Хмельницький національний університет
Факультет інформаційних технологій
Кафедра інженерії програмного забезпечення

КВАЛІФІКАЦІЙНА РОБОТА

«Програмне забезпечення інформаційної системи кафедри для обліку студентів та викладачів»
Назва теми

Рівень вищої освіти Перший (бакалаврський)

Галузь знань 12 «Інформаційні технології»

Спеціальність 121 «Інженерія програмного забезпечення»

Освітня програма Освітньо-професійна програма «Інженерія програмного забезпечення»

Шифр КвРІПЗ.190139.01.16.ПЗ

Виконав студент IV курсу група ІПЗ-19-1 _____ (Підпис, дата) В. О. Прилуцька (Ініціали, прізвище)

Керівник канд. пед. наук, доцент _____ (Підпис, дата) Н. І. Праворська (Ініціали, прізвище)

Нормоконтролер канд. техн. наук, доцент _____ (Підпис, дата) М. О. Яшина (Ініціали, прізвище)

До захисту допускаю:
Завідувач кафедри інженерії програмного забезпечення _____ (Підпис) Л. П. Бедратюк (Ініціали, прізвище)

7 червня 2023 р.

Хмельницький 2023

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

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

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
Бедратюк І.П.
"2" 01 2023 р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

Прилуцькій Вікторії Олександрівні
Прізвище, ім'я, по батькові студента

1. Тема проєкту «Програмне забезпечення інформаційної системи кафедри для обліку студентів та викладачів»

Керівник проєкту Праворська Наталія Іванівна канд. пед. наук, доцент
Прізвище, ім'я, по батькові науковий ступінь, вчене звання

Затверджено наказом ректора університету від 01 03 2023 р. № ___
2. Строк подання студентом роботи на кафедру 01.06.2023 р.
3. Вихідні дані до роботи Опис інформації щодо контингенту студентів та персоналу кафедри
4. Зміст пояснювальної записки (перелік питань, які потрібно розробити) Аналіз систем для університетів, опис їх переваг та недоліків; вибір засобів розробки програмного продукту; розробка програмного забезпечення клієнтської та серверної частин вебзастосунку; підготовка характеристики програмного продукту для користувача, опис функціоналу
5. Перелік графічного матеріалу (із зазначенням обов'язкових креслень) три креслення формату А3

6. Консультанти розділів кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Яшина О.М., к.т.н, доцент	31.05.23	6.06.23
Антиплагіат	Гурман Т.В., к.т.н, доцент	4.06.2023	4.06.2023

7. Дата видачі завдання «02» січня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

Назва етапів (розділів) кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1. Збір матеріалу за темою кваліфікаційної роботи (КвР), дослідження предметної області, в якій планується використання програмного забезпечення (ПЗ), визначення здач та вимог, розробка технічного завдання	02.01–31.01 2023	
Проектування програмного забезпечення	01.02–28.02 2023	
3. Програмна реалізація програмного забезпечення	01.03–10.04 2023	
4. Тестування програмного забезпечення	11.04–30.04 2023	
5. Написання вступу, загальних висновків, оформлення переліку джерел та посилання на додатків. Оформлення пояснювальної записки КвР згідно вимог	01.05–25.05 2023	
6. Передзахист кваліфікаційної роботи	18.05.2023	
7. Gідготовка КвР для розміщення у репозитарії ХНУ; підготовка до захисту та захист КвР	01.06–05.06 2023	

Студент
Ініціали, прізвище _____ Прилуцька В. О.

Керівник роботи
Ініціали, прізвище _____ Праворська Н. І.

АНОТАЦІЯ

Тема кваліфікаційної роботи: «Програмне забезпечення інформаційної системи кафедри для обліку студентів та викладачів».

Автор роботи: Прилуцька Вікторія Олександрівна.

Керівник роботи: Праворська Наталія Іванівна.

Пояснювальна записка: 54 с., 40 рис., 13 дод., 40 джерел.

Графічна частина: 3 креслення формату А3.

Мета кваліфікаційної роботи полягає у розробці веб-застосунку для ведення обліку студентів та викладачів кафедри.

У кваліфікаційній роботі проведено аналіз предметної області та її інформаційного забезпечення, визначені вимоги до системи електронної комерції, розроблена загальна архітектура застосунку, спроектована структура бази даних та структура застосунку. Для розробки програмної системи використано мову програмування JavaScript, мову розмітки XML, Кластер MongoDB.

Впровадження розробленого програмного продукту дозволяє автоматизувати ведення та використання бази даних кафедри та значно полегшити працю навчального персоналу.

06.06
Дата

Підпис

ABSTRACT

The topic of the qualification work: «Software support of the information system of the department for accounting of students and teachers».

Author of the project: Prylutska Victoria Oleksandrivna.

Project manager: Nataliya Ivanovna Pravorska.

Explanatory note: 54 p., 40 fig., 13 add., 40 sources.

Graphic part: 3 drawings of A3 format.

The purpose of the qualification work is to develop a web application for keeping records of students and teachers of the department.

In the qualification work, an analysis of the subject area and its information support was carried out, the requirements for the electronic commerce system were determined, the general architecture of the application was developed, the structure of the database and the structure of the application were designed. The programming language JavaScript, the XML markup language, and the MongoDB cluster were used to develop the software system.

The implementation of the developed software product allows to automate the management and use of the database of the department and to significantly facilitate the work of the educational staff.

The topic of the qualification work: "Software support of the information system of the department for accounting of students and teachers."

06.06
date

signature

ВІДОМІСТЬ ДОКУМЕНТІВ

№ рядка	Формат	Позначення документа	Найменування документа	К-сть аркушів	№ екз.	Примітка
			Текстові документи	54		
1	А4	КвРІПЗ.190125.01.2.ПЗ	Пояснювальна записка			
2	А4		Завдання на кваліфіка-	1		
			ційну роботу			
3	А4		Анотація	1		
			Графічні документи			
4	А3	КвРІПЗ.190125.01.2.Е8	Діаграма варіантів	1		
			використання			
5	А3	КвРІПЗ.190125.01.2.Е8	Діаграма активності	1		
6	А3	КвРІПЗ.190125.01.2.Е8	Діаграма класів	1		

Змн.	Арк.	№ докум.	Підпис	Дата	КвРІПЗ.190139.01.16.ВД		
Виконав		Прилуцька В.О.		06.06	«Програмне забезпечення інформаційної системи кафедри для обліку студентів та викладачів»	Літ.	Арк. 1 / Аркушів 1
Керівник		Праворська Н.І.		06.06			
Рецензент						ХНУ, ІПЗ-19-1	
Н. контр.		Яшина О.М.		6.06			
Зав. Каф.		Бедратюк Л.П.		7.06	Відомість документів		

ЗМІСТ

ВСТУП

Протягом останніх двадцяти років процеси накопичення, обробки та використання знань постійно пришвидшується. Обсяг інформації та її обробка неабияк виросли у кожній сфері життєдіяльності, а саме в економічній, науковій, політичній та духовній. Вчені стверджують, що кожного десятиліття обсяг обробки інформації подвоюється. У результаті чого, завжди виникає потреба у використанні автоматизованих систем для зберігання, обробки та розповсюдження даних.

Відповідно до сучасних вимог вищих навчальних вузів, не можливо не відзначити, що ефективність їхньої роботи неабияк залежить від рівня оснащеності інформаційними засобами саме на базі автоматизованих комп'ютерних систем обліку.

Багато процесів сьогодення пов’язані зі зберіганням та обробкою інформації студентів і викладачів, які проводяться за допомогою офісного програмного забезпечення або безпосередньо на папері. У двох цих методів є деякі недоліки. Паперові документи достатньо знажують швидкість передачі й обробки інформації, а саме програмне забезпечення виконує лише роль первинної обробки та вводу даних.

Актуальність теми полягає у тому, що на ринку існує невелика кількість застосунків, заточених саме під роботу університету та, якщо вони і є, то не безкоштовні. Для розв’язання задачі було вирішено розробити безкоштовний застосунок, який значно полегшить роботу кафедри.

Мета кваліфікаційної роботи полягає у розробці веб-застосунку для ведення обліку студентів та викладачів кафедри.

Практична значимість даної роботи полягає в тому, що викладачі зможуть легко керувати даними, так як вся введена інформація буде зберігатися у створеній інформаційній системі. Саме ж формування зведеної документації буде відбуватися за запитом адміністратора автоматично.

Завдання (цілі) кваліфікаційної роботи:

– вивчити роботу кафедри, а саме – методику ведення інформації;

– провести аналіз методів повідомлення студентів та персоналу про головні новини вузу, заходи та події;

– розглянути та вивчити концепцію об’єктно-орієнтовного програмування для реалізації програми, дослідити деякі технології програмування та проектування програмних систем;

– реалізувати клієнт-серверний застосунок і провести його тестування та налагодження.

В даній роботі необхідно створити клієнт-серверний застосунок з обліку студентів та персоналу. Всю інформацію, яка до цього зберігалася в паперовому вигляді, потрібно структурувати і розташувати в базі даних та виводити її і редагувати за допомогою колекцій у самій програмі. Це дозволить зменшити час на пошук потрібної інформації про студентів та викладачів і збільшить ступінь її захисту, що також не менш важливо.

1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Змістовний аналіз предметної області, її структурні та функціональні особливості

Предметною областю даної розробки є діяльність кафедри, яка виконує такі функції:

– перегляд та редагування списку студентів групи;

– перегляд та редагування журналу відвідування студентів;

– перегляд та редагування списку предметів;

– перегляд та редагування розкладу занять;

– повідомлення про заходи університету, розклад екзаменів;

– наділення кожного студента групи певними обов’язками;

– розподіл студентів по підгрупам;

– перегляд та редагування розкладу дзвінків;

– налаштування інформації про групу (редагування назви групи, ПІП куратора, назви факультету та університету).

В інтернеті є електронні журнали контролю обліку академічної групи, проте усі вони зв’язані з якимось конкретним вузом чи школою та є загальною інформаційною системою. Вирішено розробити застосунок, який повністю задовольнить усі потреби і не орієнтований на певний навчальний заклад, а є загальним для усіх ВНЗ України.

Користувачами є викладачі кафедри, які заповнюватимуть інформацію під час навчального року. Застосунок допоможе користувачам швидко отримати відомості про студентів груп та викладачів, про їх відвідування занять, про розклад занять, іспитів та індивідуальний навчальний план студента. Він може бути використаний клієнтом для більш зручної роботи з необхідною їм інформацією. У програмі мають бути реалізовані такі функції: додавання записів; видалення записів; редагування (оновлення) даних; пошук даних у програмі; друк даних.

Кваліфікаційна робота створюється на основі поставленої задачі, яка називається «Програмне забезпечення інформаційної системи кафедри для обліку студентів та викладачів».

При розв'язанні задачі розподіл дій між працівниками кафедри та технічними засобами за різних ситуацій такий. Головними функціями працівників кафедри є: введення контролю правильності введення первинних даних; складання, перевірка і підписання первинних документів; друкування вихідних результатних документів, контроль і підписування їх; робота із програмою (додавання, вилучення, корегування записів у застосунку), пошук даних. Функції технічних засобів: робота із програмою «Інформаційна система обліку студентів та викладачів», обробка даних та друкування вихідної інформації згідно з прикладними програмами, виведення повідомлень про будь-які помилки на екран.

У результаті обробки вхідних даних, формується вихідна інформація, представлена у вигляді: машинограм тобто відомостей, реєстрів, звітів, які використовуються для виконання функцій, а також файлів на машинних носіях, що формуються для розв'язання конкретної задачі в наступному місяці або розв'язання інших задач з управління об'єктом.

Мета кваліфікаційної роботи: розробити програмний продукт для обліку студентів та викладачів кафедри. Для досягнення даної цілі потрібно вирішити наступні задачі:

1. Розглянути існуючі аналоги.

2. Провести огляд наявних рішень, що дають можливість обліку студентів та викладачів кафедри. Вказати наявні переваги та недоліки наявних програмних рішень, а також довести актуальність створення даного програмного продукту для розв’язання поставлених завдань.

3. Обрати засоби для розробки сервісу, а саме: мову програмування, технічні інструменти, які спростять саму розробку; запропонувати можливі альтернативи, пояснити вибір.

4. Зробити архітектурні рішення з приводу структури роботи, призначити користувацький інтерфейс із самим застосунком, описати

функціонал роботи програми.

5. Здійснити тестування веб-застосунку та продемонструвати користувачеві приклади використання функцій даного веб-застосунку.

6. Детально описати інтерфейс користувача та навести приклади запитів до веб-застосунку.

Розроблений сервіс повинен надати можливість для запуску на стороні користувача, мати звичайний користувацький інтерфейс, можливість глобального пошуку та сортування по категоріям.

Система повинна запуститися у вигляді веб-застосунку та надавати користувачеві зручний інтерфейс користувача з можливістю переходу по директоріям, завантажувати файли будь-якого типу, створювати нові директорії, видаляти існуючі файли та директорії, зручний пошук, сортування та відображення.

1.2 Аналіз наявного програмно-технічного забезпечення предметної області

Сервіс управління навчанням, тобто система дистанційної освіти (Learning management system) — сервіс адміністрування навчальною діяльністю, яка спеціалізується на розробці та поширенні навчальних інтернет-матеріалів із забезпеченням спільного доступу.

Сучасні LMS вирізняються зручним та доступним для розуміння інтерфейсом, функціональними можливостями, що дозволяє вивести онлайн-навчання на новий якісний рівень. Системи дистанційного навчання дають можливість організовувати навчальні процеси та відстежувати здобутки учнів з допомогою інтернет-курсів або віртуальних уроків, які є доступні будь-де та будь-коли при наявності Інтернету. Усі навчальні документи при цьому зберігаються, їх зручно переглядати у відповідності до цілі навчання та сфери діяльності організації чи компанії.

LMS доволі часто порівнюють з віртуальним класом, в якому навчаються студенти з різних точок світу, а також відслідковується їх успішність.

ERP (Enterprise Resource Planning) — «планування ресурсів підприємства». Дане програмне забезпечення об'єднує фінанси, поставки, операції, звітність, виробництво, кадри та дозволяє маніпулювати ними. Більшість таких компаній у своєму розпорядженні мають системи управління фінансами та бізнес-процесами, проте можливості цього програмного забезпечення обмежуються звичайною рутиною або ж планами розвитку у майбутньому.

Такі системи спеціалізуються у процесі навчання або ведення підприємницької звітності, не дозволяючи зобразити всю потрібну інформацію про студентів та викладачів навчального закладу, тому облік університетів та їхніх організаційних підрозділів здебільшого ведеться завдяки офісному програмному забезпеченню, також на папері або ж за допомогою спеціальних програмних рішень. Далі розглянемо програмні рішення, що можуть задовільняти задачі обліку для університетів.

Пакет програм Microsoft Office 365

У складі Microsoft Office 365 містяться такі програми, як Word та Excel, хмарне середовище One Drive, крім того служби Exchange Online, Teams та Share Point Online.

Даний пакет є продуктом компанії Microsoft, а також фізично знаходиться в центрах Microsoft, які є доступними лише довіреним спеціалістам. До того ж, компанія забезпечує різнобічну підтримку та обслуговування. До прикладу, якщо знижується продуктивність робочого процесу, необхідно звернутися в службу підтримки компанії Microsoft. Відгук скоріше за все затримається, ймовірно, не вдасться усунути проблему терміново. До речі, коли застосовувати оновлення та вводити нові функції, вирішує тільки Microsoft. Це значить, що Microsoft Office 365 буде змінюватися, що не залежить від вашого бажання, на відміну від локального рішення, яке може не змінюватися протягом років.

Обмежений контроль реалізації може спровокувати зниження продуктивності, а це неабиякий недолік в роботі Office 365. Microsoft надає безперервну роботу і гарантує фінансові зобов'язання у рамках SLA. Якщо ж час безперервної роботи за місяць нижче 99,8%, компанія зобов'язується компенсувати втрати за допомогою сервісних кредитів. Звучить непогано, але практика доказує, що недоліки у продуктивності все-таки можливі.

Безпека – головне питання, якому Microsoft надає неабияку увагу. Проте саме через питання з безпекою Microsoft переносять перехід в так звану «хмару». Компанія забезпечує різнорівневий захист Office 365 матеріальними ресурсами, вбудованими функціями і клієнтськими елементами управління, проте компанії досить часто ризикують втратити дані, які збережені в «хмарі». Сумніви щодо безпеки даних – одна з проблем Microsoft Office 365.

Слідує відмітити, що використовування Microsoft Office 365 є не безкоштовним та має ціну пакету програм, що надаються. На рисунку 1.1 показано вартість пакетів Microsoft Office 365.

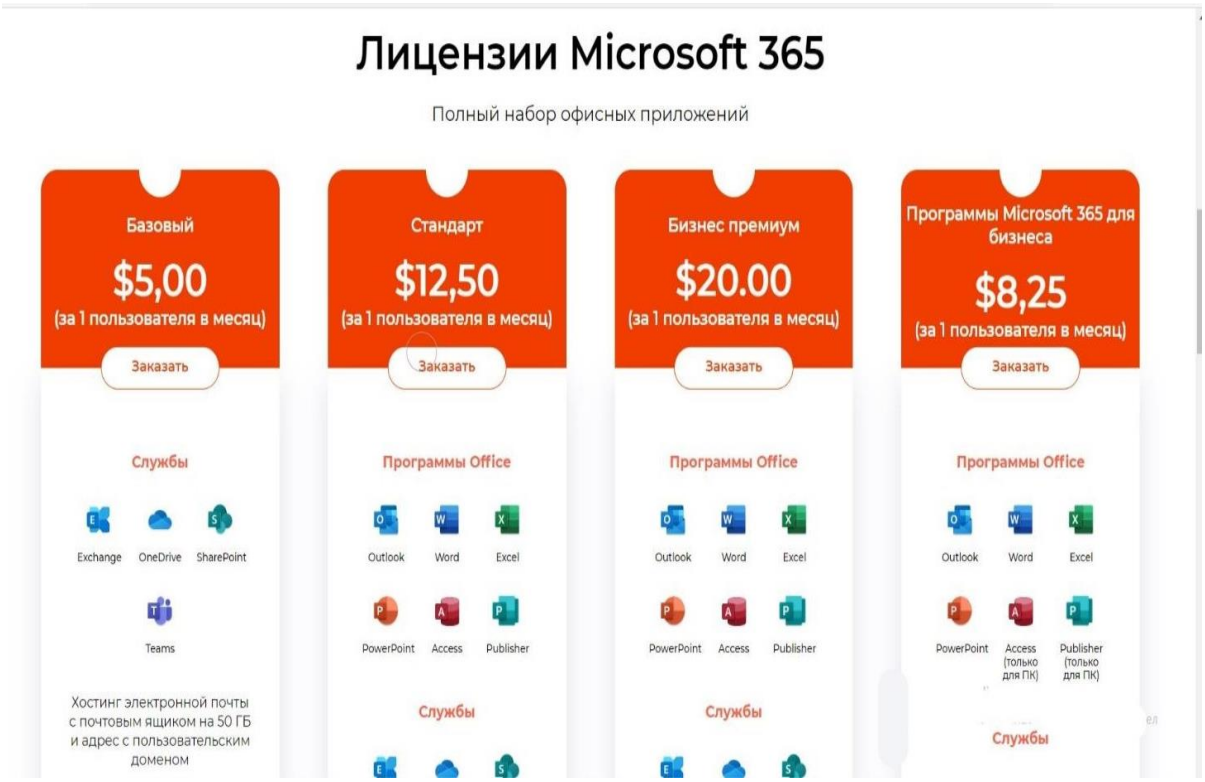


Рисунок 1.1 – Вартість пакету Microsoft Office 365

З рисунка 1.1 виходить, що користувач має можливість вибрати один з дійсних пакетів, які різняться між собою допоміжними наявними службами та програмами компанії. Оплата за пакет служб та програм є помісячною, це означає, що кожного місяця необхідно оплачувати його вартість.

Office 365 не є зручною платформою для налаштування. Звісно, вона містить різноманітні застосунки і служби, з яких компанії можуть вибрати найбільш доречні. Але якщо потрібно налаштувати конкретний застосунок, то питання може бути важко вирішеним. До того ж, оновлення від Microsoft можуть впливати на встановлені налаштування.

В Office 365 введення обліку відбувається в програмах, що входять до пакету, наприклад Word і Excel. На малюнку 1.2 наведено приклад шаблону обліку.

Office 365 вміщує багато шаблонів для ведення обліку. Дані можуть зберігатися на серверах Microsoft, що має бути реалізовано в застосунку.

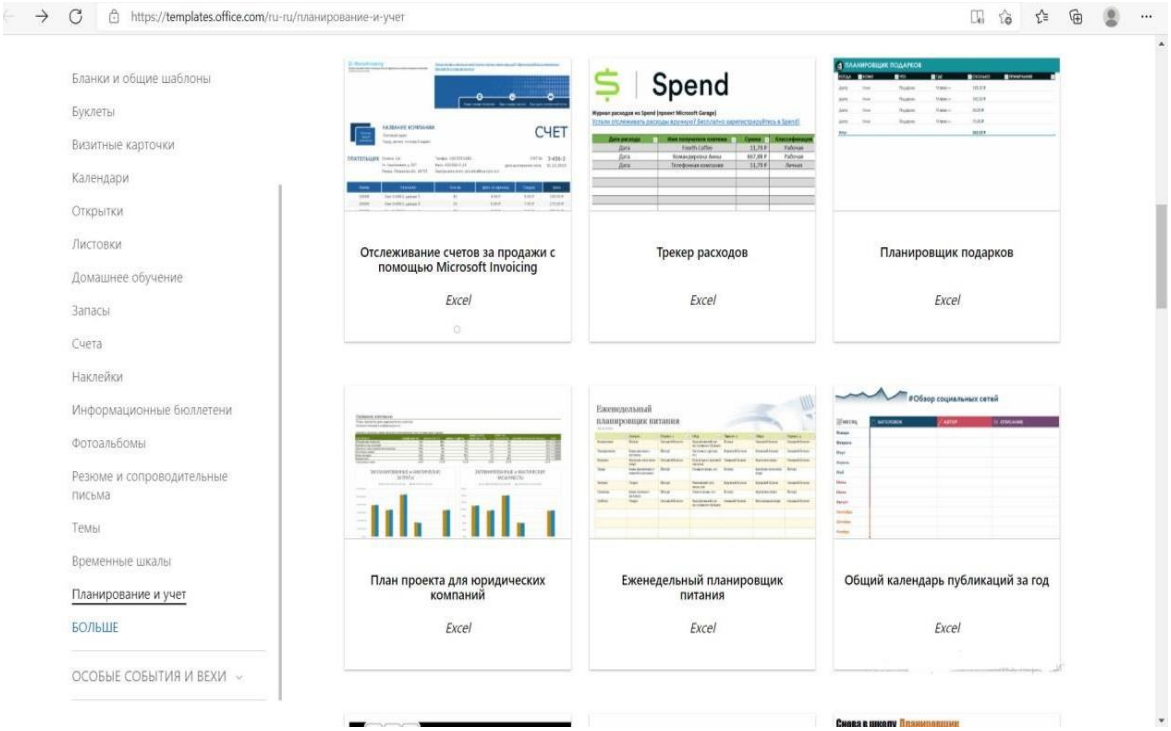


Рисунок 1.2 – Приклад шаблонів Office 365

Однак система не дає створювати та змінювати дані в режимі онлайн, тому варто розробити нову програму, щоб мати можливість створювати та змінювати файли з даними у режимі онлайн та завантажувати їх.

Система управління 1С: Підприємство

Програма «1С:Підприємство» (також пишеться як 1С, С або ES) є продуктом компанії «1С» для автоматизації підприємства. Всі сервісні рішення для розробників створюються на єдиній основі технологічної платформи і діє на загальних принципах.

