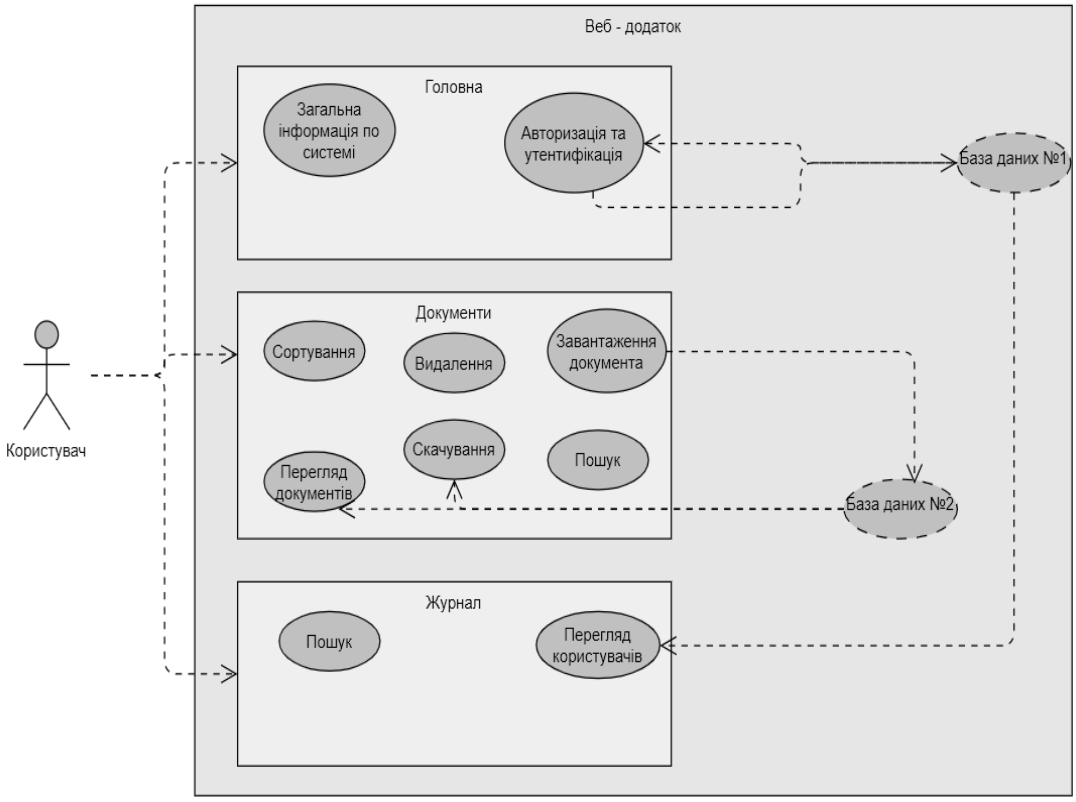

Рисунок 2.3 - UseCase діаграма

Ця UseCase діаграма відображає основні функції та взаємодію користувачів з вашим веб-додатком електронного документообігу.

2.2 WBS - модель проекту

WBS (Work Breakdown Structure) - це модель проектування, яка включає в себе декомпозицію проекту на менші, більш керовані елементи, з метою більш ефективного управління проектом. Основна ідея полягає в тому, щоб розбити весь проект на менші компоненти, які можна керувати окремо.
Наша модель відображена на рисунку 2.4.

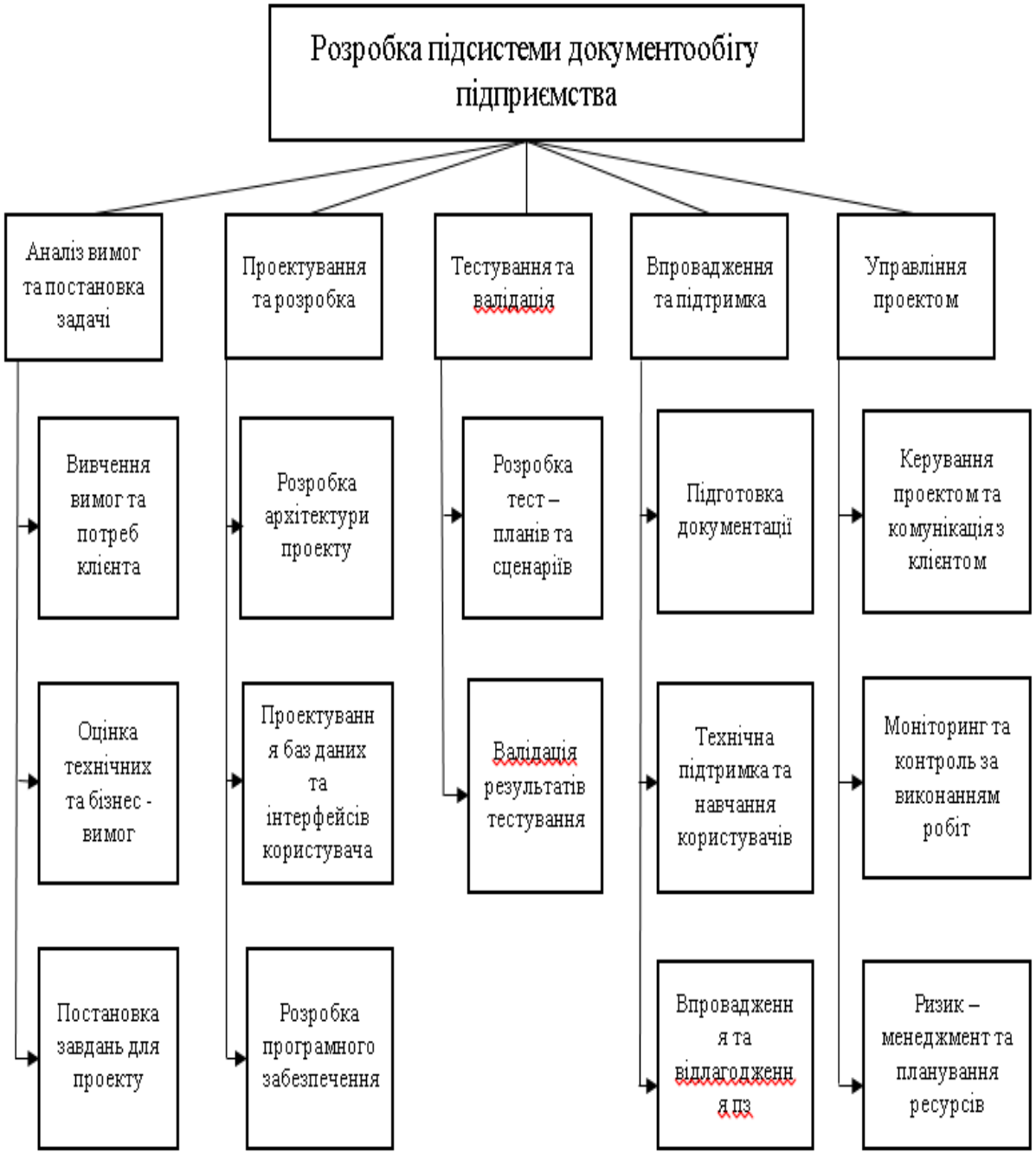

Рисунок 2.4 - WBS модель проекту

2.3 OBS - модель проекту

OBS (Organizational Breakdown Structure) - це модель, що відображає організаційну структуру проекту та розподіл функцій та відповідальності між учасниками проекту.

У контексті нашої теми, ми можемо скласти OBS наступним чином рисунок 2.5:

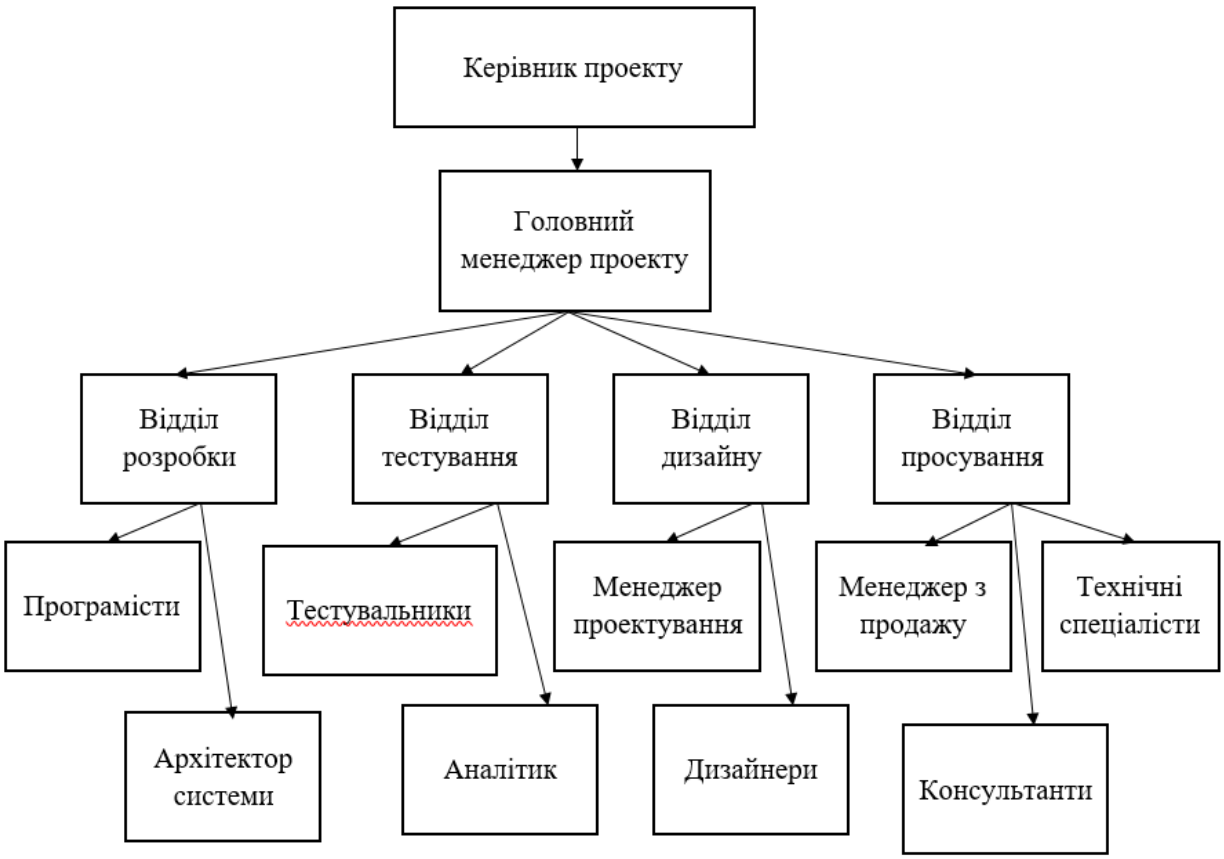

Рисунок 2.5 - OBS модель проекту

2.4 Вибір моделі життєвого циклу проекту

На даний момент у світі існує багато моделей управління проектами та різних методів їх реалізації. Дивлячись на типи методологій управління, в тому числі в сфері IT, можна побачити, що деякі є просто набором принципів, інші встановлюють структуру на основі тем або процесів. Деякі детально визначають стандарти процесу, а інші описують сам процес. Як точно визначити методологію та вибрати оптимальну для проекту?

Вибираючи методологію, слід враховувати наступні фактори:

1) Методи управління IT-проектами – традиційні, так звані каскадні

2) Складність, тобто простота або складність проекту – відноситься як до самого проекту, так і до потреб клієнта, наявних ресурсів, часу, інструментів і людей.

3) Характеристики робочого середовища – якщо воно динамічне з прагненням до еволюції та змін, гнучка методологія буде працювати; якщо робота виконується відповідно до суворих вимог, графіка та бюджету, каскадний підхід є кращим. Варто запитати себе, чи повинна ваша організація бути жорсткою чи гнучкою.

4) Цінність компанії та відносини з клієнтами – метод повинен гармоніювати зі стратегічними цілями компанії, дбаючи про найбільш значну вигоду з мінімальними втратами.

5) Організаційна цінність і потенціал команди – хороші стосунки в команді проекту та її мотивація можуть сформувати справді стійку методологію.

У нашому проекті ми будемо використовувати саме каскадну модель життєвого циклу у розробці програмного забезпечення підсистеми, оскільки каскадна модель є однією з найбільш розповсюджених та старих моделей розробки програмного забезпечення.

Основні переваги каскадної моделі життєвого циклу включають:

1. Простота та легкість у розумінні: Каскадна модель є дуже простою та легкою для розуміння, оскільки вона має прямолінійну структуру, що включає фази визначення, проектування, реалізації, тестування та випуску програмного забезпечення.

2. Визначеність: У каскадній моделі кожна фаза має чітко визначені метрики та критерії завершення, що робить процес розробки прозорим та легким у відстеженні.

3. Строгий контроль якості: У каскадній моделі процес розробки проходить через послідовні стадії, в кожній з яких проводиться тестування та верифікація програмного забезпечення, що гарантує високу якість виконаної роботи.

4. Використання забезпечує стабільність: Каскадна модель є стабільною та надійною, оскільки вона має чітко визначені стадії розробки та завершення, що забезпечує зменшення ризику помилок та недоліків у програмному забезпеченні.

5. Легко зберігати документацію: Каскадна модель є дуже організованою та структурованою, що дозволяє легко зберігати документацію, яка створюється протягом процесу розробки.

Загалом, каскадна модель життєвого циклу є дуже ефективною для простих та стабільних проектів, які мають чітко визначені вимоги.

Другою перевагою каскадної моделі є те, що вона дозволяє більш точно прогнозувати терміни завершення проекту та визначати бюджет. Це стає можливим завдяки тому, що кожен етап проекту детально планується, а також має чітко визначені критерії завершення. Тому менеджер проекту може контролювати прогрес виконання проекту та вчасно реагувати на можливі затримки.

Третьою перевагою є те, що кожен етап проекту може бути чітко розділений між командами розробників, що дозволяє підтримувати високий рівень спеціалізації та підвищувати ефективність роботи. Крім того, кожна команда може працювати над своїм етапом незалежно від інших команд, що також дозволяє підвищувати швидкість розробки.

Четвертою перевагою каскадної моделі є те, що вона забезпечує чітку документацію кожного етапу проекту. Це дозволяє підтримувати якість розробки та полегшує зміну проекту в майбутньому. Крім того, ця документація може використовуватися для навчання нових членів команди та підтримки роботи з програмним забезпеченням у майбутньому.

Більш зрозуміло показано нашу модель на рисунку 2.6.

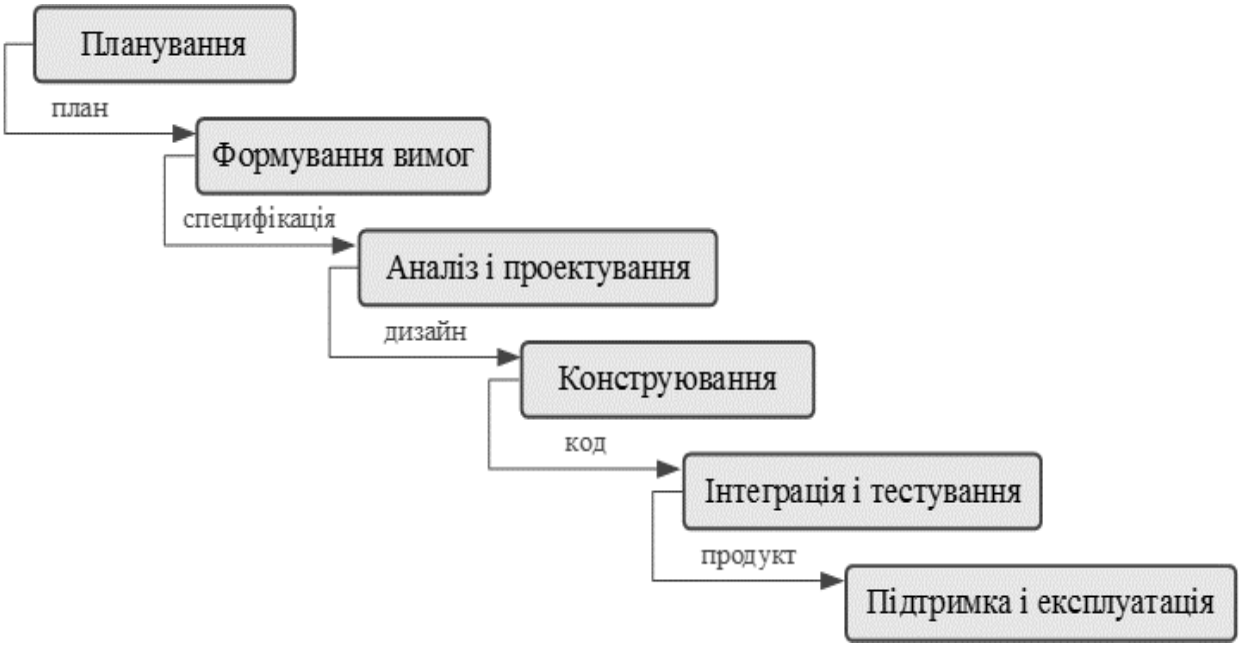

Рисунок 2.6 – Каскадна модель життєвого циклу.

2.5 Вибір та рогляд концепції підсистеми

В нашій роботі можна використовувати різні концепції, залежно від вимог та потреб користувачів. Деякі можливі концепції включають:

- Концепція документообігу в локальній мережі: програма може бути розроблена для роботи в локальній мережі, що дозволить користувачам зберігати та обробляти документи на сервері кафедри.

- Концепція гнучкого документообігу: програма може бути розроблена для підтримки різних форматів документів та забезпечення можливості зберігання документів в різних форматах.

