ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Розробка електронної системи звітності викладацької діяльності мовами PHP, JAVASCRIPT та MYSQL»

на здобуття освітнього ступеня бакалавра
зі спеціальності 121 Інженерія програмного забезпечення
освітньо-професійної програми «Інженерія програмного забезпечення»

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Кіріл ЛАДУТА
(підпис)

Виконав: здобувач вищої освіти групи ПД-43

Кіріл ЛАДУТА

Керівник: Ірина ЩЕБИНА
к.т.н., доцент

Рецензент: _____

Київ 2024

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Інженерії програмного забезпечення

Ступінь вищої освіти Бакалавр

Спеціальність 121 Інженерія програмного забезпечення

Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного забезпечення
_____ Ірина ЗАМРІЙ
« ____ » _____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Ладуті Кірілу Олексійовичу

1. Тема кваліфікаційної роботи: «Розробка електронної системи звітності викладацької діяльності мовами PHP, JavaScript та MySQL»
керівник кваліфікаційної роботи к.т.н., доц. доцент кафедри ІПЗ Ірина ЩЕРБИНА, затверджені наказом Державного університету інформаційно-комунікаційних технологій від «27» лютого 2024 р. № 36.

2. Строк подання кваліфікаційної роботи «28» травня 2024 р.

3. Вихідні дані до кваліфікаційної роботи: опис процесу ведення обліку та звітності у викладацькій діяльності, аналіз існуючих систем звітності та управління викладацькою діяльністю, план розробки системи з включенням етапів розробки, тестування та впровадження.

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

1. Огляд та аналіз існуючих існуючих проблем в системі звітності та визначення вимог до нової системи.

2. Проектування застосунку для ведення звітності викладачів початкової школи.
3. Програмна реалізація та опис функціонування електроннної системи.
4. Тестування застосунку.

5. Перелік графічного матеріалу: *презентація*
1. Аналіз аналогів.
2. Вимоги до програмного забезпечення.
3. Програмні засоби реалізації.
4. Діаграма варіантів використання.
5. Діаграма послідовності.
6. Екранні форми
7. Відео з демонстрацією web-застосунку.
8. Апробація результатів дослідження.

6. Дата видачі завдання«28» лютого 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Підбір та аналіз науково-технічної літератури	28.02.-06.03.2024	
2	Аналіз та дослідження існуючих аналогів	07.03-13.03.2024	
3	Вибір інструментів та технологій для розробки застосунку	14.03-20.03.2024	
4	Проектування застосунку для ведення звітності викладачів початкової школи	21.03-29.03.2024	
5	Програмна реалізація застосунку	30.03-08.04.2024	
6	Тестування застосунку	09.04-28.04.2024	
7	Оформлення роботи: вступ, висновки, реферат	29.04-05.05.2024	
8	Розробка демонстраційних матеріалів	06.05-12.05.2024	
9	Попередній захист роботи	13.05-31.05.2024	

Здобувач вищої освіти _____ Кіріл ЛАДУТА
(підпис)

Керівник
кваліфікаційної роботи _____ Ірина ЩЕРБИНА
(підпис)

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 54 стор., 2 табл., 18 рис., 33 джерел.

Мета роботи – підтримка ведення звітності викладачів початкової школи про результати своєї діяльності за рахунок використання електронної системи.

Об'єкт дослідження – звітність викладачів початкової школи про результати їхньої діяльності.

Предмет дослідження – електронна система для ведення звітності викладачів початкової школи за результатами їхньої діяльності.

Короткий зміст роботи: В роботі проаналізовано наявні проблеми паперової системи звітності, яка є неефективною та трудомісткою, обмеженою у доступності та прозорості інформації, неекологічною та небезпечною з точки зору конфіденційності даних. Досліджено кращі практики зарубіжного досвіду у сфері електронної звітності та визначено функціональні вимоги до нової системи. Розроблено електронна система, яка включає модулі введення даних, генерації звітів, налаштування профілю та аутентифікації користувачів, програмно реалізовані ключові функціональні можливості, зокрема автоматизацію процесів збору, зберігання та обробки даних про навчальну, методичну, наукову та організаційну діяльність викладачів. Проведено тестування системи на відповідність вимогам безпеки, масштабованості та продуктивності. В роботі використано мови програмування PHP, JavaScript та систему керування базами даних MySQL для створення динамічних веб-сторінок, інтерактивних інтерфейсів та надійного зберігання даних.

Сферою використання застосунку є навчальні заклади, де необхідно впровадити ефективну систему звітності викладацької діяльності.

КЛЮЧОВІ СЛОВА: ЕЛЕКТРОННА СИСТЕМА ЗВІТНОСТІ, ВИКЛАДАЦЬКА ДІЯЛЬНІСТЬ, PHP, JAVASCRIPT, MYSQ

ЗМІСТ

ВСТУП

В сучасному світі інформаційних технологій електронні системи звітності набувають все більшого значення в різних сферах діяльності, зокрема в освітній галузі. Розробка ефективної системи звітності викладацької діяльності є актуальним завданням, яке дозволяє оптимізувати процеси управління навчальним закладом, покращити контроль якості освіти та забезпечити прозорість і доступність інформації про роботу викладачів.

Для реалізації електронної системи звітності викладацької діяльності необхідно обрати відповідні технології та інструменти розробки. Серед популярних мов програмування для веб-розробки виділяються PHP та JavaScript, а для роботи з базами даних часто використовується MySQL. Поєднання цих технологій дозволяє створити потужну та гнучку систему, яка відповідатиме вимогам навчального закладу та забезпечуватиме зручний доступ до звітності викладачів .

Постановка завдання

Завданням дослідження є розробка електронної системи звітності викладацької діяльності з використанням мов програмування PHP, JavaScript та бази даних MySQL. Система повинна забезпечувати можливість введення, зберігання, обробки та аналізу даних про навчальну, методичну, наукову та організаційну роботу викладачів.

Мета дослідження

Метою дослідження є створення ефективної та зручної у використанні електронної системи звітності викладацької діяльності, яка дозволить підвищити якість управління навчальним процесом, спростити процедуру формування звітності та забезпечити доступність інформації для всіх зацікавлених сторін.

Результати дослідження

Електронна система звітності викладацької діяльності - це комплексне програмне рішення, яке дозволяє автоматизувати процеси збору, зберігання та обробки даних про роботу викладачів навчального закладу. Така система

забезпечує централізоване управління інформацією про навчальну, методичну, наукову та організаційну діяльність викладачів, що значно спрощує процес формування звітності та прийняття управлінських рішень .

Для розробки електронної системи звітності було обрано мови програмування PHP та JavaScript, а також базу даних MySQL. PHP є популярною мовою веб-програмування, яка дозволяє створювати динамічні веб-сторінки та взаємодіяти з базами даних. JavaScript, в свою чергу, використовується для розробки інтерактивних користувацьких інтерфейсів та забезпечення зручності роботи з системою. MySQL - це потужна реляційна база даних, яка підтримує ефективне зберігання та обробку великих обсягів інформації .

Архітектура електронної системи звітності передбачає клієнт-серверну модель, де веб-додаток, розроблений з використанням PHP та JavaScript, взаємодіє з базою даних MySQL, розташованою на сервері. Така архітектура забезпечує високу масштабованість, продуктивність та безпеку системи.

Електронна система звітності викладацької діяльності включає в себе кілька основних модулів: модуль введення даних, модуль генерації звітів, модуль налаштування профілю, та модуль аутентифікації користувачів. Модуль введення даних дозволяє викладачам вносити інформацію про свою навчальну, методичну, наукову та організаційну роботу через зручний веб-інтерфейс. Модуль генерації звітів забезпечує автоматичне формування різноманітних звітів на основі введених даних, звіти про виконання навчального навантаження, звіти про методичну діяльність тощо.

Після впровадження електронної системи звітності викладацької діяльності необхідно забезпечити її супровід та підтримку. Це включає регулярне оновлення програмного забезпечення, усунення виявлених помилок, надання консультацій та навчання користувачів. Ефективна підтримка системи дозволяє забезпечити її стабільну роботу та адаптацію до мінливих потреб навчального закладу.

Висновки та перспективи

Розробка електронної системи звітності викладацької діяльності з використанням мов програмування PHP, JavaScript та бази даних MySQL є ефективним рішенням для автоматизації процесів управління навчальним закладом. Така система дозволяє централізовано зберігати та обробляти дані про роботу викладачів, генерувати різноманітні звіти та забезпечувати доступність інформації для всіх зацікавлених сторін.

Використання сучасних веб-технологій, таких як PHP та JavaScript, дозволяє створити зручний та інтуїтивно зрозумілий користувацький інтерфейс, який спрощує процес введення та редагування даних. База даних MySQL забезпечує надійне зберігання та ефективну обробку великих обсягів інформації, що є критично важливим для функціонування системи звітності.

Розроблена електронна система звітності викладацької діяльності відповідає вимогам безпеки, масштабованості та продуктивності. Вона дозволяє автоматизувати рутинні процеси формування звітності, зменшити ймовірність помилок та підвищити прозорість роботи викладачів. Крім того, система передбачає можливість інтеграції з іншими інформаційними системами навчального закладу, що дозволяє створити єдиний інформаційний простір та оптимізувати управлінські процеси.

Впровадження електронної системи звітності викладацької діяльності вимагає ретельного планування, тестування та відлагодження. Необхідно забезпечити якісну підтримку та супровід системи, регулярно оновлювати програмне забезпечення та адаптувати його до мінливих потреб навчального закладу. Лише за таких умов електронна система звітності зможе ефективно виконувати свої функції та сприяти підвищенню якості освітнього процесу.

1. АНАЛІЗ ІСНУЮЧИХ ПРОБЛЕМ В СИСТЕМІ ЗВІТНОСТІ ТА ВИЗНАЧЕННЯ ВИМОГ ДО НОВОЇ СИСТЕМИ

1.1. Аналіз наявних недоліків паперової системи звітності

Традиційна паперова система звітності, яка використовується у багатьох навчальних закладах, має низку суттєвих недоліків, які можуть негативно впливати на ефективність роботи викладачів, адміністрації та загальний процес управління навчальним закладом. Ці недоліки можна розділити на кілька основних категорій. Розгялнемо їх:

- Неефективність та трудомісткість процесу;
- Обмежена доступність та прозорість інформації;
- Екологічні проблеми;
- Ризики конфіденційності та безпеки;
- Обмежені можливості для співпраці та спільної роботи;
- Складність внесення змін та оновлень.

Розглянемо ці недоліки більш детальніше:

Неефективність та трудомісткість процесу.

Заповнення паперових форм вимагає значних витрат часу та зусиль з боку викладачів. Традиційні паперові звіти часто містять численні поля, які потрібно ретельно заповнити вручну. Це може бути особливо трудомістким, якщо викладач має багато курсів, великі групи студентів або складні навчальні плани. Окрім того, заповнення форм від руки підвищує ризик помилок та неточностей через людський фактор. Викладачам доводиться витрачати значний час на перевірку та виправлення помилок, замість того, щоб зосередитися на більш важливих аспектах своєї роботи, таких як підготовка до занять чи наукова діяльність.

Збір та обробка великої кількості паперових документів є складним та трудомістким процесом для адміністративного персоналу. У великих навчальних закладах кількість звітів, які потрібно обробити, може бути величезною. Співробітники адміністрації змушені витрачати значну частину свого робочого

часу на сортування, перевірку та введення даних із паперових форм у комп'ютерні системи або електронні таблиці. Це не тільки неефективно з точки зору використання людських ресурсів, але й збільшує ризик допущення помилок під час ручного введення даних.

Ризик втрати або пошкодження важливих документів через фізичну природу паперових носіїв є ще однією серйозною проблемою паперової системи звітності. Паперові документи можуть бути легко пошкоджені через потрапляння води, вогню або інших факторів навколишнього середовища. Крім того, їх можна випадково загубити або вкрасти, що призведе до втрати важливої інформації. Навіть якщо звіти зберігаються в захищених місцях, з часом папір може вицвісти або розірватися, що ускладнює зчитування даних.

Складність аналізу та узагальнення даних, що міститься в паперових звітах, є ще однією значною перешкодою для ефективного управління навчальним закладом. Паперові звіти зазвичай містять величезні обсяги даних, що робить їх аналіз та виявлення тенденцій і закономірностей надзвичайно трудомістким процесом. Адміністрації доводиться вручну обробляти звіти, виконувати складні обчислення та створювати зведені таблиці для отримання загальної картини. Це не тільки забирає багато часу, але й збільшує ризик помилок та неточностей в аналізі.

Обмежена доступність та прозорість інформації.

Доступ до паперових звітів обмежений фізичним розташуванням документів, що ускладнює їх перегляд та аналіз. У традиційній паперовій системі звітності, документи зазвичай зберігаються в певному фізичному місці, наприклад, в офісі або архіві. Це означає, що для перегляду або аналізу звітів необхідно фізично бути присутнім у цьому місці. Така обмеженість доступу створює значні незручності та перешкоди для ефективної роботи.

Викладачам, які можуть працювати з різних місць або навіть віддалено, буде складно отримати доступ до необхідної інформації. Їм доведеться витрачати час та зусилля на переміщення до місця зберігання документів або чекати, поки адміністрація надішле необхідні копії, що зазвичай призводить до затримок у роботі та зниження продуктивності.