Система програм «1С: Підприємство» містить технологічну платформу (ядро) і розроблені на її основі прикладні рішення (конфігурації). Така архітектура системи принесла високу популярність, Тому що це забезпечує доступність прикладних рішень, організаційність та її гнучкість, короткі цикли впровадження, хорошу продуктивність, масштабованість від одного робочого місця до десятків тисяч на «хмарних» сервісних моделях і на мобільних пристроях.

Для забезпечення високої продуктивності системи та її відповідності зростаючим вимогам ринку 1С розробляє технологічну платформу «1С. Підприємство 8», реалізує комплекс передових технологій, інструментів та інноваційних можливостей.

Фірма масово виробляє прикладні рішення для автоматизації типових бухгалтерських та адміністративних завдань на комерційних підприємствах і бюджетних організаціях у світі бізнесу. Усі програмні продукти поєднують у собі використання стандартних рішень (загальних для всіх або кількох програм) з максимальним урахуванням деталей завдань певної галузі чи виду діяльності самого підприємства.

Також слід відзначити конфігурацію платформи 1С, яка є у відкритому доступі, що є перевагою. Для користувачів це означає, що за потреби вони можуть легко внести необхідні корективи та вдосконалення для вирішення конкретних проблем у межах свого підприємства за допомогою відповідних спеціалістів 1С.

До того ж, легко виправляти помилки програми та помилки, що виникають, які теж трапляються. Це є беззаперечним плюсом «1С:Підприємство», як для користувачів, так і для фахівців з обслуговування системи 1С. Слід зазначити, що присутні системні вимоги до встановлення та

використання.

При цьому «1С:Підприємство» більше орієнтоване на комерційний сектор і для його використання необхідно знати або хоча б бути ознайомленим з мовою програмування 1С, тому створювати новий програмний продукт буде цікаво.

Система обліку «Кадри Плюс Україна»

Кадри Плюс Україна – програма, яка створена AnDeeSoft компанією для автоматизації обліку і документообігу у відділі кадрів. Застосунок погоджує створення звітів, заяв, наказів, також відстежувати співробітників і проводити облік робочих днів. Є можливість систематизувати групи і посади, використовувати класифікатор посад, вести бази персоналу, що включає особисту інформацію співробітників, їхніх сімей і дійсних соціальних пільг. Дані в базі можна редагувати. Також підтримуються бази даних працівників, які працюють як за трудовим договором, так і за цивільним правом.

Ціна даного програмного продукту є одноразовою, у порівнянні з іншими сервісами.

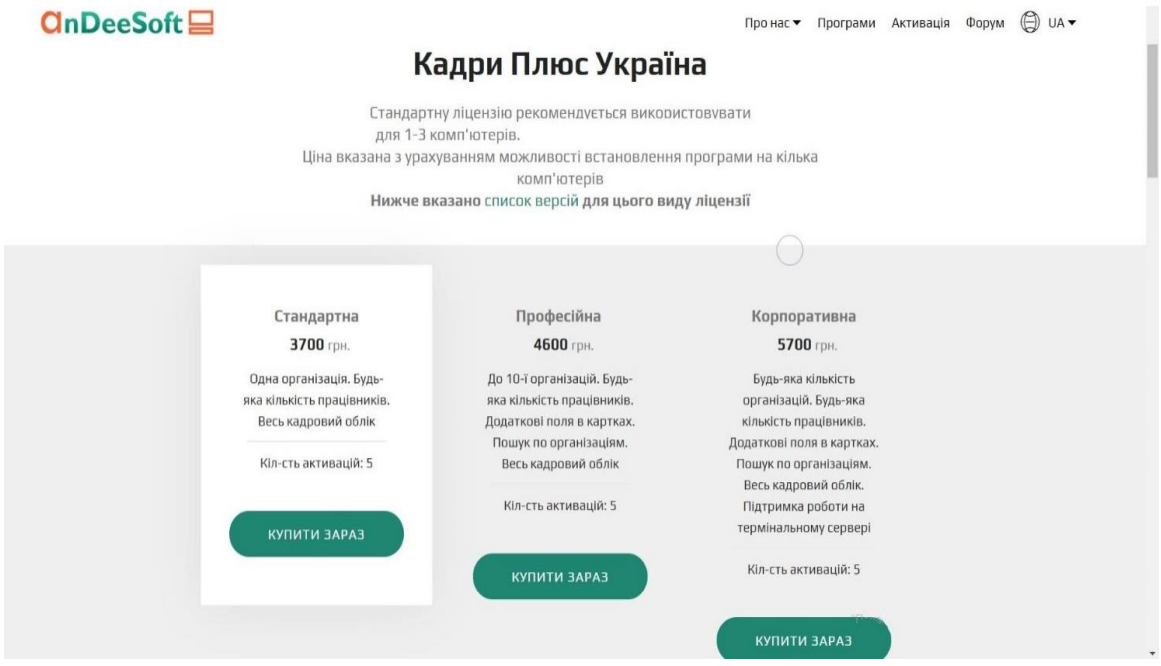


Рисунок 1.3 – Ціна ліцензії програми Кадри Плюс Україна

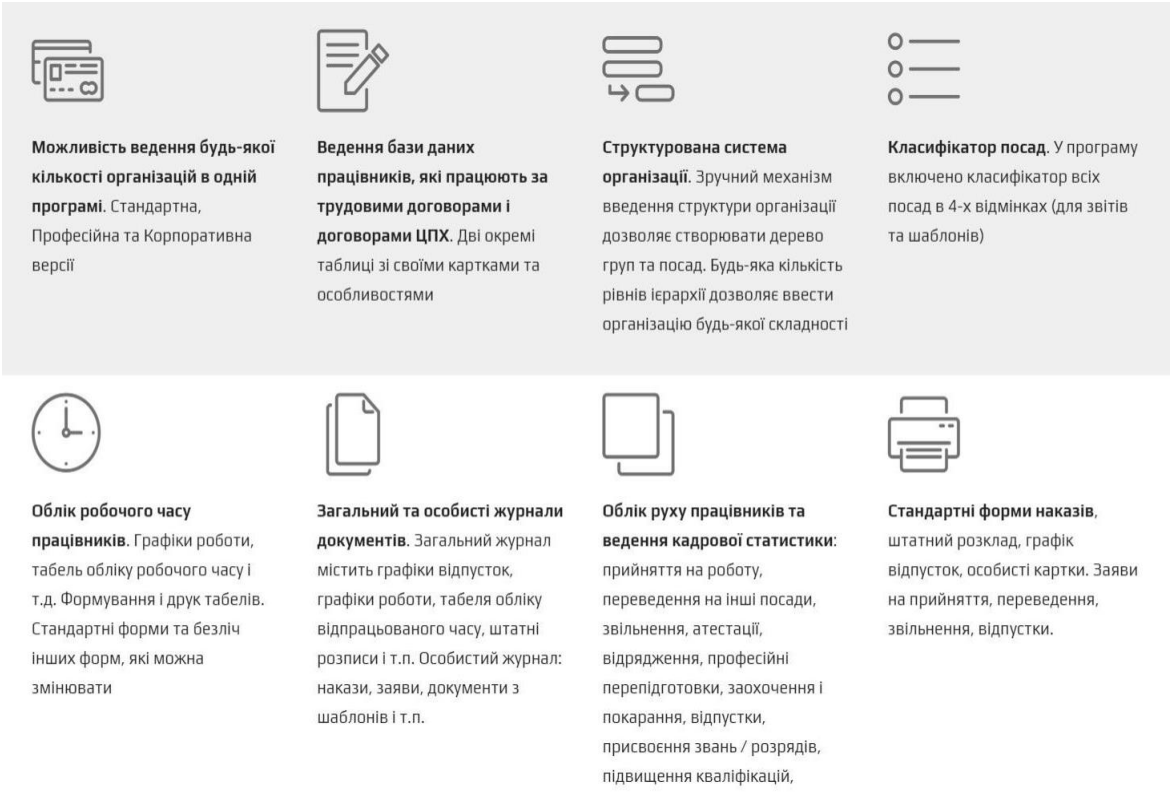


Рисунок 1.4 – Функціонал програми Кадри Плюс Україна

На рис. 1.3, пропонує свій програмний продукт представлена компанія Andeesoft в різних версіях та з різною кількістю функціоналу, при тому, що програма спеціалізується на обліку.

На рис. 1.4 продемонстровано функціональність сервісу Кадри Плюс Україна. Програма дає можливість проводити облік для різних організацій на структурованій основі, з веденням кадрової статистики та класифікатором посад, при цьому ж сама програма не дозволяє користувачеві працювати в режимі онлайн, в такому випадку варто створити новий програмний продукт.

1.3 Визначення вимог до програмного забезпечення та розробка технічного завдання

Вимоги до програмного продукту:

- загальне оформлення сайта;
- перелік курсів на головній сторінці;
- виведення інфомації про лекторів та студентів;

– виведення дисциплін;

– можливість створювати та редагувати інформацію;

– можливість пошуку предметів, студентів та викладачів.

Візуальне моделювання з використанням нотації UML можна візуалізувати, як процес переходу від найбільш абстрактної та загальної концептуальної моделі вихідної системи підприємства до логічної, а потім фізичної моделі, відповідну програмній системі. Для досягнення цілей насамперед будується модель у вигляді діаграми варіантів використання, яка описує функціональну роль системи або, іншими словами, те, що система мусить робити в рамках своєї діяльності.

Діаграма варіантів використання (use case diagram) – діаграма, яка зображує взаємодію між актором і варіантами використання.

Діаграма варіантів використання — це початкове концептуальне представлення або концептуальна модель системи під час проектування та розробки. Розробка діаграм варіантів використання має такі цілі:

– визначення загальних границь та контекст предметної області на початковому етапі проектування системи;

– формулювання загальних вимог до функціональних дій проектованої системи;

– розробка вихідної концептуальної моделі системи для її подальшої деталізації у формі фізичних та логічних моделей;

– підготовка вихідної документації для взаємодії між розробниками системи з її замовниками та користувачами.

Розглянемо діаграму варіантів використання, яка побудована відповідно до даної програмної системи у середовищі моделювання діаграм Edraw Max.

Актор – це користувач, має зв’язки типу простої асоціації і зв’язаний з варіантами використання. Опишемо сценарій виконання – певна послідовність дій, яка описує дії акторів і поведінку модельованої системи в формі звичайного тексту, та розділимо його на окремі розділи. Перший розділ описує загальну поведінку модельованої системи та кожен варіант використання.

Діаграма використання подана на рисунку 1.5.

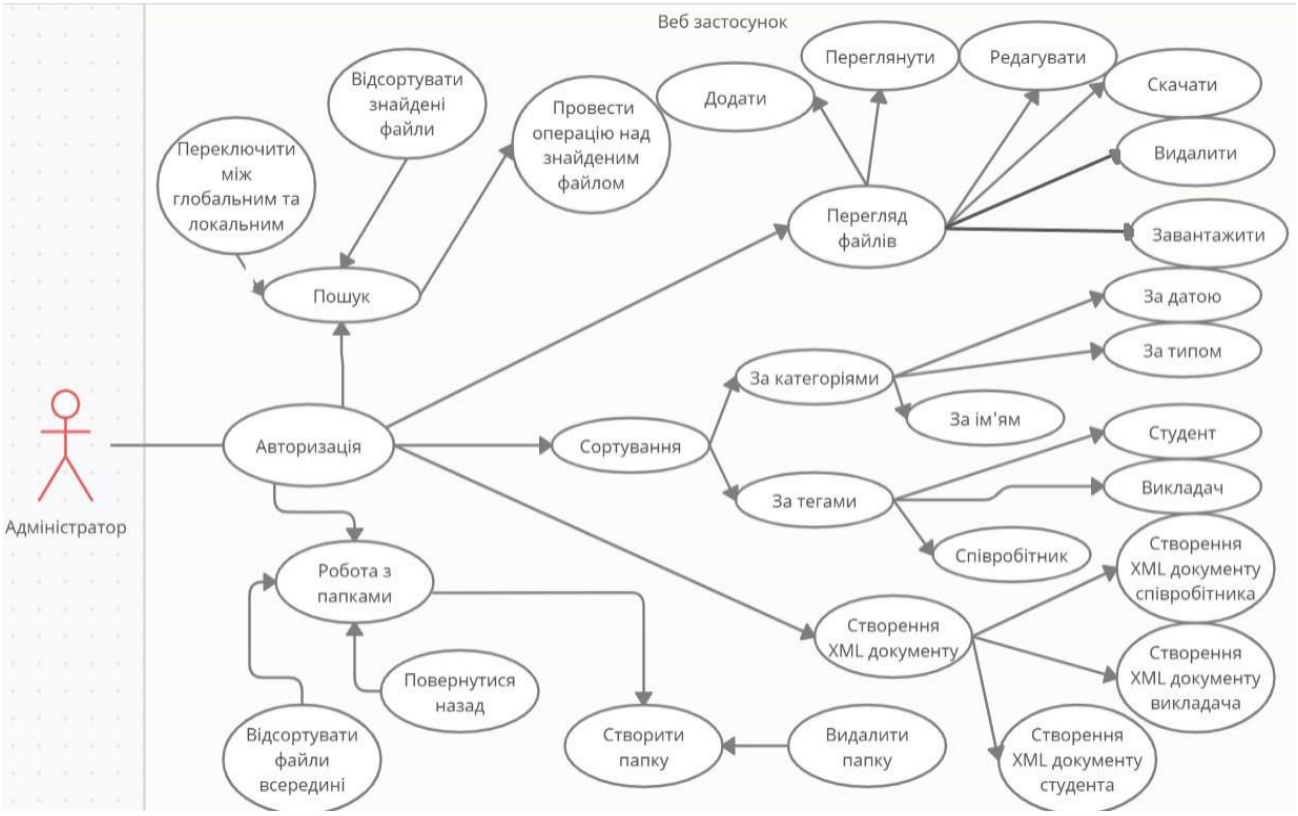


Рисунок 1.5– Діаграма варіантів використання

1.4 Висновок. Постановка задачі

У першому розділі було проведено детальний аналіз предметної області, її структурні та функціональні можливості. Було визначено, що предметною областю є робота кафедри та було встановлено функції, які вона виконує, і які у свою чергу, повинні бути реалізовані у майбутньому.

Відбувся аналіз систем, які мають схожу компетенцію. Для цього, взято три програми. В ході аналізу, були розглянути переваги та недоліки, що допоможе розробити потрібний застосунок.

Також, було визначено вимоги до майбутнього програмного забезпечення, що дає можливість створити максимально функціональний застосунок.

2 ПРОЕКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Вибір типу архітектури та шаблонів використання

Патерни, також відомі як шаблони, використовуються для багатьох завдань архітектурної розробки. Вони дають можливість швидко вирішувати завдання, що позитивно впливає на час розробки та спрощує кількість процесів. Патерн характеризує певну задачу, і якщо вона знову виникає в певному контексті, його застосування не дозволяє «винаходити колесо», а використовувати готове рішення.

Шаблон — це конкретний опис взаємодії різних елементів і рівнів, які складають архітектуру програмного рішення, пристосований до контексту конкретного завдання та вимог. Оскільки модель дозволяє визначати ключові об’єкти в структурі та використовувати їх знову і знову, процес побудови архітектури спрощується.

Кожен шаблон має свої переваги та недоліки. Нижче наведено декілька прикладів вже готових архітектурних шаблонів та їх опис роботи:

Багаторівнева патерн — його принцип полягає в тому, що вся прикладна система поділена на рівні. У той же час, кожен рівень може викликати лише наступний рівень нижче нього. Це дозволяє змінювати певні компоненти програми, не впливаючи на інші області. Проте це ускладнює структуру архітектури та робить її доволі навантаженою, що безпосередньо впливає на продуктивність.

Проміжна модель. Якщо застосунок складається з великої кількості модулів, то взаємодія виглядає доволі складно і заплутано. Для полегшення цього створено посередника, який дозволяє модулям встановлювати просту взаємодію. Функціональна сумісність компонентів підвищується відразу, але посередник є слабкою ланкою в системі. Якщо щось не вдається, вся система може перестати працювати.

Контролер перегляду моделі. Природа моделі полягає в окремому інтерфейсі для даних. Це дозволяє змінити його, не порушуючи вказівки системи додатків. Він використовується в програмах, які потребують частої зміни інтерфейсу.

Клієнт-серверна модель. Архітектури програмного забезпечення, що використовують модель клієнт-сервер, досить поширені в додатках, яким потрібно обмежити доступ споживачів до певних ресурсів. Такий підхід дозволяє програмі розвиватися та робить систему доступною та легкою для розуміння.

Шаблон «Заводський метод». Дозволяє додавати додаткові об'єкти різних типів. У разі використовування стандартних методів додавання, код буде розростатися, що знизить швидкість роботи програми, крім того, компоненти будуть розкидані по всьому коду. Ця модель полегшує додавання нових об’єктів і підтримує незалежність системи.

В архітектурі розробки програмного забезпечення використовується багато шаблонів. Це не значить, що вони всі працюють в одній структурі і дублюються. За основу береться тільки концепція, а сама архітектура будується індивідуально, залежно від контексту.

2.2 Опис декомпозиції та залежностей

Діаграми активності є окремим випадком діаграм стану. Вони допомагають реалізувати в мові UML функції синхронного і процедурного управління, за рахунок реалізації внутрішніх дій і операцій. Головним напрямком використання діаграм активності є візуалізація характеристик реалізації операцій класу, коли потрібно представити алгоритми їх реалізації. Крім того, кожен стан може бути реалізацією операції певного класу або його частини, що дає можливість користуватися діаграмами діяльності, щоб описати реакції на події в системі.

Діаграми діяльності показують логіку або покроковий перехід від однієї до іншої діяльності, зосереджуючись при цьому на результаті. Вихідні результати можуть змінити стан системи або повернути значення. Діаграми діяльності призначені для моделювання поведінки систем, хоча на цій діаграмі чітко час не вказано.

Для опису роботи даної програмної системи було прийнято рішення скласти діаграму діяльності. У діаграму включено різні компоненти, які називаються діяльностями. Опишемо її покроково.

Для відкриття cfqne необхідно спочатку перевірити підключення до серверу. Якщо підключення встановлено виконати перевірку наявності групи у базі даних. Якщо ж підключення не встановлено – припинити роботу програми.

Для успішного входу до cfqne необхідно перевірити наявність групи у базі даних. Якщо групу знайдено – виконується перевірка доступних тижнів навчання. Якщо групу не знайдено – користувачу пропонується відновити акаунт по паролю або перевірочному коду, який прийшов на пошту під час реєстрації групи. Якщо відновлення пройшло успішно – виконуємо вхід до журналу і перевіряємо наявність навчальних тижнів. Якщо відновлення акаунту не виконано – користувачу пропонується зареєструвати свою групу. Під час реєстрування групи, перевіряються дані на коректність вводу. Якщо дані введено правильно – пропонується заповнити список студентів, після цього пропонується заповнити розклад занять і потім виконується перевірка наявності тижнів ведення журналу. В разі невірно введених даних, користувачу пропонується зареєструвати групу ще раз.

Після перевірки наявності тижнів ведення журналу, програма, якщо тижні ведення журналу існують, приступає до перевірки наявності створеного журналу на поточний день тижня. Якщо списку тижнів не існує, користувачу пропонується створити новий тиждень.

Після перевірки наявності журналу на поточний день поточного тижня програма формує структуру журналу на поточний день тижня, у разі існування відповідних записів у БД, програма заповнює журнал даними з БД. Якщо

журнал на поточний день ще не створено, користувачу пропонується створити журнал на поточний день.

Діаграма активності представлена на рисунку 2.1

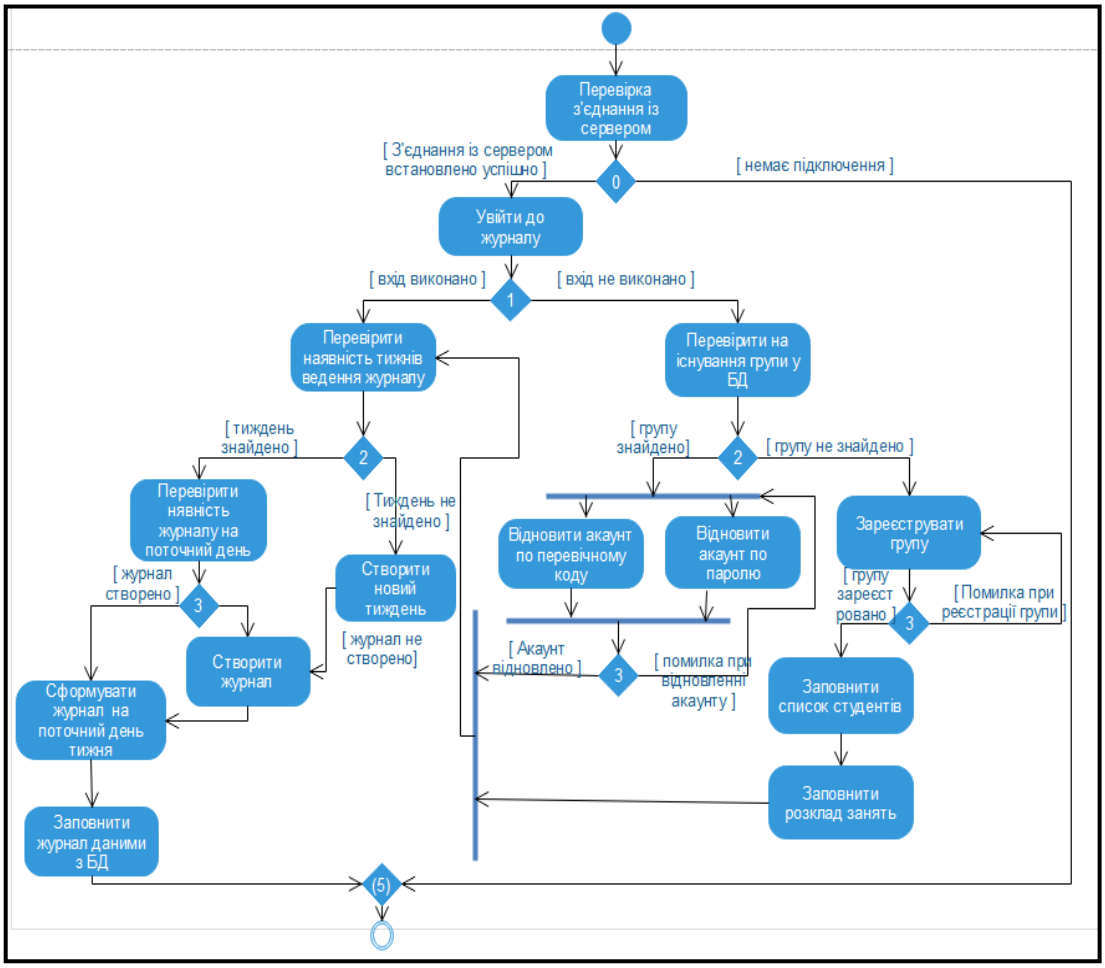


Рисунок 2.1 – Діаграма активності (діяльності)

Діаграма послідовностей показує послідовність, в якій об'єкти в процесі взаємодії обмінюються повідомленнями. Схеми послідовностей можна використовувати в різних цілях і на різних рівнях деталізації програми. Як правило, схема послідовностей створюється в таких двох випадках:

– якщо використовується схема варіантів використання, яка узагальнює відомості про користувачів системи і їх мети, можна створити схеми послідовностей, щоб описати, як основні компоненти системи взаємодіють для досягнення мети будь-якого виду використання.

– якщо визначені повідомлення, що надходять в інтерфейс компонента, можна створити схеми послідовностей, щоб описати, як внутрішні частини компонента взаємодіють для досягнення результату,
необхідного для кожного вхідного повідомлення.

Створення схем послідовностей має кілька переваг:

– можна легко побачити, як завдання розподіляються між компонентами.