- Концепція документообігу з підтримкою мережевої безпеки: програма може бути розроблена з урахуванням вимог щодо мережевої безпеки та забезпечення конфіденційності документів.

- Концепція інтеграції з іншими системами: програма може бути розроблена для інтеграції з іншими системами, такими як система електронного розкладу, електронна бібліотека, система управління студентами.

Програмне забезпечення для реалізації концепції документообігу повинно включати в себе можливості створення, редагування та відправки документів, створення та редагування списків користувачів та груп користувачів з правами доступу до документів, а також детальне відображення поточного стану документу та його маршрутування.

Крім того, для забезпечення швидкості роботи програми, необхідно використовувати оптимальні алгоритми обробки даних та мінімізувати час на передачу даних по мережі. Для забезпечення безпеки даних, необхідно використовувати надійні механізми автентифікації та авторизації користувачів, а також забезпечити захист даних від несанкціонованого доступу.

Не всі концепції можуть підходити для багатьох кафедр, наприклад щодо концепції документообігу в локальній мережі вона має деякі обмеження. А саме доступ до документів може бути обмежений лише для користувачів, які мають фізичний доступ до сервера. Крім того, при збільшенні кількості користувачів може виникнути проблема з розподілом прав доступу, також варто звернути увагу на питання забезпечення резервного копіювання та захисту даних. При використанні локального сервера, необхідно мати механізм резервного копіювання даних, щоб у випадку втрати даних користувачі не втратили важливі документи. Виходячи з цього розуміємо, що концепція документообігу в локальній мережі може бути ефективною для кафедри, яка має свій власний локальний сервер та невелику кількість користувачів, але потребує обмеженого доступу до документів. При реалізації такої системи необхідно враховувати питання безпеки даних, швидкості роботи та зручності використання.

Область застосування локальних мереж дуже широка, нині такі системи є практично кожному офісі (наприклад, встановлений один принтер кілька комп'ютерів, чи кілька комп'ютерів використовують одне ПО, припустимо 1С:Бухгалтерія та інших). З кожним днем потоки інформації стають більшими, використовуване програмне забезпечення складніше і

функціональніше, географія діяльності організацій розширюється. Застосування засобів ЛОМ стає не просто бажаним, а необхідним для успішної діяльності та розвитку бізнесу, науки, навчання студентів, школярів, підготовки та перепідготовки фахівців, виконання державних програм та функцій та ін.

Структура локальної мережі визначається принципом управління і типом зв'язку, найчастіше вона ґрунтується на структурі організації, що обслуговується. Використовуються види топології: шинна, кільцева, радіальна, деревоподібна. Найбільш поширені перші два види, за рахунок ефективного використання каналів зв'язку, простоти управління, гнучких можливостей розширення та зміни.

Топологія «шина» — усі комп’ютери, з’єднані в ланцюжок шляхом з’єднання з основним сегментом кабелю (ствола) із «термінаторами», розміщеними на кінцях для усунення сигналів, що проходять в обох напрямках. Комп'ютери в мережі з'єднуються коаксіальними кабелями з Т-роз'ємами. Пропускна здатність мережі становить 10 Мбіт/с, чого недостатньо для сучасних програм, які активно використовують відео та мультимедійні дані. Перевагою такої топології є низькі витрати на розводку та рівномірність розводки. Відображення такої мережі показано на рисунку 2.7.

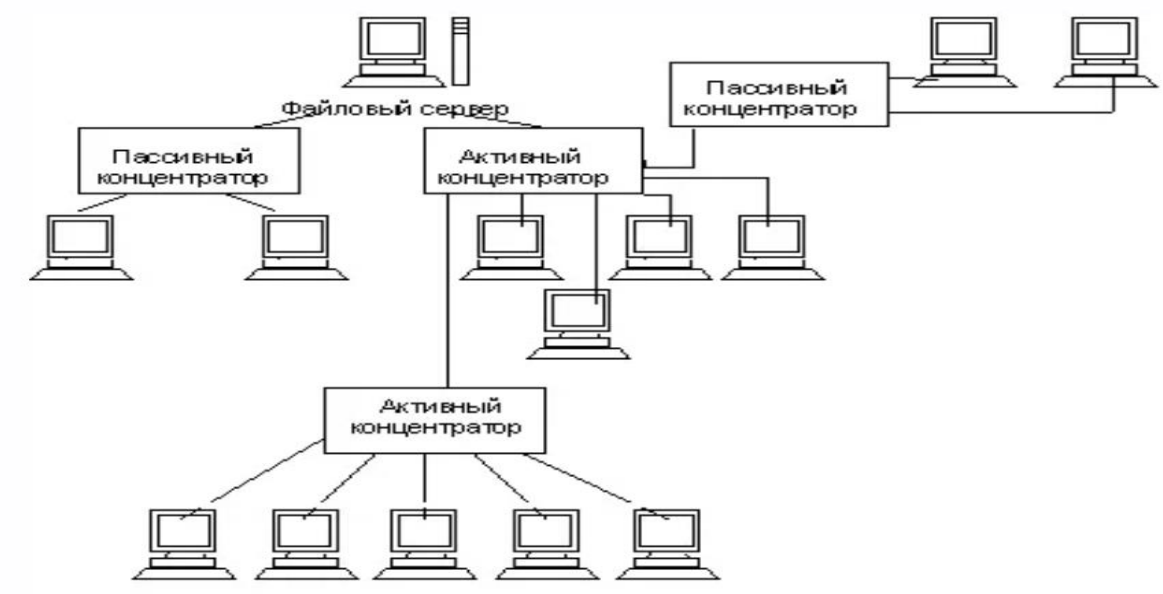

Рисунок 2.7 - Топологія "шина"

Шинна топологія – пасивна, так як через збій одного з комп’ютерів не переривається робота інших комп’ютерів і мережі в цілому. При пошкодженні шини (магістрального/основного кабелю) вся мережа стає непрацездатною. Для підключення або вимкнення системи необхідний розрив шини, що поведе за собою як наслідок зависання серверу та порушення циркулювання інформації.

2.6 Принцип документообігу на вибраному підприємству

Оскільки в нашій роботі ми взяли як підприємство вищий навчальний заклад, розглядаємо невелику ланку цього підприємства, а саме кафедру.

Розглянемо основні складові принципу електронного документообігу кафедри інформаційних технологій включають:

- Електронна пошта - використовується для передачі електронних листів з інформацією, яка стосується навчального процесу та ділових питань кафедри.
- Електронні документи - створюються та обробляються у цифровій формі. Це можуть бути електронні заявки, розклади занять, протоколи засідань, відомості про студентів тощо.
- Електронний підпис - використовується для підтвердження автентичності та цілісності електронних документів. Це забезпечує їх правову значимість та можливість використання в офіційному обігу.
- Електронний архів: це централізоване сховище для зберігання електронних документів кафедри. В архіві можуть бути створені папки або категорії, що відповідають різним типам документів, наприклад, навчальні матеріали, наукові статті, методичні рекомендації тощо. Це допомагає зручно організувати та знайти необхідні документи.
- Електронна система реєстрації документів: система реєстрації документів дозволяє зареєструвати нові документи, присвоїти їм унікальний

ідентифікатор та відстежувати їх статус. Це спрощує процес контролю та моніторингу документів на кафедрі.

Загалом, принцип електронного документообігу кафедри інформаційних технологій сприяє автоматизації бізнес-процесів та раціоналізації документообігу. Застосування цього принципу дозволяє зменшити час на обробку документів, уникнути помилок при введенні даних, покращити якість документації та забезпечити безпеку і конфіденційність інформації.

Окрім цього, електронний документообіг дає змогу забезпечити ефективну взаємодію між різними діловими партнерами та зменшити витрати на паперову документацію. Також, застосування цього принципу дозволяє підвищити рівень автоматизації та інформатизації процесів, що стає важливим у контексті сучасних вимог до якості освіти та науки.

Що стосується схеми обігу документів на кафедрі, то він представлений на рисунку 2.8 де зображено як документи передаються працівникам та за межі кафедри.

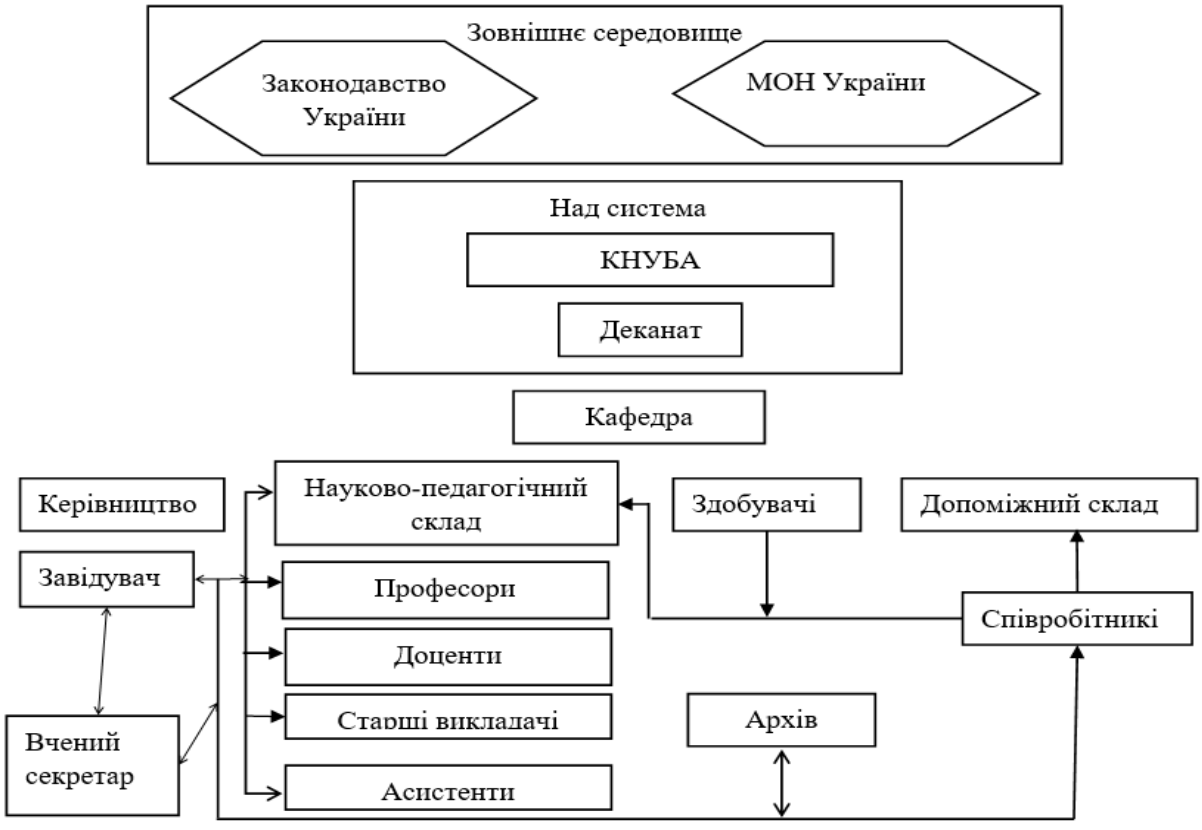

Рисунок 2.8 - Організаційно-технічна структура кафедри вищого навчального закладу

Організаційна структура визначає спосіб організації та розподілу функцій, обов'язків і повноважень всередині кафедри. Основні принципи формування організаційної структури кафедри включають:

1) Функціональна розділеність: Заснована на розподілі завдань та функцій між різними підрозділами кафедри. Кожен підрозділ спеціалізується на виконанні конкретних завдань, що допомагає ефективно організувати роботу та підвищує продуктивність.

2) Ієрархічна структура: Включає рівні керівництва та підлеглості. У кафедрі можуть бути різні рівні посад, наприклад, завідувач кафедри, професори, доценти, викладачі тощо. Це дозволяє забезпечити чітку систему керівництва та визначення повноважень.

3) Делегування повноважень: Залучення співробітників до процесу прийняття рішень та делегування повноважень сприяє розподілу відповідальності та розвитку командної роботи. Це може покращити швидкість та ефективність прийняття рішень.

Ефективна організаційна структура кафедри сприяє кращій координації роботи, зменшенню дублювання завдань та конфліктів між працівниками. Вона сприяє забезпеченню зручного спілкування та обміну інформацією, поліпшенню роботи в команді, а також сприяє досягненню загальних цілей та завдань кафедри. Крім того, чітка організаційна структура спрощує процес управління та покращує прозорість та відповідальність всередині кафедри.

Технічна інфраструктура включає комп'ютери, програмне забезпечення, мережеві з'єднання, сервери, електронні бази даних та інші технічні засоби, які підтримують організаційну діяльність кафедри. Використання таких технічних засобів має декілька переваг для поліпшення роботи кафедри, спілкування та обміну інформацією. Нижче наведені деякі висновки щодо використання цих засобів:

1) Комп'ютери: Використання комп'ютерів дозволяє виконувати різноманітні завдання, включаючи обробку даних, розробку матеріалів, ведення документації, аналіз даних та інше. Комп'ютери забезпечують

швидкий доступ до інформації і сприяють автоматизації багатьох процесів, що робить роботу кафедри більш ефективною та продуктивною.

2) Програмне забезпечення: Використання спеціалізованого програмного забезпечення, такого як системи управління навчальним процесом, електронні журнали, системи документообігу тощо, допомагає впорядковувати та автоматизувати багато рутинних задач. Це сприяє збереженню часу, зменшенню помилок та поліпшенню організації роботи.

3) Мережеві з'єднання: Мережеві з'єднання дозволяють спілкуватися та обмінюватися інформацією між різними працівниками кафедри. Це сприяє покращенню комунікації, спільній роботі над проектами, обміну дослідницькими матеріалами та координації роботи.

4) Сервери та електронні бази даних: Використання серверів та електронних баз даних дозволяє зберігати та організовувати великі обсяги інформації, такі як матеріали для навчання, наукові публікації, студентські роботи тощо. Це сприяє зручному доступу до інформації, її швидкому пошуку та сприяє збереженню та обміну знаннями всередині кафедри.

Загалом, використання технічних засобів у організаційно-технічній структурі кафедри поліпшує продуктивність, забезпечує швидку передачу інформації, полегшує спілкування та сприяє ефективній організації роботи кафедри. Враховуючи швидкий технологічний розвиток, важливо постійно оновлювати технічну інфраструктуру, впроваджувати нові технології та забезпечувати навчання співробітників для ефективного використання цих засобів.

Під час розгляду організаційно-технічної структури кафедри вищого навчального закладу було детально розглянуто різні аспекти цієї структури.

Перш за все, було пояснено, що організаційна структура кафедри включає функціональну розділеність та ієрархічну структуру, що допомагають управляти кафедрою та забезпечити координацію роботи. Делегування повноважень також є важливим аспектом, що сприяє ефективному розподілу завдань та відповідальності.

Технічна інфраструктура, включаючи комп'ютери, програмне забезпечення, мережеві з'єднання, сервери та електронні бази даних, грає значну роль у підтримці організаційної діяльності кафедри. Ці технічні засоби сприяють покращенню комунікації, обміну інформацією та співпраці між співробітниками. Використання сучасних технологій допомагає ефективніше організувати робочі процеси, сприяє швидкому доступу до необхідної інформації та полегшує ведення наукових досліджень та методичної підтримки.

2.7 Проектування баз даних

Оскільки в нашій роботі відбувається розробка підсистеми електронного документообігу методом реалізації веб – додатку, потрібно підключати бази даних для обробки та зберігання інформації, як саме працюють бази даних зображено схематично на рисунках 2.9 та 2.10.

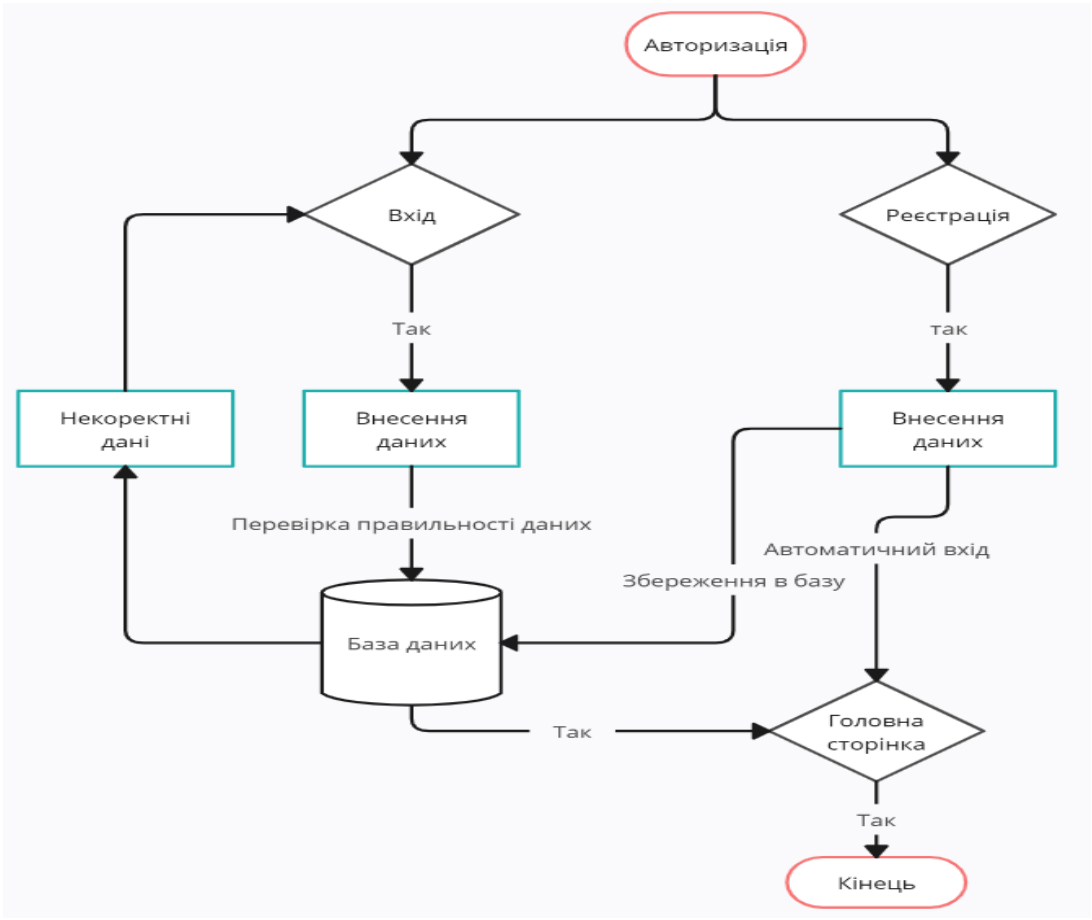

Рисунок 2.9 – робота БД із авторизацією

Як бачимо, що під час авторизації при внесені коректних даних після перевірки користувача в базі даних, відбувається вхід на головну сторінку, при некоректних даних появляється сповіщення про помилку, та повернення до процесу авторизації. Якщо відбувається процес реєстрації, то після введення даних, вони відправляються до БД, та вібувається автоматичний вхід до веб - додатку. Автоматичний вхід зроблений для більшої зручності, оскільки в такому випадку відбуваються менші затрати часу на реєстрацію, що робить підсистему більш зручною в користуванні.