Адміністративний персонал також стикається з проблемами доступу до інформації. Якщо їм потрібно переглянути звіти з різних підрозділів або кафедр, вони повинні фізично збирати всі необхідні документи, що може бути дуже трудомістким процесом. Крім того, це ускладнює порівняння та аналіз даних з різних джерел.

Відсутність централізованого сховища даних ускладнює пошук та обмін інформацією між різними підрозділами та викладачами. У паперовій системі звітності дані розпорошені по численних документах та місцях зберігання. Це робить практично неможливим швидкий пошук та доступ до необхідної інформації.

Викладачам буде складно отримати доступ до звітів своїх колег або інших відділів, що може перешкоджати співпраці та обміну досвідом. Адміністрація також матиме труднощі з консолідацією даних з різних джерел для загального аналізу та прийняття рішень.

Складність відстеження змін та історії звітів може призвести до плутанини та непорозумінь. У паперовій системі звітності важко відстежити, хто вносив зміни в документ, коли ці зміни були внесені та які саме зміни були зроблені. Це може призвести до плутанини та конфліктів, коли різні версії звітів циркулюють між різними особами або підрозділами.

Крім того, відсутність чіткої історії змін ускладнює аудит та перевірку звітів. Адміністрації буде важко відстежити, які дані були змінені або хто несе відповідальність за певні записи.

Екологічні проблеми.

Використання великої кількості паперу в традиційній системі звітності негативно впливає на навколишнє середовище, сприяючи вирубці лісів та збільшенню відходів. Папір виробляється з деревної маси, отриманої від вирубки дерев. Хоча деревина є відновлюваним ресурсом, її споживання в промислових масштабах призводить до масштабного знищення лісових насаджень.

Вирубка лісів має серйозні екологічні наслідки, такі як втрата біорізноманіття, порушення природних циклів та балансу в екосистемах, а також

збільшення викидів вуглекислого газу, що сприяє глобальному потеплінню. Ліси відіграють суттєве значення у регулюванні клімату, очищенні повітря та збереженні ґрунтових і водних ресурсів. Їх знищення може мати катастрофічні наслідки для планети.

Крім того, після використання паперові документи часто викидаються, що призводить до збільшення обсягів твердих відходів. Папір, який потрапляє на звалища або спалюється, не лише забруднює навколишнє середовище, але й безповоротно втрачає цінні ресурси, які могли б бути перероблені та використані повторно.

Додаткові витрати на придбання паперу, картриджів для принтерів та інших витратних матеріалів також є значною екологічною проблемою паперової системи звітності. Виробництво паперу вимагає великих обсягів енергії, води та хімічних речовин, що також негативно впливає на навколишнє середовище.

Крім того, виробництво картриджів для принтерів часто включає використання токсичних матеріалів, таких як важкі метали та пластмаси, які важко утилізувати. Ці відходи можуть забруднювати ґрунт і водойми, завдаючи шкоди екосистемам та здоров'ю людей.

Великі обсяги паперу та витратних матеріалів для принтерів також вимагають значних фінансових витрат з боку навчальних закладів. Ці кошти могли б бути інвестовані в інші важливі сфери, такі як навчальні ресурси, дослідження або розвиток інфраструктури.

Перехід на електронну систему звітності дозволить значно скоротити використання паперу та пов'язаних з ним витратних матеріалів, тим самим зменшуючи негативний вплив на навколишнє середовище. Замість друкування звітів, дані можуть зберігатися в електронному вигляді, що значно зменшить потребу в паперових носіях.

Ризики конфіденційності та безпеки.

Паперові документи легше втратити або викрасти, що може поставити під загрозу конфіденційність даних. Фізична природа паперових документів робить їх вразливими до різних загроз, таких як крадіжка, втрата або неправильне поводження. На відміну від електронних даних, які можуть бути захищені паролями, шифруванням та іншими заходами безпеки, паперові документи не мають такого рівня захисту.

Втрата або крадіжка конфіденційних паперових звітів може мати серйозні наслідки для навчального закладу та всіх залучених сторін. Наприклад, якщо будуть втрачені або викрадені звіти з особистими даними студентів чи викладачів, це може призвести до порушення законодавства про захист персональних даних та завдати шкоди репутації закладу.

Крім того, доступ до конфіденційної інформації, такої як оцінки, академічні досягнення чи дослідницькі проєкти, може бути використаний для отримання неправомірної вигоди або завдати шкоди зацікавленим сторонам. Це може підірвати довіру до навчального закладу та його цілісність.

Відсутність належного контролю доступу до паперових звітів також є серйозною загрозою конфіденційності. У багатьох випадках паперові звіти зберігаються в офісах або архівах, до яких можуть мати доступ різні особи, включно з тими, хто не має повноважень переглядати конфіденційну інформацію.

Навіть якщо папери зберігаються в замкнених приміщеннях, завжди існує ризик, що хтось із дозволеним доступом може випадково або навмисно розголосити конфіденційні дані. Це може статися через людську помилку, недбалість або зловмисні наміри.

Крім того, відсутність чіткого відстеження та аудиту доступу до паперових документів ускладнює виявлення та розслідування випадків порушення конфіденційності. Якщо конфіденційна інформація була розголошена, буде важко встановити, хто саме мав доступ до відповідних документів і коли це сталося.

Ризики конфіденційності та безпеки паперової системи звітності можуть мати серйозні наслідки для навчального закладу, такі як втрата репутації, юридична

відповідальність та фінансові збитки. Вони також можуть підірвати довіру студентів, викладачів та інших зацікавлених сторін до закладу.

Впровадження електронної системи звітності з належними заходами безпеки та контролем доступу може значно знизити ці ризики. Електронні дані можуть бути захищені паролями, шифруванням та іншими сучасними технологіями безпеки, що обмежить доступ лише авторизованим користувачам.

Крім того, електронна система може забезпечити детальне відстеження та аудит доступу до даних, що дозволить швидко виявляти та розслідувати будь-які порушення конфіденційності. Це підвищить підзвітність та довіру до системи.

Загалом, ризики конфіденційності та безпеки є серйозною проблемою паперової системи звітності, яку необхідно вирішити для захисту цінної інформації та збереження репутації навчального закладу. Впровадження безпечної та надійної електронної системи звітності є важливим кроком для подолання цих ризиків та забезпечення належного рівня конфіденційності та безпеки даних.

Обмежені можливості для співпраці та спільної роботи.

Складність одночасного доступу та редагування паперових документів кількома користувачами є однією з головних перешкод для ефективної співпраці та спільної роботи в рамках паперової системи звітності. Паперові документи за своєю природою є фізичними об'єктами, які можуть перебувати лише в одному місці одночасно. Це означає, що якщо один користувач працює з паперовим документом, інші не можуть отримати до нього доступ або вносити зміни.

Така ситуація може призвести до значних затримок та неефективності в процесі співпраці. Наприклад, якщо викладач потребує внести зміни у звіт, який на даний момент знаходиться в адміністрації, йому доведеться чекати, поки адміністратор закінчить роботу з документом і передасть його назад. Це може затримати важливі процеси, такі як планування навчальних програм, оцінювання або розробка нових курсів.

Крім того, відсутність можливості одночасного редагування може призвести до виникнення декількох версій одного документа, що в результаті може

спричинити плутанину та непорозуміння. Це ускладнює відстеження змін та узгодження остаточної версії документа.

Відсутність можливості для дистанційної роботи та співпраці між викладачами та адміністрацією є ще однією серйозною проблемою паперової системи звітності. У сучасному світі, де гнучкість та мобільність стають все більш важливими, багато викладачів та адміністративних працівників можуть потребувати можливості працювати віддалено або поза межами навчального закладу.

Однак, паперова система звітності фактично унеможливлює таку дистанційну роботу та співпрацю. Викладачі, які працюють віддалено, не можуть легко отримувати доступ до паперових документів або вносити зміни в них. Це може перешкоджати їхній здатності ефективно виконувати свої обов'язки, такі як підготовка до занять, оцінювання студентських робіт або звітування про свою діяльність.

Аналогічно, адміністративний персонал, який працює віддалено, матиме труднощі з доступом до необхідних паперових звітів, що може уповільнити процес прийняття рішень та управління навчальним закладом.

Відсутність можливості для дистанційної співпраці також може перешкоджати ефективному обміну інформацією та досвідом між викладачами різних відділів, факультетів або навіть навчальних закладів, що може обмежувати можливості для професійного розвитку, обміну передовим досвідом та впровадження інновацій в освітньому процесі.

Обмежені можливості для співпраці та спільної роботи, пов'язані з паперовою системою звітності, можуть мати далекосяжні наслідки для якості освіти та ефективності управління навчальним закладом. Вони можуть уповільнити процеси прийняття рішень, знизити продуктивність роботи викладачів та адміністрації, а також обмежити можливості для професійного розвитку та інновацій.

Впровадження електронної системи звітності з можливостями одночасного доступу, редагування та дистанційної співпраці може вирішити ці проблеми.

Електронні документи можуть бути легко доступними для кількох користувачів одночасно, дозволяючи їм працювати над ними спільно в режимі реального часу. Це підвищить ефективність та швидкість співпраці, а також дозволить уникнути непорозумінь та плутанини, пов'язаних з різними версіями документів.

Крім того, електронна система забезпечить можливість дистанційної роботи та співпраці, дозволяючи викладачам та адміністративному персоналу отримувати доступ до необхідної інформації та вносити зміни незалежно від їхнього місцезнаходження, що підвищить гнучкість та мобільність роботи, а також полегшить обмін досвідом та передовими практиками між різними підрозділами та навчальними закладами.

Складність внесення змін та оновлень.

Внесення змін або виправлень в паперові звіти є складним та трудомістким процесом. Паперові документи за своєю природою є статичними та негнучкими, що робить внесення змін або оновлень досить проблематичним. На відміну від електронних документів, де можна легко редагувати текст або дані, в паперових документах будь-які зміни потребують значних зусиль.

Перш за все, процес виправлення помилок або внесення змін у паперовий звіт є надзвичайно трудомістким. Якщо необхідно виправити певні дані або відомості, викладачу або адміністратору доведеться переписувати весь документ від руки або за допомогою друкарської машинки, що не тільки забирає багато часу, але й збільшує ризик додаткових помилок або неточностей під час переписування.

Крім того, внесення змін у паперові звіти може призвести до плутанини та непорозумінь, оскільки виникають різні версії одного й того самого документа, що ускладнює відстеження та контроль над змінами, що може мати серйозні наслідки, особливо коли йдеться про важливі звіти або дані, пов'язані з оцінюванням або навчальними програмами.

1.2. Дослідження кращих практик зарубіжного досвіду у сфері електронної звітності

Впровадження електронної системи звітності у сфері освіти є важливим кроком для підвищення ефективності та оптимізації процесів управління навчальним закладом. Багато країн у всьому світі вже успішно перейшли на використання подібних систем, і їхній досвід може стати цінним джерелом знань та ідей для розробки власної електронної системи звітності.

У Сполучених Штатах Америки багато університетів та коледжів впровадили електронні системи звітності, що дозволяє викладачам та адміністрації ефективно керувати навчальними програмами, оцінюванням та іншими аспектами освітнього процесу. Одним з прикладів є система Blackboard, яка використовується в багатьох провідних навчальних закладах. Вона забезпечує централізоване сховище даних, де викладачі можуть завантажувати навчальні матеріали, розклади, оцінки та звіти, а студенти мають доступ до своїх курсів та результатів навчання в режимі онлайн.

У Великобританії широко використовується система CELCAT (College Electronic Lecturer Capture and Tracking), яка допомагає викладачам та адміністрації відстежувати відвідуваність, оцінки студентів, планувати навантаження та складати звіти. Ця система забезпечує централізоване зберігання даних, автоматичне генерування звітів та можливість дистанційного доступу для викладачів та адміністраторів.

У Німеччині багато університетів використовують електронну систему під назвою HIS (Hochschul-Informations-System), яка дозволяє керувати навчальними програмами, розкладами, оцінками та звітами. Ця система забезпечує зручний доступ для викладачів та студентів до необхідної інформації, а також полегшує процеси адміністрування та звітності.

У Австралії поширеною є система Moodle, яка використовується в багатьох університетах та коледжах. Вона забезпечує не лише електронну звітність, але й можливості для дистанційного навчання, завантаження навчальних матеріалів,

форумів та спільної роботи. Викладачі можуть легко оновлювати навчальні програми, завантажувати оцінки та складати звіти в межах однієї системи.

У Канаді багато університетів використовують електронну систему під назвою Desire2Learn (D2L). Вона забезпечує централізоване сховище для навчальних матеріалів, розкладів, оцінок та звітів, а також дозволяє викладачам і студентам взаємодіяти в режимі онлайн. Система D2L також має функції для відстеження активності студентів, оцінювання та генерування звітів.

Тому, на нашу думку, дослідження кращих практик зарубіжного досвіду є важливим етапом у процесі розробки електронної системи звітності для українських навчальних закладів. Це дозволить отримати цінні знання, ідеї та рекомендації, які можуть бути використані для створення ефективної, зручної та сучасної системи, що відповідатиме потребам українського освітнього середовища.

1.3. Розробка функціональних вимог до нової системи

Для забезпечення ефективної роботи електронної системи звітності викладацької діяльності необхідно ретельно спланувати та визначити її функціональні вимоги. Функціональні вимоги описують основні функції та можливості, які повинна мати система для задоволення потреб користувачів та досягнення поставлених цілей.