– є можливість визначити шаблони взаємодії, що ускладнюють оновлення програми.

Діаграми послідовності показують лише об’єкти, які безпосередньо взаємодіють і не показують ймовірні статичні асоціації з іншими об’єктами. Для діаграм послідовності критичним моментом є динаміка взаємодії об’єктів у часі. У той же час можна сказати, що діаграма послідовності є двовимірною. Зліва направо у вигляді вертикальних ліній. Кожна представляє лінію життя свого об'єкта, який бере участь у взаємодії. Кожен об'єкт графічно представлений прямокутником і розташований у верхній частині його життєвого циклу.

Всередині прямокутника (об’єкт) записуються ім'я об'єкта та класу, які розділені двокрапкою. Тоді запис підкреслюється, що є ознакою об'єкта, який, як вже відомо, характеризує екземпляр класу.

Для даного програмного проекту було прийнято рішення створититри діаграми послідовностей: діаграма реєстрації групи, відновлення акаунту та діаграма редагування журналу відвідування.

Опишемо послідовності у діаграмі реєстрації групи. Користувач вводить назву групи та адресу своєї електронної пошти. Якщо дані введено коректно та якщо присутнє інтернет-з’єднання, відправляється лист на пошту із перевірочним кодом, який потім користувач вводить для підтвердження акаунту і, якщо воучер співпадає із тим, що прийшов на пошту – після встановлення з’єднання із сервером відбувається реєстрація групи у базу даних. При помилках користувачу виводиться відповідне повідомлення на екран. Дана діаграма подана на рисунку 2.2.

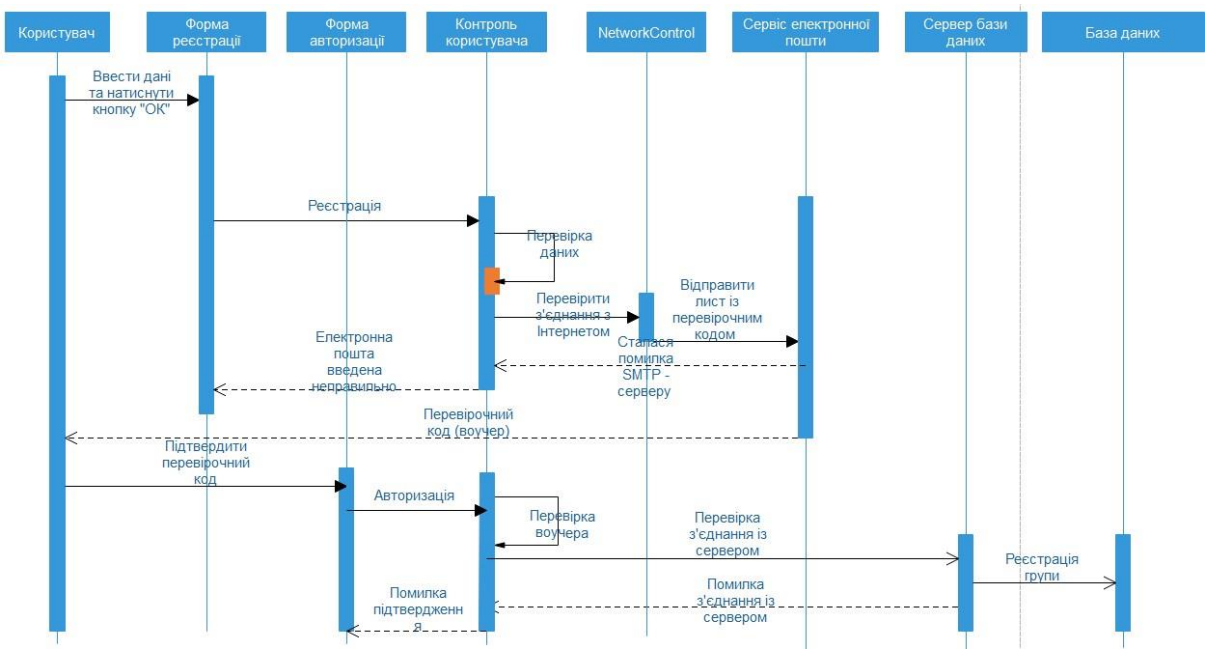


Рисунок 2.2 – Діаграма послідовності (Реєстрація групи)

Наступна діаграма послідовності – Відновлення акаунту. Користувач вводить дані у вікні відновлення акаунту. Якщо такі дані знайдені у базі даних – виконується вхід до системи, якщо ні – відбувається повернення на попередню форму. Діаграма подана на рисунку 2.3.

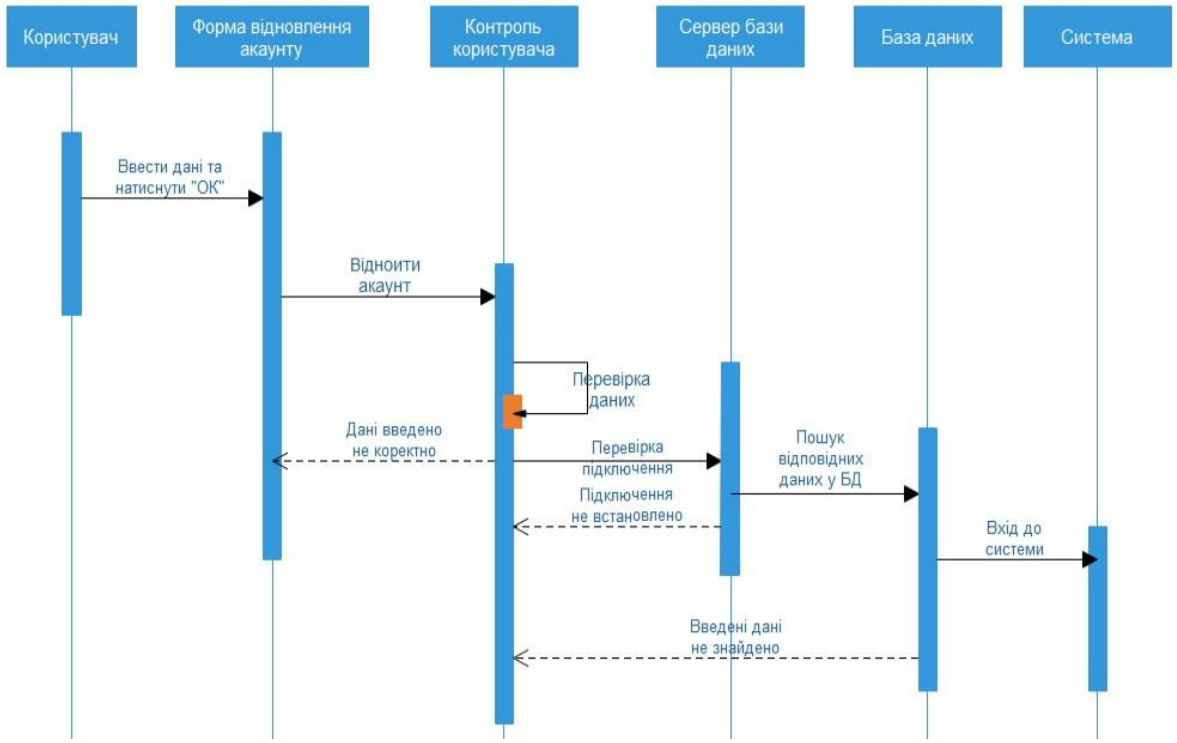


Рисунок 2.3 – Діаграма послідовності (Відновлення акаунту)

Наступна діаграма – послідовність створення журналу. Користувач, якщо використовує пароль для входу, вводить його, та, якщо пароль правильний – відбувається вхід у систему. Далі користувач на формі «Тижні навчання» створює новий тиждень. Потім користувач зможе створити журнал на вибраний день і заповнити його. Діаграма подана на рисунку 2.4.

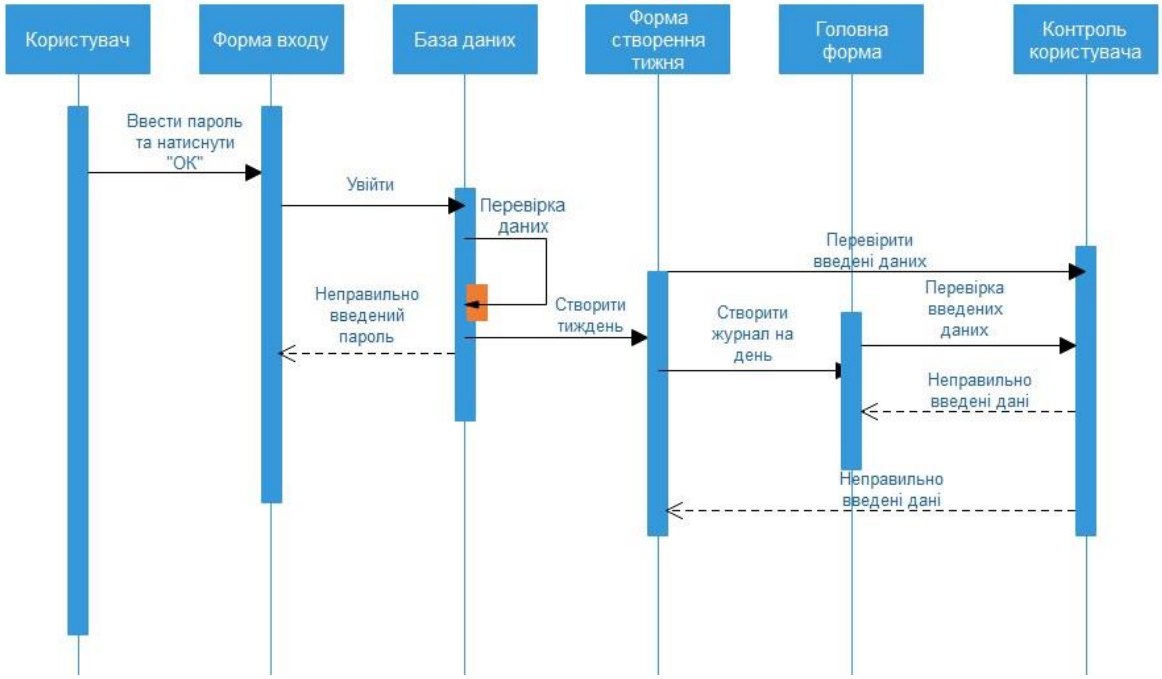


Рисунок 2.4 – Діаграма послідовності (Створення журналу)

2.3 Опис інтерфейсів

Центральне місце в ООАП займає розробка логічної моделі системи у вигляді діаграми класів.

Нотація UML дає багато можливостей для відображення додаткової інформації (абстрактні операції та класи, шаблони, узагальнені та приватні методи, детальні інтерфейси, параметризовані класи).

Крім того, графіку можна використовувати для зв’язків і її специфічних властивостей, таких як зв’язки набору, коли класи можуть бути частиною

інших класів.

Діаграма класів може бути використана для різних цілей, таких як:

1. Опис структури класів системи та їх взаємодії.

2. Визначення інтерфейсів між класами та підсистемами.

3. Показ взаємозв’язків між класами, таких як агрегація, композиція, спадкування, асоціація.

4. Визначення поведінки класів та їх методів.

Діаграма класів складається з класів, інтерфейсів, абстрактних класів, зв’язків між класами та анотацій. Класи можуть мати атрибути та методи. Зв’язки між класами показують взаємозв’язки між ними, такі як асоціацію, композицію, агрегацію та залежність.

Діаграма класів є важливим інструментом для проектування системи та розуміння її структури. Вона дозволяє описати взаємозв’язки між класами та їх поведінку, що допомагає розробникам зрозуміти, як система працює та як її можна покращити.

Діаграма класів – це такий граф, вершинами якого є елементи типу «категорія», які пов'язані між собою різними типами структурних відносин. Потрібно зазаначити, що діаграми класів також можуть містити інтерфейси, пакети, відносини і навіть окремі екземпляри, такі як об'єкти і зв'язку. Коли говорять про дану діаграму, мають на увазі статичну структурну модель проектованої системи.

Для програмної системи запропоновано скласти діаграму класів та умовно розділити її на 5 логічних частин: ієрархія класів бізнес-логіки – головні класи програми, що описують об’єктну логіку; класи моделі представлення – призначені для заповнення колекцій, типи яких (колекцій) базуються на класах бізнес-логіки та редагування записів у базі даних, тобто реалізує двосторонню комунікацію з представленням; класи для роботи з базою даних – введені з метою спростити звернення до бази даних із класів моделі представлення; класи-форми – безпосередньо форми для відображення графічних компонентів користувачу на екран для змоги

взаємодіяти із цими компонентами; класи-контролери – створюють функціональний каркас системи, що дозволяє відділити певні обчислення та контроль користувача від загальної логіки програми. Діаграми розміщені на рисунках 2.6 – 2.9.

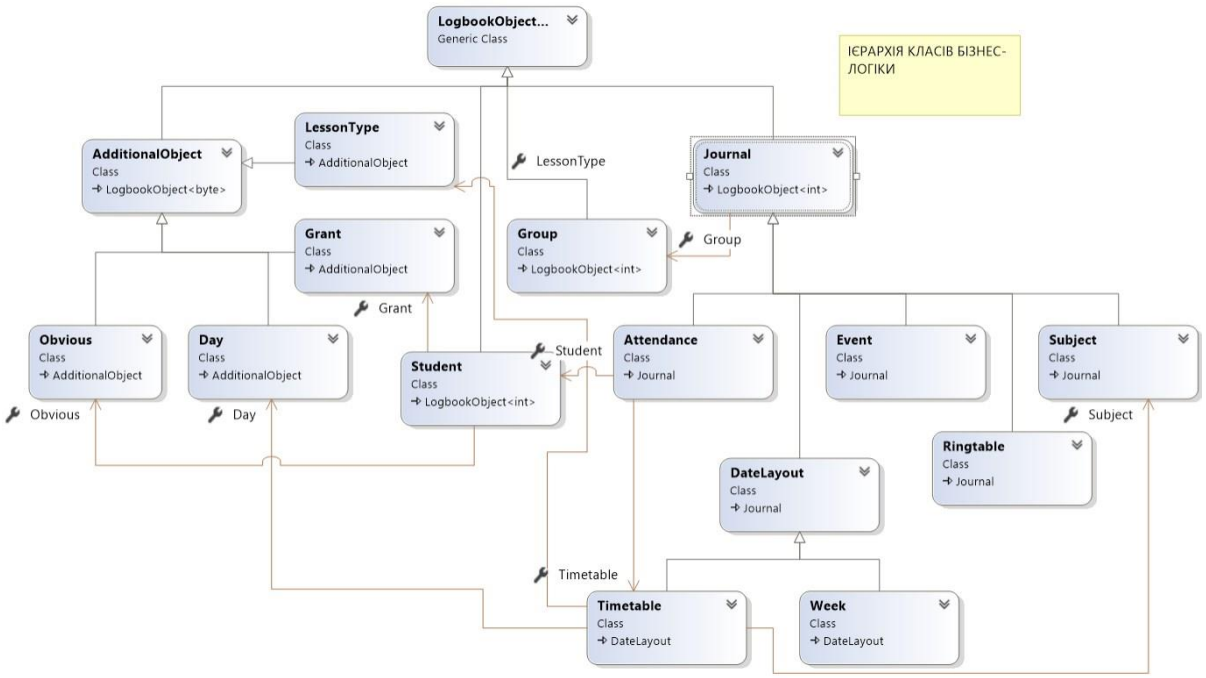


Рисунок 2.5 – Діаграма класів (Бізнес-логіка)

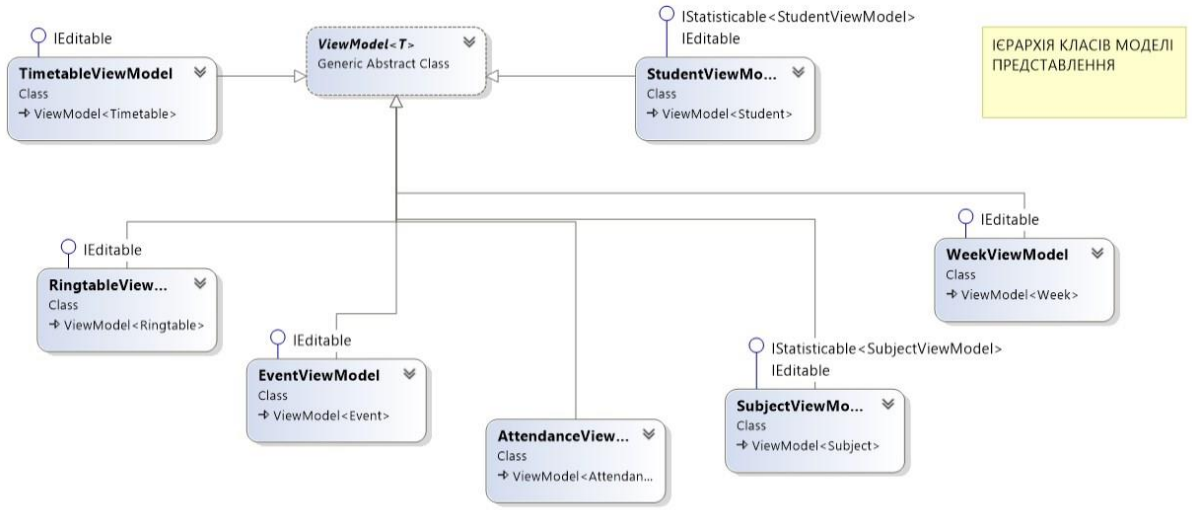


Рисунок 2.6 – Діаграма класів (Модель представлення)

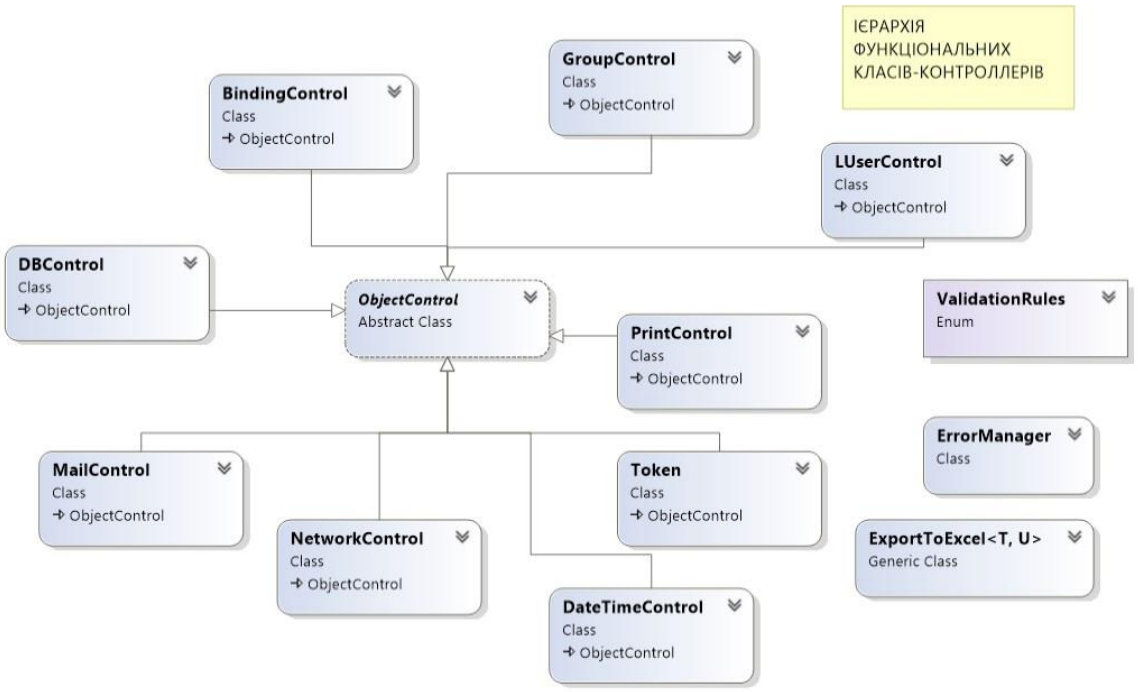


Рисунок 2.7 – Діаграма класів (Функціональні класи)

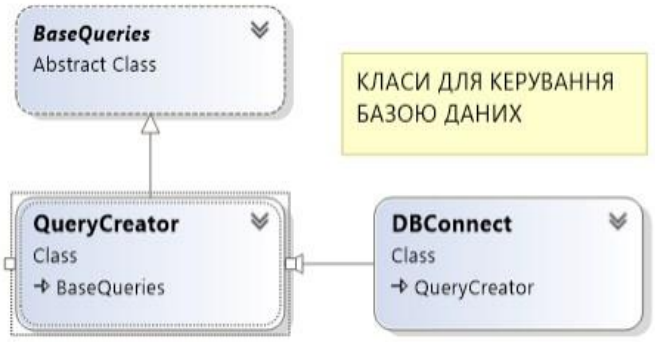


Рисунок 2.8 – Діаграма класів (Керування базою даних)

2.4 Аналіз та вибір технологій і методів реалізації застосунку

Після проведеного аналізу, було вирішено розробити обліково-інформаційну систему для студентів і викладачів на клієнт-серверній архітектурі з використанням мови програмування JavaScript, стеку MERN — JavaScript-стеку, призначену для спрощення самого процесу розробки. MERN включає в собі чотири компоненти з відкритим вихідним кодом, а саме: MongoDB, Express, React і Node.js:

- MongoDB: кросплатформенна БД документів;
- Express: базова платформа веб-застосунку;
- React: бібліотека JavaScript для створення користувацького інтерфейсу;
- Node.js: кросплатформенне середовище розробки на основі JavaScript.
- додаткові компоненти та утиліти для зручнішої розробки та вірного функціонування.

На рисунку 2.9 показано взаємодію цих компонентів між собою.

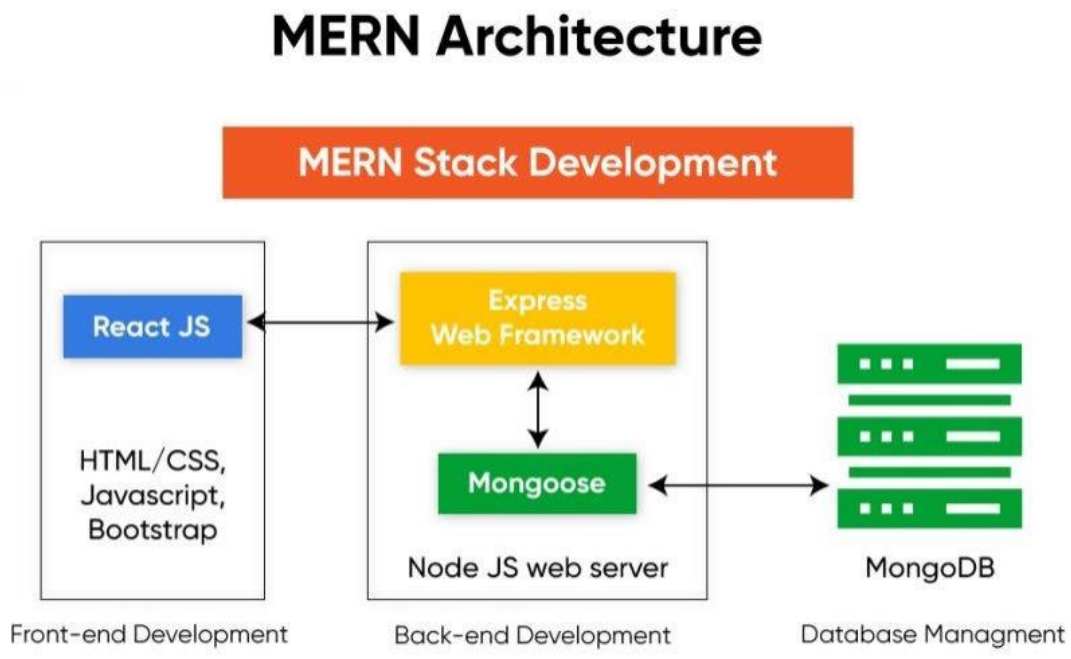


Рисунок 2.9 – Взаємодія компонентів стеку MERN між собою

Як показано на рисунку 2.9, стек має трирівневу архітектуру та дозволяє використовувати одну мову програмування для клієнтської та серверної частин.