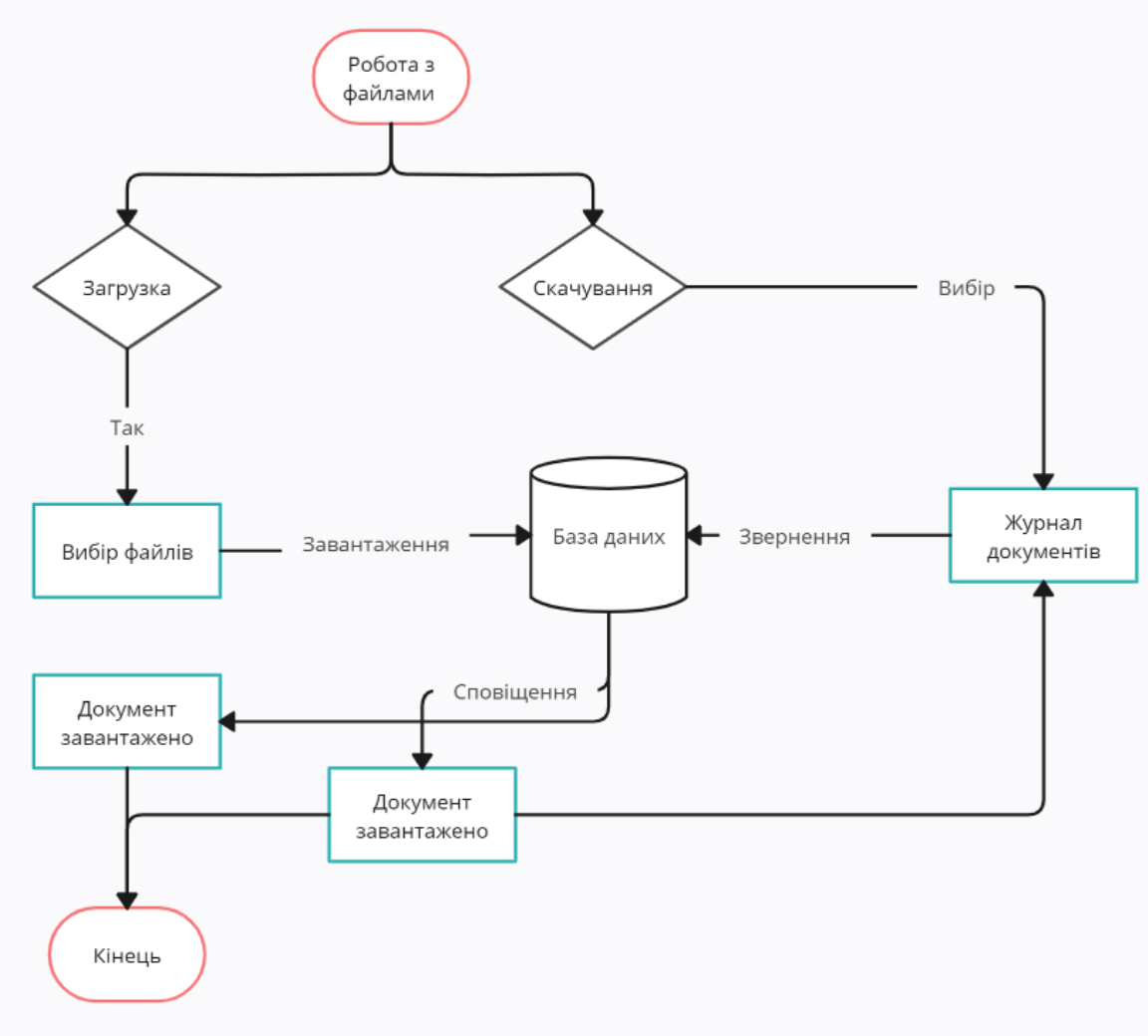

Рисунок 2.10 - робота БД з файлами

Як бачимо, що після вибору завантаження випливає вікно про вибір файлу який потрібно завантажити, в нашу підсистему, після успішного результату нашого користувача буде про це повідомлено, та дані про сам файл відправляються до журналу документів, що вже зберігаються в БД.

Якщо нам потрібно завантажити файл із підсистеми на наш пристрій, потрібно вибрати його в журналі файлу, далі йде звернення до БД та відбувається скачування файлу, після успішного результату користувача також буде про це повідомлено.

У рамках даної роботи ми плануємо використовувати NoSQL базу даних типу RESTful API, яка буде зберігати дані для авторизації користувачів. Для цього ми будемо використовувати сервіс mockAPI, який надає можливість зберігання та отримання даних за допомогою HTTP-запитів до визначених ендпоінтів. Такий тип бази даних є зручним і гнучким для розробки веб-додатків, оскільки він дозволяє швидко встановлювати та налаштовувати взаємодію з даними, не вимагаючи складної інфраструктури та налаштування. З використанням RESTful API, ми зможемо ефективно обмінюватись даними між нашим веб-додатком електронного документообігу та базою даних, що забезпечить нам потрібну функціональність та забезпечить зручну роботу з даними.

База даних типу RESTful API, як у випадку сервісу mockAPI, має свої переваги та недоліки:

Переваги:

1. Гнучкість: NoSQL бази даних забезпечують гнучкість у структурі та схемі даних. Вони дозволяють зберігати різноманітні типи даних, іноді навіть без жорсткої схеми, що дозволяє легко адаптуватися до змін вимог та розширювати функціональність додатку.

2. Простота розробки та використання: RESTful API дозволяє легко взаємодіяти з базою даних за допомогою HTTP-запитів. Це зменшує складність розробки та дозволяє швидко виконувати операції з даними.

3. Масштабованість: NoSQL бази даних можуть легко масштабуватися горизонтально, дозволяючи збільшувати обсяг даних та оброблювати велику кількість запитів.

Недоліки:

1. Відсутність складних операцій: У порівнянні з реляційними базами даних, NoSQL бази даних можуть бути обмеженими у функціональності, такі як складні операції зв'язків між даними або транзакції.

2. Відсутність стандартизації: Оскільки NoSQL бази даних мають різні реалізації, вони можуть мати відмінність у синтаксисі та можливостях. Це може призвести до викликів при переході від одного сервісу до іншого.

3. Відсутність SQL: Багато розробників вже знайомі з SQL для роботи з даними. Використання NoSQL бази даних може вимагати вивчення нових запитів та підходів.

Також нижче зображено концептуальну та логічну модель БД нашої підсистеми, та загальна структура програмного продукту на рисунках 2.11 - 2.13.

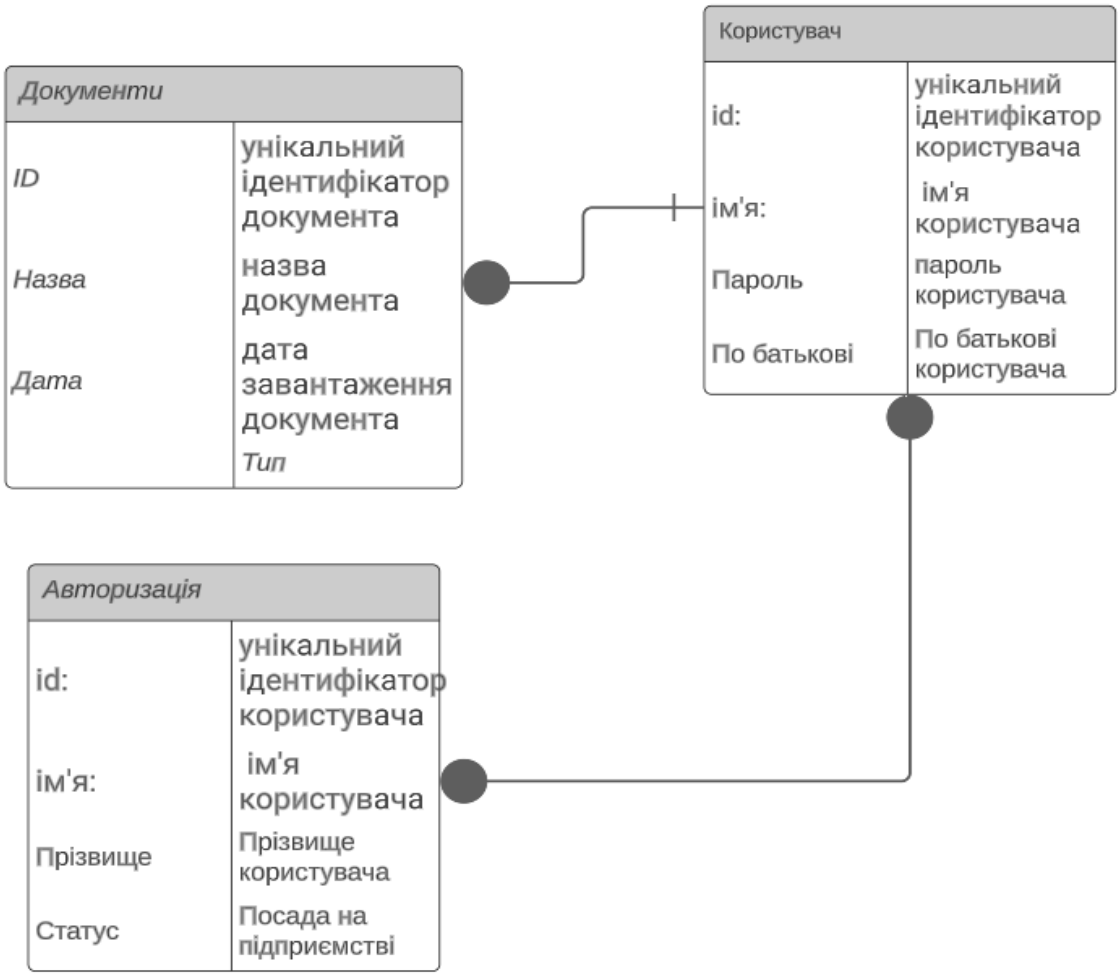

Рисунок 2.11 – концептуальна модель

Рисунок 2.12 - логічна модель

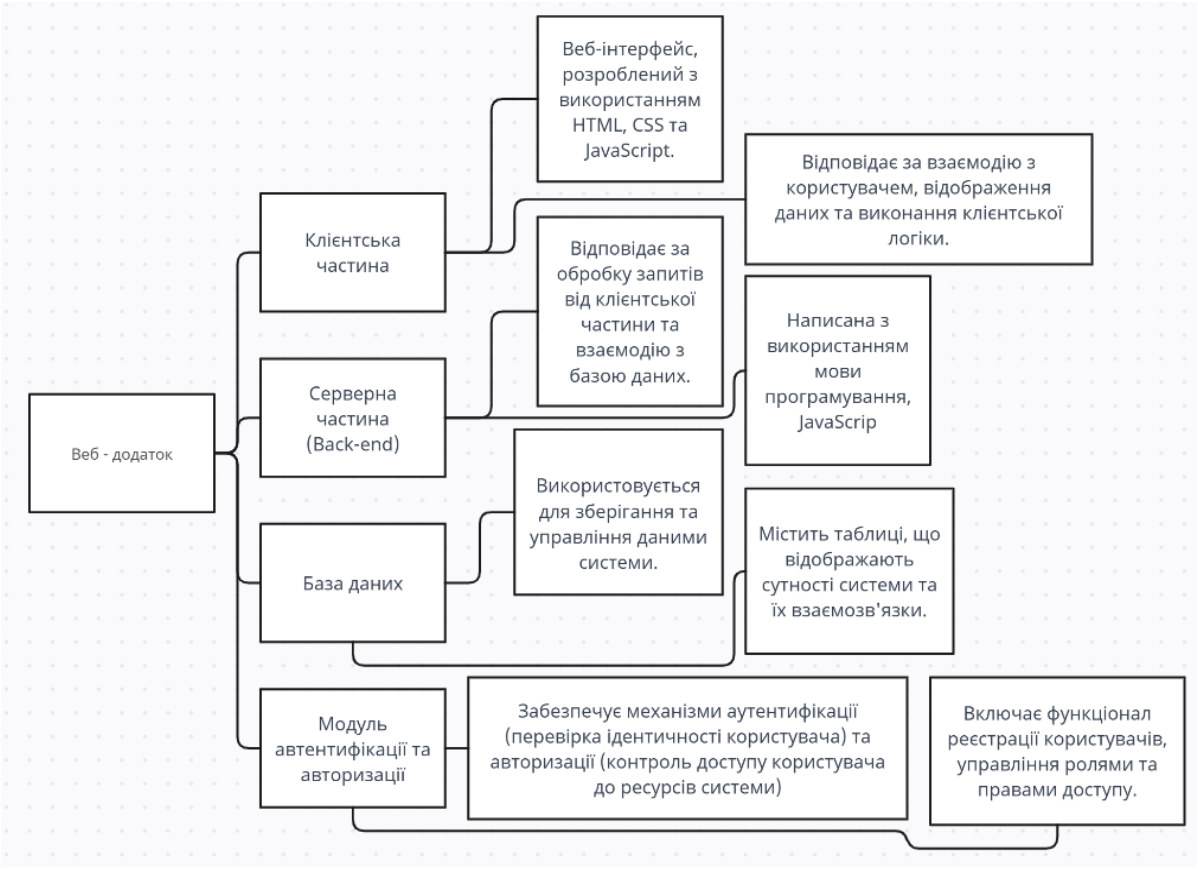

Рисунок 2.13 - загальна структура програмного продукту

ПРОЕКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Вибір мови програмування

Для початку потрібно вибрати мову програмування для реалізації нашої підсистеми у вигляді веб - додатку, як і з будь-якою мовою, важливо ретельно проаналізувати вимоги до проекту, переваги та недоліки кожної мови програмування і прийняти рішення на основі цієї інформації.

Jan 2023	Jan 2022	Change	Programming Language	Ratings	Change
1	1		Python	16.36%	+2.78%
2	2		C	16.26%	+3.82%
3	4	↑	C++	12.91%	+4.62%
4	3	↓	Java	12.21%	+1.55%
5	5		C#	5.73%	+0.05%
6	6		Visual Basic	4.64%	-0.10%
7	7		JavaScript	2.87%	+0.78%
8	9	↑	SQL	2.50%	+0.70%
9	8	↓	Assembly language	1.60%	-0.25%
10	11	↑	PHP	1.39%	-0.00%
11	10	↓	Swift	1.20%	-0.21%
12	13	↑	Go	1.14%	+0.10%
13	12	↓	R	1.04%	-0.21%
14	15	↑	Classic Visual Basic	0.98%	+0.01%
15	16	↑	MATLAB	0.91%	-0.05%
16	18	↑	Ruby	0.80%	-0.08%
17	14	↓	Delphi/Object Pascal	0.73%	-0.27%
18	26	⇈	Rust	0.61%	+0.11%
19	20	↑	Perl	0.59%	-0.12%
20	23	↑	Scratch	0.58%	-0.01%

Рисунок 3.1 – 20 найпопулярніших мов програмування на січень 2023.

У будь-якому випадку, важливо ретельно проаналізувати вимоги до проекту і визначити, яка мова програмування найбільш підходить для розробки програми електронного документообігу.

Потрібно враховувати в нашій роботі, що при розробці програмного забезпечення для роботи з базами даних існує кілька мов програмування, які мають добру підтримку та зручні інструменти для роботи з базами даних. Ось декілька мов програмування, які вважаються популярними та зручними для цього:

1. SQL: SQL (Structured Query Language) - це спеціалізована мова запитів для роботи з базами даних. Вона призначена для створення, модифікації та керування базами даних. SQL є стандартом для роботи з реляційними базами даних і має широку підтримку.

2. Python: Python є популярною мовою програмування для роботи з базами даних. Вона має багато бібліотек, таких як SQLAlchemy та Django ORM, які спрощують взаємодію з базами даних. Python також має розширення для роботи з реляційними базами даних, такими як MySQL та PostgreSQL, а також для нереляційних баз даних, таких як MongoDB.

3. Java: Java є мовою програмування, яка має потужну підтримку для роботи з базами даних. Вона має вбудовану підтримку для JDBC (Java Database Connectivity), що дозволяє взаємодіяти з різними типами баз даних. Крім того, Java має популярні фреймворки, такі як Hibernate та Spring Data, які спрощують роботу з базами даних.