Модуль реєстрації та входу:

- Введення імені користувача.
- Введення пароля та його підтвердження.
- Введення електронної пошти.
- Хешування пароля перед збереженням у базі даних для забезпечення безпеки.
- Валідація форми для перевірки коректності введених даних (наприклад, перевірка довжини пароля, унікальності імені користувача).
- Введення імені користувача та пароля.

- Перевірка наявності введеного користувача у базі даних.
- Верифікація пароля шляхом порівняння хешу введеного пароля з хешем, збереженим у базі даних.
    - Створення сесії для авторизованого користувача та збереження його даних у сесії.

    Особистий кабінет викладача:
    - Відображення основної інформації про викладача.
    - Можливість редагування особистих даних.

    Модуль звітності:
    - Вибір типу звіту (навчальна, методична, організаційна, виховна діяльність).
    - Вибір періоду звітування (тиждень, місяць, весь час, користувацький період).
    - Вибір предмета та класу, якщо це необхідно для звіту.
    - Відображення створених звітів у зручному для користувача форматі.
    - Можливість фільтрації та сортування звітів за різними параметрами.
    - Експорт звітів у формат Excel (XLSX).

    Інтерфейс користувача
    - Інтерфейс, що підлаштовується під різні розміри екранів та пристроїв.
    - Проста та зрозуміла структура меню.
    - Швидкий доступ до основних функцій системи.
    - Використання сучасних технологій для покращення користувацького досвіду (наприклад, AJAX для асинхронного завантаження даних).

Ці функціональні вимоги охоплюють основні потреби для простої та базової електронної системи звітності викладацької діяльності. Система дозволить викладачам зручно вводити дані про свою роботу, генерувати звіти та надавати їх керівництву в електронному вигляді.

2. ВИБІР ТЕХНОЛОГІЙ РОЗРОБКИ ТА АРХІТЕКТУРНОГО РІШЕННЯ

2.1 Аналіз сучасних технологій веб-розробки

HTML5 є сучасною відкритою платформою, розробленою для створення веб-додатків з використанням аудіо, відео, графіки та анімації. Основною метою розробки HTML5 було підвищення рівня підтримки мультимедійних технологій, зберігаючи зворотну сумісність, читабельність коду для людини та простоту аналізу для парсерів.

HTML5 створений як єдина мова розмітки, яка поєднує синтаксичні норми HTML та XHTML. Він удосконалює та оптимізує розмітку документів, а також додає єдиний API для розробки складних веб-додатків. Завдяки новим елементам, HTML5 усуває потребу у використанні сторонніх розширень для відображення сайту в браузері. Зокрема, елементи <audio> та <video> забезпечують інтеграцію та відтворення мультимедійного контенту без необхідності у Flash Player.

Основні Переваги HTML5 такі.

Семантична структура документа - HTML5 включає ряд нових семантичних тегів, які допомагають створити більш змістовну організацію веб-сторінок. До них належать теги <header>, <article>, <footer>, <nav>, <aside>, <section>. Деякі теги HTML4 були визнані застарілими, а інші отримали нові значення та атрибути.

Графічні можливості - новий елемент <canvas> створений для роботи з 2D-графікою, надаючи безліч нових можливостей для впровадження на сторінки. <canvas> є динамічною поверхнею для програмного малювання та різних операцій з графікою.

Мультимедіа - HTML5 додав підтримку мультимедійного контенту за допомогою тегів <video> та <audio>.

Геолокація - функції getCurrentPosition і watchPosition дозволяють зчитувати геолокацію користувачів. Браузери запитують дозвіл перед відправленням геоданих, забезпечуючи конфіденційність користувачів.

JavaScript API - нові API для роботи з графікою та мультимедіа, методи перетягування об'єктів (Drag & Drop) та інші можливості розширюють функціональність веб-додатків.

Нові елементи веб-форм - HTML5 представив нові елементи для веб-форм, що розширюють можливості для розробників та дизайну веб-сторінок.

Локальне сховище - HTML5 впроваджує локальне сховище на стороні клієнта, яке замінює застарілі куки, що дозволяє зберігати більше даних на пристрої користувача з більшими можливостями.

Багато нових функцій HTML5 вже підтримуються сучасними браузерами, що дозволяє прискорити завантаження сторінок і додати нові можливості на сайт. Якщо якась із функцій не підтримується браузером, це не впливає на роботу сайту, адже HTML5 ґрунтується на HTML4, і непідтримувані елементи просто ігноруються.

PHP є скриптовою мовою програмування, створеною для генерації HTML-сторінок на стороні веб-сервера. PHP є однією з найпоширеніших мов у сфері веб-розробок, підтримується більшістю хостинг-провайдерів і базується на відкритому програмному коді. Це робить PHP популярним вибором серед розробників для створення динамічних та інтерактивних веб-додатків.

PHP інтерпретується веб-сервером в HTML-код, який передається на сторону клієнта. На відміну від таких скриптових мов програмування, як JavaScript, користувач не має доступу до PHP-коду, що є перевагою з точки зору безпеки, але може знизити інтерактивність сторінок. Однак PHP може використовуватися для генерування JavaScript-коду, який виконується вже на стороні клієнта.

PHP може бути вбудований безпосередньо в HTML-код сторінок, які коректно обробляються PHP-інтерпретатором. PHP починає виконувати код після першої екрануючої послідовності (<?php) і продовжує виконання до зустрічі з парною екрануючою послідовністю (?>). Цей механізм дозволяє легко інтегрувати PHP в HTML.

Основні Можливості PHP наступні.

Наявність інтерфейсів до багатьох баз даних - PHP має вбудовані бібліотеки для роботи з MySQL, PostgreSQL, mSQL, Oracle, dbm, Hyperware, Informix, InterBase, Sybase. Через стандарт ODBC (Open Database Connectivity Standard) можна підключатися до всіх баз даних, для яких існує драйвер.

Традиційність - PHP здаватиметься знайомою мовою для програмістів, які працюють з різними мовами. Багато конструкцій PHP запозичені з C та Perl, що знижує зусилля при вивченні мови. PHP поєднує переваги Perl та C, має зрозумілий синтаксис і спеціально спрямований на роботу в Інтернеті.

Наявність вихідного коду та безкоштовність - PHP є проєктом з відкритим програмним кодом, що означає, що користувачі можуть вільно використовувати, модифікувати та розповсюджувати PHP. Підтримка користувачів зі всього світу стала важливим чинником у розвитку PHP.

Ефективність - PHP є інтерпретованою мовою, що дозволяє обробляти сценарії з високою швидкістю. За деякими оцінками, PHP-сценарії обробляються швидше за аналогічні програми, написані на Perl. Хоча виконувані файли, отримані шляхом компіляції, працюють швидше, продуктивність PHP є достатньою для створення серйозних веб-додатків.

Bootstrap — це фреймворк, який містить набір HTML і CSS інструментів та шаблонів для швидкого створення веб-сайтів і додатків. Він є доступним для використання з відкритою ліцензією.

Основні Переваги Bootstrap.

Економія часу - використання готових класів і шаблонів дозволяє значно скоротити час розробки, що економить ресурси.

Адаптивність - динамічні макети Bootstrap якісно відображаються на різних пристроях без необхідності внесення змін у розмітку.

Дизайн: Bootstrap забезпечує єдиний стиль для всіх елементів макета, що створює гармонійний вигляд сайту. Фреймворк добре відображається у всіх браузерах.

Простота і відкритість - Bootstrap настільки простий у використанні, що з ним справляються навіть початківці. Відкритий вихідний код дозволяє користувачам брати участь у розробці та модифікувати фреймворк під свої потреби.

Код HTML, JavaScript і CSS в Bootstrap розроблений та перевірений сотнями розробників з усього світу, що забезпечує його надійність та ефективність. Bootstrap використовує мову стилів LESS, яка розширює можливості CSS.

JavaScript є об'єктно-орієнтованою скриптовою мовою програмування, розробленою компанією Netscape, яка використовується мільйонами веб-сторінок та серверних додатків по всьому світу. Вона є розширенням мови ECMA-262 Edition 3 (ECMAScript) з незначними відмінностями від опублікованого стандарту. Попри поширену думку, JavaScript не є "інтерпретатором Java".

JavaScript — це динамічна скриптова мова, яка підтримує прототипне створення об'єктів. Її базовий синтаксис навмисно схожий на Java і C++, щоб зменшити кількість нових концепцій, необхідних для вивчення мови. Такі мовні конструкції, як if, for, while, switch, try...catch схожі на аналогічні конструкції цих мов.

JavaScript може функціонувати як процедурна, так і як об'єктно-орієнтована мова. На відміну від синтаксичних визначень класів у компільованих мовах, таких як C++ чи Java, в JavaScript об'єкти можуть бути створені програмно під час виконання. Після створення об'єкта він може бути використаний як прототип для створення подібних об'єктів.

Динамічні можливості JavaScript включають створення об'єктів під час виконання, змінне число параметрів, динамічне створення скриптів за допомогою eval, інтроспекцію об'єктів за допомогою for...in, а також відновлення вихідного коду (програми на JavaScript можуть декомпілювати тіла функцій назад у вихідний код).

Для того, щоб повідомити браузеру про вбудований у документ HTML сценарій JavaScript, використовують тег <script>. У вихідному коді HTML ця команда повинна бути розміщена між дескрипторами мови.

Сучасні гіпертекстові інформаційні системи можна умовно уявити як сукупність декількох компонентів:

- Системи збереження гіпертекстових об'єктів.
- Системи відображення гіпертекстових об'єктів.
- Системи підготовки гіпертекстових об'єктів.
- Системи програмування перегляду сукупності гіпертекстових об'єктів.

Першими були розроблені системи збереження та відображення (1989-1991 рр.), які продовжують розвиватися. Після 1991 р. з'явилися перші системи підготовки документів, а після 1995 р. — перші мови управління сценаріями перегляду.

Програми перегляду гіпертекстових сторінок традиційно називають скриптами. В програмуванні перегляду гіпертекстових документів web існує два методи:

- Створення скриптів, які інтерпретуються програмою перегляду (технологія JavaScript).
- Компіляція байткоду (технологія Java).

Перший метод дозволяє розробляти гіпертекстові сторінки за допомогою звичайного текстового редактора, що робить їх читабельними для людини-оператора. Другий підхід підвищує ефективність виконання програм та захист коду від несанкціонованих змін. Байткоди забезпечують технологію програмування на Java.

Таблиця 2.1

Порівняльна характеристика сучасних веб-технологій

Критерії	HTML	PHP	JavaScript
Об’єктно-орієнтована	0	1	1
Простота у використанні	1	0	0
Чутливість до регістру змінних	0	1	1
Чутливість до регістру функцій	0	0	1
Оновлюваність файлів на сервері	0	1	0
Підтримка фреймворків	1	1	1
Підтримка баз даних	0	1	0
Сумарний коефіцієнт	2	5	3

Ця таблиця демонструє порівняльні характеристики трьох основних веб-технологій: HTML, PHP та JavaScript, допомагаючи зрозуміти їхні сильні та слабкі сторони у різних аспектах веб-розробки.

Також сучасні технології веб-розробки представлені широким спектром систем керування вмістом сайтів (CMS), серед яких найбільш поширені Joomla, Drupal та WordPress. Розглянемо кожну з цих систем детальніше, зосереджуючись на особливостях, що роблять їх привабливими для розробників.

WordPress є однією з найпопулярніших і найпростіших у встановленні та використанні систем керування вмістом. Вона написана на мові програмування PHP і використовує базу даних MySQL. Ліцензується за GNU General Public License. Застосовується для створення веб-сайтів різної складності — від простих блогів до складних корпоративних сайтів. Вбудована система «тем» і «плагінів» дозволяє значно розширити функціональні можливості WordPress, що робить його універсальним інструментом для веб-розробки.

Drupal є потужною модульною системою керування вмістом з відкритим кодом, також написаною на PHP. Ця CMS відома своєю безпекою та гнучкістю, що робить її популярною серед великих організацій, таких як урядові установи, великі

корпорації, ООН та НАТО. Модулі для Drupal зазвичай розробляються висококваліфікованими програмістами, що забезпечує високу якість і стабільність роботи системи.

Joomla є ще однією популярною системою керування вмістом, написаною на PHP та JavaScript, що використовує базу даних MySQL для збереження даних. Вона є безкоштовним програмним забезпеченням, захищеним ліцензією GPL. Joomla багато переваг, які роблять її зручною у використанні як для розробників, так і для кінцевих користувачів:

Joomla підтримує різні рівні доступу для зареєстрованих користувачів, що дозволяє гнучко керувати правами доступу до адміністративної та фронтальної частини сайту.

Зручна структура розділів і категорій, що дозволяє легко структурувати вміст сайту, що сприяє зручній навігації та управлінню.

Користувачі можуть швидко додавати та редагувати вміст як через адміністративну, так і через фронтальну частину сайту.

Joomla дозволяє налаштовувати відображення різних елементів сайту, таких як останні новини, популярні матеріали, пошук, форми авторизації та голосування.

Матеріали можна редагувати за допомогою вбудованого візуального редактора, схожого на текстові редактори, такі як Microsoft Word.

Joomla підтримує використання додаткових програмних продуктів інших розробників, що дозволяє розширити функціональність сайту.