Внизу наведено обґрунтування використання цих інструментів.

Node.js є програмною платформою, що робить мову JavaScript мовою загального призначення. Вона дозволяє розробникам створювати не лише фронт-енд частину веб-додатків, але й бек-ендову частину. Node.js виконує функції веб-серверу і може зв'язуватися з зовнішніми бібліотеками та викликати команди з коду. Це дозволяє створювати повноцінні веб-додатки та застосунки без безпосереднього використання браузера. Node.js є потужним інструментом для розробки веб-додатків, оскільки він дозволяє розробникам створювати швидкі, ефективні та масштабовані додатки з використанням мови JavaScript.

Неблокуюча модель введення-виведення є головною перевагою Node.js перед іншими середовищами розробки, оскільки вона дозволяє керувати великою кількістю з'єднань та забезпечувати ефективний розподіл ресурсів. Крім того, Node.js працює асинхронно, що дозволяє створювати пріоритетні черги, що забезпечує більш ефективну роботу з багатокористувацькими додатками та великою кількістю запитів. За рахунок цього Node.js дозволяє створювати швидкі та ефективні веб-додатки з високим рівнем масштабованості та надійності. Крім того, на відміну від аналогічних мов, розробник має можливість використовувати мову безпосередньо на клієнті та на сервері. Якщо програміст має досвід роботи з JavaScript, йому простіше буде вивчити «надбудову», аніж зовсім іншу технологію. На рисунку 2.10 показано статистику використання середовищ розробки в 2020 році.

Як можна побачити на рисунку 2.10, завдяки своїм перевагам, середовище розробки Node.js стало найпопулярнішим в 2020 році.

Неблокуюча модель введення-виведення дозволяє керувати великою кількістю з'єднань та розподіляти ресурси ефективно. Крім того, Node.js є популярним середовищем розробки, що дозволяє розробникам швидко вивчити його та створити робочий прототип системи за короткий час.

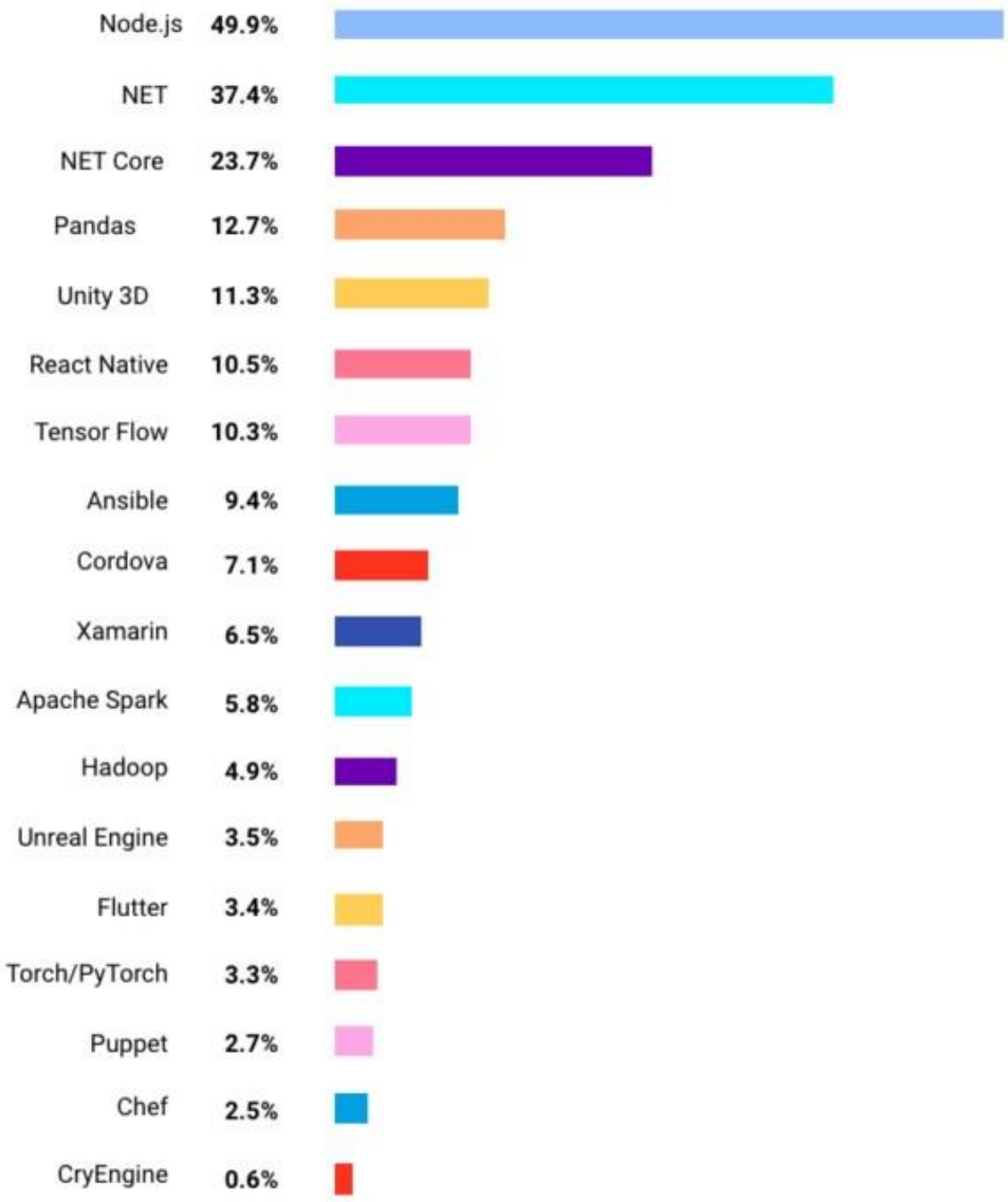


Рисунок 2.10 – Статистика використань середовищ розробки

Технологія досить швидко розвивається, адже, над нею працюють тисячі спеціалістів по всьому світу.

React.js — бібліотека JavaScript з відкритим кодом, який відкрила компанія Facebook у 2013 році. Цей фреймворк ідеально підходить для того, щоб створювати великі веб-застосунки, де дані можуть часто змінюватися.

На відміну від аналогів, React.js легкий у вивченні. Він вчиться без труднощів через простоту його синтаксису. Розробникам достатньо буде згадати свої навички написання HTML. Немає потреби в поглибленому вивченні TypeScript, наприклад, як у випадку з Angular.

По результатам опитування, React.js є найпопулярнішим фреймворком на сьогоднішній день. На рисунку 2.11 показано результати опитування в 2020 році.

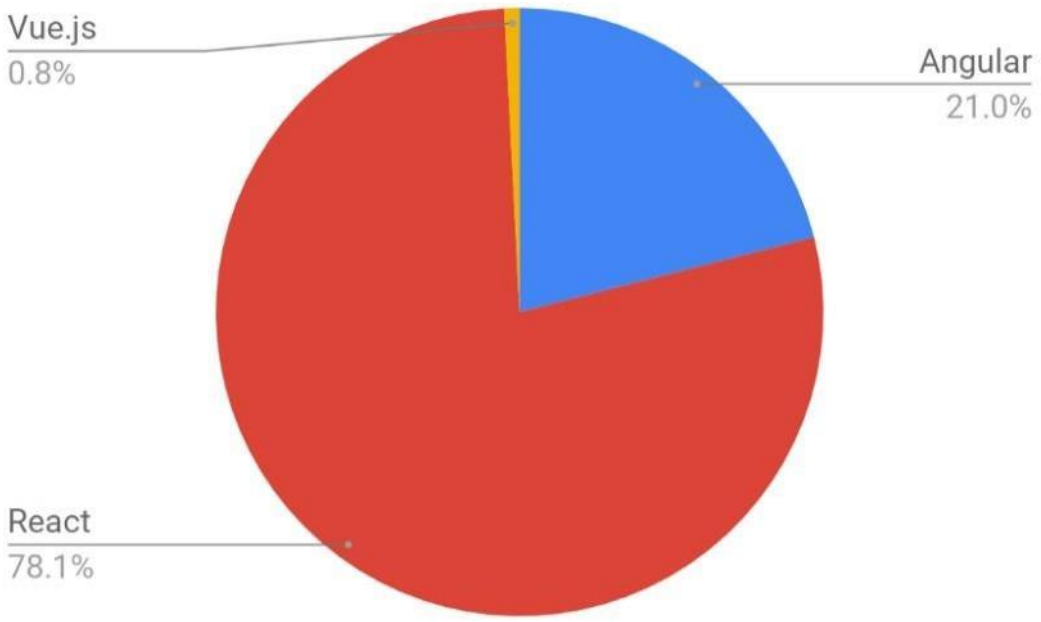


Рисунок 2.11 – Популярність фреймворків для фронтенду

що з 60 000 респондентів 78,1% розробників JavaScript обирають фреймворк React.js для розробки програм клієнтської частини.

До того ж, React.js має хороший рівень гнучкості та віртуальну DOM (document object model), яка дозволяє маніпулювати документами форматів HTML, XHTML або XML в дереві, що краще всього підходить веб-браузерам у аналізі деяких елементів веб-застосунку. В поєднанні з ES6/7 React.js легко працює з сильними навантаженнями.

Ще однією перевагою React.js є MVVM (Model-View-ViewModel), що дозволяє розробникам працювати окремо в одній частині, використовуючи той самий набір даних.

Слід зазначити, що зв’язування даних у React.js від більшого до меншого. Це визначає такий потік даних, де дочірні елементи не можуть впливати на вихідні дані. React.js – це бібліотека JavaScript з відкритим кодом, яка отримує багато оновлень і вдосконалень на основі відгуків розробників по всьому світу.

На останок, React.js є легким, тому що дані, які виконуються на стороні користувача, можуть бути легко відображені на стороні серверу одночасно. React.js краще співпрацює з іншими компонентами, ніж інші аналогічні компоненти, тому було вирішено вибрати саме його.

Будь-яка реляційна база даних має стандартну схему, яка показує число таблиць, а також, зв’язок між ними. В MongoDB ж немає такої схеми зі зв’язком між таблицями.

На відміну від інших аналогів, MongoDB побудована на колекціях різноманітних документів. Число полів, вміст і розмір цих документів можуть відрізнятися. MongoDB підтримує динамічні запити на основі документів, а дані зберігаються як документи JSON.

Все це дозволяє працювати з документами в рази швидше, ніж аналогічні документи з чіткою структурою кожного об'єкта. На рисунку 2.12 показано, як візуалізувати дані колекції документів за допомогою Mongo Atlas.

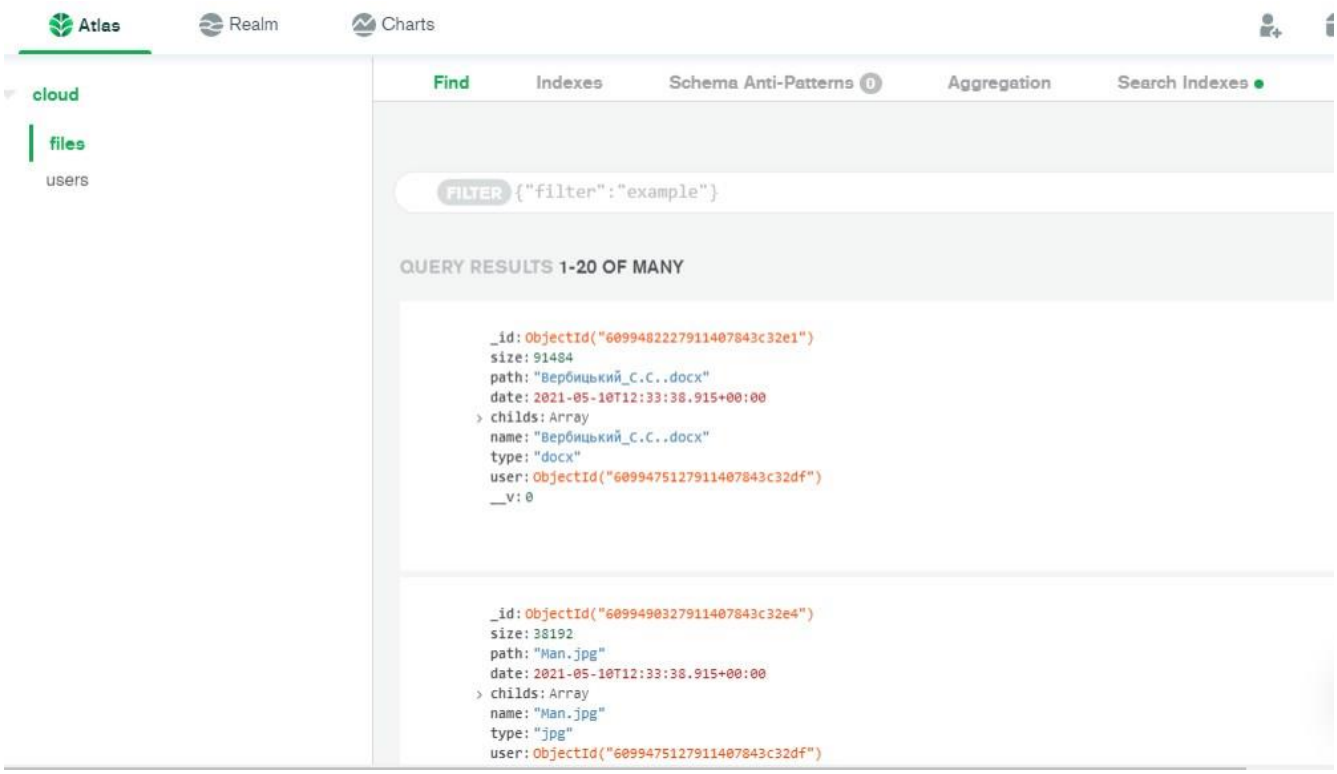


Рисунок 2.12 – Перегляд інформації про колекції даних

Як показано на рисунку 2.12, можна переглянути інформацію про колекції документів в справжній час. Для зберігання використаних в момент часу даних використовується внутрішня пам’ять, яка дозволяє отримати швидший доступ.

Крім того, слід зазначити, що MongoDB легко масштабується, вона не має Запити JOIN складні, і розробникам не потрібне відображення об’єктів застосування в об’єктах бази даних, тому воно підходить для розгортання програмного забезпечення Продукт кращий за інші аналоги.

Основна функція цього фреймворку заключається в тому, що для Express є своєрідним невелика кількість основних функцій.

Express.js дозволяє створювати потужні структури проекту, приклад показано на рисунку 2.13

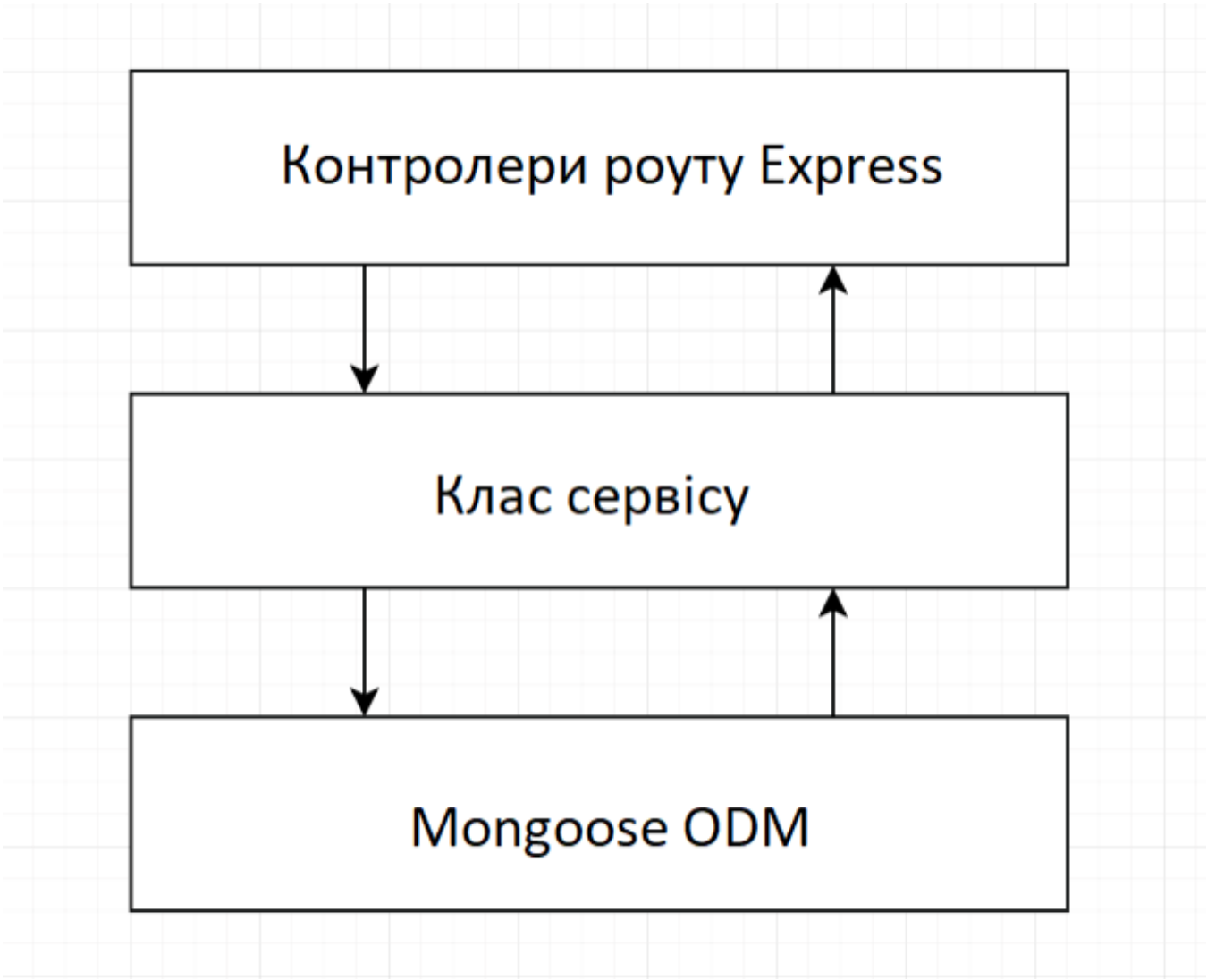


Рисунок 2.13 – Функціональність Express.js в проекті

На рисунку 2.13 продемонстровано використання Express.js в комбінації з другими компонентами. Ця структура проекту є досить надійною та з нею легко працювати, якщо є досвід у розробкці на мові JavaScript.

Всі інші необхідні розробнику функції слід вибирати за рахунок зовнішніх модулів. По суті, Express у чистому вигляді — це сервер, і він може не мати жодного модуля. Завдяки такому мінімалізму розробник на самому початку отримує інструмент, який легкий і швидкий у використанні. Він може поглиблюватися та вдосконалюватися. Express практично не має аналогів і при роботі з іншими комплектуючими замінити його практично неможливо.

В ході виконання проєкту також використовувалися для зручної розробки та правильного функціонування наступні засоби:

– бібліотека Axios; для подання асинхронних запитів. Вона містить підтримку запитів, отримання відповідей від серверу, їх трансформація та конвертація в JSON, що є автоматичною;

– Bcrypt – адаптивна криптографічна хеш-функція для формування ключа, яка використовується для захищення паролів;

– Бібліотека Redux є способом управління станом застосунку. Вона дозволяє зберігати стан застосунку в одній зберігальній області, що дозволяє зробити його більш прогнозованим та легким для розуміння;

– Mongoose є ODM-бібліотекою (Object Data Modelling) для роботи з MongoDB. Вона дозволяє зіставляти об'єкти класів та документи колекцій з бази даних MongoDB. Mongoose дозволяє використовувати схеми для опису даних та їх валідації, а також забезпечує можливість використання різних типів запитів до бази даних.

– Nodemon – утиліта інтерфейсу командної строки (CLI), що відстежує файлову систему програми Node та зразу перезавантажує процес, якщо є така необхідність.

2.5 Висновки до розділу

Розроблюване програмне забезпечення використовує великі апартні ресурси. Створення системи повинно бути правильно спроектовано. Було досліджено основну концепцію технологій і методів реалізації застосунку, оцінено потужність інструменту.

У результаті проектування було створено основні діаграми, на основі яких будується програмне забезпечення: діаграми класів, діаграми активності та послідовності.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Створення серверної частини

Для простоти та зручності у розробці, щоб не доводилося багато разів прописувати одну і ту ж саму інформацію, використовується файл конфігурації, який зручно і швидко змінює дані для застосунку. На рисунку 3.1 показано файл конфігурації.

server > config > {} default.json > secretKey

```
{
  "serverPort" : 3000,
  "dbUrl": "mongodb+srv://admin:admin@cluster0.qkcau.mongodb.net/cloud?retryWrites=true&w=majority",
  "secretKey": "mern-secket-key",
  "filePath": "F:\\MERN\\server\\files",
  "staticPath": "F:\\MERN\\server\\static"
}
```

Рисунок 3.1 – Приклад реалізації файлу конфігурації

На рисунку 3.1, показана інформація про директорії файлів, секретний ключ шифрування даних, що передаються, порт серверу, а також посилання на безпосереднє підключення до MongoDB, яке знаходиться у json файлі конфігурації default.json.

Необідно створити маршрут користувачу для авторизації в систему. На рисунку 3.2 показано приклад реалізації маршруту авторизації для застосунку.

```
router.post('/login',
    async (req, res) => {
        try {
            const {email, password} = req.body
            const user = await User.findOne({email})
            if (!user) {
                return res.status(404).json({message: "User not found"})
            }
            const isPassValid = bcrypt.compareSync(password, user.password)
            if (!isPassValid) {
                return res.status(400).json({message: "Invalid password"})
            }
            const token = jwt.sign({id: user.id}, config.get("secretKey"), {expiresIn: "3h"})
            return res.json({
                token,
                user: {
                    id: user.id,
                    email: user.email,
                    diskSpace: user.diskSpace,
                    usedSpace: user.usedSpace,
                    avatar: user.avatar
                }
            })
        } catch (e) {
            console.log(e)
            res.send({message: "Server error"})
        }
    })
```

Рисунок 3.2 – Маршрут авторизації

Як видно на рисунку 3.2, процес авторизації користувача є асинхронним і виконується методом post. Коли дані вводяться користувачем, здійснюється перевірка на присутність цього користувача в системі шляхом авторизації. Якщо перевірка не виконана, користувач отримає відповідне повідомлення «користувача не знайдено, тобто «user is not found». Потім виконується перевірка зашифрованого пароля за допомогою криптографічної функції Bcrypt, яка розшифровує пароль на сервері та порівнює їх. Якщо пароль не збігається, користувач повинен побачити повідомлення «Невірний пароль», або «Invalid password». Після успішної авторизації користувач отримає токен jwt, дійсний протягом 3 годин, це означає, що через 3 години користувач автоматично вийде зі свого профілю у веб-застосунку. Користувач отримує доступ до веб-застосунку та всієї його інформації. Користувач отримає повідомлення «Помилка сервера», якщо сталася помилка під час авторизації.