4. PHP: PHP є мовою програмування, яка широко використовується для веб-розробки і має вбудовану підтримку для багатьох баз даних, зокрема MySQL. PHP має простий синтаксис для взаємодії з базами даних та розширення, такі як PDO (PHP Data Objects), які допомагають забезпечити безпеку та ефективну роботу з базами даних.

Це лише деякі з популярних мов програмування для роботи з базами даних, і вибір конкретної мови залежатиме від вашої задачі, особистих уподобань та попереднього досвіду. Важливо також враховувати підтримку бази даних, з якою ми плануємо працювати, і наявність відповідних драйверів та бібліотек для роботи з нею. Оскільки підсистема буде веб –

додатком, потрібно розібрати, що таке веб - додаток, для більшого розуміння які мови програмування нам використовувати.

Веб-додаток – це програма або ж програмне забезпечення, яку можливо відкрити в будь-якому браузері. Зовнішній інтерфейс веб-додатку розроблено з використанням таких мов програмування як CSS, HTML, JavaScript, які підтримуються всіма браузерами: Chrome, Mozilla, Opera). Для написання серверної частини (back-end) можна використовувати інші мови: Python, PhP, Ruby, Java.

Основні переваги веб-додатку:

- Використовуються на будь-якій операційній системі (Mac, Windows, Linux), оскільки ці системи підтримують сучасні браузери.
- Додаток легше програмується оскільки не залучається багато компонентів ПК такі як ядро, процесор, відеокарта.
- Легша підтримка завдяки використанню веб-додатком одного й того самого коду як настільні програми.
- Не вимогливість у схваленні платформи для публікації своїх програм, на відміну від мобільного додатку.
- Більш економічні для бізнесу, так як не вимагають реєстрації та купівлі ліцензії, але використовуються як служба SaaS. До того ж набагато дешевші.

Аналізуючи цю інформацію, при розробці веб-додатку для електронного документообігу на нашому підприємстві було вирішено використовувати комбінацію мов програмування JavaScript (JS), Cascading Style Sheets (CSS) та HyperText Markup Language (HTML), а також бібліотеку React. Кожна з цих мов та інструментів має свої переваги та недоліки, які розглянемо нижче:

1. JavaScript (JS): Переваги:

- Широке поширення та підтримка: JS є однією з найпопулярніших мов програмування для розробки веб-додатків. Вона має велику спільноту розробників, що забезпечує багато ресурсів, бібліотек та фреймворків для спрощення розробки.

- Клієнтська та серверна розробка: JS може використовуватись як на клієнтській стороні (браузер), так і на серверній стороні (Node.js). Це дозволяє забезпечити повну стекову розробку веб-додатку.
- Динамічність та інтерактивність: JS дозволяє динамічно змінювати сторінки, обробляти події користувача та забезпечувати інтерактивність веб-додатку.

Недоліки:

- Швидкодія: у порівнянні з іншими мовами програмування, JS може бути менш швидкодійним, особливо при обробці великих об'ємів даних. Однак, це питання можна вирішити за допомогою оптимізації та використання спеціалізованих бібліотек.

2. CSS: Переваги:

- Розділення стилів та зовнішнього вигляду: CSS дозволяє відокремити оформлення веб-сторінок від їхнього вмісту. Це спрощує редагування та підтримку стилів та зовнішнього вигляду.
- Розширені можливості оформлення: CSS надає широкий набір можливостей для керування виглядом веб-елементів, включаючи розташування, кольори, шрифти, анімацію тощо.
- Підтримка респонсивного дизайну: CSS дозволяє створювати адаптивні веб-сторінки, що підлаштовуються під різні розміри екранів.

Недоліки:

- Складність: при розробці складних макетів, CSS може стати складним для управління та підтримки. Це може потребувати додаткового зусилля для організації та структурування стилів.

3. HTML: Переваги:

- Семантична розмітка: HTML дозволяє визначати семантику веб-сторінки, що полегшує сприйняття та інтерпретацію вмісту пошуковими системами та розробниками.

- Легкість вивчення: HTML має простий синтаксис та логічну структуру, що дозволяє швидко вивчити його основи та почати розробку веб-додатків.

Недоліки:

- Обмеженість функціональності: HTML забезпечує основну структуру та розмітку веб-сторінки, але має обмежену можливість взаємодії з користувачем та обробки подій. Для цих завдань зазвичай використовуються JavaScript та інші технології.

4. Бібліотека React: Переваги:

- Компонентний підхід: React дозволяє розбити веб-додаток на незалежні компоненти, що полегшує управління станом та перевикористання коду.
- Віртуальний DOM: React використовує віртуальний DOM для ефективного оновлення та маніпуляції елементами веб-сторінки.
- Широкі можливості: React має багато розширень та плагінів, що дозволяють розширити його функціональність та прискорити розробку.

Недоліки:

- Навчання: Використання React може вимагати додаткового часу та зусиль для вивчення його концепцій та парадигм.
- Великий розмір: Використання React та його бібліотек може збільшити розмір веб-додатку, що може вплинути на швидкодію завантаження сторінок.

Враховуючи плюси та мінуси кожної мови та інструменту, вибір комбінації JavaScript, CSS, HTML та React для розробки веб-додатку електронного документообігу є найбільш оптимальним рішенням. Використання JavaScript та React дозволить створити інтерактивний та динамічний додаток, CSS надасть можливість керувати оформленням, а HTML забезпечить структуру сторінок. Такий вибір забезпечить зручну та ефективну розробку веб - додатку для електронного документообігу.

3.2 Вибір платформи розробки

Існує безліч середовищ розробки, які можна використовувати для розробки програмного забезпечення, тому потрібно розглянути декілька середовищ розробки, враховуючи їх переваги та недоліки, для оптимального рішення при виконані роботи. Декілька популярних середовищ розробки включають:

1) intelliJ IDEA: Це одне з найпопулярніших інтегрованих середовищ розробки для мови Java і багатьох інших мов програмування, таких як Kotlin, Scala, JavaScript та інші. IntelliJ IDEA надає широкий набір інструментів, що полегшують розробку програмного забезпечення, включаючи підказки коду, відлагодження, автоматичне форматування, підтримку систем контролю версій та багато іншого. Воно також має потужні можливості для рефакторингу коду та автоматичної перевірки якості коду.

Переваги:

- Має потужні функції підказок коду та автодоповнення.
- Забезпечує швидке та ефективне відлагодження коду.
- Має багато розширень та плагінів для розширення функціональності.

Недоліки:

- Вимагає великого обсягу ресурсів комп'ютера та пам'яті.
- Платна версія має обмеження на використання для комерційних проектів.

2) Eclipse: Це інше популярне середовище розробки, яке в основному спеціалізується на розробці на мові Java, але також підтримує розробку на інших мовах програмування. Воно надає широкий набір інструментів для розробки, включаючи редактор коду, відлагоджувач, систему управління проектами та плагіни для розширення його можливостей. Eclipse також відомий своєю гнучкістю та можливістю налаштування під потреби розробника.

Переваги:

- Має широкий вибір плагінів та розширень.
- Може працювати з багатьма мовами програмування.
- Має потужну систему управління проектами та залучення до спільної роботи.

Недоліки:

- Може бути складним для налаштування та використання для початківців.
- Інтерфейс користувача може виглядати застарілим порівняно з іншими середовищами.

3) Visual Studio: Це інтегроване середовище розробки, розроблене компанією Microsoft, що спеціалізується на розробці програмного забезпечення для платформи Windows. Visual Studio підтримує різні мови програмування, такі як C#, VB.NET, JavaScript, TypeScript та багато інших. Воно має потужні інструменти для розробки, включаючи відлагоджувач, підказки коду, систему контролю версій та багато інших функцій. Visual Studio також надає інтегроване середовище для розробки додатків для платформи .NET.

Переваги:

- Має потужну підтримку мови C# та інших мов програмування Microsoft.
- Надає широкі можливості для розробки на платформі .NET.
- Інтегроване середовище для розробки додатків для платформи Windows.

Недоліки:

- Обмежена підтримка для розробки на інших платформах, окрім Windows.
- Вимагає багато ресурсів системи та пам'яті.

4) PyCharm: Це інтегроване середовище розробки, спеціалізоване на розробці програмного забезпечення на мові Python. PyCharm надає потужні

інструменти для розробки, включаючи підказки коду, автоматичне завершення, відлагодження, систему контролю версій та інші. Воно також підтримує розширення через плагіни і має спеціальні можливості для розробки веб-додатків та наукових проектів на Python.

Переваги:

- Спеціалізоване середовище для розробки на мові Python.
- Має потужні функції аналізу коду та відлагодження.
- Підтримує веб-розробку та наукові проекти на Python.

Недоліки:

- Платна версія має обмеження на використання для комерційних проектів.
- Може бути важким для початківців через велику кількість функцій та налаштувань.

5) Xcode: Це середовище розробки, розроблене компанією Apple, яке спеціалізується на розробці програмного забезпечення для платформи iOS та macOS. Xcode надає інструменти для розробки мобільних додатків, включаючи відлагоджувач, візуальний редактор інтерфейсу користувача та інші. Воно також підтримує розробку на мові Swift та Objective-C.

Переваги:

- Спеціалізоване середовище розробки для платформи iOS та macOS.
- Інтегровані інструменти для розробки мобільних додатків.
- Підтримка мов Swift та Objective-C.

Недоліки:

- Обмежена підтримка для розробки на інших платформах.
- Може бути вимогливим до ресурсів комп'ютера та пам'яті.

Це лише декілька прикладів середовищ розробки, які можуть використовуватися в залежності від мови програмування та типу проекту. Кожне з цих середовищ має свої особливості та переваги, і вибір залежить від наших потреб та особистих вподобань.

Після ретельного розгляду різних середовищ розробки, ми прийшли до висновку, що для нашої дипломної роботи найкращим вибором є середовище PyCharm. Враховуючи особливості нашого проекту, який пов'язаний з розробкою підсистеми електронного документообігу обираючи PyCharm для розробки, ми можемо бути впевнені, що отримаємо потужне та зручне середовище, яке допоможе нам ефективно розробляти підсистему з використанням мови програмування Python.

3.3 Визначення технічних вимог та архітектура продукту

Для повного функціонування програми електронного документообігу для кафедри необхідно мати комп'ютер з досить потужними характеристиками та операційною системою, яка підтримує встановлення .NET Framework, який необхідний для роботи програми. Крім того, програма може вимагати наявності деяких додаткових програм та бібліотек для роботи з базою даних. Основні вимоги до ПК користувача:

- Операційна система: Windows10 або вище
- Процесор: Intel Core i5 або вище
- Оперативна пам'ять: 8 ГБ або більше
- Відеокарта: не менше 1 ГБ відеопам'яті
- Монітор: роздільна здатність 1280x1024 пікселів або більше
- .NET Framework версії 4.7.2 або вище
- SQL Server 2016 або вище або інша Система керування базами даних (СКБД), яка підтримує LINQ.

Також розуміючи те, що програма має працювати з конфіденційною інформацією, рекомендується забезпечити достатній рівень захисту інформації на комп'ютері та в мережі, щоб уникнути можливих загроз безпеці даних.

Потрібно також розуміти архітектуру нашого веб-додатку, його опис структури, функцій та їх реалізацію.

1. Загальна архітектура:

- Клієнт-серверна архітектура: Наш веб-додаток базується на моделі клієнт-сервер, де клієнти (користувачі) здійснюють взаємодію з сервером (веб-додаток) для обробки запитів та отримання відповідей.
- Багатошарова архітектура: Ми використовуємо багатошарову архітектуру, де різні компоненти додатку (презентаційний шар, логічний шар, шар доступу до даних) функціонують незалежно один від одного, що сприяє модульності та розширюваності системи.

2. Файлова структура проекту:

- Папка "src": В цій папці містяться всі вихідні файли нашого веб-додатку.
- Папка "components": Тут розміщені компоненти, які використовуються для будівництва інтерфейсу користувача, такі як кнопки, форми, таблиці тощо.
- Папка "pages": Вона містить окремі сторінки нашого додатку, які об'єднані відповідно до їхньої функціональності.
- Папка "services": Тут знаходяться сервіси, які забезпечують взаємодію з сервером, виконують запити та обробляють дані.
- Папка "utils": Вона містить допоміжні утиліти, які використовуються для загальних функцій, таких як робота з датами, рядками тощо.

3. Структура функцій та їх реалізація:

- Аутентифікація та авторизація: Наш додаток надає можливість користувачам авторизуватися і отримувати доступ до відповідних функціональних можливостей. Для цього ми реалізуємо механізми аутентифікації, які дозволяють користувачам входити в систему зі своїми обліковими записами і перевіряти їхні права доступу.
- Управління документами: Ми реалізуємо функціональність для створення, перегляду, редагування та видалення документів. Користувачі зможуть завантажувати файли, шукати та фільтрувати документи згідно їхніх потреб.
- Повідомлення та сповіщення: Ми реалізуємо систему сповіщень, яка надсилатиме користувачам повідомлення про дії які відбуваються.

- Звіти та аналітика: Ми надамо користувачам можливість генерувати звіти та аналітичні дані з документообігу для підтримки прийняття рішень.

Кожна функція має відповідний модуль або компонент, який відповідає за її реалізацію. Ми використовуємо мови програмування JavaScript, HTML та CSS для створення інтерфейсу та реалізації функцій веб-додатку.

Аналіз загальної архітектури та структури функцій дає нам чітке розуміння структури нашого програмного продукту та покладає основу для подальшої розробки та реалізації функціональності веб-додатку.

3.4 Проектування дизайну підсистеми

Основним завданням дизайну є забезпечення зручності використання системи користувачами, що зменшить кількість помилок при взаємодії з системою та збільшить продуктивність роботи.

Основні етапи розробки дизайну підсистеми:

- Аналіз вимог користувачів. Дизайн повинен відповідати потребам користувачів, тому на першому етапі необхідно визначити їх вимоги та потреби.

- Розробка концепції дизайну. На основі вимог користувачів можна розробити концепцію дизайну, яка буде відповідати їхнім потребам. На цьому етапі необхідно вирішити питання щодо кольорової гами, шрифтів, розмірів та форматів елементів інтерфейсу.

- Розробка макетів. На основі концепції дизайну можна створити макети інтерфейсу, які будуть використовуватися для розробки функціональності підсистеми. Макети повинні відображати всі елементи інтерфейсу та їх розташування.

- Тестування та вдосконалення дизайну. Після створення макетів необхідно провести тестування їх з користувачами, щоб виявити можливі недоліки та проблеми з використанням. На основі результатів тестування можна вдосконалити дизайн та внести необхідні зміни.

Отже, розробка дизайну підсистеми є важливим етапом в процесі розробки програмного забезпечення. Його успішне виконання забезпечує зручність та ефективність використання системи користувачами. Крім того, важливо пам'ятати про те, що дизайн підсистеми має бути логічним та простим у використанні. Користувачі повинні з легкістю знаходити необхідні елементи інтерфейсу та здійснювати необхідні дії.

У процесі розробки дизайну зручний дизайн підсистеми забезпечує швидке та ефективне використання системи користувачами, що позитивно впливає на роботу кафедри в цілому.

В нашому проекті ми будемо використовувати мови CSS та HTML, оскільки CSS відповідає за оформлення та вигляд веб-сторінок, анімації та розміщення елементів та інші аспекти візуального представлення, а HTML буде використовуватись для створення структури сторінки, включаючи заголовки, текстові блоки, посилання, форми та інші елементи, що складають веб-сторінку. Буде зроблено всього три сторінки, а саме сторінки – “Мої документи”, “Документи” та “Журнал”, рисунок 3.4.

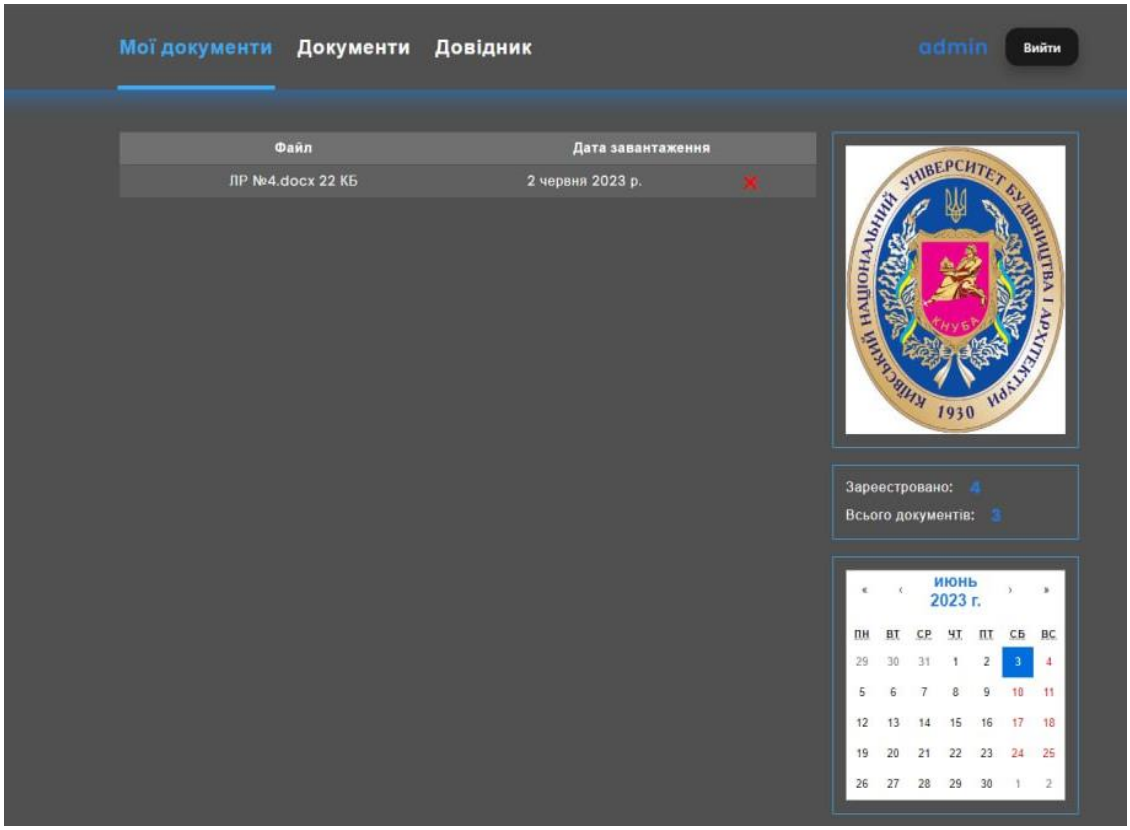

Рисунок 3.4 - дизайн основної сторінки.