Тепер розглянемо основні бібліотеки, що використовуються у веб-розробці, та вибір бази даних для сучасних веб-додатків:

Основні бібліотеки у веб-розробці.

jQuery – це швидка, мала і багатофункціональна JavaScript-бібліотека. Вона спрощує такі речі, як обхід та маніпулювання HTML-документами, обробку подій, анімацію та Ajax із набагато простішим API, яке працює у великій кількості браузерів. За допомогою комбінації універсальності та розширюваності, jQuery змінила спосіб написання JavaScript-у мільйонами розробників по всьому світу.

React – це бібліотека для побудови користувацьких інтерфейсів, розроблена Facebook. Вона дозволяє створювати компоненти, які можна використовувати повторно, що значно полегшує процес розробки. React використовує концепцію "віртуального DOM", що дозволяє ефективно оновлювати та рендерити лише ті компоненти, які змінилися. Це робить React швидким та продуктивним інструментом для створення сучасних веб-додатків.

Angular – це платформа та фреймворк для побудови клієнтських додатків за допомогою HTML та TypeScript. Angular забезпечує архітектурні підходи для побудови масштабованих додатків, включаючи потужні інструменти для рендерингу та підтримку багатого функціоналу. Angular розроблений та підтримується Google, що гарантує його стабільність і актуальність.

Vue.js – це прогресивна JavaScript-фреймворк для побудови користувацьких інтерфейсів. На відміну від монолітних фреймворків, Vue.js створений з самого початку, щоб бути поступово прийнятним. Основна бібліотека зосереджується лише на шарі представлення, і її легко інтегрувати з іншими бібліотеками або існуючими проєктами. Vue.js також відмінно підходить для створення складних одно-сторінкових додатків при використанні в комбінації з сучасними інструментами та додатковими бібліотеками.

Ember.js - повний фреймворк для створення амбітних веб-додатків за допомогою архітектурних шаблонів та концепцій, таких як маршрутизація, обмеження доступу та життєвий цикл об'єкта. Ember.js зосереджений на продуктивності та простоті розширення.

Бази даних:

СУБД MySQL є однією з найвідоміших, надійніших і найшвидших з усього сімейства існуючих СУБД. Однією з причин її популярності є правила розповсюдження – MySQL безкоштовна та розповсюджується разом із початковими текстами. Інша причина – це її відносна швидкість. PostgreSQL, наприклад, також розповсюджується під подібною ліцензією, але не набула такого

широкого поширення через помітну повільність. Отже, дві головні причини популярності MySQL: ціна і продуктивність.

MySQL підтримується на близько десяти видах операційних систем, включаючи FreeBSD, OpenBSD, MacOS, OS/2, SunOS, WinXP і Linux. Сьогодні MySQL особливо поширена на платформах Linux і Windows, причому на останніх зустрічається набагато рідше.

Принцип роботи СУБД MySQL аналогічний принципу роботи будь-якої СУБД, що використовує SQL (Structured Query Language, мова структурованих запитів) як командну мову для створення та видалення баз даних і таблиць, для поповнення таблиць даними, для здійснення вибірки даних.

Характеристики СУБД MySQL:

- Написана на мовах C і C++;

- Протестована на широкому спектрі різних компіляторів;

- Працює на багатьох різних платформах;

- Для забезпечення переносимості використовує інструменти GNU;

- Доступні API-інтерфейси для C, C++, Eiffel, Java, Perl, PHP, Python, Ruby;

- Повністю багатопотокова з використанням потоків ядра, може працювати в багатопроцесорних системах;

- Забезпечує транзакційні і нетранзакційні механізми зберігання;

- Використовує дуже швидкі дискові таблиці (MyISAM) зі стисненням індексів;

- Дуже швидка система розподілу пам'яті, заснована на потоках;

- Код MySQL протестований за допомогою інструментів пошуку витоку пам'яті;

- Сервер доступний як окрема програма для використання в клієнт-серверному мережевому середовищі.

Мережева зв'язність СУБД MySQL:

- Клієнти можуть підключатися до сервера MySQL, використовуючи сокети TCP/IP на будь-якій платформі. У Windows-системах клієнти можуть підключатися

з використанням іменованих каналів. У системах на базі UNIX клієнти можуть підключатися через файли сокетів UNIX-доменів.

- Інтерфейс CONNECTOR/ODBC дозволяє MySQL підтримувати клієнтські програми, які використовують ODBC-з'єднання. Наприклад, для підключення до сервера MySQL можна використовувати MS Access. Клієнтське програмне забезпечення може виконуватися під управлінням Windows або UNIX. Початкові тексти інтерфейсу CONNECTOR/ODBC доступні. Підтримуються всі функції ODBC, а також багато інших.

- Інтерфейс CONNECTOR/JDBC дозволяє MySQL взаємодіяти з клієнтськими програмами на Java, в яких використовуються JDBC-підключення. Клієнтське програмне забезпечення може виконуватися під управлінням Windows або UNIX. Початкові тексти інтерфейсу CONNECTOR/JDBC доступні.

Oracle Database – перша в світі база даних, розроблена спеціально для роботи в мережах розподілених обчислень Grid, призначена для ефективного розгортання на базі різних типів устаткування, від невеликих серверів до потужних симетричних багатопроцесорних серверних систем, від окремих кластерів до корпоративних розподілених обчислювальних систем. СУБД надає можливість автоматичної настройки і управління, що робить її використання простим і економічно вигідним.

Oracle Developer Suite – повний набір інтегрованих засобів для розробки web-додатків. Він включає зручне інтегроване середовище розробки із засобами моделювання, програмування, розробки компонентів, бізнес-аналізу і складання звітів. Всі ці засоби використовують загальні ресурси, що дозволяє спільно працювати над одним проєктом групі розробників, маючи одночасний доступ до ресурсів бази даних, не заважаючи один одному, а вносячи паралельні зміни до однієї і тієї ж бази даних.

Сервер Oracle забезпечує ефективні і дієві рішення для основних засобів баз даних: управління великими базами даних і контроль управління простором. Oracle підтримує найбільші бази даних потенційного розміру до сотень гігабайт. Щоб

забезпечити дієвий контроль за використанням дорогих дискових пристроїв, він надає повний контроль розподілу простору. Підтримує велике число користувачів, що одночасно виконують різноманітні застосування, які оперують одними і тими ж даними. Сервер мінімізує суперництво за дані і гарантує узгодженість даних.

У комп'ютерному середовищі, сполучених мережами, Oracle комбінує дані, що фізично знаходяться на різних комп'ютерах, у одну логічну базу даних, до якої мають доступ всі користувачі мережі. Розподілені системи володіють таким же ступенем прозорості для користувачів і узгодженості даних, що і нерозподілені системи, надаючи при цьому переваги управління локальною базою даних.

Oracle автоматично підтримує цілісність даних, дотримуючи "організаційні правила", які диктують стандарти даних. Як наслідок, усуваються витрати на кодування і супровід перевірок у численних додатках бази даних. Програмне забезпечення Oracle дозволяє різним типам комп'ютерів і операційних систем спільно використовувати інформацію за допомогою мереж.

2.2 Обрання мов програмування (PHP, JavaScript) та СКБД (MySQL) для проєкту

При розробці веб-додатку важливим є обрання відповідних мов програмування та системи керування базами даних (СКБД). В нашому проєкті ми обрали PHP, JavaScript та MySQL з огляду на їхні переваги та відповідність вимогам проєкту.

PHP – це серверна мова програмування, яка широко використовується для створення динамічних веб-додатків. Її переваги включають:

- Легкість у вивченні та використанні.
- Відмінна інтеграція з базами даних.
- Висока швидкість обробки запитів.
- Велика кількість бібліотек і фреймворків.
- Відкритий вихідний код, що забезпечує велику спільноту розробників та підтримку.

JavaScript – це клієнтська мова програмування, яка використовується для створення інтерактивних елементів на веб-сторінках. Вона має такі переваги:

- Висока продуктивність завдяки використанню вбудованих движків браузерів.
- Можливість створення динамічних і інтерактивних інтерфейсів.
- Підтримка різних фреймворків і бібліотек (React, Angular, Vue.js), що полегшує розробку.
- Широка сумісність з іншими мовами та технологіями.

MySQL – це одна з найпопулярніших СКБД з відкритим кодом, яка забезпечує ефективне управління даними. Основні переваги MySQL включають:

- Висока продуктивність та швидкість обробки запитів.
- Підтримка різних типів даних та транзакцій.
- Широка підтримка спільноти та наявність великої кількості документації.
- Можливість інтеграції з різними мовами програмування, включаючи PHP.
- Відкритий код та безкоштовність використання.

Таблиця 2.2

Порівняльна характеристика СУБД (5 бальна)

Критерії	MySQL	Oracle
Кількість функцій	3	5
Ціна	5	3
Гнучкість	5	4
Надійність	4	5
Технічна підтримка	4	3
Сумарний коефіцієнт	21	20

Як видно з таблиці, MySQL демонструє вищі оцінки в аспектах ціни та гнучкості, що робить її оптимальним вибором для нашого проєкту. Однак Oracle

також має свої переваги, такі як більша кількість функцій та надійність, що може бути корисним для інших типів проєктів.

2.3. Розробка архітектури клієнт-серверної системи

Проєкт "TeachReport" передбачає розробку клієнт-серверної системи, яка забезпечить користувачам можливість реєстрації, аутентифікації та перегляду звітів про викладацьку діяльність. Запланована архітектура дозволить розділити функціональні можливості системи між клієнтською частиною, яка працюватиме на пристроях користувачів, та серверною частиною, яка оброблятиме дані та забезпечуватиме збереження і безпеку інформації. У цьому розділі буде розглянуто основні етапи та аспекти розробки нашої клієнт-серверної архітектури.

Першим етапом у розробці нашої клієнт-серверної системи стане проєктування бази даних. База даних слугуватиме сховищем для всієї інформації про користувачів, звіти та інші важливі дані. Ми плануємо використовувати MySQL як систему керування базами даних (СКБД) завдяки її надійності та широким можливостям. Структура бази даних буде включати таблиці для зберігання інформації про користувачів (користувацькі дані, хешовані паролі, електронні адреси), звіти (типи звітів, періоди, пов'язані дані) та інші необхідні дані. Особлива увага буде приділена забезпеченню цілісності даних та оптимізації запитів для підвищення швидкодії системи.

Клієнтська частина системи буде розроблена з використанням сучасних веб-технологій, таких як HTML, CSS та JavaScript. Ми плануємо створити інтуїтивно зрозумілий інтерфейс, який дозволить користувачам легко взаємодіяти з системою. Використання JavaScript дозволить додати динамічні елементи, які покращать користувацький досвід, такі як валідація форм в режимі реального часу та інтерактивні елементи інтерфейсу.

На серверній частині планується використовувати PHP як основну мову програмування для обробки запитів від клієнтської частини та взаємодії з базою даних. PHP є популярним вибором для розробки веб-додатків завдяки своїй

простоті у використанні та широким можливостям. Серверна логіка включатиме обробку реєстрації та аутентифікації користувачів, генерацію звітів, а також забезпечення безпеки даних. Ми приділимо особливу увагу захисту від таких загроз, як SQL-ін'єкції та міжсайтовий скриптинг (XSS).

З основних функцій нашої системи буде можливість генерації та перегляду звітів. Для цього планується розробити модуль генерації звітів, який дозволить користувачам вибирати типи звітів, періоди та інші параметри. Звіти будуть формуватися на основі даних, що зберігаються в базі даних, і надаватимуться у зручному форматі для перегляду та експорту. Ми розглядаємо можливість використання бібліотек для генерації звітів у форматі Excel, що дозволить користувачам зберігати та аналізувати дані у звичному вигляді.

Взаємодія між клієнтською і серверною частинами буде здійснюватися за допомогою HTTP-запитів. Клієнтська частина надсилатиме запити на сервер для отримання даних або виконання певних дій, таких як реєстрація або генерація звіту. Серверна частина оброблятиме ці запити, взаємодіятиме з базою даних і повертатиме результати у вигляді HTML-сторінок або JSON-об'єктів. Така архітектура забезпечить гнучкість та масштабованість системи, дозволяючи додавати нові функції та модулі в майбутньому.

Значну увагу буде приділено безпеці системи. Реєстрація та аутентифікація користувачів включатимуть використання хешування паролів з використанням сучасних алгоритмів, таких як bcrypt, щоб забезпечити безпеку збережених паролів. Крім того, буде реалізовано механізм сесій для підтримки авторизації користувачів протягом їхньої роботи в системі. Також планується впровадження захисту від основних веб-загроз, включаючи SQL-ін'єкції та XSS-атаки, шляхом використання підготовлених запитів і валідації введених даних.

Ще одним важливим аспектом розробки буде забезпечення масштабованості та продуктивності системи. Ми плануємо використовувати кешування для зменшення навантаження на базу даних і підвищення швидкодії системи. Для цього розглядається можливість використання таких технологій, як Memcached або Redis. Крім того, буде розроблено оптимізовані запити до бази даних і впроваджено індексацію таблиць для прискорення доступу до даних.

Завершальним етапом розробки буде тестування та налаштування системи. Ми плануємо провести всебічне тестування, включаючи модульні тести, інтеграційні тести та тестування користувацького інтерфейсу, щоб забезпечити високу якість та надійність нашого додатку. Після завершення тестування система буде розгорнута на виробничому сервері, де буде налаштована для забезпечення високої доступності та безпеки.