Файловий контролер є основою серверної програми і користувачів. Модуль контролера містить клас FileController, який має асинхронні методи у роботі з файлами. Методи цього класу працюють з файлами асинхронно, тобто коли є відповідь від сервера на клієнта. На рисунку 3.3 зображено реалізацію сортування на серверній частині.

server > controllers > JS fileController.js > FileController > getFiles

```
31    async getFiles(req, res) {
32        try {
33            const {sort} = req.query
34            let files
35            switch (sort) {
36                case 'name':
37                    files = await File.find({user: req.user.id, parent: req.query.parent}).sort({name:1})
38                    break
39                case 'type':
40                    files = await File.find({user: req.user.id, parent: req.query.parent}).sort({type:1})
41                    break
42                case 'date':
43                    files = await File.find({user: req.user.id, parent: req.query.parent}).sort({date:1})
44                    break
45                default:
46                    files = await File.find({user: req.user.id, parent: req.query.parent})
47                    break;
48            }
49            return res.json(files)
50        } catch (e) {
51            console.log(e)
52            return res.status(500).json({message: "Файли не знайдено"})
53        }
54    }
```

Рисунок 3.3 – Реалізація сортування

Як показано на рисунку 3.3, методом getFiles виконується сортування файлів за типом, ім’ям та датою. Якщо файли не є наявні, то користувач отримає повідомлення «response status 500».

Методом createDir створюються директорії у вигляді папок та це дозволяє зберігати файли саме всередині неї. На рисунку 3.4 зображено реалізацію методу створення директорій.

server > controllers > JS fileController.js > FileController > createDir

```
9     async createDir(req, res) {
10        try {
11            const {name, type, parent} = req.body
12            const file = new File({name, type, parent, user: req.user.id})
13            const parentFile = await File.findOne({_id: parent})
14            if(!parentFile) {
15                file.path = name
16                await fileService.createDir(file)
17            } else {
18                file.path = `${parentFile.path}\\${file.name}`
19                await fileService.createDir(file)
20                parentFile.childs.push(file._id)
21                await parentFile.save()
22            }
23            await file.save()
24            return res.json(file)
25        } catch (e) {
26            console.log(e)
27            return res.status(400).json(e)
28        }
29    }
```

Рисунок 3.4 – Метод створення директорій

Для скачування файлу необхідно створити окремий метод класу. На рисунку 3.5 продемонстровано реалізацію методу додавання файлу.

На рисунку 3.5, метод uploadFile скачує файл та виконує перевірку на наявність місця на самому диску та на вже існуючий файл, це означає, що, якщо файл з цією назвою та типом вже наявний, то користувач отримає відповідне повідомлення про помилку. Метод зберігає файл у директорії, в якій безпосередньо знаходиться користувач, а також зберігається інформацію про назву, вагу, дату, директорію, тип, батьківський елемент (директорія, в якій знаходиться файл) та про користувача, що його додав.

server > controllers > JS fileController.js > FileController > uploadFile

```
56      async uploadFile(req, res) {
57          try {
58              const file = req.files.file
59
60              const parent = await File.findOne({user: req.user.id, _id: req.body.parent})
61              const user = await User.findOne({_id: req.user.id})
62
63              if (user.usedSpace + file.size > user.diskSpace) {
64                  return res.status(400).json({message: 'Немає місця на диску'})
65              }
66
67              user.usedSpace = user.usedSpace + file.size
68
69              let path;
70              if (parent) {
71                  path = `${config.get('filePath')}\\${user._id}\\${parent.path}\\${file.name}`
72              } else {
73                  path = `${config.get('filePath')}\\${user._id}\\${file.name}`
74              }
75
76              if (fs.existsSync(path)) {
77                  return res.status(400).json({message: 'Файл вже існує'})
78              }
79              file.mv(path)
80
81              const type = file.name.split('.').pop()
82              let filePath = file.name
83              if (parent) {
84                  filePath = parent.path + "\\" + file.name
```

Рисунок 3.5 – Метод додавання файлів

Також інформація про користувача, а також завантажені файли повинні зберігатися. На рисунку 3.6 зображено приклад інформації про завантаженні файли.

Як видно з рисунку 3.6, надана інформація про завантажений файл на веб-застосунок, або ж створену директорію, зберігається в БД.

Крім того, потрібний метод, який відповідає за завантажування файлу та перевірку його на наявність, якщо даного файлу немає в директорії.

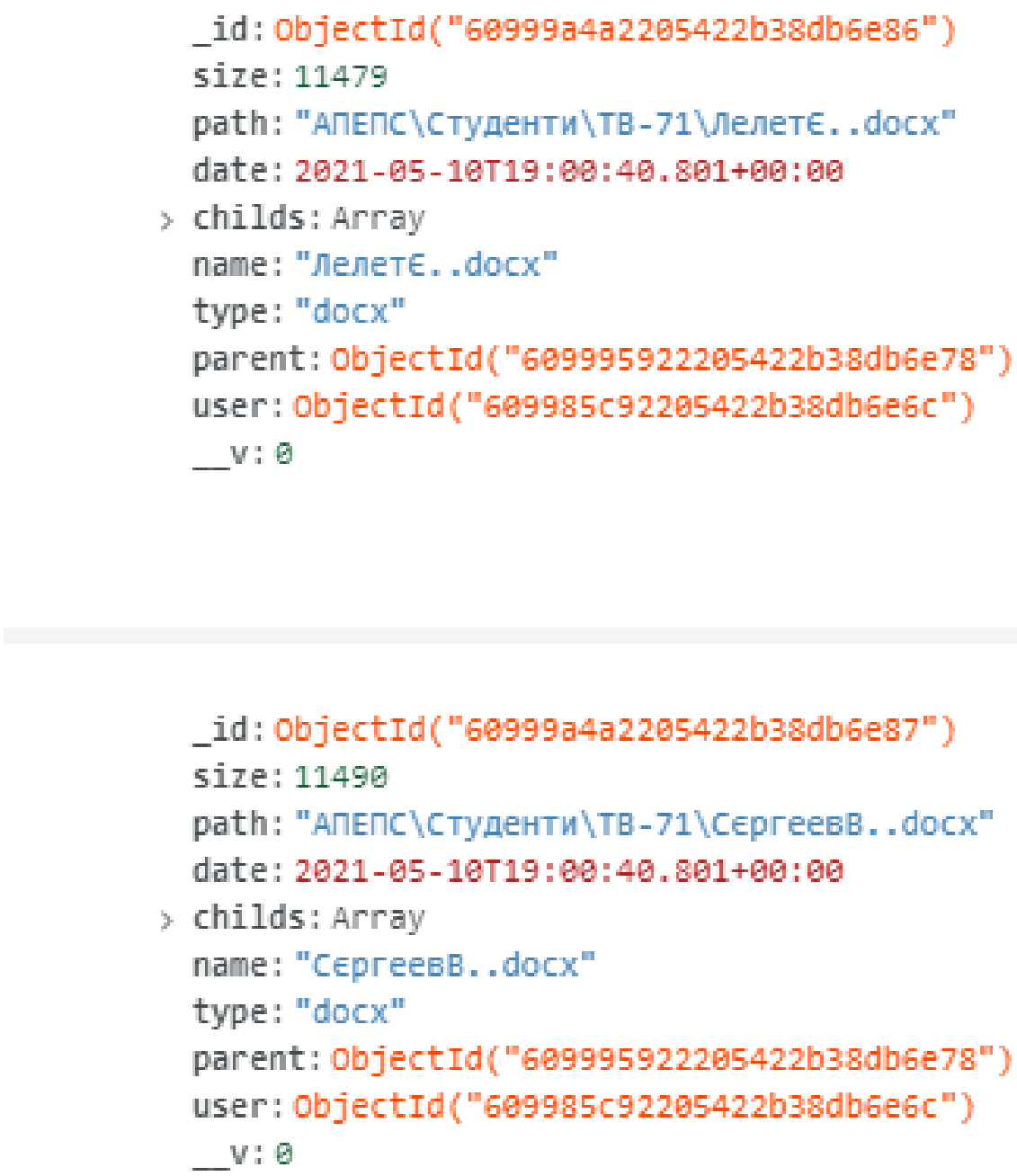


Рисунок 3.6 – Інформація про завантаженні на сайт файли та створенні директорії

3.2 Створення клієнтської частини

Для перегляду стану зміни застосунку використовують редусери з бібліотеки Redux, в якій зберігаються деякі дані, а також стани користувацького інтерфейсу в самому дереві станів. На рисунках 3.7 та 3.8 показано реалізацію редусера у роботи з файлами та кореневий редусер.

client > src > reducers > JS fileReducer.js > ...

```
export default function fileReducer(state = defaultState, action) {
    switch (action.type) {
        case SET_FILES: return {...state, files: action.payload}
        case SET_CURRENT_DIR: return {...state, currentDir: action.payload}
        case ADD_FILE: return {...state, files: [...state.files, action.payload]}
        case SET_POPUP_DISPLAY: return {...state, popupDisplay: action.payload}
        case PUSH_TO_STACK: return {...state, dirStack: [...state.dirStack, action.payload]}
        case DELETE_FILE: return {...state, files: [...state.files.filter(file => file._id != action.payload)]}
        case SET_VIEW: return {...state, view: action.payload}
        default:
            return state
    }
}

export const setFiles = (files) => ({type: SET_FILES, payload: files})
export const setCurrentDir= (dir) => ({type: SET_CURRENT_DIR, payload: dir})
export const addFile = (file) => ({type: ADD_FILE, payload: file})
export const setPopupDisplay = (display) => ({type: SET_POPUP_DISPLAY, payload: display})
export const pushToStack = (dir) => ({type: PUSH_TO_STACK, payload: dir})
export const deleteFileAction = (dirId) => ({type: DELETE_FILE, payload: dirId})
export const setFileView = (payload) => ({type: SET_VIEW, payload})
```

Рисунок 3.7 – Редусер для роботи з файлами

client > src > reducers > JS index.js > rootReducer > files

```
import {applyMiddleware, combineReducers, createStore} from "redux";
import {composeWithDevTools } from 'redux-devtools-extension'
import thunk from "redux-thunk";
import userReducer from "./userReducer";
import fileReducer from "./fileReducer";
import uploadReducer from "./uploadReducer";
import appReducer from "./appReducer";


const rootReducer = combineReducers({
    user: userReducer,
    files: fileReducer,
    upload: uploadReducer,
    app: appReducer
})

export const store = createStore(rootReducer, composeWithDevTools(applyMiddleware(thunk)))
```

Рисунок 3.8 – Кореневий редусер

Для меню навігації, файлів, їхніх списків, скачування та іншого функціоналу користувача створено окремі компоненти і файли з описом стилю завдяки препроцесору less. На рисунках 3.9 та 3.10 зображено приклади реалізації компонентів сортування та їх стилі.

client > src > components > disk > fileList > FileList.jsx > FileList

```
27
28      if (fileView === 'list') {
29          return (
30              <div className='filelist'>
31                  <div className="filelist__header">
32                      <div className="filelist__name">Назва</div>
33                      <div className="filelist__date">Дата</div>
34                      <div className="filelist__size">Розмір</div>
35                  </div>
36                  <TransitionGroup>
37                      {files.map(file =>
38                          <CSSTransition
39                              key={file._id}
40                              timeout={500}
41                              classNames={'file'}
42                              exit={false}
43                          >
44                              <File file={file}/>
45                          </CSSTransition>
46                      )}
47                  </TransitionGroup>
48              </div>
49          );
50      }
51
52  };
```

Рисунок 3.9 – Компонента сортування за датою, типом та назвою

client > src > components > disk > fileList > {} fileList.less > ...

```
3   .filelist {
4     margin: 20px 0;
5
6     &__header {
7       display: grid;
8       grid-template-columns: 1fr 4fr repeat(4, 1fr);
9     }
10
11    &__name {
12      grid-column-start: 2;
13    }
14    &__date {
15      grid-column-start: 5;
16      justify-self: center;
17    }
18    &__size {
19      grid-column-start: 6;
20      justify-self: center;
21    }
```

Рисунок 3.10 – Стилі компоненти

Як видно з рисунків 3.9 та 3.10, для того, щоб запобігти повторювань коду, компоненти мають окремі файли зі стилями.

3.3 Тестування

Для забезпечення безвідмовної роботи системи потрібно дотримуватися основних вимог та рекомендацій щодо використання.

Встановлювати систему немає потреби, так як вона створена у вигляді веб-застосунку і є у відкритому доступні за посиланням. Даний веб-застосунок дає можливість користувачу підключитися з будь-якого пристрою та користуватися можливими функціями.

У користувача має бути простий інтерфейс і легкий для розуміння інструментарій. Крім того, дані, які вводить користувач, мають бути надійно захищені. Користувач взаємодіє з графічним інтерфейсом, що є у доступі, як веб-застосунок. Так як у системі є можливість роботи, розподілена по ролям, тому початкове вікно це реєстрація/авторизація. На рисунку 3.11 зображено початкове вікно авторизації.

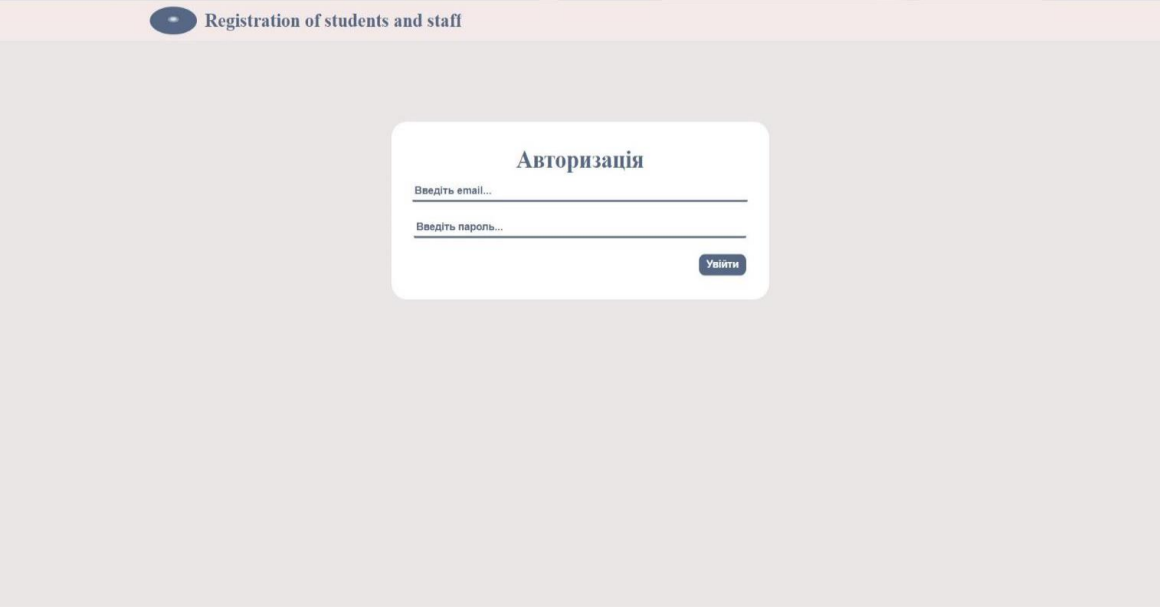


Рисунок 3.11 – Вікно авторизації

На рисунку 3.11, при вході в розроблений застосунок, видно меню авторизації разом з полями для вводу логіну у вигляді електронної, поле для введення паролю, а також кнопку «увійти» для відправки даних, які користувач ввів на перевірку. Паролі надійно шифруються за допомогою хеш-функції Bcrypt. Далі відбувається перевірка на наявність користувача в базі даних. У випадку успішної авторизації користувач потрапить на сторінку з директорією. Через 3 години потрібно залогінитися, тобто пройти авторизацію ще раз, оскільки час сесії роботи користувача всього лише 3 години. На рисунку 3.12 зображена початкова сторінка.

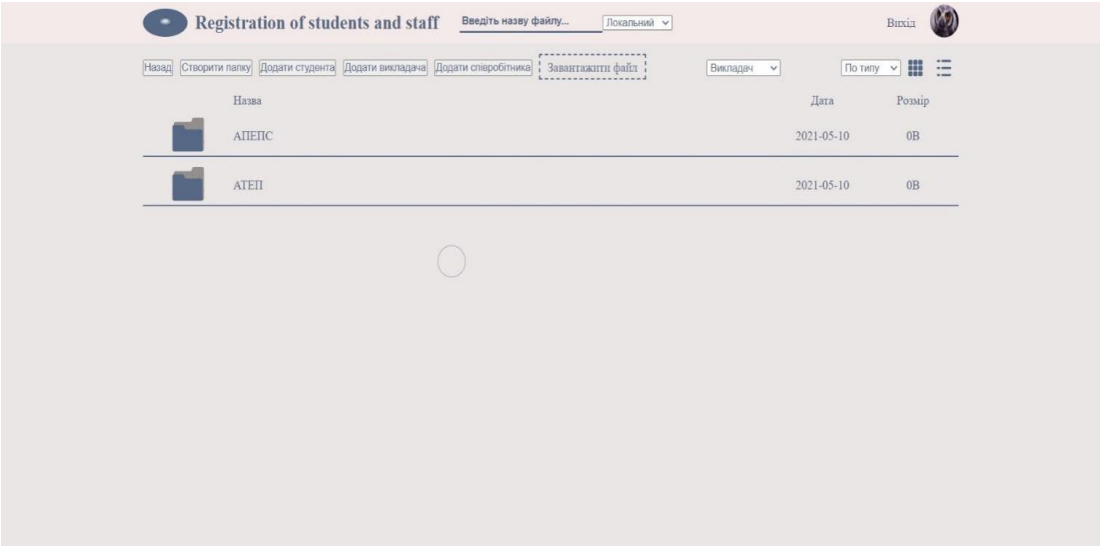


Рисунок 3.12 – Початкова сторінка з кореневою директорією

На рисунку 3.12, на головній сторінці знаходяться директорії для переходу. Для того, щоб перейти за директорією необхідно натиснути на папку, а для повернення на один рівень назад – необхідно клікнути кнопку «Назад».

Для швидкого локального та глобального пошуку потрібного файлу треба скористатися пошуком, натиснувши на нього та ввівши потім назву файлу або жпочаткові літери назви. На рисунку 3.13 зображено приклад використання функції глобального та локального пошуку.

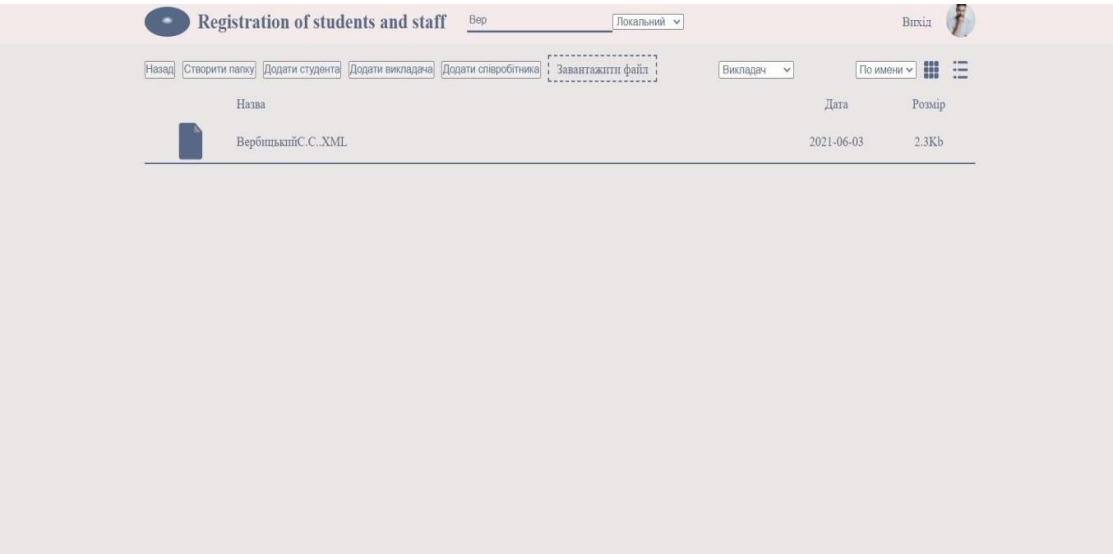


Рисунок 3.13 – Приклад використання локального пошуку

Рисунок 3.14 – Приклад використання глобального пошуку

Як видно з рисунків 3.13 та 3.14, для зручності користувач має можливість ввести перші літери необхідного файлу або директорії на панелі пошуку з підписом «Введіть назву файлу». Крім того, можливо переключити

пошук з глобального на локальний, тобто пошук тільки в папці, в якій знаходиться користувач в момент часу. Справа від панелі пошуку є вибір типу пошуку. Для того, щоб змінити пошук необхідно обрати один з двох варіантів: «Глобальний» або «Локальний». Якщо пошук глобальний, то не потрібно переходити до папки з необхідним файлом або ж директорією.

Користувач має можливість створити нову директорію в будь-якій з інших директорії. На рисунках 3.15 та 3.16 зображено приклад створення нової директорії (папки).

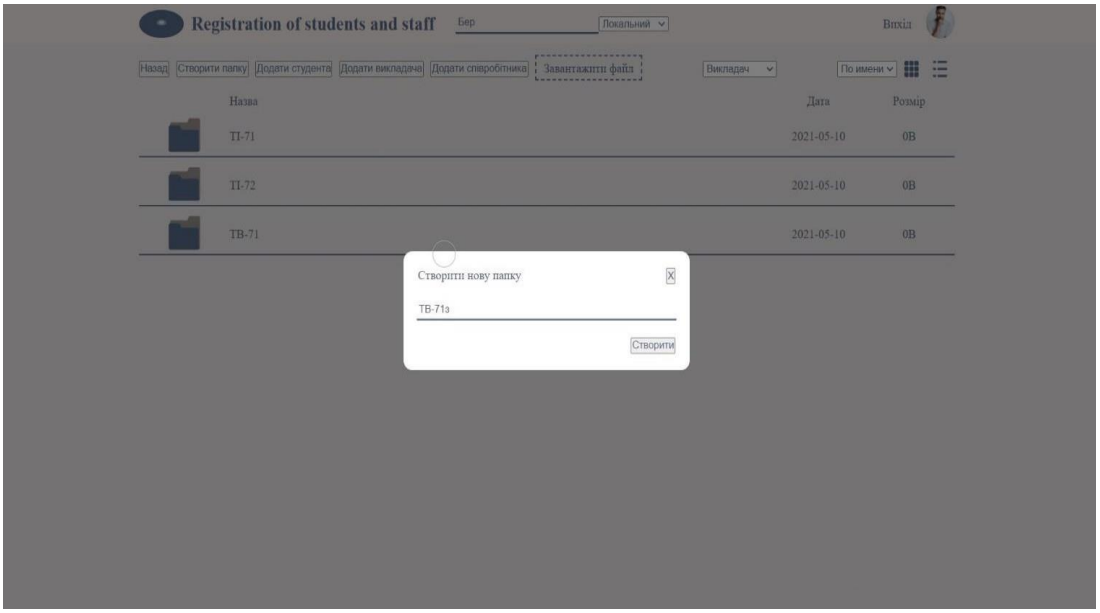


Рисунок 3.15 – Введення назви папки

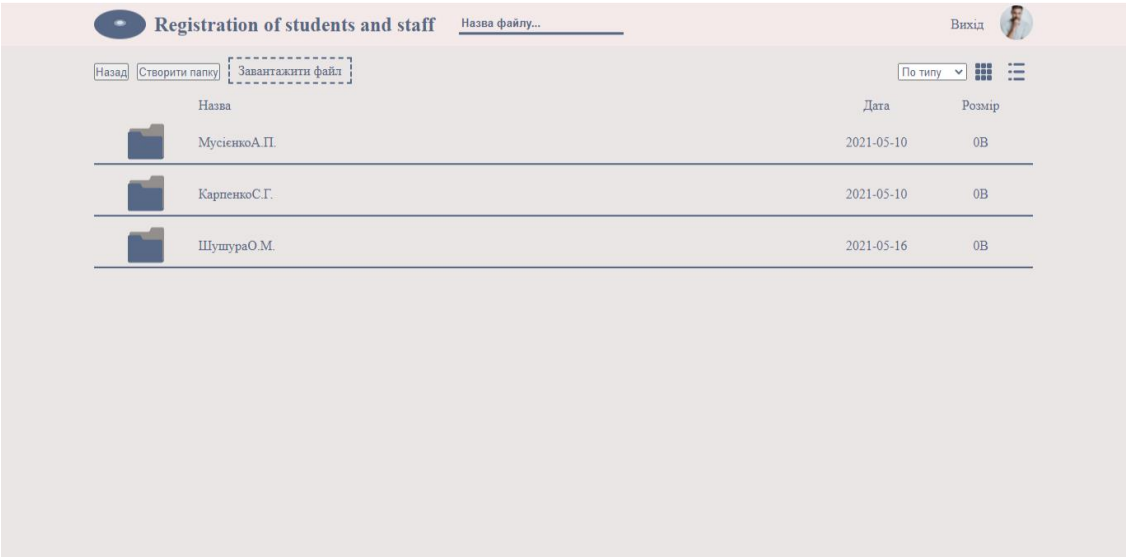


Рисунок 3.16 – Успішне відображення створеної папки

Як видно з рисунків 3.15 та 3.16, для того, щоб створити в поточній папці нової директорії (папки) необхідно клікнути на кнопку під назвою «Створити папку», далі ввести її назву і натиснути кнопку «Створити». Після успішної операції отримуємо нову директорію з назвою, яку було задано.