Наш дизайн має сірий колір із логотипом вибраного ВНЗ. Продемонстровано як саме розміщені та виглядають наші сторінки. Дизайн є досить простим для користувачів та інтуітивно зрозумілим для зручного користування нашим веб–додатком.

4. ПРАКТИЧНА РЕАЛІЗАЦІЯ

У даному розділі буде розглянуто більш детально розробку інтерфейсної частини веб-додатку нашої підсистеми та алгоритми, що були використані. Оскільки ця частина є невід’ємною складовою у розробці для зручного користування.

4.1 Зручний інтерфейс

При розробці нашого веб-додатку електронного документообігу було створено зручний та зрозумілий інтерфейс, який складається з трьох основних сторінок. Кожна сторінка має свою унікальну функціональність і призначення, що дозволяє користувачам легко взаємодіяти з системою.

Мої документи: Ця сторінка є вихідною точкою для користувачів. На ній вони зустрічаються зі зручним інтерфейсом, який надає швидкий доступ до основних функцій системи. На цій сторінці розміщені такі блоки: список усіх документів які завантажив даний користувач, функції видалення файлів, інформування користувача по його завантаженням та інформування загальною статистикою по підсистемі.

Сторінка документів: Ця сторінка присвячена управлінню документами. Користувачі можуть переглядати, завантажувати, сортувати та шукати документи, з використанням зручних інтерфейсних елементів. Тут можна також дізнатись хто саме завантажив документ, для того щоб знати у кого можна задати питання в разі їх виникнення.

Сторінка довідника: Ця сторінка дозволяє користувачам знайти потрібного їм користувача по системі та мати змогу зв’язатися з ним. Також для більшої зручності присутній пошук по статусу на підприємстві або по П.І.Б.

Кожна сторінка має інтуїтивний дизайн, простий та зрозумілий для користувачів інтерфейс, який спрощує їхню роботу з додатком. Таким чином,

наші користувачі можуть ефективно працювати з додатком і досягати своїх цілей з мінімальними зусиллями.

4.2 Авторизація користувачів

Вище в проекті у нас є відображена структура роботи вікна авторизації з роботою баз даних, у 2 розділі 7 пункт про проектування баз даних.

Як саме виглядає процес авторизації та реєстрації зображено на рисунку 4.1 – 4.2.

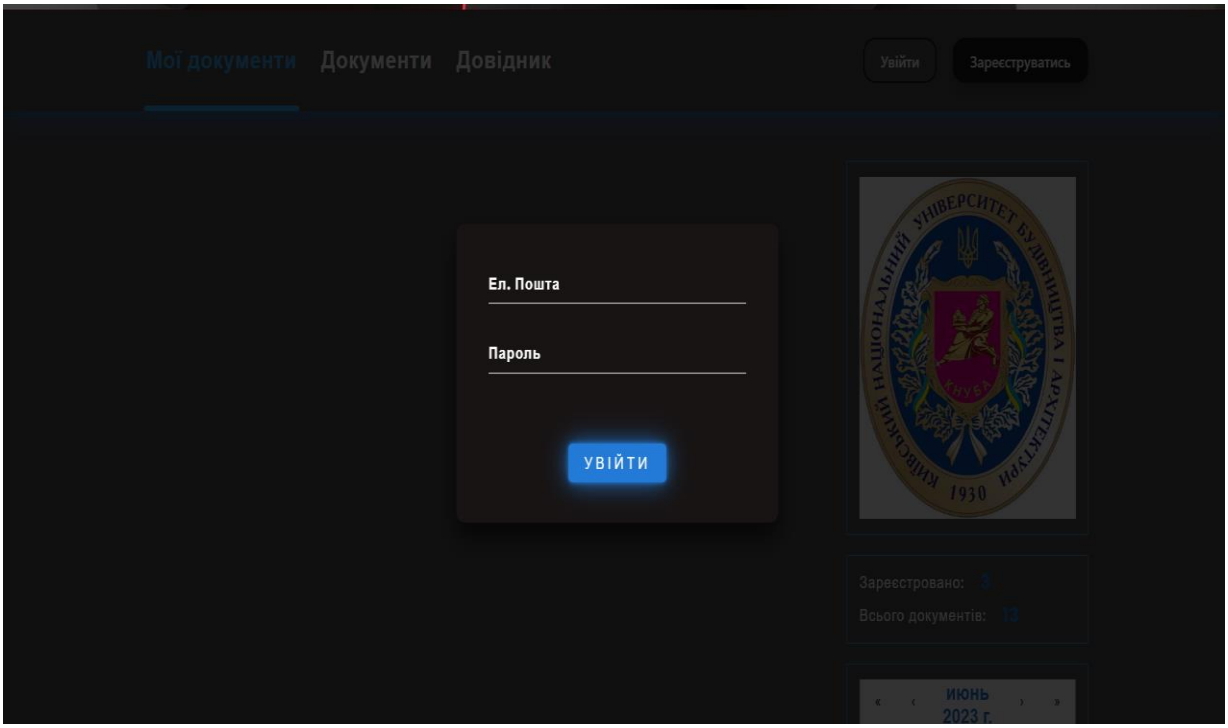

Рисунок 4.1 - процес входження у веб - додаток

Користувачу потрібно ввести свою електронну пошту та пароль, після чого його направить на сторінку “ Мої документи ” та появиться сповіщення про успішний вхід, якщо данні не вірні то помилку.

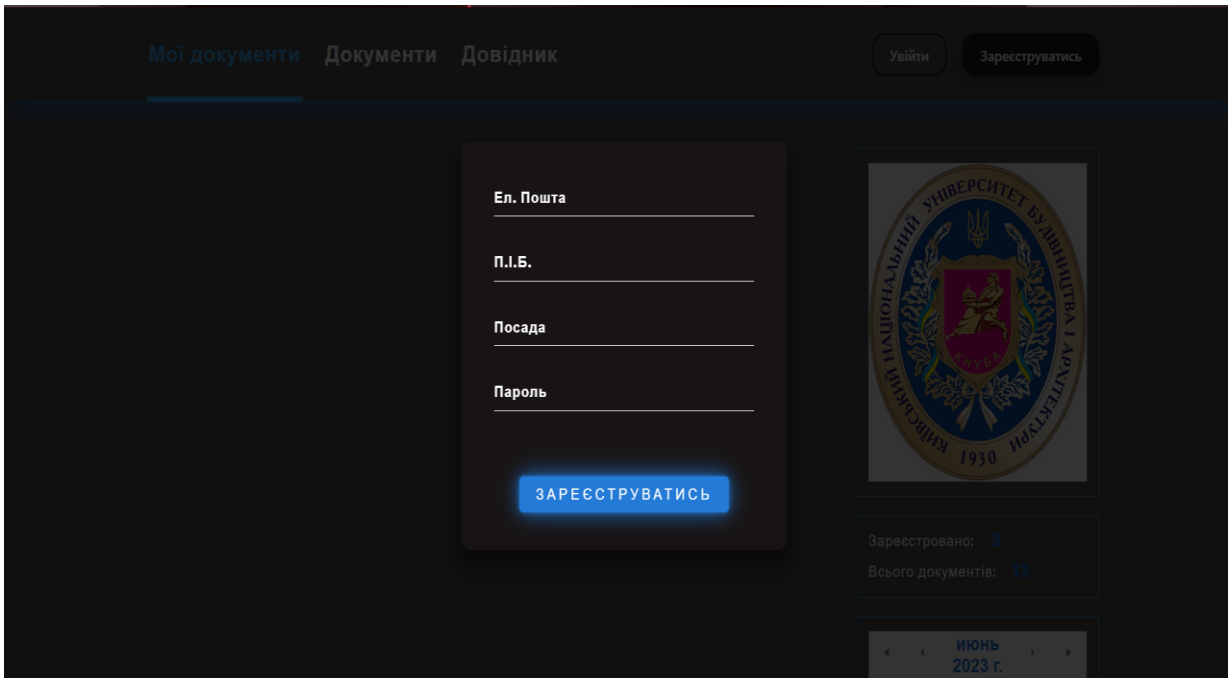

Рисунок 4.2 – процес реєстрації

При реєстрації користувачу потрібно ввести електронну пошту яка буде потім відображатись у довіднику користувачів, для змоги з ними звязатись, та вказати посаду на підприємстві.

Як бачимо дизайн досить простий, що робить його зручним у використанні.

4.3 Тип пошуку

У нашому проекті присутня функція пошуку на двух сторінках, на кожній з них є три варіанти пошуку, як саме вони розміщенні та по яким критеріям відбувається пошук відображено на рисунку 4.3 та 4.4.

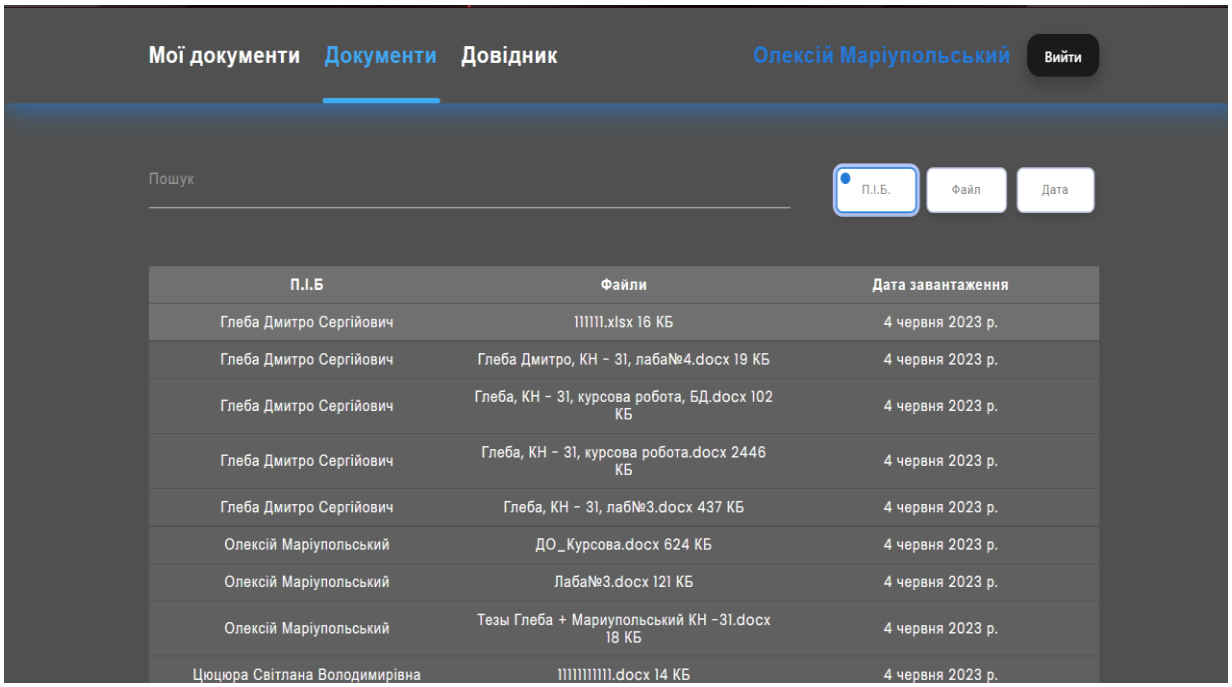

Рисунок 4.3 – пошук на сторінці “ Документи ”

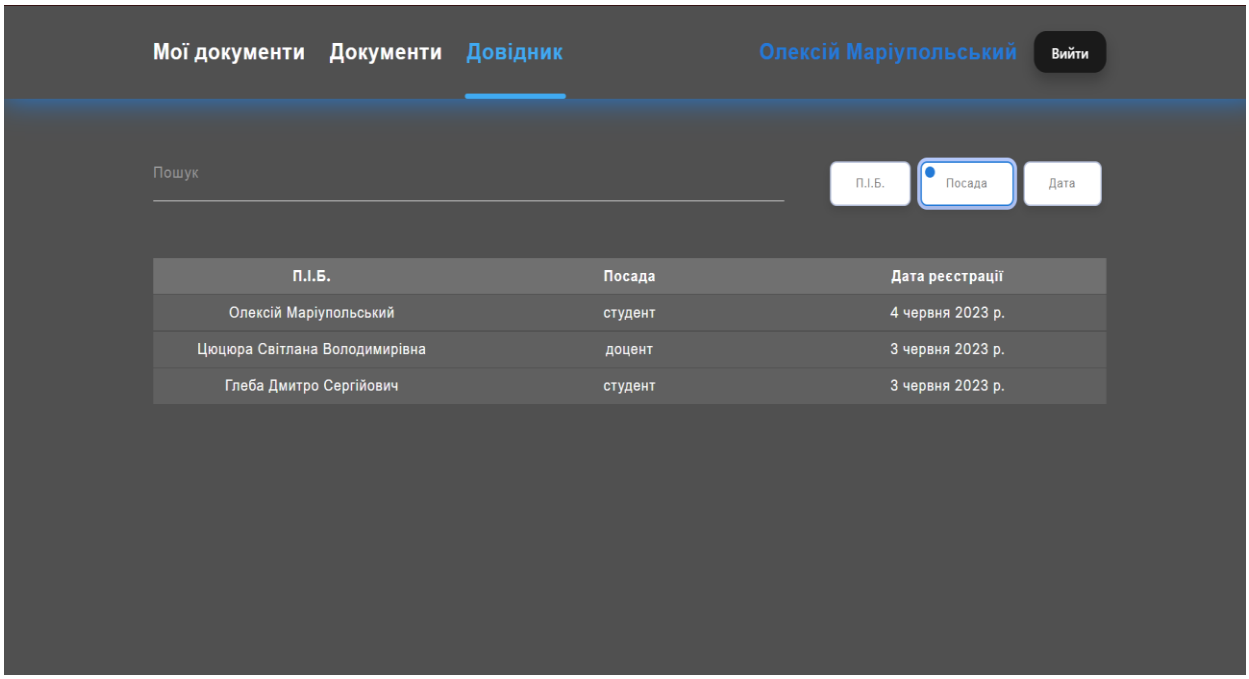

Рисунок 4.4 - пошук на сторінці “ Довідник ”

Розуміємо, що пошук відбувається за трьома критеріями, після вибору критерія потрібно ввести дані в поле пошуку, все відбувається автоматично і під час пошуку одразу відбувається видача даних.