3. РОЗРОБКА ЕЛЕКТРОННОЇ СИСТЕМИ ЗВІТНОСТІ

3.1. Розробка модуля введення даних

Розробка модуля введення даних для проєкту "TeachReport" була виконана з метою забезпечення зручного та ефективного способу введення та зберігання інформації про звітність викладацької діяльності. Цей модуль дозволяє користувачам вводити дані, які потім обробляються і зберігаються у базі даних. Процес розробки модуля включає декілька ключових етапів: створення форми введення даних, обробка введених даних на сервері, валідація даних, а також збереження їх у базі даних.

Форма введення даних була створена з використанням HTML та CSS для забезпечення зручного інтерфейсу користувача. Основні елементи форми включають поля для введення тексту, вибору дат, випадаючі списки та кнопки. Ось приклад нашої форми введення даних:

```
<!DOCTYPE html>
<html lang="uk" data-theme="light">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <link rel="stylesheet" href="../css/report.css">
  <title>Введення даних</title>
</head>
<body>
<div class="container">
  <form id="dataEntryForm" method="POST">
    <label for="category">Категорія:</label>
    <select name="category" id="category" required>
      <option value="">--Виберіть категорію--</option>
      <option value="teaching">Навчальна</option>
      <option value="methodical">Методична</option>
      <option value="organizational">Організаційна</option>
      <option value="educational">Виховна</option>
    </select>

    <label for="period">Період:</label>
    <input type="date" name="period" id="period" required>
```

```
        <label for="subject">Предмет:</label>
        <input type="text" name="subject" id="subject" required>

        <label for="class">Клас:</label>
        <input type="text" name="class" id="class" required>

        <button type="submit">Ввести дані</button>
    </form>
</div>
</body>
</html>
```

Після того, як користувач вводить дані і натискає кнопку "Ввести дані", форма відправляє дані на сервер за допомогою методу POST. Сервер обробляє ці дані, використовуючи PHP. Основні етапи обробки включають отримання даних з форми, валідацію даних та збереження їх у базі даних. Ось приклад серверної обробки:

```
<?php
include '../config.php';

if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    $category = $_POST['category'];
    $period = $_POST['period'];
    $subject = $_POST['subject'];
    $class = $_POST['class'];

    // Валідація даних
    if (empty($category) || empty($period) || empty($subject) || empty($class)) {
        echo "Всі поля є обов'язковими для заповнення.";
    } else {
        // Збереження даних у базі даних
        $query = "INSERT INTO reports (category, period, subject, class) VALUES
('$category', '$period', '$subject', '$class')";
        if (mysqli_query($conn, $query)) {
            echo "Дані успішно введені.";
        } else {
            echo "Помилка: " . $query . "<br>" . mysqli_error($conn);
        }
    }
}
?>
```

На стороні сервера перевіряється, чи всі обов'язкові поля були заповнені. Якщо будь-яке з полів не було заповнено, користувачу відображається відповідне повідомлення про помилку.

Після успішної валідації дані зберігаються у базі даних за допомогою SQL-запиту. У даному прикладі використовується база даних MySQL, де дані вводяться в таблицю reports (буде розглянуто в наступному підрозділі).

3.2. Розробка модуля формування звітів

Розробка модуля формування звітів в рамках прокту "TeachReport" була складним і багатогранним процесом, який вимагав детального планування, розробки та тестування. Основною метою цього модуля було забезпечення користувачів можливістю генерувати різноманітні звіти про викладацьку діяльність, що допомогло б в автоматизації і оптимізації управлінських процесів в освітніх установах. Для досягнення цієї мети була створена система, яка включає декілька основних компонентів: інтерфейс користувача, серверну обробку, взаємодію з базою даних та генерацію звітів у форматі Excel.

На першому етапі розробки була створена структура проєкту, яка включала розробку класів та функцій для обробки запитів користувачів, а також для взаємодії з базою даних. Основними класами в цьому модулі стали ReportGenerator, DatabaseHandler, ReportForm та ExcelExporter. Клас ReportGenerator відповідав за логіку формування звітів, включаючи вибір даних з бази даних та підготовку їх до експорту. Клас DatabaseHandler був створений для забезпечення надійної та ефективної взаємодії з базою даних, включаючи виконання SQL-запитів та обробку результатів. Клас ReportForm відповідав за створення і валідацію форми введення даних, а ExcelExporter - за експорт даних у формат Excel.

Процес розробки модуля формування звітів розпочався з проєктування інтерфейсу користувача. Важливою частиною цього етапу було створення інтуїтивно зрозумілої і зручної форми введення даних, яка дозволяла б користувачам вибирати категорії звітів, періоди, предмети та інші параметри. Для

цього був створений клас ReportForm, який включав методи для генерації HTML-форми та її валідації на стороні клієнта. Важливим аспектом було забезпечення динамічної зміни форми в залежності від вибраних параметрів, що було реалізовано за допомогою JavaScript.

Наступним кроком було створення серверної логіки для обробки запитів на формування звітів. Клас ReportGenerator був основним компонентом на цьому етапі. Він включав методи для отримання даних з бази даних, їх обробки та підготовки до експорту. Важливою частиною роботи цього класу була обробка різних запитів користувачів і генерація відповідних SQL-запитів для вибору даних з таблиці reports. Для забезпечення ефективної роботи з базою даних був розроблений клас DatabaseHandler, який включав методи для встановлення з'єднання з базою даних, виконання SQL-запитів та обробки результатів. Це дозволило зробити код більш організованим та зручним для підтримки.

Особливу увагу було приділено забезпеченню коректної валідації введених даних. Валідація виконувалася як на стороні клієнта, так і на стороні сервера. На стороні клієнта була реалізована перевірка заповненості всіх обов'язкових полів форми, а також коректності введених даних. На стороні сервера були реалізовані додаткові перевірки для забезпечення безпеки та надійності системи, включаючи перевірку на SQL-ін'єкції та інші потенційні загрози.

З основних аспектів модуля, як вже було визначено, було формування звітів була генерація звітів у форматі Excel. Для цього був розроблений клас ExcelExporter, який включав методи для створення та налаштування Excel-файлів, додавання даних та форматування. Для цього був розроблений клас ExcelExporter, який включав методи для створення та налаштування Excel-файлів, додавання даних та їх форматування. Використання JavaScript-бібліотеки XLSX, доступної через CDN (наприклад, з використанням ajax/libs/xlsx), дозволило реалізувати всі необхідні функції для експорту даних у формат Excel. Це забезпечило користувачам зручний спосіб отримання звітів у популярному форматі, який легко відкривається і редагується у більшості офісних додатків.

Для забезпечення коректної роботи модуля формування звітів була проведена серія тестувань. Це включало як ручне тестування, так і автоматизоване тестування з використанням PHPUnit. Основна увага була приділена перевірці коректності генерації звітів, обробці помилок, а також забезпеченню безпеки системи. Результати тестувань дозволили виявити і виправити помилки, а також оптимізувати роботу модуля.

Останнім етапом розробки модуля формування звітів була його інтеграція в загальну систему "TeachReport". Це включало налаштування маршрутизації, забезпечення взаємодії з іншими модулями системи, а також додавання відповідних посилань і кнопок в інтерфейсі користувача. Важливою частиною цього етапу було забезпечення зручного доступу до модуля формування звітів для всіх користувачів системи.

3.3. Реалізація інтерфейсу системи

Основними завданнями цього етапу було створення привабливого дизайну, забезпечення легкої навігації та інтерактивності, а також оптимізація для різних пристроїв і екранів. Процес розробки інтерфейсу включав декілька освновних факторів:

- Проєктування користувацького інтерфейсу (UI);
- Розробка користувацького досвіду (UX);
- Використання сучасних веб-технологій, інтеграція з серверною частиною;
- Тестування.

На першому етапі була проведена детальна робота з проєктування користувацького інтерфейсу (UI). Основна увага приділялася створенню інтуїтивно зрозумілих форм, кнопок, меню та інших елементів, які забезпечували б зручний доступ до функцій системи. Дизайн інтерфейсу базувався на сучасних тенденціях веб-дизайну, з акцентом на мінімалізм, читабельність та привабливість.

Важливою частиною процесу розробки було забезпечення зручної навігації по системі. Для цього був створений зрозумілий і логічно структурований інтерфейс, який включав головне меню, підменю, кнопки швидкого доступу та інші навігаційні елементи. Основною метою було забезпечити користувачам можливість швидко знаходити необхідні функції та інформацію без зайвих зусиль. Особливу увагу приділили організації розділів та категорій, що дозволило зробити інтерфейс більш структурованим і зрозумілим.

Для забезпечення інтерактивності інтерфейсу використовувалися сучасні веб-технології, такі як HTML5, CSS3 та JavaScript. За допомогою CSS3 були реалізовані анімації та переходи, що зробило інтерфейс більш динамічним і привабливим. JavaScript використовувався для створення інтерактивних елементів, таких як випадаючі списки, модальні вікна, динамічні форми та інші функції, які покращували користувацький досвід. Використання бібліотек та фреймворків, таких як jQuery та Bootstrap, дозволило значно прискорити процес розробки та забезпечити крос-браузерну сумісність.

Інтеграція з серверною частиною була для того, щоб дії користувачів, такі як введення даних, створення звітів, перегляд інформації, були пов'язані з серверною логікою. Для цього використовувалися AJAX-запити, які дозволяли оновлювати вміст сторінок без перезавантаження, що значно покращувало продуктивність і зручність використання. Дані передавалися у форматі JSON, що забезпечувало швидкий і ефективний обмін інформацією між клієнтом і сервером.

Тестування інтерфейсу було, щоб дозволити виявити і виправити помилки, а також покращити зручність використання системи. Тестування проводилося як вручну, так і автоматично, з використанням інструментів для тестування, таких як Selenium. Особлива увага приділялася перевірці коректності роботи на різних браузерах і пристроях, а також тестуванню інтерактивних елементів і валідації форм.

З основних факторів реалізації інтерфейсу була його інтеграція з іншими модулями системи. Це включало забезпечення коректної роботи всіх функцій, таких як реєстрація користувачів, авторизація, введення даних, формування звітів,

перегляд і редагування інформації. Всі ці функції повинні були працювати в єдиному інтерфейсі, що забезпечувало зручність використання та цілісність системи. Для цього були розроблені відповідні API-запити, що дозволило забезпечити надійну взаємодію між клієнтом і сервером.

Процес розробки інтерфейсу включав також роботу над безпекою системи. Всі форми введення даних були захищені від SQL-ін'єкцій та інших потенційних загроз. Для цього використовувалися методи валідації та очищення введених даних, а також налаштування прав доступу для різних користувачів, що забезпечило безпеку зберігання і обробки даних, а також захист системи від несанкціонованого доступу.

3.4. Візуалізація структури та процесів системи

Основними завданнями цього етапу було створення діаграм, що ілюструють структуру системи, її функціональні можливості та взаємодію компонентів. Процес візуалізації включав декілька основних аспектів:

- Розробка діаграми сценаріїв використання (Use Case діаграми);
- Створення діаграми діяльності (Activity Diagram);
- Створення діаграми послідовності (Sequence Diagram);

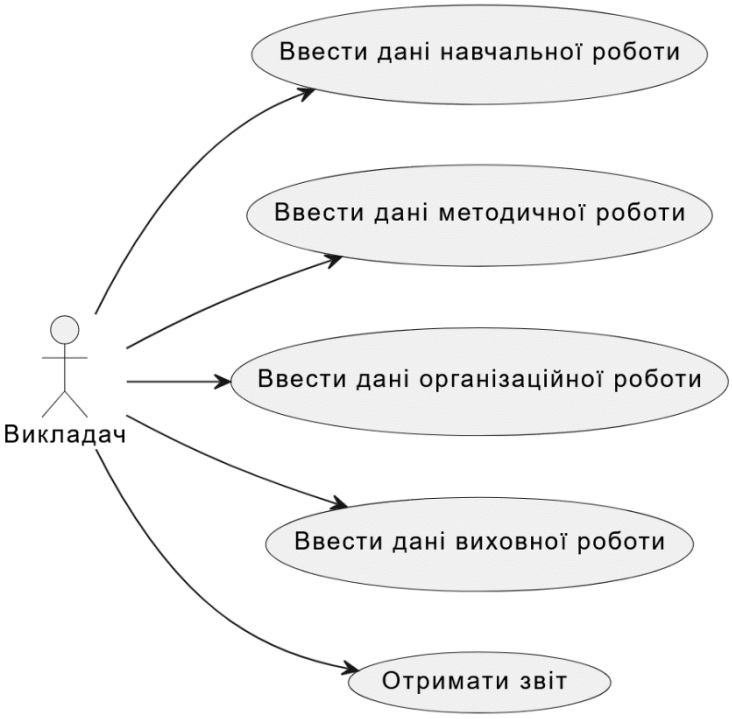


Рис. 3.1 Діаграма варіантів використання

На першому етапі була проведена детальна робота з розробки діаграми сценаріїв використання (Use Case діаграми). Основна увага приділялася визначенню ключових функцій системи та взаємодії користувачів з нею.

Ця діаграма сценаріїв використання писує взаємодію викладача з системою для виконання наступних завдань: Викладач може вводити дані навчальної роботи, дані методичної роботи, дані організаційної роботи та дані виховної роботи. Крім того, викладач має можливість отримати звіт на основі введених даних.