Користувач може скачати файли в потрібній йому кількості і будь-якого типу, за розміром до 250 Мегабайтів та з максимальною швидкістю, що пропонує провайдер користувачу.

Користувач має можливість створити XML файли в застосунку, де вводиться інформація про студентів і викладачів. На рисунку 3.8 показано меню з полями для заповнення інформації про студента.

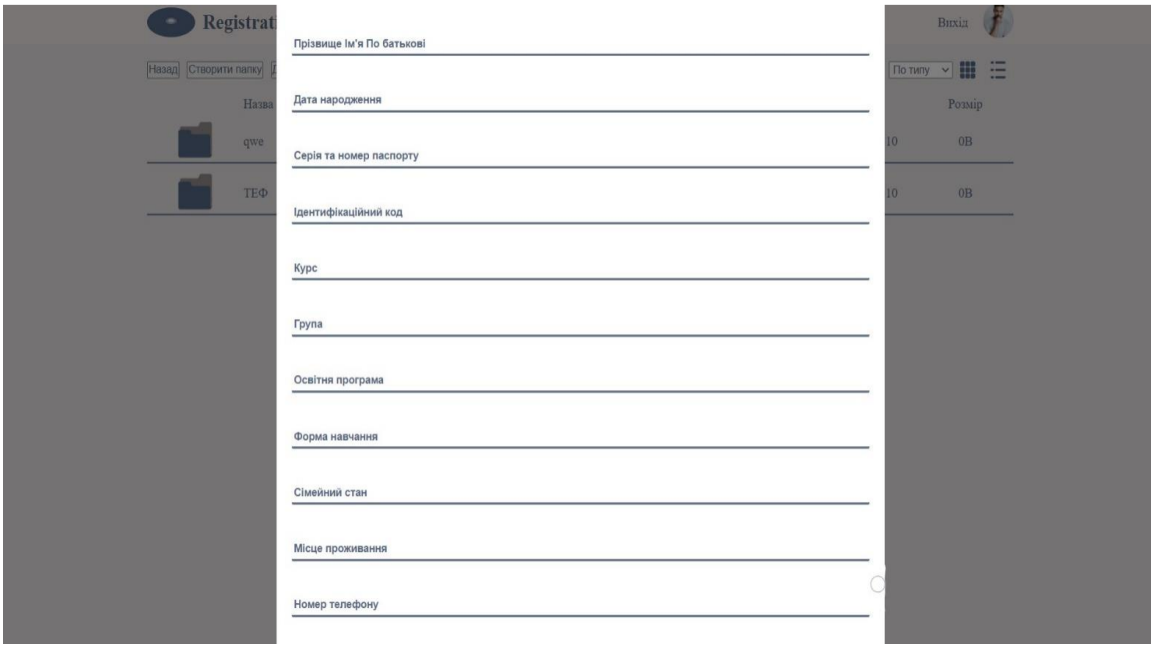


Рисунок 3.17 – Додавання інформації про студента

Як видно з рисунку 3.17, для створення відповідного XML файлу потрібно натиснути кнопку на вибір «додати студента», «додати викладача» або «додати співробітника», які розташовані по праву сторону функції «створити папку» та заповнити дані у відповідних полях. Після вводу даних необхідно клікнути кнопку «створити». Після цього добавлений користувачем новий XML файл повинен з’явитися в директорії, де він знаходиться в той момент часу.

Користувач має можливість завантажувати, редагувати та видаляти створені ним XML файли. На рисунку 3.18 показано функціонал при роботі зі створеними XML документами.

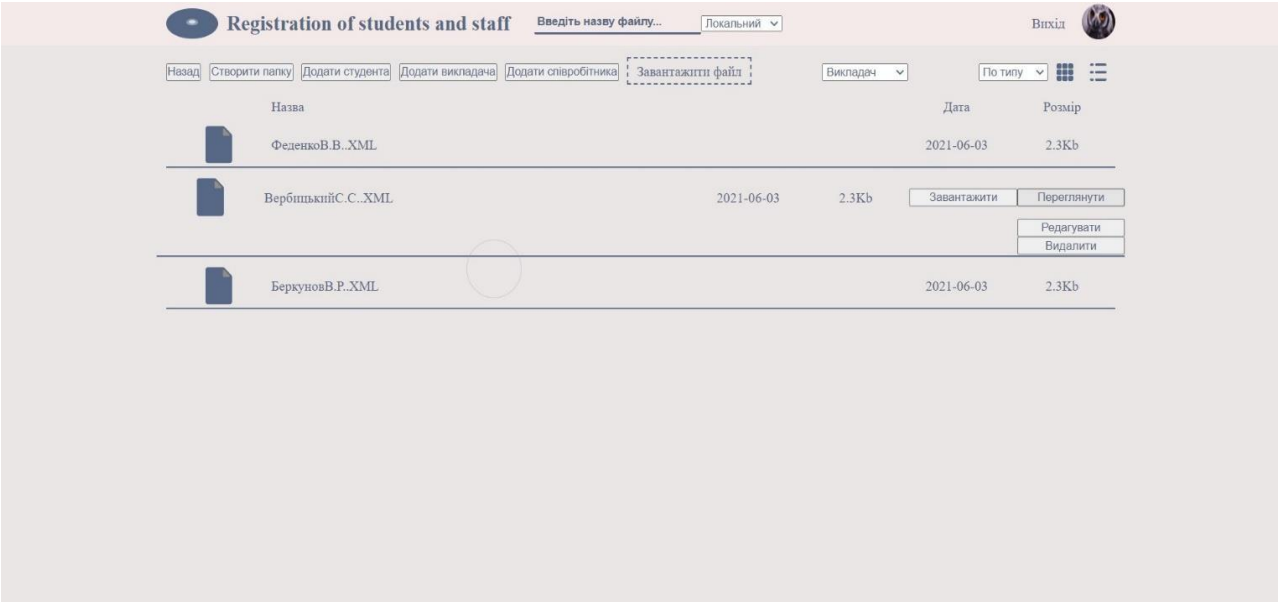


Рисунок 3.18 – Функціонал роботи

Як видно з рисунку 3.18, користувач може переглядати, скачувати, редагувати або видаляти створені файли, натиснувши на одну з наведених кнопок: «завантажити», «редагувати», «видалити» або ж «переглянути». На рисунку 3.19 показано приклад редагування XML документу.

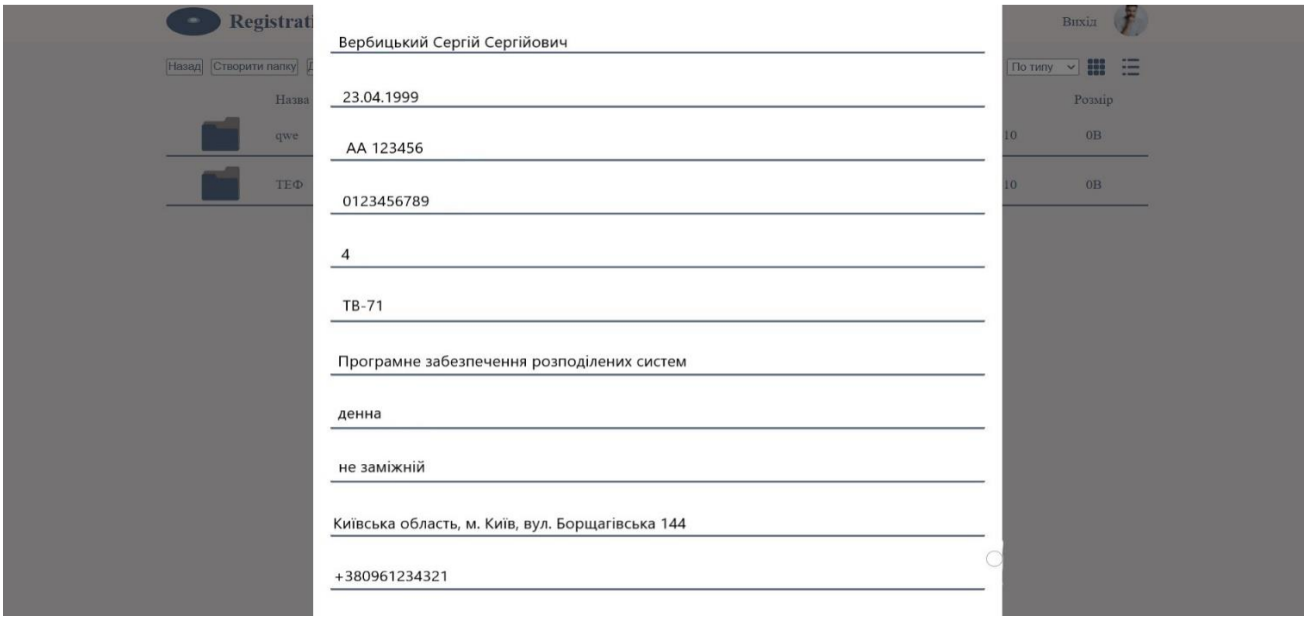


Рисунок 3.19 – Редагування інформації про студента

Як показано на рисунку 3.19, при натисканні кнопки «редагувати» у створеному файлі, користувач може побачити перелік даних у створеному вже документі та зможе відредагувати його, натиснувши на потрібне поле та змінивши дані в ньому.

На рисунках 3.20 та 3.21 показано додавання файлів на веб-сторінку.

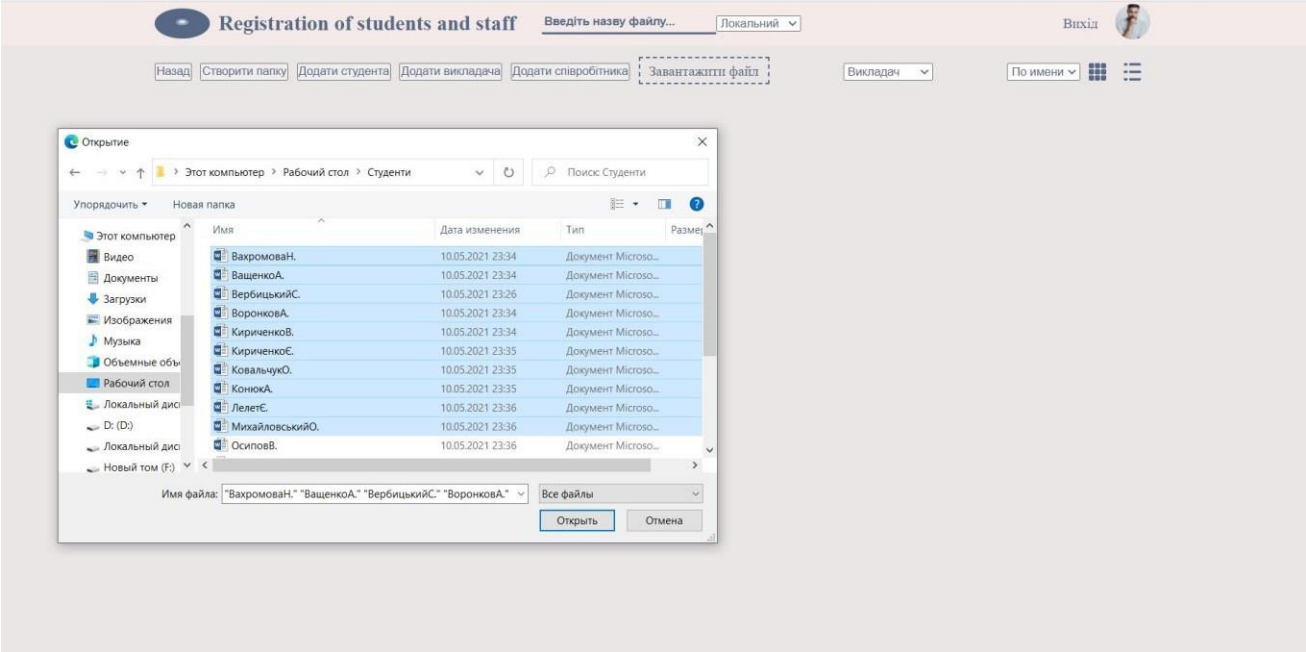


Рисунок 3.20 – Додавання файлів за допомогою кнопки «Завантажити файл»

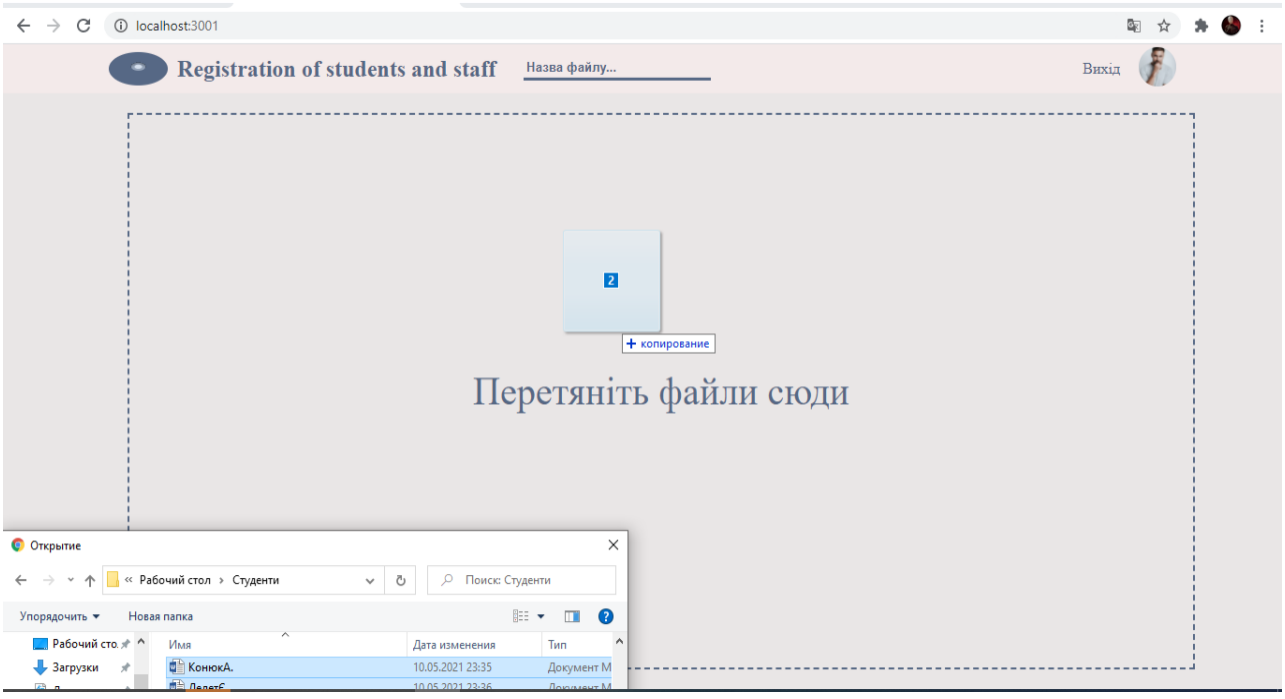


Рисунок 3.21 – Завантаження файлів за допомогою перетягування(drag-and-drop)

3.4 Висновки

В даному розділі здійснювалася розробка програмного забезпечення, а саме серверної та клієнтської частини, з використанням мови JavaScript. Розробка програмної частини та інтерфейсу здійснювалася покроково, паралельно одна одній.

Було описано основні класи і їх методи застосунку. Також, описано їх взаємодію з компонентами розроблюваного програмного забезпечення.

В розділі продемонстровано скріншоти, а також функціонал програми (деякі вікна), надано інструкцію для подальшого зручного використання. Конктерно описано, як авторизуватися, здійснювати пошук (як глобальний, так і локальний), завантажувати, редагувати та видаляти файли, а також, як переглядати інформацію про студентів, викладачів та предмети.

ВИСНОВКИ

У процесі виконання поставлених завдань розроблено програму продукт, призначений для ведення обліку студентів та викладачів кафедри, що дозволяє додавати, редагувати, переглядати та видаляти документи для вказаних осіб у файлі XML. У ході роботи ми отримали наступний результат:

1. Проведено огляд існуючих програмних рішень в області обліку студентів та викладачів кафедри. Це дало можливість розробити оптимальний програмний продукт, який відповідає потребам користувачів. Огляд існуючих програмних рішень в області обліку студентів та викладачів кафедри дозволяє визначити, які функції та можливості повинні бути реалізовані у програмному продукті, а також визначити оптимальну архітектуру програми та вибрати найкращі технології для розробки.

2. Було вибрано засоби розробки ПЗ: стек MERN, який складається з таких компонентів, як: середовище розробки Node.js (через неблокуючу модель вводу-виводу та одну мову на клієнті та на сервері); бібліотеку React.js (через прозорість синтаксису, високий рівень гнучкості та легку вагу); СУБД MongoDB (через високу швидкість у роботі з файлами); фреймворк Express (через швидку).

3. Розроблено архітектурні рішенння, які дозволили визначити, які компоненти програми повинні бути реалізовані та як вони повинні взаємодіяти між собою. Це забезпечило ефективну та оптимальну роботу програми. Тестування програми є не менш важливим етапом, який дозволив перевірити роботу програми та виявити можливі помилки та недоліки. Показ тестового прикладу використання деяких окремих функцій програми дозволив перевірити, чи відповідає програма поставленим вимогам та чи забезпечує вона необхідний функціонал. Результати тестування, що підтвердили працездатність сервісу та відповідність до поставлених вимог, переконали, що програма працює правильно та відповідає вимогам користувачів.

4. Детальний опис інтерфейсу взаємодії користувача з програмою

дозволив зробити його більш зручним та простим у використанні. Наведення прикладів взаємодії з функціоналом та їх опис дозволяє користувачам швидко зрозуміти, як користуватися програмою та який функціонал вона має.

5. Розроблюваний застосунок дає створити, переглядати, редагувати та видаляти XML файл з даними про студентів та викладачів, а також завантажувати, скачувати та видаляти файли різних типів, крім того, створювати структуру папки для більшої зручності обліку. Є можливість здійснити глобальний пошук, сортувати за категоріями та змінити зображення у вигляді тексту або іконок для файлів або директорій.

Результати даної роботи можуть бути доцільно використані співробітниками університету з відділу кадрів для ведення обліку студентів та викладачів.

В майбутніх дослідженнях варто розширити роботу веб- застосунку, розширити функціонал для зручності користування, додати можливість пошуку за тегами, розглянути можливості інтегрування в інші сервіси.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. D. L. Yuan Fang Li, Paramjit K. Das, «Два десятиліття тестування веб-додатків – огляд останніх досягнень», Information Systems, vol. 43, С. 20 – 54, 2014.

2. Microsoft Office 365 Documentation [Electronic resource] - Access mode: https://docs.microsoft.com/en-us/microsoft-365/?view=o365-worldwide/ .

3. Documentation 1C: Enterprises [Electronic resource] — Access mode: https://its.1c.ua/db/v838doc/.4. Wonderware platform, http://global.wonderware.com/EN/Pages/default.aspx, last retrieved on Jan 31st, 2011.

5. MongoDB documentation [Electronic resource] - Access mode: https://docs.mongodb.com/ .

6. Express documentation [Electronic resource] — Access mode: https://expressjs.com/ .

7. React Documentation [Electronic Resource] - Access Mode: https://reactjs.org/tutorial/tutorial.html/ .

8. Node.js documentation [Electronic resource] - Access mode: https://nodejs.org/en/docs/ .

9. Axios Documentation [Electronic resource] - Access mode: https://www.npmjs.com/package/axios/ .

10. Bcrypt documentation [Electronic resource] - Access mode: https://www.npmjs.com/package/bcrypt/ .

11. Redux Documentation [Electronic resource] - Access mode: https://redux.js.org/introduction/getting-started/ .

12. Mongoose documentation [Electronic resource] - Access mode: https://mongoosejs.com/ .

13. Nodemon documentation [Electronic resource] - Access mode: https://github.com/remy/nodemon#nodemon/ .

14. E. Dustin, T. Garrett і B. Gauf, Implementing Automated Software Testing. Pearson Education, 2009.

15. JavaScript Documentation [Electronic resource] – Режим доступу до ресурсу: https://devdocs.io/javascript/.

16. Лобок О.П. Організація баз даних та знань. Теоретичні основи проектування, реалізації та використання баз даних: Навч. посіб. – К.: НУХТ, 220133 2262 с.

17. Методичні вказівки до виконання КвР. [Electronic resource] // Офіційний сайт Хмельницького національного університету. – Режим доступу: https://msn.khnu.km.ua/pluginfile.php/227349/mod_resource/content/1/Kusr_proekt_metodicka.pdf

18. Опис функціоналу 1С:Підприємства [Electronic resource] — Режим доступу: http://1c.ua/ua/v8/RegionalSolutions_UA_ZUP.php/ .

19. Порівняння версій та опис функціоналу Кадри Плюс Україна [Електронний ресурс] — Режим доступу: https://andeesoft.com/ua/kpu/.

20. Порівняння версій Microsoft Office 365 [Electronic resource] — Режим доступу: https://www.microsoft.com/uk-ua/microsoft-365/buy/compare-all-microsoft-365-products/

21. Java Documentation [Electronic resource] – Режим доступу до ресурсу: https://docs.oracle.com/en/java/

22. Форкун Ю. В. Інформатика : навч. Посіб. / Ю. В. Форкун, Н. А. Длугунович. – Л. : Новий світ-2000, 2011. – 567 с.

23. JavaServer Pages Document [Electronic resource] – https://docs.oracle.com/javaee/5/tutorial/doc/bnajo.html

24. Коротка історія .NET Core; [Electronic resource]: https://dotnetcore.show/episode-1-a-brief-history-of-net-core/25. Buckett C. Dart in Action // Chris Buckett, 2013. – 424 с.

26. Strephonsays [Electronic resource] – Режим доступу до ресурсу: https://uk.strephonsays.com/logical-and-vs-physical-data-model-11564

27. Mitchell D. Dart: Scalable Application Development // D. Mitchell, S. Akopkokhyants, I. Balbaert, 2017. – 1298 с.

28. Н. Панкая, «Порівняльне дослідження популярних онлайн-платформ для електронного навчання», Ph.D. дис., Центр дослідження освітнього медіа, Ун-т. Майсур, Майсур, 2015.

29. Sabharwal, R, Chugh, R, Hossain MR & Wells, M, 2018, «Системи управління навчанням на робочому місці: огляд літератури», 2019 IEEE International Conference on Teaching, Assessment and Learning of Engineering (TALE), 4-7 Грудень, Вуллонгонг, Австралія, стор. 387-393, doi: 10.1109/TALE.2018.8615158.