4.4 Зручне завантаження

Завантаження у нас відбувається тільки на сторінці з нашими документами, а вже від нашої сторінки відправляється до списку усіх документів, при завантажені у нас відкривається вікно з доступом усіх файлів на пристрої користувача, після вибору файла та підтвердження завантаження у нас вспливає вікно про успішність завантаження. Як це реалізовано зображено на наступних рисунках:

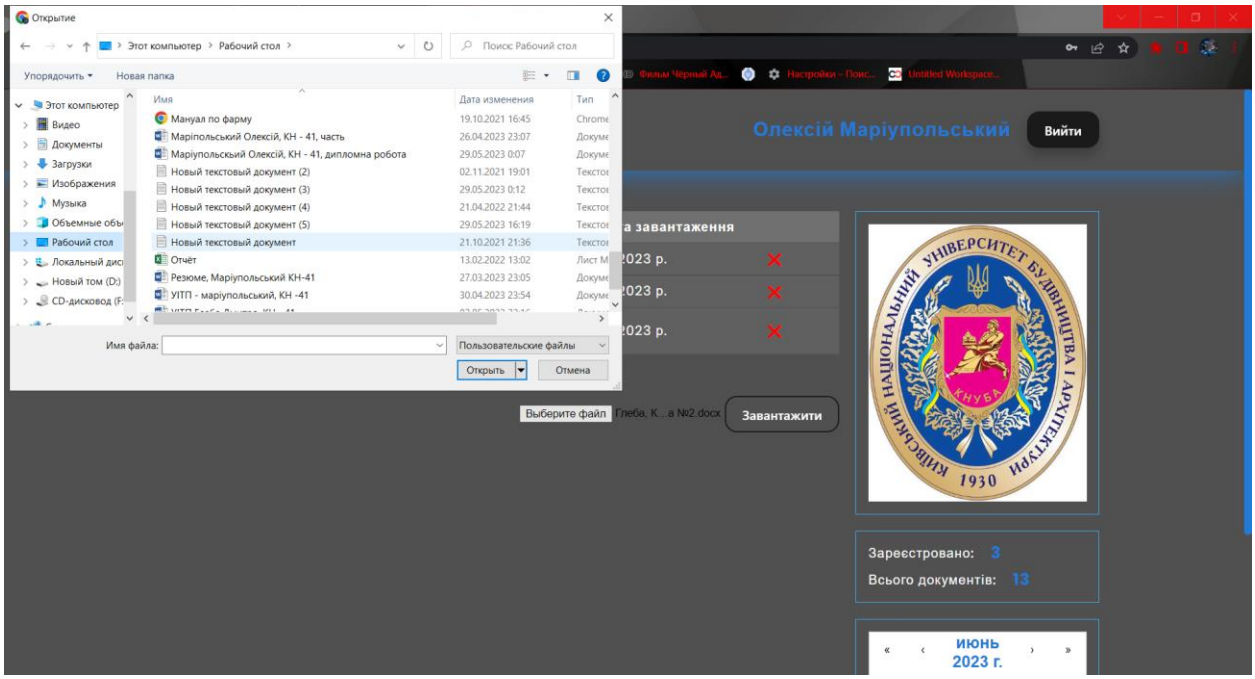

Рисунок 4.5 – функція завантаження

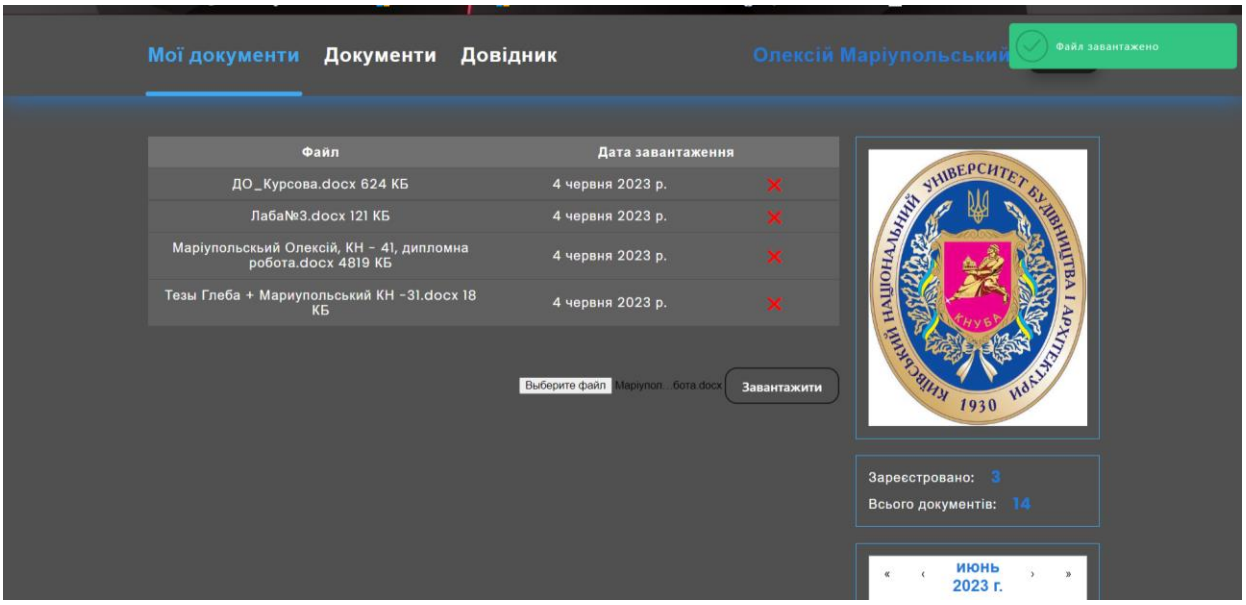

Рисунок 4.6 – інформування

4.5 Видалення документів

Також добавлена функція видалення документів із системи, але реалізована так, що видалити користувач може тільки ті які сам і завантажив, під час видалення також вспливає інформування про підтвердження данної дії рисунок 4.7.

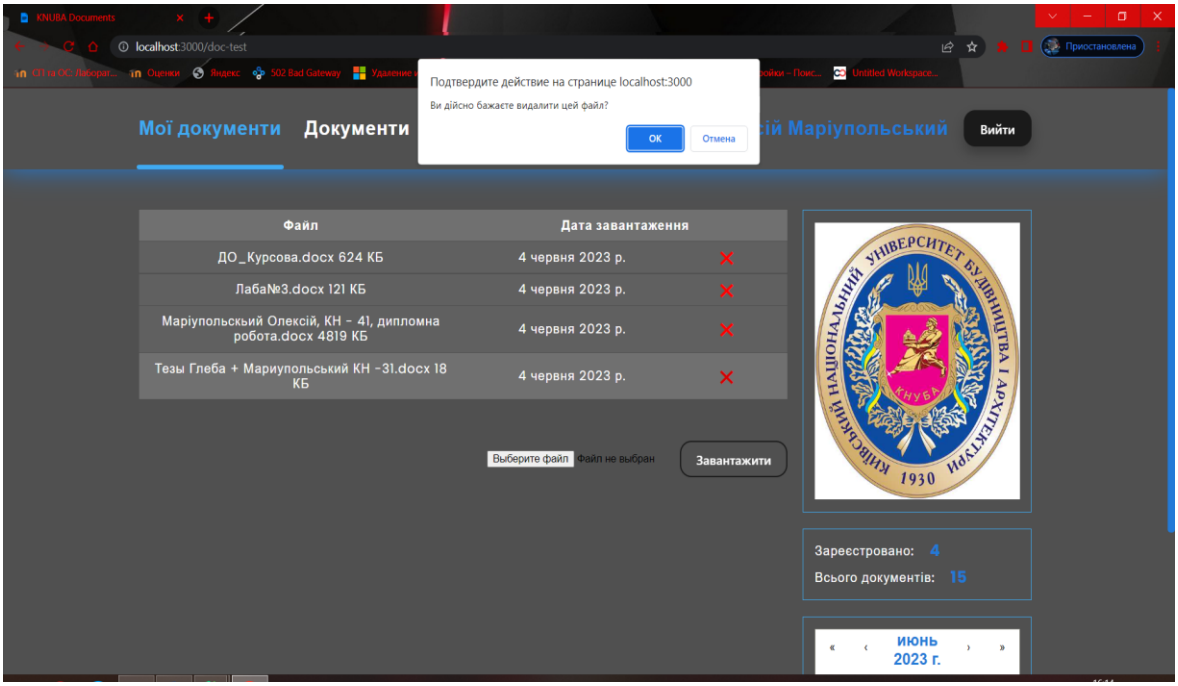

Рисунок 4.7 – видалення документів

4.6 Алгоритм стиснення документів

Для кожного пікселя присутній колір, який формується за допомогою моделі ARGB. ARGB (Alpha, Red, Green, Blue) - основні кольори завдяки яким відбувається формування кольору пікселя. 1 колір = 1 байт інформації, отже щоб зберегти в пам’яті 1 піксель потрібно 4 байти.

$$1 \text{ піксель} = 10010100 = 32 \text{ біт} (4 \text{ байта })$$

Документи зазвичай у чорному-білому форматі зображення, тобто зберегти їх можна у електронному вигляді у чорному-білому кольорі. Середнє арифметичне значення кожного з кольорів R,G,B слугує для переведення одного кольорового пікселю у сірий відтінок, а потім присвоїти

отримане значення кожному з кольорів. Таким чином значеннння R, G, B - однакові, при цьому прозорість A можна опустити.

$$1 \text{ кольоровий піксель} = \frac{R + G + B}{3} = 8 \text{ Біт (1 байт)}$$

В пам’яті збереження переводу пікселя у сірий відтінок займає 1 байт, що значно зменшує розмір файлу.

З приводу чорно-білих документів, вони не всі є чорно-білими, адже в деяких є кольоровий підпис та кольорова печатка. Алгоритм виділяє кольору зону і залишає її без змін аби зберегти такий документ.

4.7 Фінансовий план виконання проекту

Цей розділ надає загальний огляд фінансових аспектів проекту, включаючи його мету, об'єктиви та основні аспекти, які будуть враховуватися при розробці фінансової стратегії.

У нашому проекті до фінансового плану можна включити наступну інформацію:

Мета фінансового плану: полягає у забезпеченні фінансової стабільності та успішності проекту. В основі цієї мети лежить бажання забезпечити ефективне використання фінансових ресурсів, зробити обґрунтовані фінансові рішення та досягнути поставлених фінансових цілей.

Що до основної мети фінансового плану включаємо:

1. **Забезпечення фінансової стабільності**: Фінансовий план допомагає забезпечити належне фінансове планування та управління ресурсами, щоб уникнути фінансових труднощів, недостачі коштів або заборгованості.

2. **Оцінка фінансової вигоди**: Фінансовий план дає можливість оцінити фінансову вигоду проекту, визначити його потенційну прибутковість та рентабельність. Це допомагає визначити, чи є проект економічно доцільним та прийняти відповідні рішення.

3. **Прогнозування фінансових результатів**: Фінансовий план дозволяє зробити прогноз фінансових результатів проекту на майбутні періоди. Це

включає прогноз продажів, доходів, витрат, прибутку та інших фінансових показників, що допомагає планувати дії і вживати заходів для досягнення поставлених цілей.

4. **Ефективне управління фінансовими ресурсами**: Фінансовий план надає можливість керівництву проекту ефективно управляти фінансовими ресурсами, розподіляти кошти між різними потребами, контролювати витрати та забезпечувати оптимальне використання фінансових ресурсів.

5. **Залучення інвестицій**: Фінансовий план може використовуватися для привернення інвестицій або фінансування проекту. Він має демонструвати потенційним інвесторам або кредиторам перспективи проекту та його фінансову привабливість.

Загальна мета фінансового плану полягає в забезпеченні фінансової стійкості, ефективного управління ресурсами, прогнозуванні результатів та досягненні фінансових цілей проекту. Відповідно до цього, фінансовий план розробляється з урахуванням специфічних потреб та особливостей проекту для досягнення найкращих результатів.

Об'єктиви: Об'єктиви фінансового плану включають наступні конкретні цілі:

1. **Збільшення обсягу продажів**: Одним з головних об'єктивів може бути збільшення обсягу продажів або доходів компанії. Це може досягатися шляхом впровадження маркетингових стратегій, розширення ринків збуту, підвищення конкурентоспроможності продукту або послуги.

2. **Досягнення прибутковості**: Однією з ключових цілей може бути досягнення прибутковості проекту або підприємства. Це включає досягнення позитивного фінансового результату, забезпечення рентабельності та виконання фінансових показників.

3. **Ефективне управління фінансовими ресурсами**: Одним з об'єктивів може бути ефективне управління фінансовими ресурсами, включаючи оптимізацію витрат, контроль над бюджетом, мінімізацію фінансових ризиків та оптимальне використання доступних коштів.

4. **Підвищення фінансової стійкості**: Метою може бути забезпечення фінансової стійкості та стабільності проекту або організації. Це включає виявлення та забезпечення достатнього рівня фінансових резервів, забезпечення надійного фінансового планування та контролю за фінансовими процесами.

5. **Мінімізація фінансових ризиків**: Одним з об'єктивів може бути мінімізація фінансових ризиків і забезпечення фінансової безпеки. Це включає аналіз та оцінку ризиків, розробку стратегій зменшення ризиків, застосування фінансових інструментів для забезпечення стабільності та захисту від небажаних фінансових втрат.

Конкретні об'єктиви фінансового плану будуть залежати від конкретних потреб та цілей проекту чи організації. Важливо сформулювати їх чітко і конкретно, щоб вони відповідали потребам і сприяли досягненню фінансової успішності та стабільності.

Цільові показники: у фінансовому плані ми будемо враховувати наступні ключові фінансові показники:

1. **Обсяг продажів**: Цей показник визначає загальний обсяг продукції або послуг, які планується реалізувати протягом певного періоду. Він відображає потенційні доходи від продажів і є одним із головних показників фінансового успіху проекту.

2. **Прибуток**: Прибуток відображає різницю між доходами і витратами. Цей показник дозволяє оцінити фінансову ефективність проекту і вказує на його прибутковість.

3. **Рентабельність**: Рентабельність визначається як відношення прибутку до загальних витрат. Вона показує, яка частина витрат повертається у вигляді прибутку і допомагає оцінити ефективність використання ресурсів.

4. **Витрати**: Витрати включають всі витрати, пов'язані з розробкою, виробництвом або наданням послуг. Цей показник дозволяє контролювати і планувати витрати, а також визначити ефективність використання ресурсів.

5. **Інвестиції**: Інвестиції відображають суму коштів, яку потрібно вкласти в проект для його реалізації. Цей показник є важливим для оцінки фінансової потреби проекту і планування джерел фінансування.

Ці цільові показники допоможуть нам оцінити фінансову вигоду проекту, забезпечити фінансову стабільність та раціональне використання ресурсів. Вони також стануть основою для прогнозування фінансових результатів і прийняття важливих рішень з планування та управління проектом.

Методи фінансового аналізу: Для аналізу фінансових даних і забезпечення обґрунтованості рішень використовуються різні методи та інструменти фінансового аналізу. Основні методи, які можуть бути використані, включають:

1. **Оцінка інвестицій**: Цей метод дозволяє оцінити ефективність і рентабельність інвестиційного проекту. Використовуються такі показники, як NPV (чиста поточна вартість), IRR (внутрішня норма доходності), ROI (рентабельність інвестицій) та інші. Ці методи допомагають визначити, чи варто реалізувати проект і яка його потенційна фінансова вигода.

2. **Аналіз рентабельності**: Цей метод дозволяє оцінити рентабельність діяльності підприємства. Використовуються такі показники, як рентабельність продажів, рентабельність активів, рентабельність капіталу тощо. Цей аналіз допомагає визначити ефективність використання ресурсів та прийняти рішення з покращення рентабельності.

3. **Прогнозування фінансових потоків**: Цей метод використовується для прогнозування майбутніх фінансових потоків підприємства. Використовуються фінансові моделі і методи, такі як метод дисконтування, аналіз трендів, сценарний аналіз тощо. Це допомагає зробити прогнози щодо прибутку, грошових потоків та інших фінансових показників для планування та прийняття рішень.

4. **Фінансовий ризиковий аналіз**: Цей метод дозволяє виявити та оцінити ризики, пов'язані з фінансовою діяльністю підприємства.

Використовуються такі інструменти, як аналіз чутливості, аналіз ймовірностей, сценарний аналіз тощо. Цей аналіз допомагає визначити ризики, які можуть вплинути на фінансові результати проекту і прийняти заходи для їх управління.

Користування цими методами та інструментами дозволить провести детальний фінансовий аналіз проекту, забезпечити обґрунтованість рішень та зменшити фінансові ризики. Вони допоможуть зрозуміти фінансову ситуацію проекту, здійснити прогнозування та планування фінансових результатів, а також визначити можливості для покращення фінансової продуктивності.

Обмеження та ризики: Під час розробки фінансового плану необхідно врахувати потенційні фінансові обмеження та ризики, які можуть впливати на проект. Деякі з них включають:

1. **Фінансові обмеження з боку інвесторів**: Можливі обмеження щодо доступності фінансування або умови, пов'язані з ним, які потрібно враховувати при розробці фінансового плану. Наприклад, інвестор може встановити максимальну суму фінансування або вимоги щодо використання коштів.

2. **Зміни в регуляторному середовищі**: Зміни в законодавстві, правилах та регуляторних вимогах можуть мати вплив на фінансову діяльність проекту. Наприклад, зміни в податкових ставках, правилах бухгалтерського обліку або регулюванні сфери діяльності проекту.

3. **Залежність від зовнішніх факторів**: Деякі проекти можуть бути вразливими до змін на ринку, економічних умов, цінової конкуренції або попиту на продукти чи послуги. Ці зовнішні фактори можуть впливати на обсяг продажів, ціни або витрати проекту.

4. **Фінансові ризики**: Включають ризик валютних коливань, процентних ставок, змін ринкових цін або невдалі інвестиції. Ці ризики можуть призвести до непередбачуваних змін у фінансових результатах проекту.

5. **Фінансові витрати**: Розробка та впровадження фінансового плану можуть вимагати додаткових витрат на залучення експертів, інвестиційний аналіз, фінансову звітність тощо. Необхідно враховувати ці витрати при плануванні бюджету проекту.

Враховуючи ці обмеження та ризики, важливо ретельно проаналізувати їх вплив на фінансовий план і прийняти відповідні заходи для мінімізації ризиків та забезпечення фінансової стабільності проекту.

Тому після аналізу розробимо список фінансового плану нашого проекту, та постараємося описати усі затрати на розробку проекту, та його прибутки.

1.Бюджет проекту:

1.1 Загальні витрати на проект: $500,000

1.2 Плановані терміни проекту: 12 місяців

1.3 Планована вартість проекту за місяць: $41,667 ($500,000 / 12 місяців)

2.Фінансування проекту:

2.1 Власні кошти компанії: $100,000

2.2 Кредитні ресурси: $200,000 (під 15% річних на 5 років)

2.3 Інвестиції: $100,000

2.4 Інші джерела фінансування: $100,000

2.5 Загальний обсяг фінансування: $500,000

3.Розподіл бюджету:

3.1 Загальні витрати на розробку ПЗ: $200,000

3.2 Витрати на аналіз ринку та конкурентів: $50,000

3.3 Витрати на маркетинг: $100,000

3.4 Витрати на рекламу та просування: $50,000

3.5 Витрати на технічну підтримку: $50,000

3.6 Витрати на вдосконалення ПЗ: $100,000

3.7 Інші витрати: $50,000

3.8 Загальні витрати проекту: $600,000

4.Планована прибутковість проекту:

4.1 Орієнтовна вартість продажу ПЗ: $200

4.2 Планована кількість продажів: 10,000

4.3 Орієнтовний прибуток від продажів ПЗ: $2,000,000

4.4 Прибутковість проекту: $1,400,000 ($2,000,000 - $600,000)

5. Керування ризиками:

5.1 Резервний фонд на непередбачувані витрати: $50,000

5.2 Фонд зменшення ризику: $100,000

6.Прогноз доходів на кожен місяць проекту:

Місяць 1: $3,000

Місяць 2: $6,000

Місяць 3: $9,000

Місяць 4: $12,000

Місяць 5: $15,000

Місяць 6: $18,000

7.Розраховані витрати на кожен місяць проекту:

Місяць 1: $22,500

Місяць 2: $22,500

Місяць 3: $22,500

Місяць 4: $22,500

Місяць 5: $22,500

Місяць 6: $22,500

8.Планована прибутковість на кожен місяць проекту:

Місяць 1: ($19,500)

Місяць 2: ($16,500)

Місяць 3: ($13,500)

Місяць 4: ($10,500)

Місяць 5: ($7,500)

Місяць 6: ($4,500)

Це повний фінансовий план проекту, який детально відображає всі витрати та доходи, що пов'язані з розробкою програмного продукту.