Наступним кроком було створення діаграми діяльності (Activity Diagram), яка показує послідовність дій та логіку виконання основних процесів у системі. Основна увага була приділена:

- Візуалізації потоку дій при введенні даних;
- Деталізації процесу генерації звітів, включаючи всі проміжні етапи;
- Опису процесу перегляду звітів користувачем;

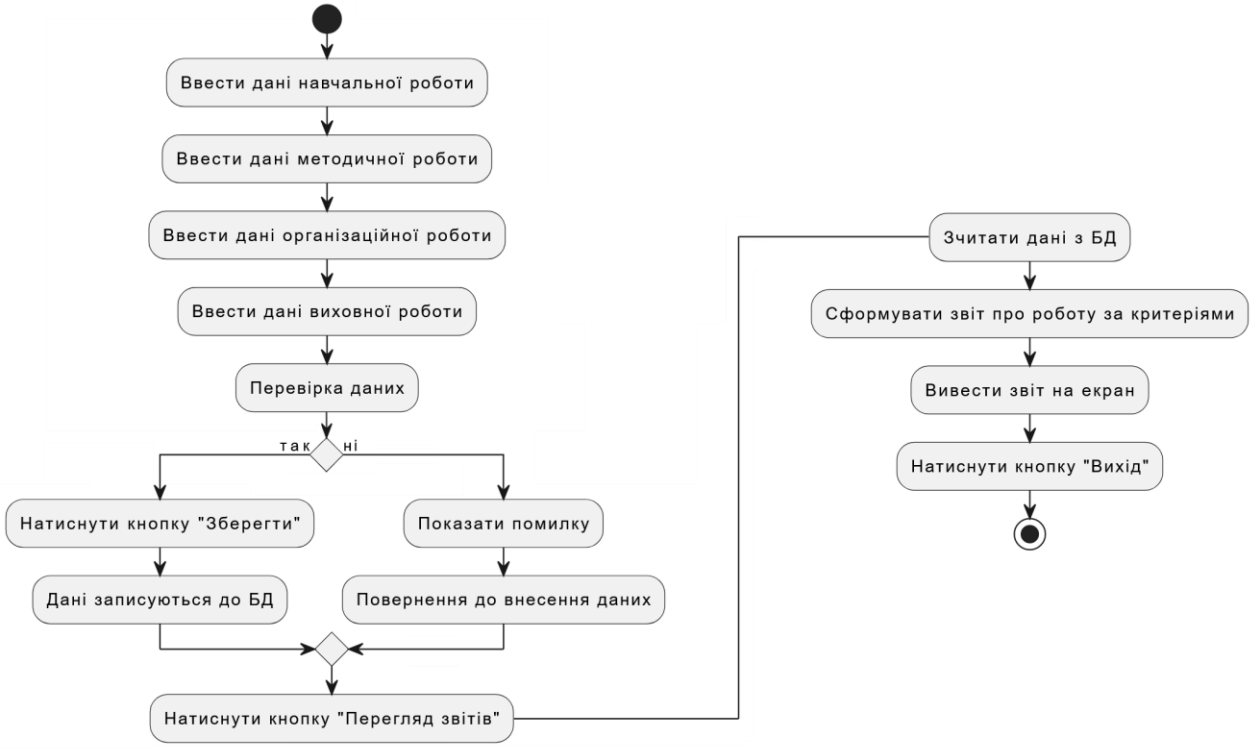


Рис. 3.2 Діаграма діяльності

Ця діаграма діяльності описує процес взаємодії викладача з системою для введення різних типів даних, їх перевірки, збереження і формування звіту.

Процес починається з введення даних навчальної роботи, потім даних методичної роботи, далі даних організаційної роботи та виховної роботи. Після введення всіх даних відбувається перевірка їхньої правильності. Якщо дані вірні, викладач натискає кнопку "Зберегти", і дані записуються до бази даних. Якщо дані містять помилки, система показує помилку, і викладач повертається до внесення даних для їх виправлення. Після успішного збереження даних викладач може натиснути кнопку "Перегляд звітів". Система зчитує дані з бази даних, формує звіт про роботу за визначеними критеріями та виводить його на екран. Взаємодія із системою завершується натисканням кнопки "Вихід".

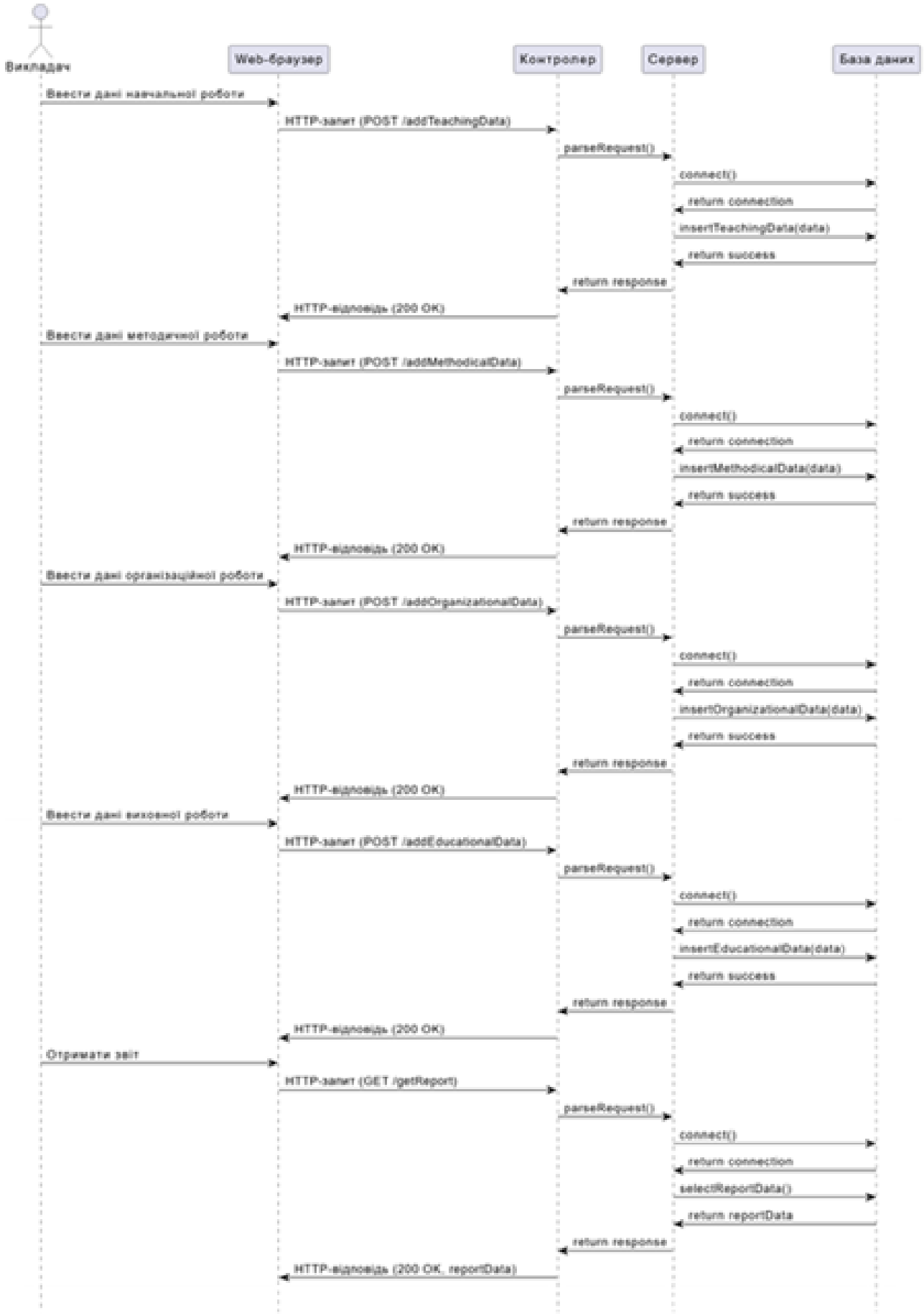


Рис. 3.3 Діаграма послідовності

Ця діаграма послідовності (Sequence Diagram) ілюструє процес обміну даними між викладачем, веб-браузером, контролером, сервером і базою даних. Викладач через веб-браузер надсилає HTTP-запити для введення даних навчальної, методичної, організаційної та виховної роботи. Контролер обробляє запити, встановлює з'єднання з базою даних через сервер і записує відповідні дані. Після успішного збереження даних сервер повертає повідомлення про успіх, і контролер відправляє HTTP-відповідь (200 OK) до веб-браузера. Для отримання звіту викладач надсилає GET-запит, який обробляється аналогічним чином: контролер отримує дані звіту з бази даних і повертає їх до веб-браузера. Діаграма демонструє цикл взаємодії між користувачем і системою, включаючи введення даних і отримання звіту.

4. ТЕСТУВАННЯ, ВПРОВАДЖЕННЯ ТА СУПРОВІД СИСТЕМИ

4.1. Тестування функціональності та інтерфейсу системи

Форма входу є першою точкою взаємодії користувача з системою. Вона забезпечує безпеку доступу до системи, дозволяючи користувачам вводити свої облікові дані для авторизації. Користувач вводить свою електронну пошту та пароль, після чого система перевіряє правильність введених даних. У разі успішного входу користувач перенаправляється на головну сторінку. Якщо введені дані неправильні, користувач отримує повідомлення про помилку. Форма входу також включає посилання на реєстрацію для нових користувачів.

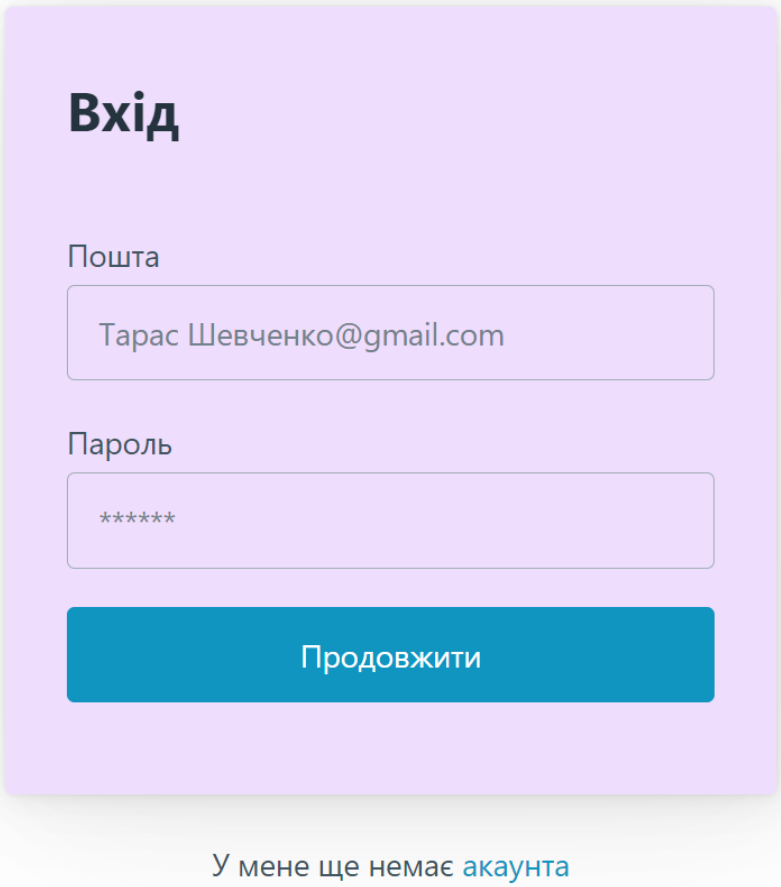


Рис. 4.1 Форма входу для користувача

Форма реєстрації дозволяє новим користувачам створювати облікові записи. Користувач вводить своє ім'я, електронну пошту, пароль та підтвердження пароля. Додатково, користувач може завантажити зображення профілю. Після заповнення форми та натискання кнопки "Продовжити", новий обліковий запис створюється і зберігається в базі даних. У разі виникнення помилок, наприклад, якщо паролі не співпадають або електронна пошта вже використовується, користувач отримує відповідне повідомлення.