30. Sikore M. Dart Essentials // Мартін Сікора, 2015. – 236 с.

31. Strom C. Dart for Hipsters // Chris Strom, 2014. – 145 с.

32. Чарльз Андерсон. 2015. Docker [програмна інженерія]. Програмне забезпечення IEEE 32, 3 (2015), 102–с3.

33. ДСТУ ГОСТ 3008-95 «Документація. Звіти у сфері науки і техніки. Структура і правила оформлення»

34. V. O. Gryaznova, S. V. Yefimenko. Fundamentals of programming methodology. - K.: Kyiv University of Ukrainian Orthodox Church, 2005.

36. T.P. Karavanova. Basics of algorithmization and programming. 750 problems with recommendations and examples. - K.: Forum, 2002

37. Л. Новіков. Введення в Rational Unified Process. - Режим доступу: http://www.interface./rational/interface/151199/rup/main.htm

38. Трьохрівневий MVC у програмуванні [Електронний ресурс] – Режим доступу до ресурсу: http://naukam.triada.in.ua/index.php/konferentsiji/45-pyatnadtsyata-vseukrajinska-praktichno-piznavalna-internet-konferentsiya/304-tririvnevij-mvc-u-veb-programuvanni

39. D. L. D. Yuan Fang Li, Paramjit K. Das, «Два десятиліття тестування веб-додатків – огляд останніх досягнень», Information Systems, vol. 43, С. 20 – 54, 2014.

40. Alan Berg “Jenkins Continuous Integration” – June 21, 2012 p. 344

ДОДАТОК А

(обов’язковий)

ПРЕЗЕНТАЦІЙНИЙ МАТЕРІАЛ

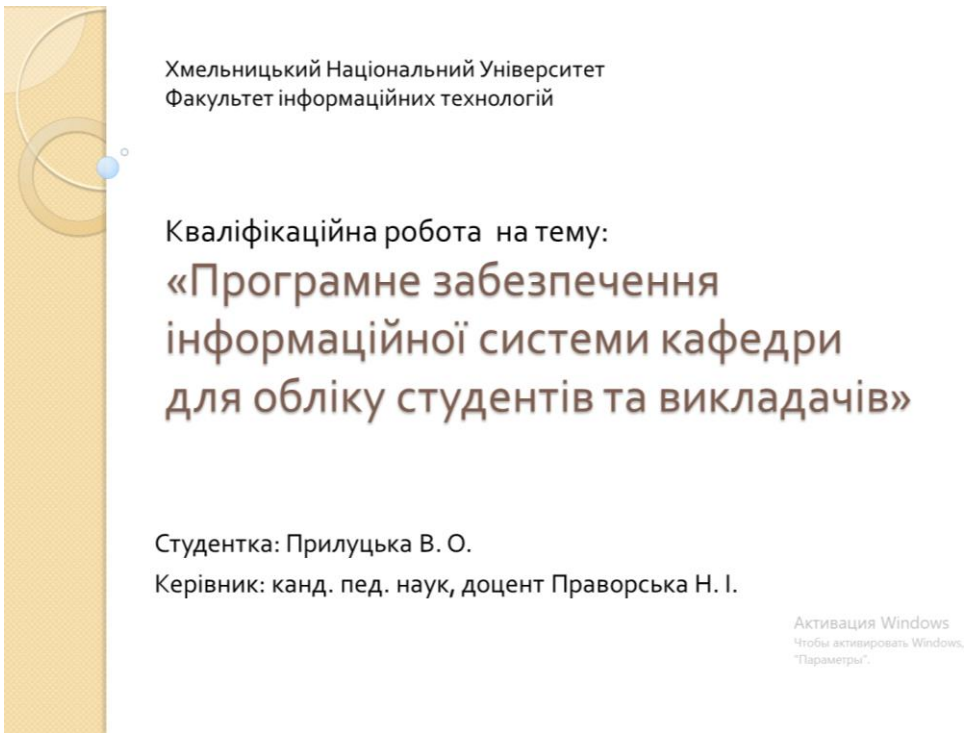


Рисунок А.1 – Слайд №1

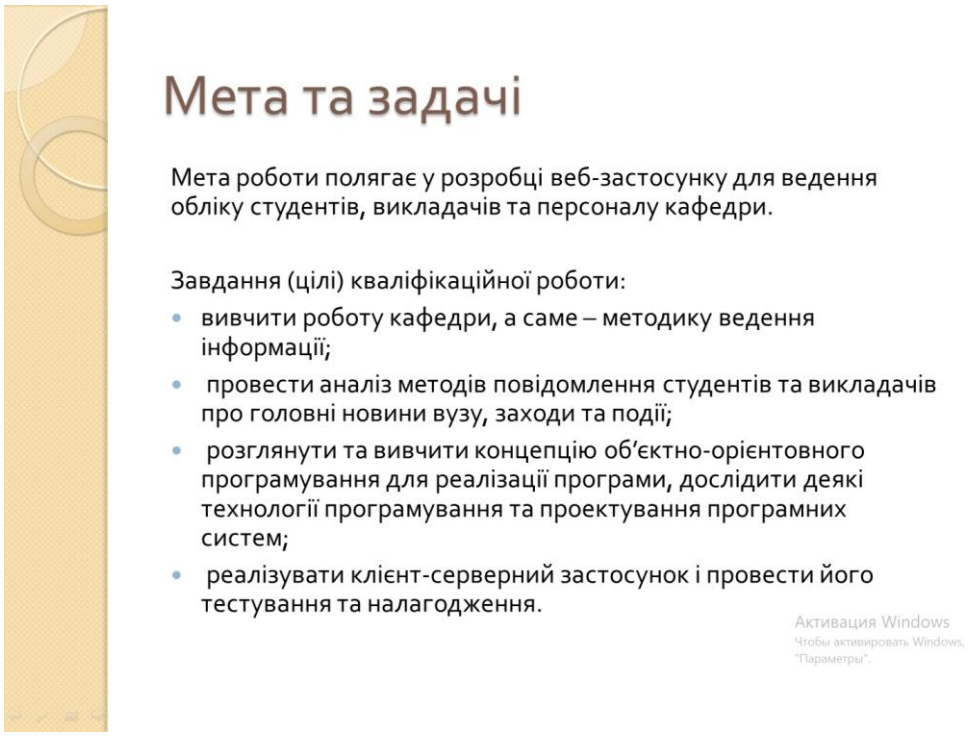


Рисунок А.2 – Слайд №2

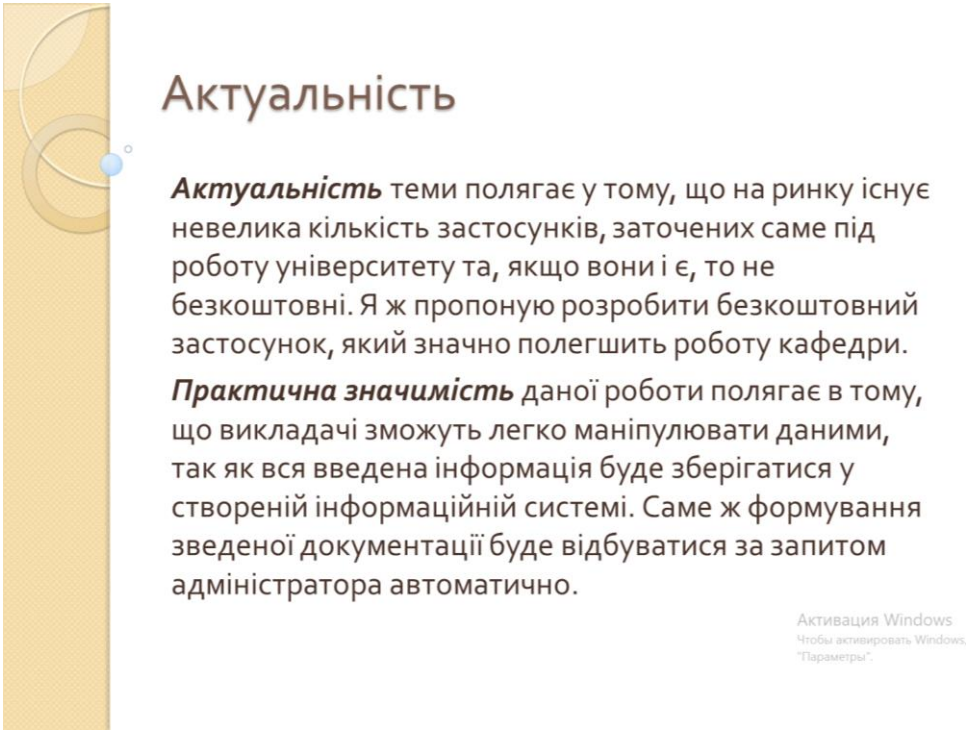


Рисунок А.3 – Слайд №3

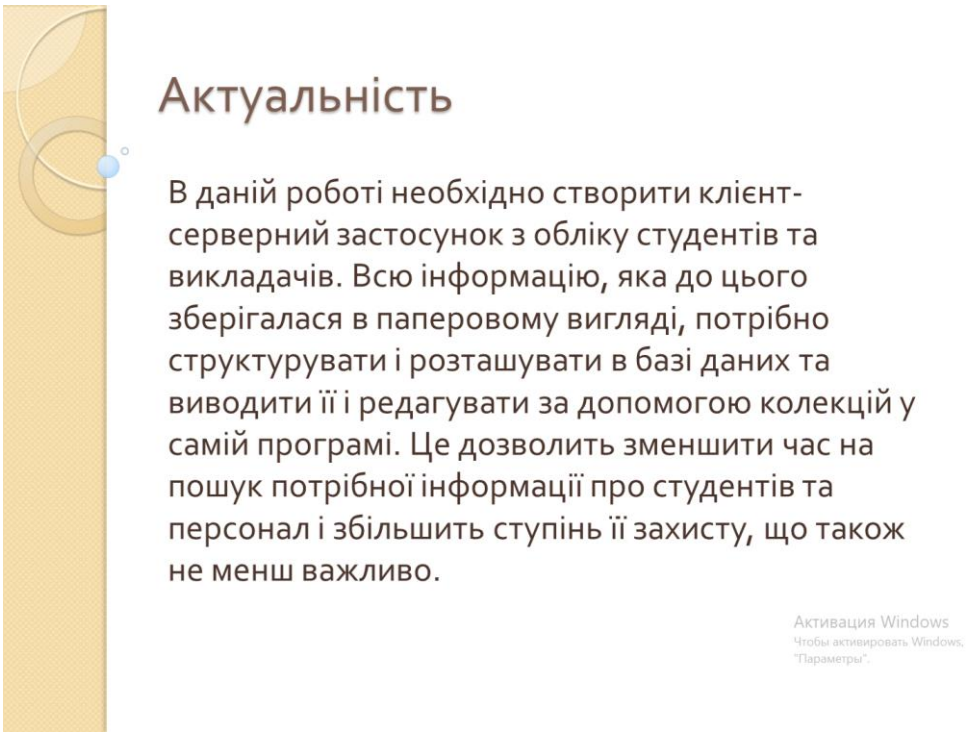
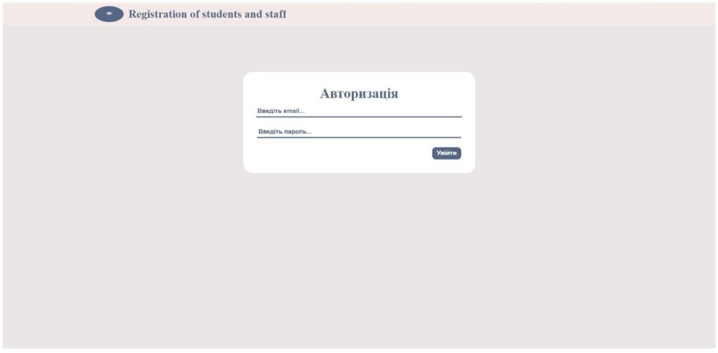


Рисунок А.4 – Слайд №4

Рисунок А.5 – Слайд №5

Рисунок А.6 – Слайд №6

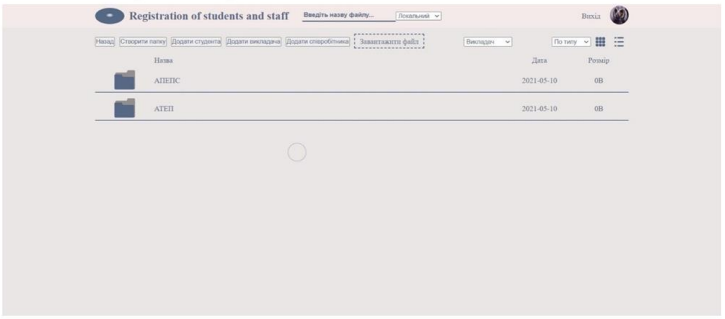
Вікно розроблюваного ПЗ



Як показано на рисунку, при вході на наш веб-застосунок, ми бачимо меню авторизації з полями для вводу електронної пошти у вигляді логіну, поле для введення паролю, а також кнопка «увійти» для відправлення даних, введених користувачем на перевірку.

Рисунок А.7 – Слайд №7

Вікно розроблюваного ПЗ

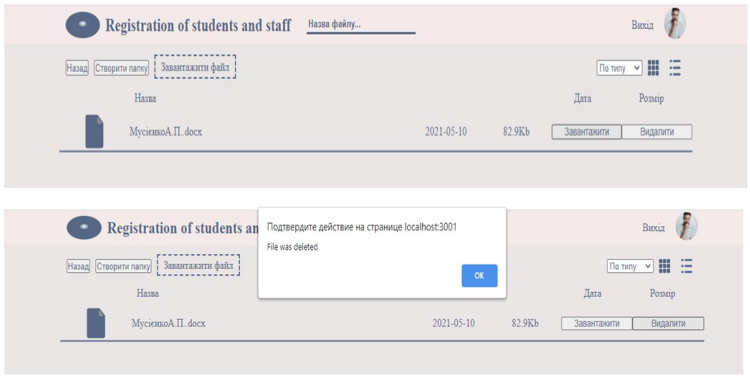


На головній сторінці ми маємо директорії папок для переходу. Для переходу за директорією папки потрібно натиснути на неї, для повернення на один рівень вгору потрібно натиснути кнопку «Назад».

Для швидкого локального або глобального пошуку необхідного файлу можна скористатися пошуком, натиснувши на нього та ввівши початкові літери назви файлу.

Рисунок А.8 – Слайд №8

Вікно розроблюваного ПЗ



Як видно на рисунках, для скачування або видалення файлів та видалення директорій (папок) потрібно навести мишкою на потрібну нам файл чи папку та натиснути кнопку «Видалити» або «Завантажити». При успішному видаленні ми отримаємо повідомлення про успіх.

Рисунок А.9 – Слайд №9

Висновки

Таким чином, розроблено програмний продукт для обліку студентів та викладачів кафедри, який дозволяє додавати, редагувати, переглядати та видаляти документи щодо вказаних осіб у вигляді XML файлів. В ході виконання даної роботи було отримано наступні результати:

1. Проведено огляд існуючих програмних рішень в області обліку студентів та викладачів кафедри, який показав актуальність створення нового програмного продукту, побудованого на архітектурі клієнт-сервер з необхідним функціоналом та зручним користувацьким інтерфейсом.

2. Вибрано засоби розробки сервісу: стек MERN, що складається з наступних компонентів: середовище розробки Node.js через неблокуючу модель вводу-виводу та одну мову на клієнті та сервері; бібліотеку React.js через простоту синтаксису, високий рівень гнучкості та легку вагу; СУБД MongoDB задля високої швидкості при роботі з файлами; фреймворк Express через швидку розробку з використанням Node.js.

Рисунок А.10 – Слайд №10

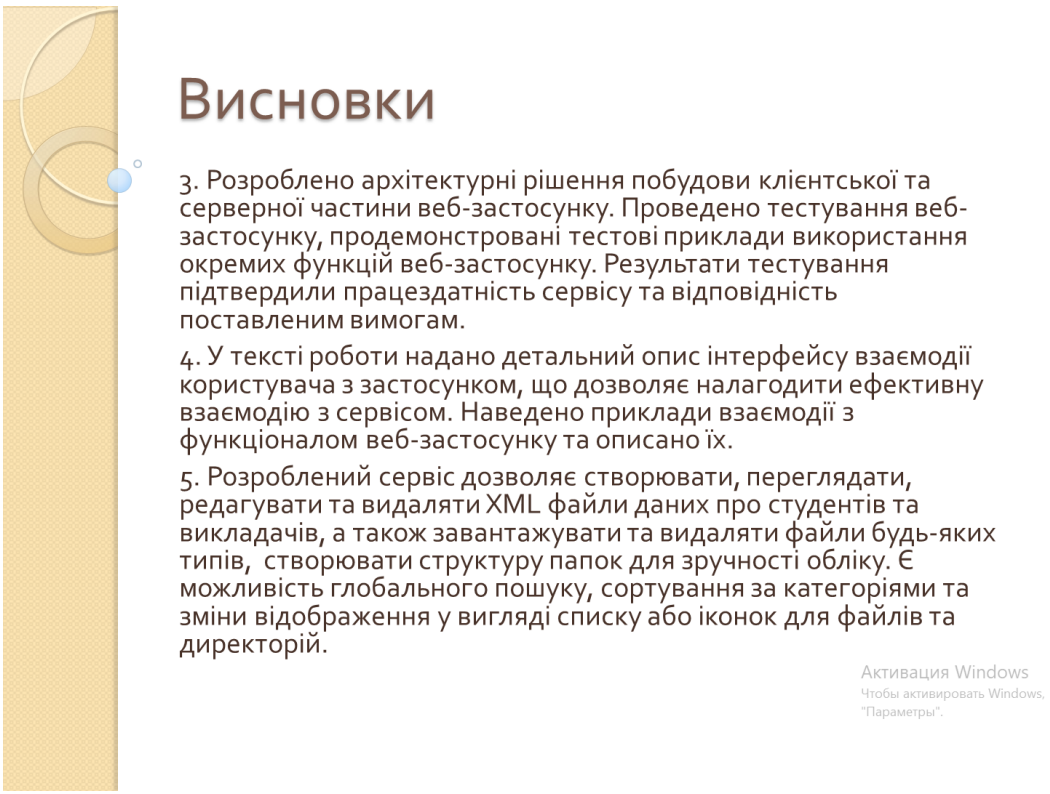


Рисунок А.11 – Слайд №11

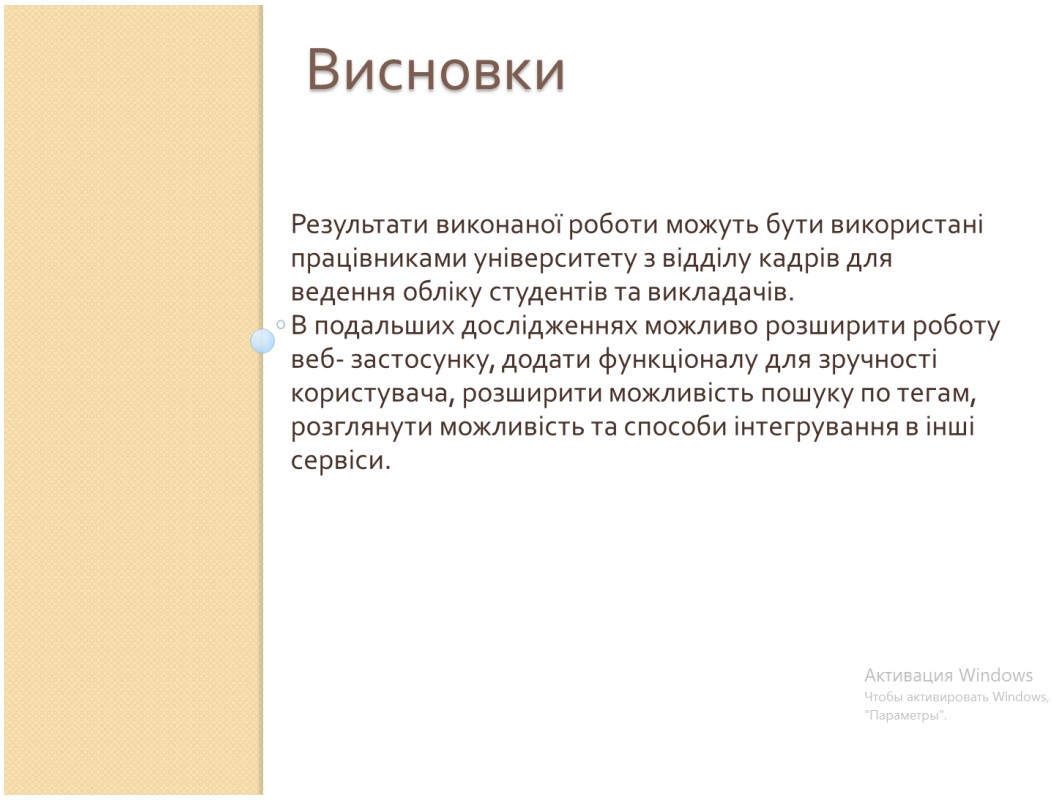


Рисунок А.12 – Слайд №12

Рисунок А.13 – Слайд №13

ДОДАТОК Б

(обов’язковий)

fileService.js const

```
fs = require('fs')

const File = require('../models/File') const
config = require('config')

class FileService {
    createDir(file) {

        const filePath = `${config.get('filePath')}\\${file.user}\\${file.path}`
        return new Promise(((resolve, reject) => {

            try {

                if (!fs.existsSync(filePath)) {
                    fs.mkdirSync(filePath)

                    return resolve({message: 'Файл було створено'}) } else
                {

                    return reject({message: "Файл уже існує"}) }

            } catch (e) {

                return reject({message: 'Помилка файлу'}) }

        })) }

    deleteFile(file) {

        const path = this.getPath(file) if
        (file.type === 'dir') {

            fs.rmdirSync(path) } else
        {

            fs.unlinkSync(path) }

    }
    getPath(file) {

return config.get('filePath') + '\\' + file.user + '\\' + file.path }

}
```

```
module.exports = new FileService()
```

auth.routes.js

```
const Router = require("express");

const User = require("../models/User")
const bcrypt = require("bcryptjs")
const config = require("config")

const jwt = require("jsonwebtoken")

const {check, validationResult} = require("express-validator")
const router = new Router()

const authMiddleware = require('../middleware/auth.middleware')
const fileService = require('../services/fileService')

const File = require('../models/File')
router.post('/auth',
   [

      check('email', "Uncorrect email").isEmail(),

      check('password', 'Password must be longer than 3 and shorter than 12').isLength({min:3, ma
x:12})

   ],

   async (req, res) => {
   try {

      const errors = validationResult(req)
      if (!errors.isEmpty()) {

         return res.status(400).json({message: "Uncorrect request", errors})
      }

      const {email, password} = req.body

      const candidate = await User.findOne({email})
      if(candidate) {
```

```
            return res.status(400).json({message: `User with email ${email} already exist`})
        }

        const hashPassword = await bcrypt.hash(password, 8)
        const user = new User({email, password: hashPassword})
        await user.save()

        await fileService.createDir(new File({user:user.id, name: ''}))
        res.json({message: "User was created"})

} catch (e) {
    console.log(e)

    res.send({message: "Server error"}) }

})
router.post('/login',

  async (req, res) => { try {

      const {email, password} = req.body   const user
      = await User.findOne({email}) if (!user) {

        return res.status(404).json({message: "User not found"}) }

      const isPassValid = bcrypt.compareSync(password, user.password) if
      (!isPassValid) {

        return res.status(400).json({message: "Invalid password"}) }

      const token = jwt.sign({id: user.id}, config.get("secretKey"), {expiresIn: "3h"})
      return res.json({

        token, user:
        {

          id: user.id,

          email: user.email, diskSpace:
          user.diskSpace, usedSpace:
          user.usedSpace, avatar:
          user.avatar
```

```
            } })

        } catch (e) {
            console.log(e)

            res.send({message: "Server error"}) }

    })
router.get('/auth', authMiddleware,
    async (req, res) => { try {

            const user = await User.findOne({_id: req.user.id})

            const token = jwt.sign({id: user.id}, config.get("secretKey"), {expiresIn: "1h"})
            return res.json({

                token, user:
                {

                    id: user.id,

                    email: user.email, diskSpace:
                    user.diskSpace, usedSpace:
                    user.usedSpace, avatar:
                    user.avatar

                } })

        } catch (e) {
            console.log(e)

            res.send({message: "Server error"}) }

    })
module.exports = router
```

fileController.js

```
const fileService = require('../services/fileService')

const config = require('config')
const fs = require('fs')

const User = require('../models/User')
const File = require('../models/File')
const Uuid = require('uuid')
```

```
class FileController {

    async createDir(req, res) {
        try {

            const {name, type, parent} = req.body

            const file = new File({name, type, parent, user: req.user.id})
            const parentFile = await File.findOne({_id: parent})
            if(!parentFile) {

                file.path = name

                await fileService.createDir(file)
            } else {

                file.path = `${parentFile.path}\\${file.name}`
                await fileService.createDir(file)
                parentFile.childs.push(file._id)

                await parentFile.save()
            }

            await file.save()
            return res.json(file)

        } catch (e) {
            console.log(e)

            return res.status(400).json(e)
        }

    }


    async getFiles(req, res) {
        try {

            const {sort} = req.query
            let files

            switch (sort) {
                case 'name':

                    files = await File.find({user: req.user.id, parent: req.query.parent}).sort({name:1})
                    break

                case 'type':

                    files = await File.find({user: req.user.id, parent: req.query.parent}).sort({type:1})
                    break

                case 'date':

                    files = await File.find({user: req.user.id, parent: req.query.parent}).sort({date:1})
                    break
```

```
            default:
                files = await File.find({user: req.user.id, parent: req.query.parent})
                break;
        }
        return res.json(files)
    } catch (e) {
console.log(e)
        return res.status(500).json({message: "Файли не знайдено"}) }
}

async uploadFile(req, res) {
    try {
        const file = req.files.file

        const parent = await File.findOne({user: req.user.id, _id: req.body.parent})
        const user = await User.findOne({_id: req.user.id})

        if (user.usedSpace + file.size > user.diskSpace) {
            return res.status(400).json({message: 'Немає місця на диску'}) }

        user.usedSpace = user.usedSpace + file.size

        let path;
        if (parent) {
            path = `${config.get('filePath')}\\${user._id}\\${parent.path}\\${file.name}` }
        else {
            path = `${config.get('filePath')}\\${user._id}\\${file.name}` }

        if (fs.existsSync(path)) {
            return res.status(400).json({message: 'Файл вже існує'}) }
        file.mv(path)

        const type = file.name.split('.').pop()
        let filePath = file.name
        if (parent) {
            filePath = parent.path + "\\" + file.name }
```