Зважаючи на врахування всіх складових пунктів плану, визначена планована прибутковість проекту, яка показує, що в перші місяці продукт може не приносити прибутків, але з часом стане дохідним. Також, в плані передбачені резервні фонди на непередбачувані витрати та фонд зменшення ризику для покриття можливих негативних наслідків.

Можливі ризики.

Під час реалізації будь-якого проекту можуть виникнути різні проблеми, які можуть вплинути на його успішність та завершення в заплановані терміни. До можливих проблем, які можуть виникнути під час реалізації проекту віднесемо:

1. **Зміна вимог клієнта**: Під час розробки проекту замовник може змінити свої вимоги, що може призвести до необхідності переробки деяких частин проекту.
2. **Недостатня кваліфікація команди**: Якщо команда, яка працює над проектом, не має достатньої кваліфікації або досвіду, то це може призвести до затримок у розробці або до невдалих рішень, які можуть пошкодити проект.
3. **Технічні проблеми**: Технічні проблеми, такі як збої обладнання або програмного забезпечення, можуть призвести до затримок у розробці та негативно вплинути на проект.
4. **Несумісність технологій**: Якщо в проекті використовуються різні технології, то можуть виникнути проблеми з їх сумісністю, що може призвести до затримок у розробці та до необхідності переробки частин проекту.
5. **Проблеми з комунікацією**: Якщо комунікація між учасниками проекту не ефективна, то це може призвести до недорозумінь, затримок у розробці та до невдалого керування проектом.
6. **Фінансові проблеми**: Недостатність фінансування або невдале управління фінансами можуть призвести до затримок у розробці та недосягнення мети проекту.

7. **Неочікувані зміни на ринку**: Неочікувані зміни на ринку можуть негативно впливати на наш проект.

Іншою можливою проблемою може бути недостатня кількість ресурсів, таких як фінансові, людські, технічні тощо. Це може призвести до затримок в графіку, низької якості розробки ПЗ та невиконання запланованих функцій.

Також, неправильне управління командою може стати проблемою. Недостатнє чи надмірне керівництво, недостатня комунікація, відсутність мотивації та недостатня відповідальність можуть призвести до конфліктів та зниження ефективності роботи, можуть виникнути проблеми з підтримкою ПЗ, яке розробляється. Потрібно передбачити достатню кількість ресурсів на підтримку, оновлення та вдосконалення підсистеми, а також на відповідну підтримку користувачів.

Серйозною проблемою може стати конкуренція на ринку. Необхідно враховувати конкурентні переваги та недоліки проекту, а також встановлювати стратегію маркетингу та просування продукту.

ВИСНОВКИ

Під час аналізу та дослідженні проблеми були виявлені такі проблеми: недостатня ефективність процесів, низьку безпеку та конфіденційність, обмежену інтеграцію з іншими системами, складність управління версіями та контролю доступу, а також обмежену мобільність та доступність. Такі проблеми мають негативний вплив на продуктивність, ефективність та якість роботи на підприємстві. Вони можуть сповільнювати процеси, збільшувати кількість помилок. Враховуючи виявлені проблеми та їх вплив, вирішення цих проблем стає важливим завданням для підприємства. Вдосконалення процесів електронного документообігу та впровадження нашого веб - додатку може покращити продуктивність та ефективність. Усі ці аспекти підкреслюють важливість розробки та необхідність подальшого дослідження та впровадження рекомендацій для вирішення виявлених проблем.

При виконанні проекту було розроблено архітектуру, яка визначає структуру та взаємозв'язки між різними компонентами підсистеми. Також використання розроблених WBS та OBS моделей проекту дозволило розділити його на окремі робочі частини та визначити ролі та відповідальність учасників проекту, такий підхід дозволяє краще управляти проектом та збільшити ефективність розробки. Було враховано різні аспекти проекту, вибір моделі життєвого циклу, концепція підсистеми, принципи документообігу та проектування баз даних, сприяло створенню ефективної та функціональної підсистеми електронного обігу документів для вибраного підприємства.

Основною метою була розробка ефективної архітектури та інтерфейсу, які задовольнять вимоги та потреби користувачів.

Клієнтська частина веб-додатку написана з використанням HTML, CSS та JavaScript, що забезпечує його інтерактивність та візуальну привабливість.

Серверна частина веб-додатку відповідає за обробку запитів користувачів, взаємодію з базою даних та забезпечення цілісності даних.

Для розробки інтерфейсної та функціональної частини, було розглянуто мову програмування JavaScript для реалізації клієнтської логіки та взаємодії з сервером. Також було використано фреймворк React, який надає зручні інструменти для розробки модульних та швидких інтерфейсів.

У результаті нашої роботи також побудовано детальні моделі баз даних концептуальну, даталогічну та фізичну тому, що ці моделі є основою для створення та управління даними електронного документообігу на підприємстві.

В результаті можна добавити, що проект був реалізований із досягненням зазначених результатів у створенні нашого веб-додатку електронного документообігу. Основні функції та декілька додаткових було успішно реалізовано, і ми створили функціональну систему, яка забезпечує обмін документами у нашому підприємстві.

Проте, в процесі реалізації проекту з'явилися певні труднощі та обмеження, які не дозволили нам повністю реалізувати всі задумані функції та можливості. Обмежений час та ресурси призвели до необхідності пріоритезації функцій та уваги на основних елементах системи.

В цілому, хоча і не реалізовано все, що задумувалось, наш проект залишається важливим кроком у вдосконаленні документообігу в нашому підприємстві тому, що він принесе значну користь та сприятиме підвищенню продуктивності та ефективності роботи.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Про електронні документи та електронний документообіг: Закон України від 22.05.2003 р. № 851-IV: станом на 31 грудня 2023 р. URL: https://zakon.rada.gov.ua/laws/show/851-15#Text (дата звернення: 04.04.2024).

2. Електронний документообіг FossDoc. Опис системи. Система електронного документообігу FossDoc. Офіційний сайт. URL: https://fossdoc.com/elektronniy-dokumentooborot (дата звернення: 04.04.2024).

3. Smyrnov I. Що таке веб додаток? | Блог WEBCASE. Webcase. URL: https://webcase.com.ua/uk/blog/cho-takoe-web-prilozhenie-vse-vidy/ (дата звернення: 04.04.2024).

4. Локально-обчислювальні мережі. Українські реферати. URL: https://reff.net.ua/28967-Lokal_no_vychislitel_nye_seti.html (дата звернення: 04.04.2024).

5. Smyrnov I. Що таке веб додаток? | Блог WEBCASE. Webcase. URL: https://webcase.com.ua/uk/blog/cho-takoe-web-prilozhenie-vse-vidy/ (дата звернення: 04.04.2024).

6. CSS Tutorial. *W3Schools Online Web Tutorials*. URL: https://www.w3schools.com/css/ (date of access: 05.04.2024).

7. Express - Node.js web application framework. *Express*. URL: https://expressjs.com/ (date of access: 05.04.2024).

8. Express 4.x - API Reference. *Express - Node.js web application framework*. URL: https://expressjs.com/en/api.html (date of access: 05.04.2024).

9. GitHub - facebook/react: The library for web and native user interfaces. *GitHub*. URL: https://github.com/facebook/react (date of access: 05.04.2024).

10. Hohpe G. Enterprise integration patterns: Designing, building, and deploying messaging solutions. Boston: Addison-Wesley, 2003. 683 p.

11. HTML Tutorial. *W3Schools Online Web Tutorials*. URL: https://www.w3schools.com/html/ (date of access: 05.04.2024).

12. Index of /docs/. *Node.js*. URL: https://nodejs.org/docs/ (date of access: 05.04.2024).

13. MDN Web Docs. *MDN Web Docs*. URL: https://developer.mozilla.org/en-US/ (date of access: 05.04.2024).

14. Node.js - cross-platform JavaScript runtime environment. *Node.js*. URL: https://nodejs.org/en (date of access: 05.04.2024).

15. React. The library for web and native user interfaces. *React*. URL: https://react.dev/ (date of access: 05.04.2024).

16. Stefanov S. React: up & running: Building web applications. 2016. 201 p.URL: https://pgasa.dp.ua/wp-content/uploads/2018/02/Tema-3.pdf [2]

17. Борщ Л. М. Інвестування: теорія і практика: навчальний посібник / Л. М. Борщ. – Київ: Знання, 2005. – 470 с.

18. Букер, Тоні. "Документообіг: реалізація та автоматизація". Видавництво "Біном". Рік видання: 2018.

19. Грабіновський, Сергій. "Електронний документообіг на основі відкритого програмного забезпечення". Видавництво "Університетська книга". Рік видання: 2019.

20. Етапи впровадження системи електронного документообігу – Wiki TNEU. *Wiki TNEU*. – 2012. URL: http://wiki.tneu.edu.ua/index.php?title=Етапи впровадження системи електронного документообігу (дата звернення: 05.04.2024).

21. Казієва Н. Критерії вибору системи електронного документообігу / Н. Казієва // Діловодство та документообіг. – 2017. – № 4. – С. 8-13

22. Мартін Фаулер Рефракторинг: покращення проекта існуючого коду / Мартін Фаулер, Кент Бек, Джон Брант, Уильям Опдайк, Дон Робертс. – Діалектика, 2019. – 448 с

23. Ніколашин А. Проблеми електронного документообігу та шляхи їх вирішення. *Облік і фінанси АПК: освітній портал*. URL: https://magazine.faaf.org.ua/problemi-elektronnogo-dokumentoobigu-ta-shlyahi-ih-virishennya.html.

24. Прийоми об'єктно-орієнтованого проектування. Патерни проектування. / Е. Гамма та ін. Харків: Б-ка програм., 2001. 368 с. URL: http://www.sugardas.lt/~p2d/books/Priemioop.pdf.

25. Про затвердження переліку обов'язкових етапів робіт під час проектування, впровадження та експлуатації засобів інформатизації: Постанова Каб. Міністрів України від 04.02.1998 р. № 121: станом на 03 вересня 2011 р. URL: https://zakon.rada.gov.ua/laws/show/121-98-п#Text (дата звернення: 05.04.2024).

26. Про інформацію: Закон України від 02.10.1992 р. № 2657-XII: станом на 27 липня 2023 р. URL: https://zakon.rada.gov.ua/laws/show/2657-12#Text (дата звернення: 05.04.2024).

27. Про місцеві державні адміністрації: Закон України від 09.04.1999 р. № 586-XIV: станом на 03 серпня 2023 р. URL: https://zakon.rada.gov.ua/laws/show/586-14#Text (дата звернення: 05.04.2024).

28. Системи управління інвестиційними проектами: конспект Ц130 лекцій / уклад.: С. В. Цюцюра, О.В. Криворучко. – К.: КНУБА, 2010. – 56 с.

29. Сомервіль І. "Інженерія програмного забезпечення" - Університетська книга, 2018.

30. Цюцюра С. В. «Методичні вказівки до виконання дипломної роботи» – // КНУБА – 2022.

31. Цюцюра Світлана Володимирівна. Управління інноваційними проектами модернізації підприємств енергоємних галузей: дис. д-ра техн. наук: 05.13.22 / Київський національний ун-т будівництва і архітектури. — К., 2007. — 342 арк. — Бібліогр.: арк. 312-333.

32. Чирський, Ю. Запровадження системи електронного документообігу в Україні // Юридичний журнал: Аналіт. матеріали. Коментарі. Судова практика. - 2006. - № 6. - С. 90-92. URL: http://old.minjust.gov.ua/7546

Додаток А

Код програмного забечення.

Клієнтська частина веб - додатку

Код сторінки “Мої документи”

```
import { memo, useEffect } from 'react';
import s from './MainPage.module.scss';

import Header from 'components/Header/Header';
import Container from 'components/Container/Container';
import MainStats from 'components/MainStats/MainStats';
import MyDocsComp from 'components/MyDocsComp/MyDocsComp';
import { useDispatch, useSelector } from 'react-redux';
import { selectCurrentUser, selectDocs } from 'redux/docs/docsSelectors';
import { getDocs } from 'redux/docs/docsOperations';

const MainPage = ({ setModalData }) => {
  const dispatch = useDispatch();
  const docs = useSelector(selectDocs);
  const user = useSelector(selectCurrentUser)

  const filteredArr = docs.filter(item => item.user === user?.name);


  useEffect(() => {
    dispatch(getDocs());
  }, [dispatch]);

  return (
    <>
      <Header setModalData={setModalData} />
      <Container className={s.container}>
        {user?.name && <MyDocsComp arr={filteredArr} />}
        <MainStats />
      </Container>
    </>
  );
};

export default memo(MainPage);
```

```
import Container from 'components/Container/Container';
import s from './Header.module.scss';
```

```
import { NavLink } from 'react-router-dom';
import clsx from 'clsx';
import { useDispatch, useSelector } from 'react-redux';
import { selectCurrentUser } from 'redux/docs/docsSelectors';
import { logOutUser } from 'redux/docs/docsOperations';

const getActiveClass = ({ isActive }) => clsx(s.link, isActive && s.active);

const Header = ({ setModalData }) => {
  const dispatch = useDispatch();
  const isCurrentUser = useSelector(selectCurrentUser);

  const handleModalOpen = action => {
    setModalData(action);
  };

  const handleLogOut = action => {
    dispatch(logOutUser());
  };

  return (
    <header className={s.header}>
      <Container className={s.headerContainer}>
        <nav>
          <ul className={s.navLinks}>
            <li>
              <NavLink className={getActiveClass} to="/">
                Мої документи
              </NavLink>
            </li>
            <li>
              <NavLink className={getActiveClass} to="/docs">
                Документи
              </NavLink>
            </li>
            <li>
              <NavLink className={getActiveClass} to="/users">
                Довідник
              </NavLink>
            </li>
          </ul>
        </nav>
```

```
        {!isCurrentUser ? (
          <div className={s.btnWrap}>
            <button onClick={() => handleModalOpen('login')}>Увійти</button>
            <button
              className={s.btnRegister}
              onClick={() => handleModalOpen('register')}
            >
              Зареєструватись
            </button>
          </div>
        ) : (
          <div className={s.btnWrap}>
            <p>{isCurrentUser.name}</p>
            <button
              className={s.btnRegister}
              onClick={() => handleLogOut('register')}
            >
              Вийти
            </button>
          </div>
        )}
      </Container>
    </header>
  );
};

export default Header;
```

```
import { delApi, downloadApi } from 'services/docsApi';
import s from './MyDocsComp.module.scss';
import { useDispatch } from 'react-redux';
import { getDocs } from 'redux/docs/docsOperations';
import { useState } from 'react';
import UploadFileComp from 'components/UploadFileComp/UploadFileComp';

const MyDocsComp = ({ arr }) => {
  const selectedOption = useState('');
  const dispatch = useDispatch();

  function formatDate(dateStr) {
    const options = { day: 'numeric', month: 'long', year: 'numeric' };
    const formattedDate = new Date(dateStr).toLocaleDateString(
```

```
    'uk-UA',
    options
  );
  return formattedDate;
 }

 const filteredArr = arr.filter(item => {
  if (selectedOption === '') return item;

  // const filterValue =
  //   selectedOption === 'date'
  //     ? formatDate(item.created)
  //     : item[selectedOption];
    return item;
  // return filterValue.toLowerCase().includes(searchText.toLowerCase());
 });

 const handleDelete = async path => {
  // eslint-disable-next-line
  const confirmation = window.confirm('Ви дійсно бажаєте видалити цей
файл?');
  await delApi(path);
  dispatch(getDocs());
 };

 return (
  <div className={s.myDocWrap}>
   <ul className={s.arr}>
    <li>
     <p className={s.title}>Файл</p>
     <p className={s.title}>Дата завантаження</p>
    </li>
    {filteredArr.map(({ name, user, size, created }) => (
     <li key={name + created} className={s.item}>
      <p>{name + ' ' + (size / 1024).toFixed(0)} КБ</p>
      <p>
       {formatDate(created)}
       <button
        type="button"
        className={s.delBtn}
        onClick={e => {
         e.stopPropagation();
```

```
                handleDelete({ user, name });
              }}
            >
              ✕
            </button>
          </p>
        </li>
      ))}
    </ul>
    <UploadFileComp />
  </div>
 );
};

export default MyDocsComp;
```