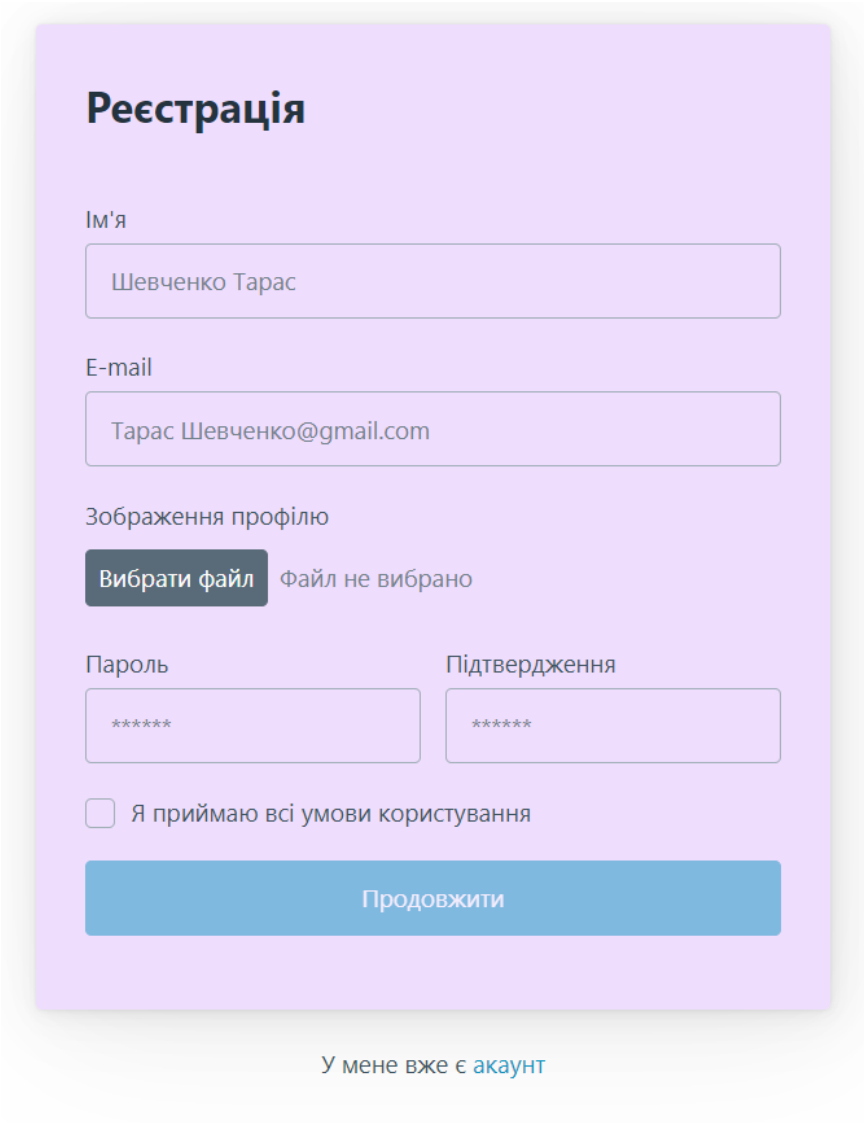


Рис. 4.2 Форма реєстрації для користувача

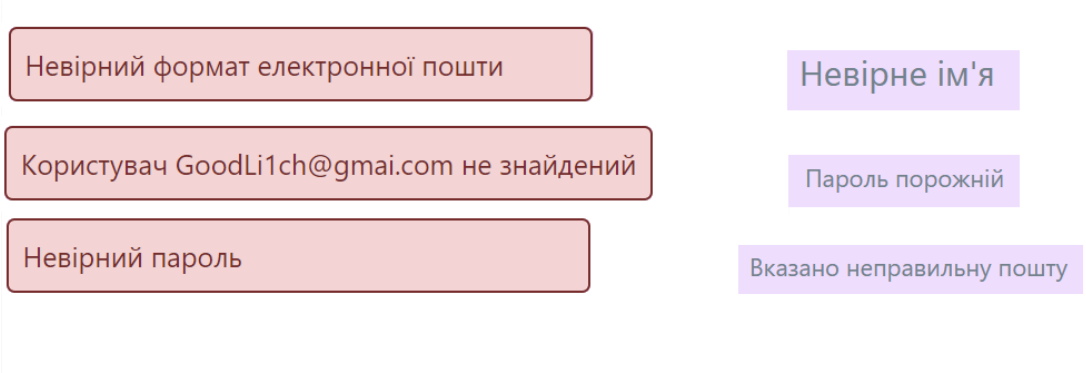


Рис. 4.3 Обробки помилок

Після успішного входу користувач потрапляє на головну сторінку системи. Головна сторінка служить центральним вузлом для навігації по системі. На ній користувач може ознайомитися з основними можливостями системи та інструкціями щодо використання. Головна сторінка містить перелік доступних розділів, таких як "Навчальна робота", "Методична робота", "Організаційна робота", "Виховна робота" та "Звіт". Користувач може перейти до будь-якого з цих розділів, натиснувши відповідне посилання у меню.

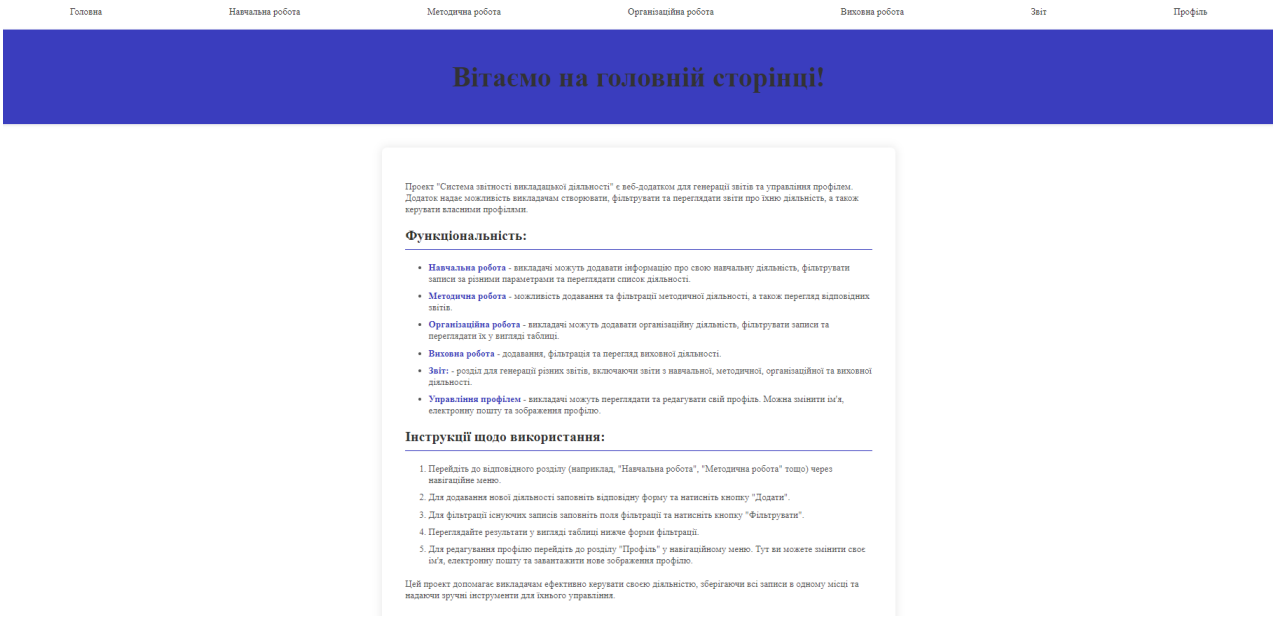


Рис. 4.4. Головна сторінка

Розділ "Навчальна робота" дозволяє викладачам додавати та фільтрувати записи про свою навчальну діяльність. У цьому розділі користувач може вказати предмет, клас, дату, деталі діяльності та іншу інформацію, яка стосується навчального процесу. Після заповнення форми користувач натискає кнопку "Додати навчальну діяльність", щоб зберегти запис у базі даних. Крім того, у цьому розділі можна фільтрувати записи за різними параметрами, такими як предмет, клас, дата та інші. Результати фільтрації відображаються у вигляді таблиці, що забезпечує зручний доступ до необхідної інформації.

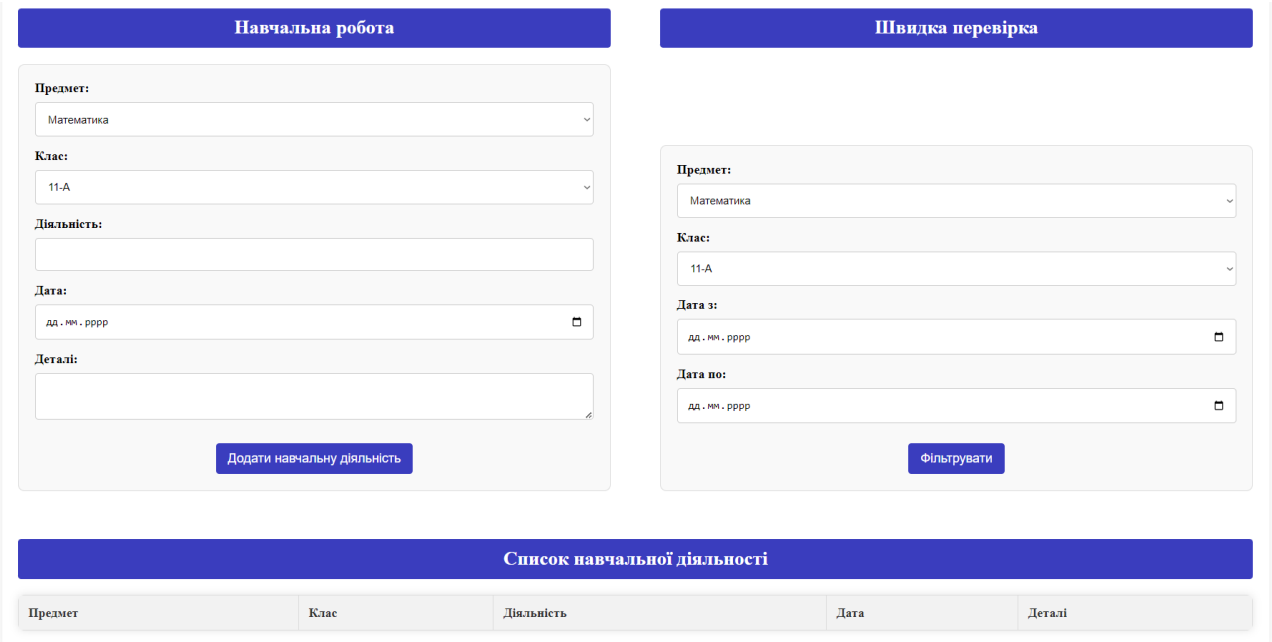


Рис. 4.5. Розділ "Навчальна робота"

Розділ "Методична робота" дозволяє викладачам додавати методичну діяльність, включаючи розробку та проведення методичних заходів, підготовку навчальних матеріалів та інше. У цьому розділі користувач може вказати деталі діяльності, дату та іншу інформацію. Після заповнення форми користувач натискає кнопку "Додати методичну діяльність", щоб зберегти запис у базі даних. Також у цьому розділі доступна функція фільтрації записів, що дозволяє швидко знаходити необхідну інформацію за певними параметрами. Результати фільтрації відображаються у вигляді таблиці.

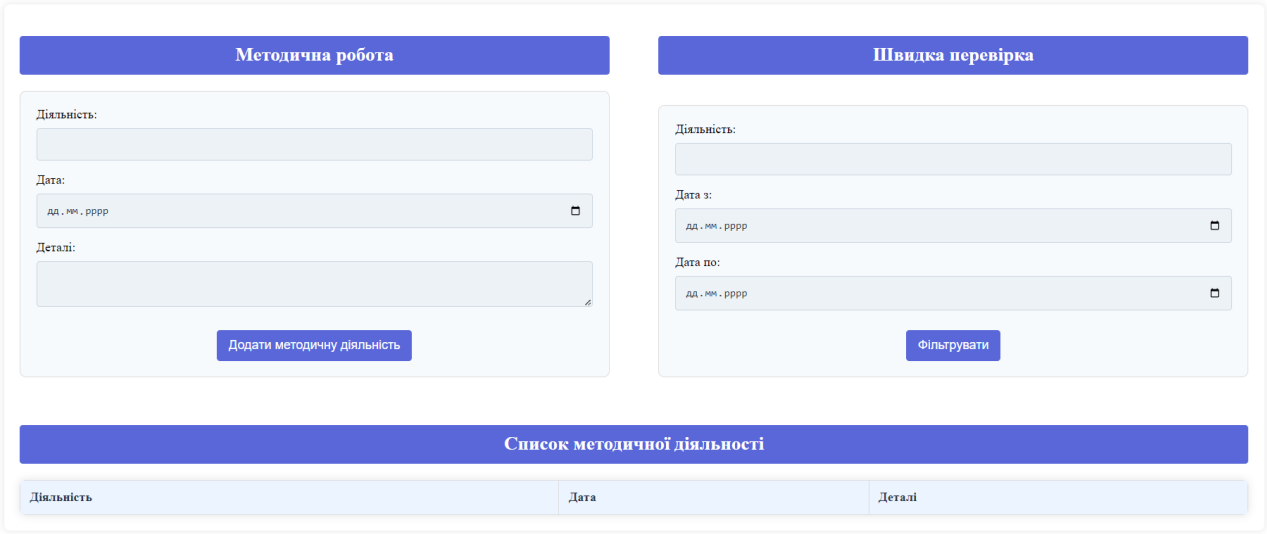


Рис. 4.6 Розділ "Методична робота"

Розділи "Організаційна робота» та "Виховна робота" є аналогічними до сторіки "Методична робота":