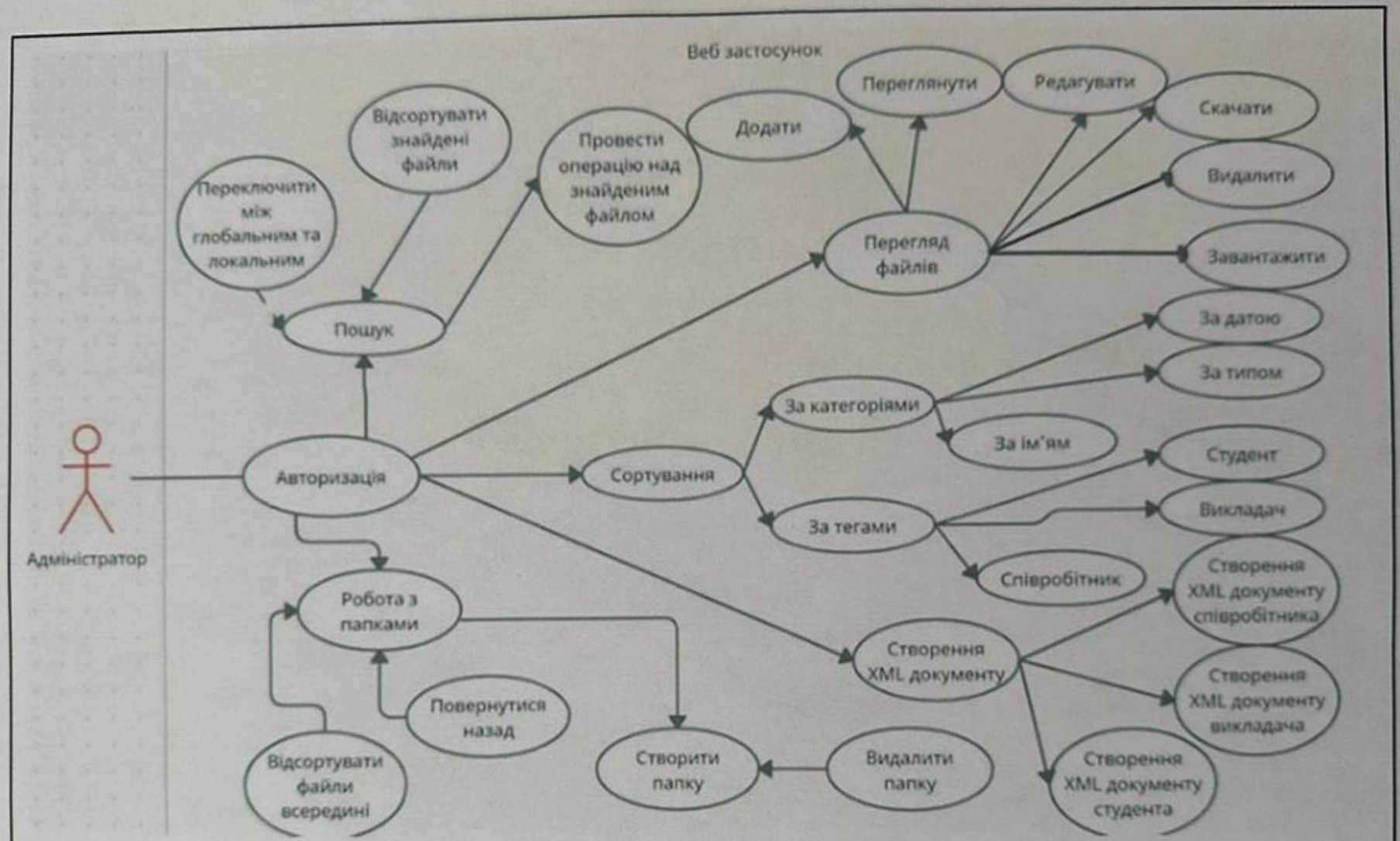
```
        const dbFile = new File({
            name: file.name,

            type,

            size: file.size,
            path: filePath,

            parent: parent?._id,
            user: user._id

        });


        await dbFile.save()
        await user.save()


        res.json(dbFile) }
    catch (e) {

        console.log(e)

        return res.status(500).json({message: "Помилка завантаження"}) }

}

async downloadFile(req, res) {
    try {

        const file = await File.findOne({_id: req.query.id, user: req.user.id})
        const path = fileService.getPath(file)

        if (fs.existsSync(path)) {

            return res.download(path, file.name) }

        return res.status(400).json({message: "Помилка скачування"}) }
    catch (e) {

        console.log(e)

        res.status(500).json({message: "Помилка скачування"}) }

}


async deleteFile(req, res) {
    try {

        const file = await File.findOne({_id: req.query.id, user: req.user.id}) if
        (!file) {

            return res.status(400).json({message: 'Файл не знайдено'}) }

        fileService.deleteFile(file)
        await file.remove()
```

```
            return res.json({message: 'Файл було видалено'}) }
        catch (e) {

            console.log(e)

            return res.status(400).json({message: 'Директорія не порожня'}) }

    }


    async searchFile(req, res) {
        try {

            const searchName = req.query.search

            let files = await File.find({user: req.user.id})

            files = files.filter(file => file.name.includes(searchName))
            return res.json(files)

        } catch (e) {
            console.log(e)

            return res.status(400).json({message: 'Помилка пошуку'}) }

    }


    async uploadAvatar(req, res) {
        try {

            const file = req.files.file

            const user = await User.findById(req.user.id)
            const avatarName = Uuid.v4() + ".jpg"

            file.mv(config.get('staticPath') + "\\" + avatarName)
            user.avatar = avatarName

            await user.save()
            return res.json(user)

    } catch (e) {

                console.log(e)

                return res.status(400).json({message: 'Upload avatar error'})
            }

        }


        async deleteAvatar(req, res) {
            try {

                const user = await User.findById(req.user.id)
                fs.unlinkSync(config.get('staticPath') + "\\" + user.avatar)
                user.avatar = null
```

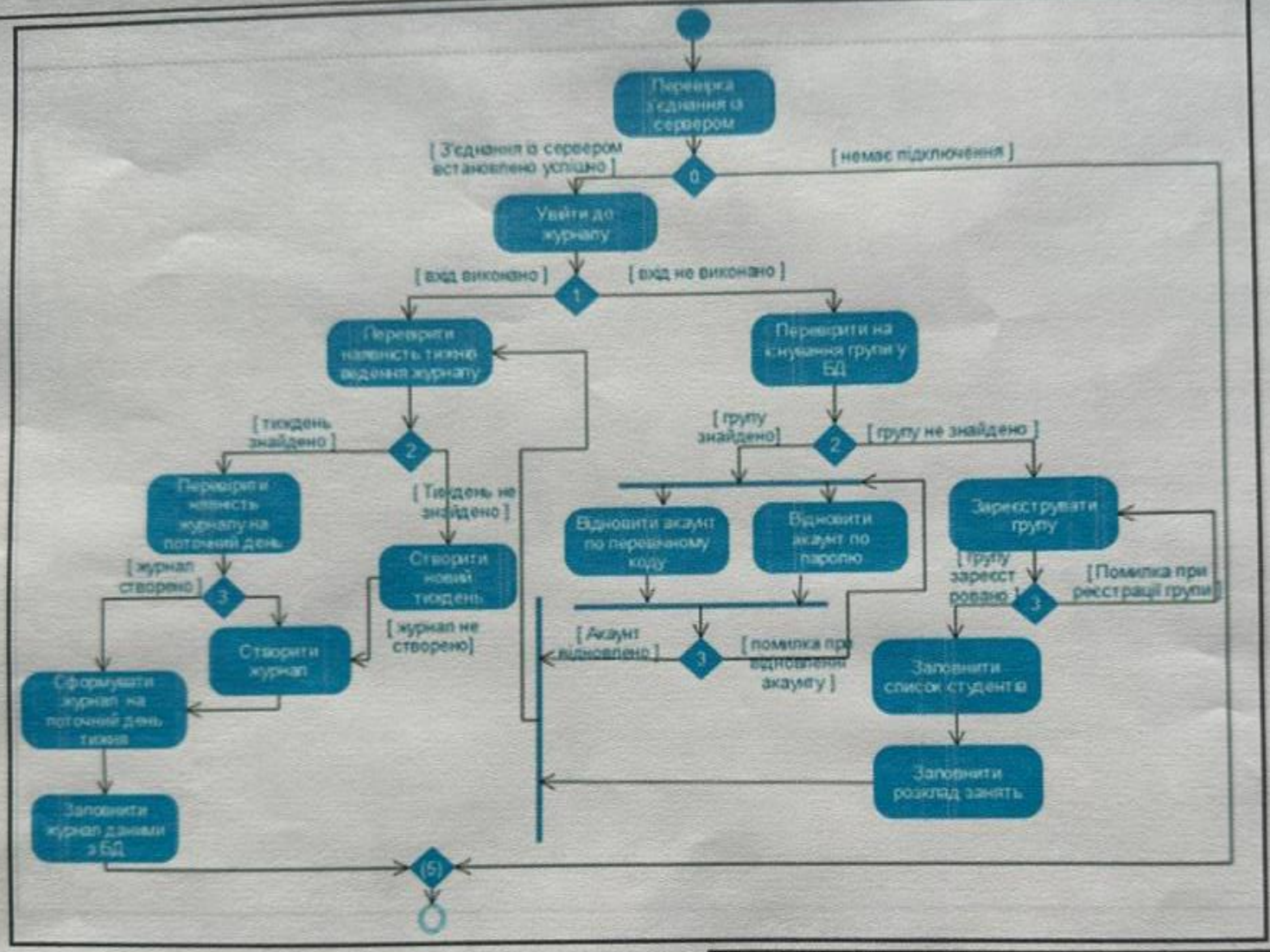
```
            await user.save()
            return res.json(user)

        } catch (e) {
            console.log(e)

            return res.status(400).json({message: 'Delete avatar error'})
        }

    }
}


module.exports = new FileController()
```

ГРАФІЧНА ЧАСТИНА

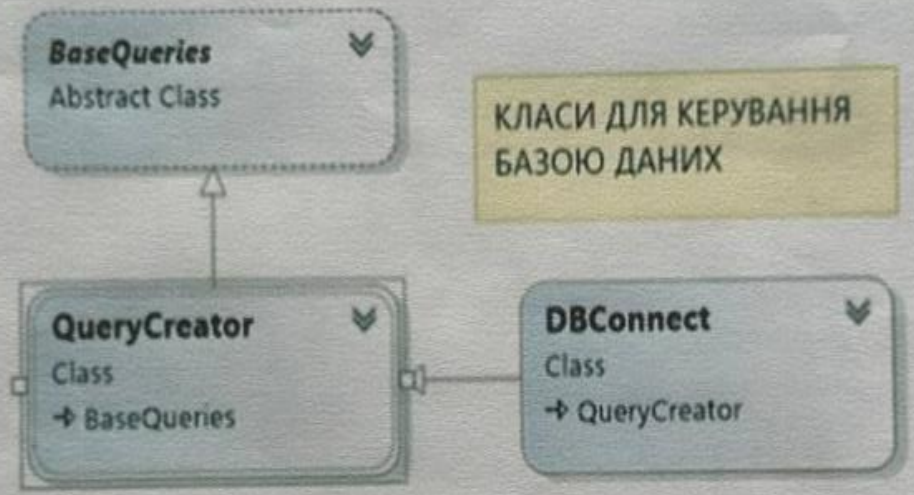
(обов’язкова)



					КвРІПЗ.190139.01.16.Е8		
Зм.	Арк.	№ докум.	Підпис	Дата	Діаграма варіантів використання	Літера	Маса	Масштаб
Розробив		Прилушка В.С.		8.06				
Керівник		Правососька К.		8.06		Аркуш	Аркушів	
Консульт.						1	3	
Н. Контр.		Яцина О.М.		8.06	ХНУ, ІПЗ-19-1			
Зав. каф.		Бадратих Ю.Ф.						



						КвРІПЗ.190139.01.16.Е8	
Зм.	Арк.	№ докум.	Підпис	Дата			
Розробив		Прилушка В.О.		8.06	Діаграма активності	Літера	Маса	Масштаб
Керівник		Правосська Н.І.		8.06				
Консульт.						Аркуш	Аркушів	
Н. Контр.		Яцина О.М.		8.06	ХНУ, ІПЗ-19-1	2	3	
Зав. каф.		Відратська Л.Ф.		8.06				



					КвРІПЗ.190139.01.16.Е8		
Зм.	Арк.	№ докум.	Підпис	Дата			
Розробив		Прилуцька В.О.			Діаграма класів	Літера	Маса	Масштаб
Керівник		Правосська Н.І.						
Консульт.						Аркуш	Аркушів
						3	3
Н. Контр.		Яцина О.М.			ХНУ, ІПЗ-19-1		
Зав. каф.		Бадраток Л.П.					

СУПРОВІДНІ ДОКУМЕНТИ

Anti-Plagiarism v-15.257

Максимальне співпадіння з одним документом 33.0%

Словники перевірки: en_US, ru_RU, ua_UA. **Помилок в документах: 10%**

ID: 115228 Назва: БКР Програмне забезпечення інформаційної системи кафедри для обліку студентів та викладачів Додано в БД: 2023-06-08 Автора: Прилуцька В.О. Керівники: Праворська Н.І. к.п.н. доц. Консультанти: Опоненти:	Документ		Сумарний збіг по Базі Даних	
	Символи	Лексеми	Символи	Лексеми
	47260	747	15741 (33%)	254 (34%)

Джерело плагіату

ID	Опис	Наявність плагіату в документі	
		Символи	Лексеми
114994	Назва: БКР Програмне забезпечення інформаційної системи кафедри для обліку студентів та викладачів Додано в БД: 2023-06-06 Автора: Прилуцька В.О. Керівники: Праворська Н.І. к.п.н. доц. Консультанти: Опоненти:	15695 (33.0%)	252 (34.0%)

UNICHECK

Ім'я користувача:
Кафедра ІПЗ

ID перевірки:
1015506636

Дата перевірки:
08.06.2023 14:08:41 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
08.06.2023 14:11:18 EEST

ID користувача:
100005589

Назва документа: **Кваліфікаційна робота остання Прилуцька**

Кількість сторінок: **56** Кількість слів: **8336** Кількість символів: **66788** Розмір файлу: **3.76 MB** ID файлу: **1015162036**

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

13.2%
Схожість

Найбільша схожість: **3.25%** з Інтернет-джерелом (https://ela.kpi.ua/bitstream/123456789/44123/1/Verbitskii_bakalavr.pd.

10.4% Джерела з Інтернету	573	Сторінка 58
5.36% Джерела з Бібліотеки	131	Сторінка 62

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

37.8%
Вилучень

Деякі джерела вилучено автоматично (фільтри вилучення: кількість знайдених слів є меншою за **8 слів** та **0%**)

Немає вилучених Інтернет-джерел

37.8% Вилученого тексту з Бібліотеки	1	Сторінка 62

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Підозріле форматування	19 сторінок

Завідувачу кафедри
інженерії програмного забезпечення
проф. Бедратюку Л. П.
студента групи ІПЗ-19-1
Прилуцька В.О.
Прізвище, ініціали

ЗАЯВА

Прошу закріпити за мною тему кваліфікаційної роботи освітнього ступеня «бакалавр» за спеціальністю 121 «Інженерія програмного забезпечення»: «Програмне забезпечення інформаційної системи кафедри для обліку студентів та викладачів»

(керівник роботи – Праворська Н.І.)
Прізвище, ім'я, по батькові

05.02.23
Дата

Підпис студента

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

ДЕКЛАРАЦІЯ УЧАСНИКА ОСВІТНЬОГО ПРОЦЕСУ

щодо дотримання академічної доброчесності

Цією декларацією я, Прилуцька Вікторія Олександрівна,

студент IV курсу спеціальності 121 – Інженерія програмного забезпечення, група ІПЗ-19-1

здобувач вищої освіти (шифр та назва спец-ті, курс, академічна група)

підтверджую, що ознайомився (-лась) з Положенням про систему забезпечення академічної доброчесності у Хмельницькому національному університеті та Кодексом академічної доброчесності і **зобов'язуюсь** дотримуватися їх вимог під час освітнього процесу, проведення наукової діяльності, виконання організаційно-адміністративних функцій тощо.

Усвідомлюю, що у разі порушення мною принципів академічної доброчесності нестиму відповідальність перед академічною спільнотою ХНУ згідно з нормами, визначеними Положенням про систему забезпечення академічної доброчесності у Хмельницькому національному університеті, законодавства України.

05 лютого 2023 р.

Підпис

Завідувачу кафедри інженерії програмного забезпечення проф. Бедратюку Л. П.

здобувача вищої освіти
Прилуцька В. О.
Прізвище, ініціали

факультет ІТ, 4 курс, група ІПЗ-19-1

ЗАЯВА

З правилами чинного Положення «Про систему забезпечення академічної доброчесності в Хмельницькому національному університеті» від 01.07.2022, згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту і застосування заходів дисциплінарної та академічної відповідальності, ознайомлена. Про використання програмно-технічних засобів для перевірки кваліфікаційних робіт здобувачів вищої освіти на плагіат оповіщена та надаю свою згоду на обробку й збереження університетом моєї роботи в інституційному репозитарії університету.

Також надаю університету право на передачу моєї роботи для обробки та збереження в базах даних програмно-технічних засобів (Unicheck та/або Anti-Plagiarism) і використання роботи для виявлення академічного плагіату в інших роботах, які перевіряються програмно-технічними засобами та користувачами, що мають доступ до цих програмно-технічних засобів, виключно в обмежених цілях для виявлення плагіату в текстах робіт.

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

06.06.2023
дата

підпис

РІШЕННЯ ЕКСПЕРНОЇ КОМІСІЇ
КАФЕДРИ ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ
ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатами звіту/звітів подібності щодо роботи, продукованими програмно-технічним засобом (ами) перевірки текстів на плагіат:

Назва: «Програмне забезпечення інформаційної системи кафедри для обліку студентів та викладачів»
Автор: Прилуцька Вікторія Олександрівна
Спеціальність: 121 – Інженерія програмного забезпечення
Освітня програма: Освітньо-професійна програма «Інженерія програмного забезпечення»
Науковий керівник: Праворська Наталія Іванівна, кандидат педагогічних наук, доцент
Після аналізу звіту подібності зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі, є законними і не є плагіатом. Робота приймається до захисту.	
2	Виявлені запозичення не є плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи. Робота приймається до захисту, але має бути відкоригована. Відкоригований варіант має бути поданий на кафедру за 2 дні до захисту, разом із заявою щодо самостійності виконання письмової роботи та ідентичності друкованої й електронної версії роботи	**відповідає**
3	Виявлені запозичення не є плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. В зв'язку з цим мета роботи та поставлені завдання не були досягнені. Робота може бути допущена до захисту (наступного року) після того, як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат.	
4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	
5	Інше:	

Підтвердження:

Виконувалася повторна перевірка роботи:

1)За результатом повторної перевірки збіжність з системою Antiplagiarism було виявлено 33,0% схожості з попереднією версією роботи;

2)При повторній перевірці системою Unicheck було виявлено 50,9% схожості з попереднією версією роботи. При виключенні джерел з попередньої версії роботи відсоток збіжності з джерела Інтернету та бібліотеки становить 13,2%;

Сумарний обсяг всіх запозичень, визначений системою виявлення збігів ідентичності/ схожості, складає 13,2% і адресується до 704 джерел, що, з урахуванням наведених обгрунтувань, відповідає характеру теми і свідчить на користь кваліфікаційної роботи.

Дата 08.06

Завідувач кафедри _____ Леонід БЕДРАТЮК

Гарант освітньої програми _____ Леонід БЕДРАТЮК

Керівник кваліфікаційної роботи _____ Наталія ПРАВОРСЬКА

5. Негативні сторони роботи У роботі використано багато текстового матеріалу.

6. Оцінка графічного оформлення та пояснювальної записки Графічне оформлення виконано відповідно до теми кваліфікаційної роботи та подано у вигляді діаграм і рисунків. Пояснювальна записка оформлена згідно вимог чинних стандартів.

7. Відгук про кваліфікаційну роботу в цілому Кваліфікаційна робота заслуговує позитивної оцінки. Матеріал пояснювальної записки структурований, послідовний, чіткий та простий, що дозволяє чітко зрозуміти викладений матеріал у рамках тематики проектування. Графічний матеріал дає можливість наочно побачити деталі проектування системи.

8. Інші зауваження

9. Оцінка кваліфікаційної роботи Кваліфікаційна робота виконана у повному обсязі, відповідає поставленій задачі та заслуговує на оцінку «добре».

РЕЦЕНЗЕНТ Бобровнікова Кіра Юліївна, кандитат технічних наук, доцент кафедри комп'ютерної інженерії та інформаційних систем (КІІС) ХНУ.

"6" 06 202__ р. _____
(підпис)