Код сторінки “Документи”

```
import { memo, useEffect, useState } from 'react';

import s from './DocsPage.module.scss';
import Header from 'components/Header/Header';
import Container from 'components/Container/Container';
import UploadFileComp from 'components/UploadFileComp/UploadFileComp';
import { useDispatch, useSelector } from 'react-redux';
import { getDocs } from 'redux/docs/docsOperations';
import { selectCurrentUser, selectDocs } from 'redux/docs/docsSelectors';
import DocsComp from 'components/DocsComp/DocsComp';
import DocFilter from 'components/DocFilter/DocFilter';
import ErrorMsg from 'components/ErrorMsg/ErrorMsg';

const DocsPage = ({ setModalData }) => {
  const dispatch = useDispatch();
  const user = useSelector(selectCurrentUser)
  const docs = useSelector(selectDocs);
  const [searchText, setSearchText] = useState('');
  const [selectedOption, setSelectedOption] = useState('');

  useEffect(() => {
    dispatch(getDocs());
  }, [dispatch]);
```

```
  return (
    <>
      <Header setModalData={setModalData} />
      <Container>
        {user?.name ? (
          <>
            <DocFilter
              setSearchText={setSearchText}
              setSelectedOption={setSelectedOption}
              searchText={searchText}
              selectedOption={selectedOption}
            />
            <DocsComp arr={docs} filter={{ searchText, selectedOption }} />
          </>
        ) : (
            <ErrorMsg />
        )}
      </Container>
    </>
  );
};

export default memo(DocsPage);
```

```
import s from './DocFilter.module.scss';

const DocFilter = ({
  setSearchText,
  setSelectedOption,
  searchText,
  selectedOption,
}) => {

  const handleInputChange = event => {
    setSearchText(event.target.value);
  };

  const handleRadioChange = event => {
    setSelectedOption(event.target.id);
  };

  return (
```

```
<>
 <div className={s.inputWrap}>
  <div className={s.formGroup}>
   <input
    type="input"
    className={s.formField}
    placeholder="Пошук"
    value={searchText}
    onChange={handleInputChange}
   />
   <label htmlFor="name" className={s.formLabel}>
    Пошук
   </label>
  </div>
  <div className={s.radioInputs}>
   <label>
    <input
     className={s.radioInput}
     type="radio"
     name="engine"
     id="user"
     checked={selectedOption === 'user'}
     onChange={handleRadioChange}
    />
    <span className={s.radioTile}>
     <span className={s.radioLabel}>П.І.Б.</span>
    </span>
   </label>
   <label>
    <input
     className={s.radioInput}
     type="radio"
     name="engine"
     id="name"
     checked={selectedOption === 'name'}
     onChange={handleRadioChange}
    />
    <span className={s.radioTile}>
     <span className={s.radioLabel}>Файл</span>
    </span>
   </label>
   <label>
```

```
            <input
              className={s.radioInput}
              type="radio"
              name="engine"
              id="date"
              checked={selectedOption === 'date'}
              onChange={handleRadioChange}
            />
            <span className={s.radioTile}>
              <span className={s.radioLabel}>Дата</span>
            </span>
          </label>
        </div>
      </div>
    </>
  );
};

export default DocFilter;
```

```
import { delApi, downloadApi } from 'services/docsApi';
import s from './DocsComp.module.scss';
import { useDispatch, useSelector } from 'react-redux';
import { getDocs } from 'redux/docs/docsOperations';
import { selectCurrentUser } from 'redux/docs/docsSelectors';

const DocsComp = ({ arr, filter }) => {
  const dispatch = useDispatch();
  const currentUser = useSelector(selectCurrentUser);

  function formatDate(dateStr) {
    const options = { day: 'numeric', month: 'long', year: 'numeric' };
    const formattedDate = new Date(dateStr).toLocaleDateString(
      'uk-UA',
      options
    );
    return formattedDate;
  }

  const { searchText, selectedOption } = filter;

  const filteredArr = arr.filter(item => {
```

```
    if (searchText === '') return arr;

    const filterValue =
      selectedOption === 'date'
        ? formatDate(item.created)
        : item[selectedOption];

    return filterValue.toLowerCase().includes(searchText.toLowerCase());
  });

  const handleDownload = path => {
    const confirmation = window.confirm('Ви дійсно бажаєте завантажити цей файл?');

    if (confirmation) {
      downloadApi(path);
    }
  };

  return (
    <ul className={s.arr}>
      <li>
        <p className={s.title}>П.І.Б</p>
        <p className={s.title}>Файли</p>
        <p className={s.title}>Дата завантаження</p>
      </li>
      {filteredArr.map(({ name, user, size, created }) => (
        <li
          key={name + created}
          onClick={() => handleDownload({ user, name })}
          className={s.item}
        >
          <p>{user}</p>
          <p>{name + ' ' + (size / 1024).toFixed(0)} КБ</p>
          <p>
            {formatDate(created)}
          </p>
        </li>
      ))}
    </ul>
  );
};
```

```
export default DocsComp;
```

Код сторінки “Довідка”

```
import { memo, useState } from 'react';

import Header from 'components/Header/Header';
import Container from 'components/Container/Container';
import ListComp from 'components/ListComp/ListComp';
import { useSelector } from 'react-redux';
import { selectCurrentUser, selectUsers } from 'redux/docs/docsSelectors';
import UserFilter from 'components/UserFilter/UserFilter';
import ErrorMsg from 'components/ErrorMsg/ErrorMsg';

const UsersPage = ({ setModalData }) => {
  const users = useSelector(selectUsers);
  const user = useSelector(selectCurrentUser);
  const [searchText, setSearchText] = useState('');
  const [selectedOption, setSelectedOption] = useState('');

  return (
    <>
      <Header setModalData={setModalData} />
      <Container>
        {user?.name ? (
          <>
            <UserFilter
              setSearchText={setSearchText}
              setSelectedOption={setSelectedOption}
              searchText={searchText}
              selectedOption={selectedOption}
            />
            <ListComp
              arr={users}
              filter={{ searchText, selectedOption }}
              titles={['П.І.Б.', 'Посада', 'Дата реєстрації']}
            />
          </>
        ) : (
          <ErrorMsg />
        )}
      </Container>
```

```
    </>
  );
};

export default memo(UsersPage);
```

```
import { current } from '@reduxjs/toolkit';
import s from './ListComp.module.scss';
import { useState } from 'react';

const ListComp = ({ arr, filter, titles }) => {
  const [currentUserData, setCurrentUserData] = useState(null);
  function formatDate(dateStr) {
    const options = { day: 'numeric', month: 'long', year: 'numeric' };
    const formattedDate = new Date(dateStr).toLocaleDateString(
      'uk-UA',
      options
    );
    return formattedDate;
  }

  const { searchText, selectedOption } = filter;

  const filteredArr = arr.filter(item => {
    if (searchText === '') return arr;
    const filterValue =
      selectedOption === 'date'
        ? formatDate(item.createdAt)
        : item[selectedOption];
    return filterValue.toLowerCase().includes(searchText.toLowerCase());
  });

  const handleUserClick = data => {
    setCurrentUserData(data);
  };

  return (
    <>
      <ul className={s.arr}>
        <li>
          <p className={s.title}>{titles[0]}</p>
          <p className={s.title}>{titles[1]}</p>
```

```
          <p className={s.title}>{titles[2]}</p>
        </li>
        {filteredArr.map(item => (
          <li
            key={item.id}
            onClick={() =>
              handleUserClick({ email: item.email, name: item.name })
            }
          >
            <p>{item.name}</p>
            <p>{item.status}</p>
            <p>{formatDate(item.createdAt)}</p>
          </li>
        ))}
      </ul>
      {currentUserData?.email && (
        <div
          onClick={e => {
            e.target === e.currentTarget && setCurrentUserData(null);
          }}
          className={s.backdrop}
        >
          <div>
            <p className={s.currentName}>{currentUserData.name}</p>
            <p className={s.currentEmail}>Email: {currentUserData.email}</p>
          </div>
        </div>
      )}
    </>
  );
};

export default ListComp;
```

Код авторизації

```
import { useState } from 'react';
import s from './LoginForm.module.scss';
import { useDispatch, useSelector } from 'react-redux';
import { Notify } from 'notiflix/build/notiflix-notify-aio';

import { selectUsers } from 'redux/docs/docsSelectors';
import { loginUser } from 'redux/docs/docsOperations';
```

```
const initialState = {
  email: '',
  password: '',
};

const LoginForm = ({ setModalData }) => {
  const [fields, setFields] = useState(initialState);
  const dispatch = useDispatch();
  const users = useSelector(selectUsers);

  function checkUser(email, password) {
    const foundUser = users.find(
      user => user.email === email && user.password === password
    );

    return foundUser ? foundUser : null;
  }

  const handleChange = e => {
    const { name, value } = e.target;
    setFields(prevFields => ({
      ...prevFields,
      [name]: value,
    }));
  };

  const handleSubmit = e => {
    e.preventDefault();
    if (checkUser(fields.email, fields.password) !== null) {
      Notify.success('Success', { timeout: 2000 });
      dispatch(loginUser(checkUser(fields.email, fields.password)));
      setModalData(null);
    } else {
      Notify.failure('Error', { timeout: 2000 });
    }
  };

  return (
    <div className={s.loginBox}>
      <form onSubmit={handleSubmit}>
        <div className={s.userBox}>
```

```
          <input
            type="text"
            name="email"
            required
            value={fields.email}
            onChange={handleChange}
          />
          <label>Ел. Пошта</label>
        </div>
        <div className={s.userBox}>
          <input
            type="password"
            name="password"
            required
            value={fields.password}
            onChange={handleChange}
          />
          <label>Пароль</label>
        </div>
        <center>
          <button type="submit">
            Увійти
            <span></span>
          </button>
        </center>
      </form>
    </div>
  );
};

export default LoginForm;
```

Код реєстрації

```
import { useState } from 'react';
import s from './RegisterForm.module.scss';
import { useDispatch } from 'react-redux';
import { Notify } from 'notiflix/build/notiflix-notify-aio';


import { registerUser } from 'redux/docs/docsOperations';

const initialState = {
  email: '',
```

```
  name: '',
  password: '',
  status: '',
};

function validateFields(obj) {
  for (let key in obj) {
    if (obj.hasOwnProperty(key) && !obj[key]) {
      return false;
    }
  }
  return true;
}

const RegisterForm = ({ setModalData }) => {
  const [fields, setFields] = useState(initialState);
  const dispatch = useDispatch();

  const handleChange = e => {
    const { name, value } = e.target;
    setFields(prevFields => ({
      ...prevFields,
      [name]: value,
    }));
  };

  const handleSubmit = e => {
    e.preventDefault();
    !validateFields(fields)
      ? Notify.failure('Error', { timeout: 2000 })
      : dispatch(registerUser(fields)).then(
          Notify.success('Success', { timeout: 2000 }),
          setModalData(null)
        );
  };

  return (
    <div className={s.loginBox}>
      <form onSubmit={handleSubmit}>
        <div className={s.userBox}>
          <input
            type="text"
```

```
    name="email"
    required
    value={fields.email}
    onChange={handleChange}
   />
   <label>Ел. Пошта</label>
  </div>
  <div className={s.userBox}>
   <input
    type="text"
    name="name"
    required
    value={fields.name}
    onChange={handleChange}
   />
   <label>П.І.Б.</label>
  </div>
  <div className={s.userBox}>
   <input
    type="text"
    name="status"
    required
    value={fields.status}
    onChange={handleChange}
   />
   <label>Посада</label>
  </div>
  <div className={s.userBox}>
   <input
    type="password"
    name="password"
    required
    value={fields.password}
    onChange={handleChange}
   />
   <label>Пароль</label>
  </div>
  <center>
   <button type="submit">
    Зареєструватись
    <span></span>
   </button>
```

```
      </center>
    </form>
   </div>
  );
};

export default RegisterForm;
```

Код запитів на сервер

```
import axios from 'axios';

export const getDocsApi = async () => {
  return axios.get('http://localhost:5000/api/docs').then(res => res.data);
};

export const addDocApi = async body => {
  return axios
    .post('http://localhost:5000/api/docs', body, {
      headers: {
        'Content-Type': 'multipart/form-data; charset=utf-8',
      },
    })
    .then(res => res.data);
};

export const downloadApi = ({ user, name }) => {
  const encodedFileName = encodeURIComponent(`${user}\\${name}`);

  fetch(`http://localhost:5000/api/docs/download?filename=${encodedFileName}`,
{
    method: 'GET',
  })
    .then(response => {
      if (!response.ok) {
        throw new Error('Ошибка при загрузке файла');
      }
      return response.blob();
    })
    .then(blob => {
      const url = window.URL.createObjectURL(blob);
      const link = document.createElement('a');
      link.href = url;
```

```
      link.setAttribute('download', name);
      document.body.appendChild(link);
      link.click();
      document.body.removeChild(link);
    })
    .catch(error => {
      console.error(error);
    });
};

export const delApi = ({ user, name }) => {
  const encodedFileName = encodeURIComponent(`${user}\\${name}`);
  fetch(`http://localhost:5000/api/docs/download?filename=${encodedFileName}`,
{
    method: 'DELETE',
  })
    .then(response => {
      if (!response.ok) {
        throw new Error('Ошибка при удалении файла');
      }
    })
    .catch(error => {
      console.error(error);
    });
};
```

Код завантаження та скачування документів

```
import React, { useState } from 'react';
import { useDispatch, useSelector } from 'react-redux';
import { selectCurrentUser } from 'redux/docs/docsSelectors';
import s from './UploadFileComp.module.scss';
import { addDocApi } from 'services/docsApi';
import { Notify } from 'notiflix';
import { getDocs } from 'redux/docs/docsOperations';

const UploadFileComp = () => {
  const [selectedFile, setSelectedFile] = useState(null);
  const user = useSelector(selectCurrentUser);
  const dispatch = useDispatch();

  const handleFileChange = event => {
    const file = event.target.files[0];
```

```
    setSelectedFile(file);
  };

  const handleUpload = async () => {
    if (selectedFile) {
      const formData = new FormData();
      const encodedFileName = encodeURIComponent(selectedFile.name);
      formData.append('file', selectedFile, encodedFileName);
      const newFile = {
        date: new Date().getTime(),
        ...user,
      };
      formData.append('user', JSON.stringify(newFile));
      Notify.success('Файл завантажено', { timeout: 2000 });
      await addDocApi(formData);
      dispatch(getDocs());
    } else {
      console.log('Файл не обран');
    }
  };

  return (
    <div className={s.inputWrap}>
      <input
        type="file"
        onChange={handleFileChange}
        accept=".pdf, .docx, .doc, .txt, .xlsx, .xls"
      />
      <button onClick={handleUpload}>Завантажити</button>
    </div>
  );
};

export default UploadFileComp;
```

Серверна частина веб - додатку

Код шляхів доступу до сервера

```
const express = require("express");
const {
    getDocs,
    removeDoc,
    addDoc,
    sendDoc,
```

```
    delDoc,
} = require("../../controllers/docsController");

const multer = require("multer");
const upload = multer({ dest: "uploads/" });

const isValidId = require("../../middlewares/isValidId");

const router = express.Router();

router.get("/", getDocs);

router.get("/download", sendDoc);

router.delete("/download", delDoc);

router.post("/", upload.single("file"), addDoc);

router.delete("/:docId", isValidId, removeDoc);

module.exports = router;
```

Код функцій роботи з документами

Отримання всіх документів

```
const getDocs = async (req, res, next) => {
  try {
    const targetDirectory = path.join(__dirname, "..", "uploads");

    fs.readdir(targetDirectory, async (err, users) => {
      if (err) {
        console.error(err);
        return res.status(500).json({ error: "Помилка сервера" });
      }

      let fileList = [];

      // Обхід користувачів
      for (const user of users) {
        const userDirectory = path.join(targetDirectory, user);

        // Проверка, является ли элемент папкой
        if (fs.statSync(userDirectory).isDirectory()) {
```

```
            const files = fs.readdirSync(userDirectory);

            // Обхід файлів у папці користувача
            for (const fileName of files) {
              const filePath = path.join(userDirectory, fileName);
              const stats = fs.statSync(filePath);

              // Додавання інформації про файл до списку
              fileList.push({
                name: fileName,
                size: stats.size,
                created: stats.birthtime,
                user: user,
              });
            }
          }
        }

        res.json(fileList);
      });
    } catch (error) {
      next(error);
    }
};
```

Завантаження на сервер

```
const addDoc = async (req, res, next) => {
    try {
      if (!req.file) {
        return res.status(400).json({ error: "Файл не знайден" });
      }

      const { originalname, path: tempFilePath } = req.file;
      const { user } = req.body;

      const parsedUser = JSON.parse(user, null, 2);

      const originalnameDecoded = decodeURIComponent(originalname);

      const userDirectory = path.join(
        __dirname,
        "..",
        "uploads",
```

```
      parsedUser.name
    );

    if (!fs.existsSync(userDirectory)) {
      fs.mkdirSync(userDirectory);
    }

    const targetFilePath = path.join(userDirectory, originalnameDecoded);

    fs.copyFileSync(tempFilePath, targetFilePath);

    fs.unlinkSync(tempFilePath);

    res.status(201).json({ message: "successfully create" });
  } catch (error) {
    next(error);
  }
};
```

Отримання з серверу

```
const sendDoc = async (req, res, next) => {
  try {
    const encodedFilePath = decodeURIComponent(req.query.filename);
    const filePath = path.join(__dirname, "..", "uploads", encodedFilePath);

    res.status(200).sendFile(filePath);
  } catch (error) {
    next(error);
  }
};
```

Видалення файлів

```
const delDoc = async (req, res, next) => {
  try {
    const encodedFilePath = decodeURIComponent(req.query.filename);
    // console.log(encodedFilePath);

    const filePath = path.join(__dirname, "..", "uploads", encodedFilePath);

    fs.unlink(filePath, (error) => {
      if (error) {
        return res.status(500).json({ message: "Помилка сервера" });
```

```
        }

        res.status(200).json({ message: "Файл успішно видалено" });
      });
    } catch (error) {
      next(error);
    }
};
```