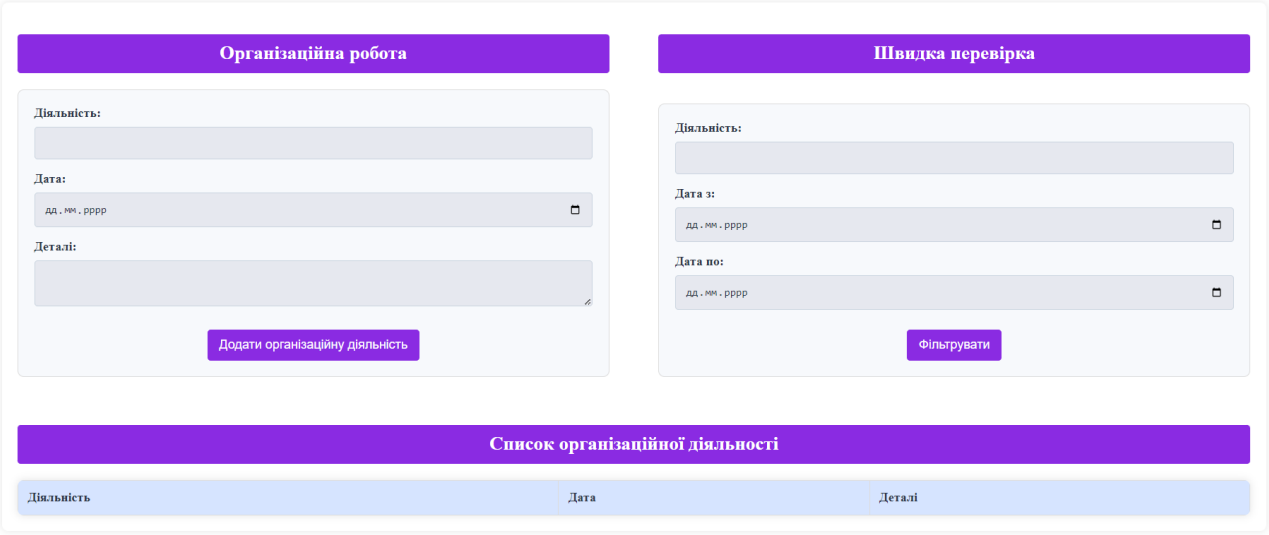


Рис. 4.7 Розділ "Організаційна робота"

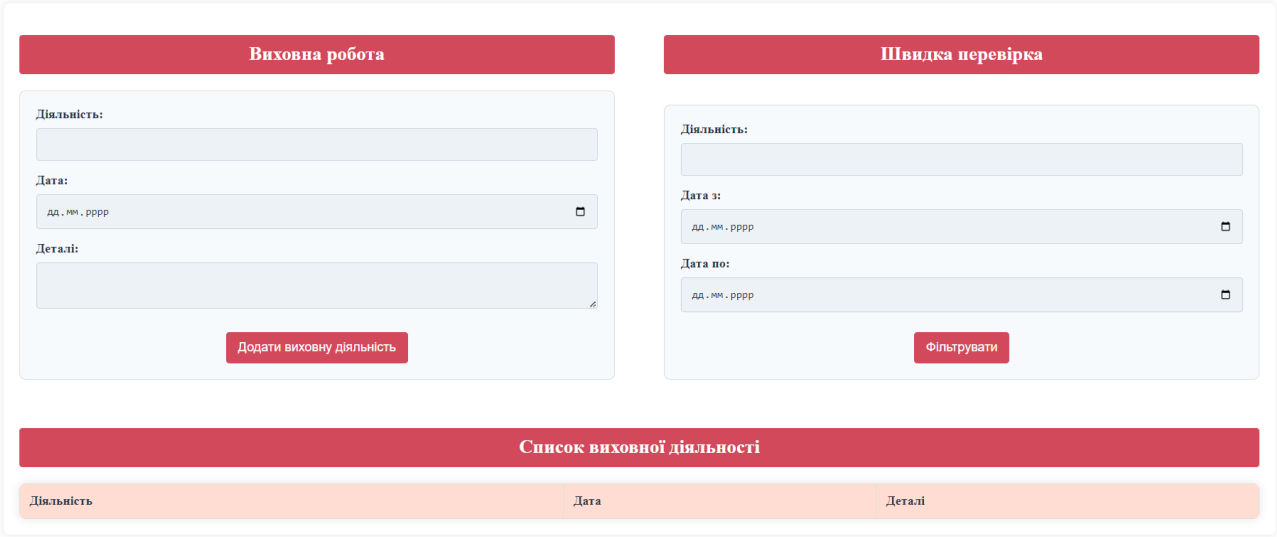


Рис. 4.8 Розділ "Виховна робота"

Кожна успішна дія користувача супроводжується повідомленням про успіх, що підтверджує, що запит був успішно виконаний. Наприклад, після додавання нової діяльності користувач бачить повідомлення про успішне додавання запису. Це забезпечує зворотний зв'язок з користувачем і підтверджує правильність виконання його дій.

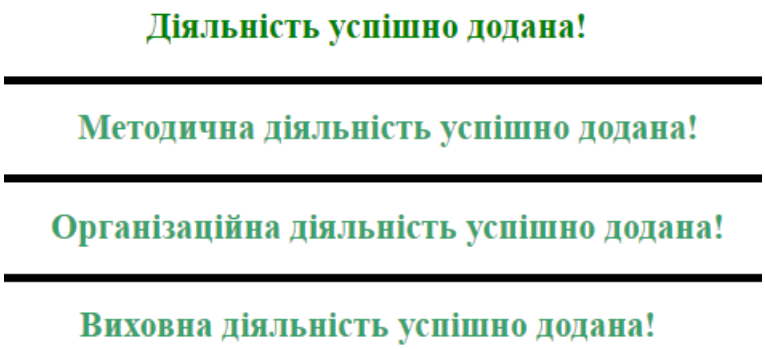


Рис. 4.9 Чотири види повідомлення про успіх

Розділ "Звіт" дозволяє користувачам генерувати звіти за різними категоріями та періодами. У цьому розділі користувач може вибрати тип звіту, період, предмет та клас. Після заповнення форми та натискання кнопки "Генерувати дані" система створює звіт, який відображається у вигляді таблиці. Користувач також може зберегти звіт у форматі Excel для подальшого використання, що забезпечує

зручний спосіб отримання детальної інформації про викладацьку діяльність за вибраними параметрами.

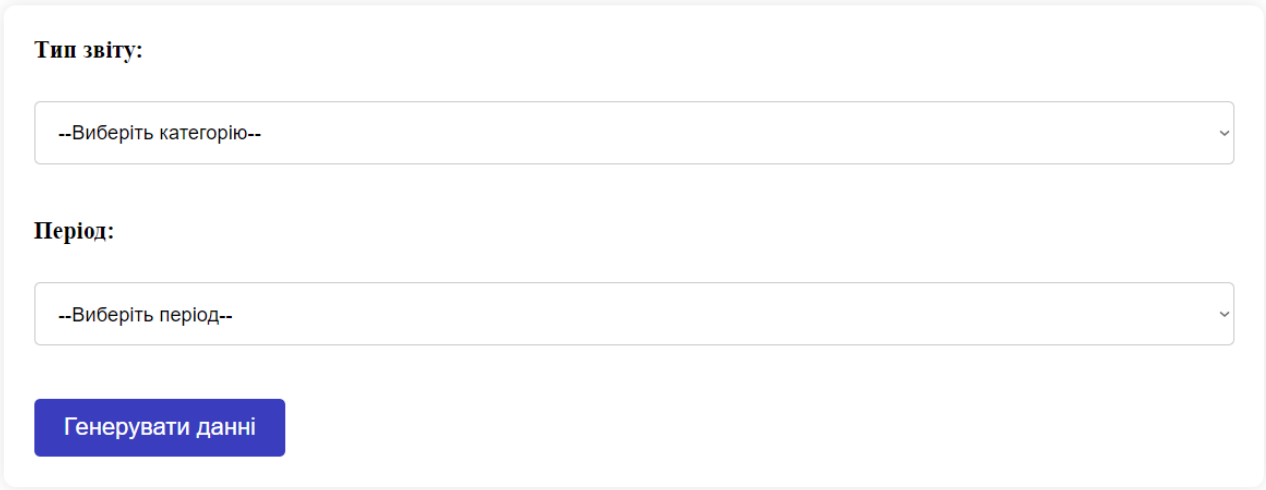


Рис. 4.10. Початкове меню в розділі «Звіт»

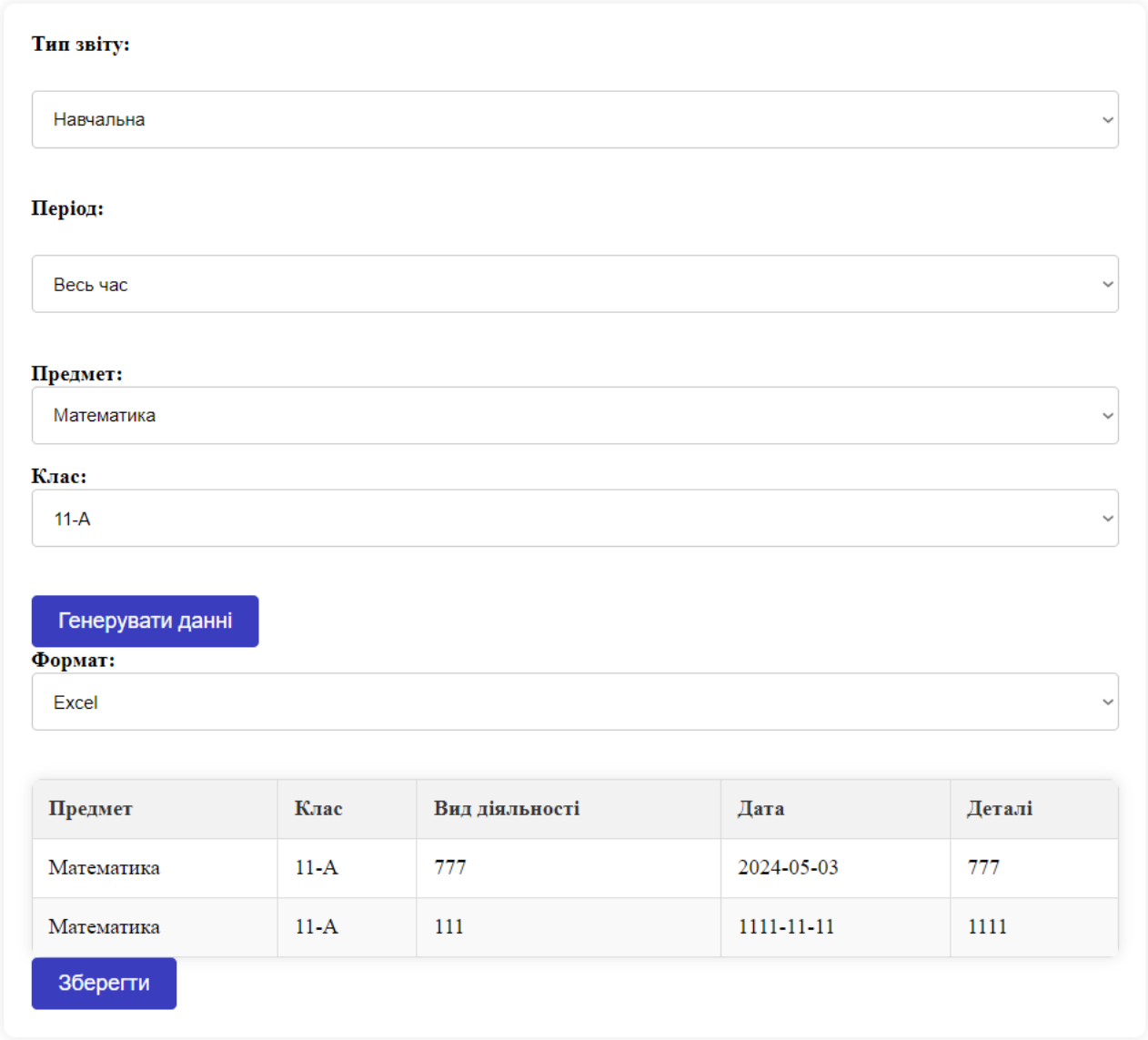


Рис. 4.11. Результат заповнення форми в розділі «Звіт»

Останні скріншоти з розділу «Звіт» демонструють результати завантаження та генерації звіту у вигляді таблиці, де користувач може переглядати всі введені дані та зберігати звіти у форматі Excel. Це дозволяє мати всю необхідну інформацію в зручному форматі для аналізу та звітності.

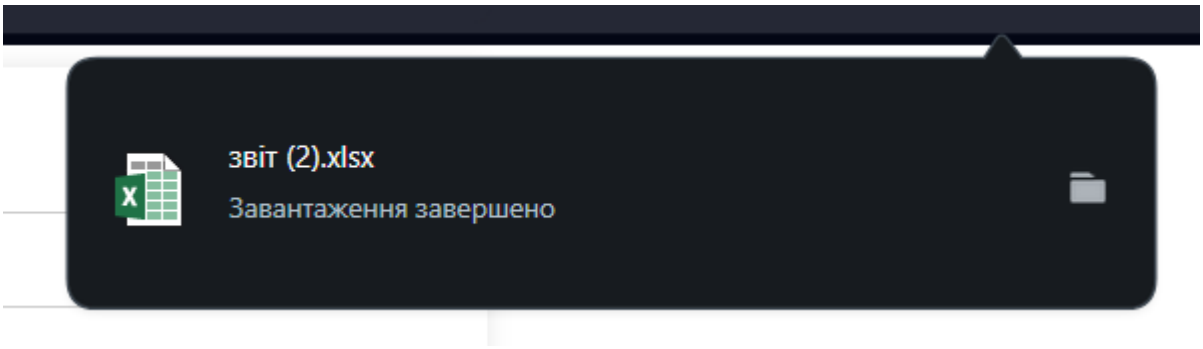


Рис. 4.12 Результат завантаження звіту в exel-формі

Предмет	Клас	Вид діяль	Дата	Деталі
Математи	11-А	777	2024-05-0	777
Математи	11-А	111	1111-11-1	1111

Рис. 4.13 Exel-форма

Також в кінці в нас є розділ управління профілем, яке дозволяє користувачам змінювати свої особисті дані, такі як ім'я, електронна пошта, пароль та зображення профілю.

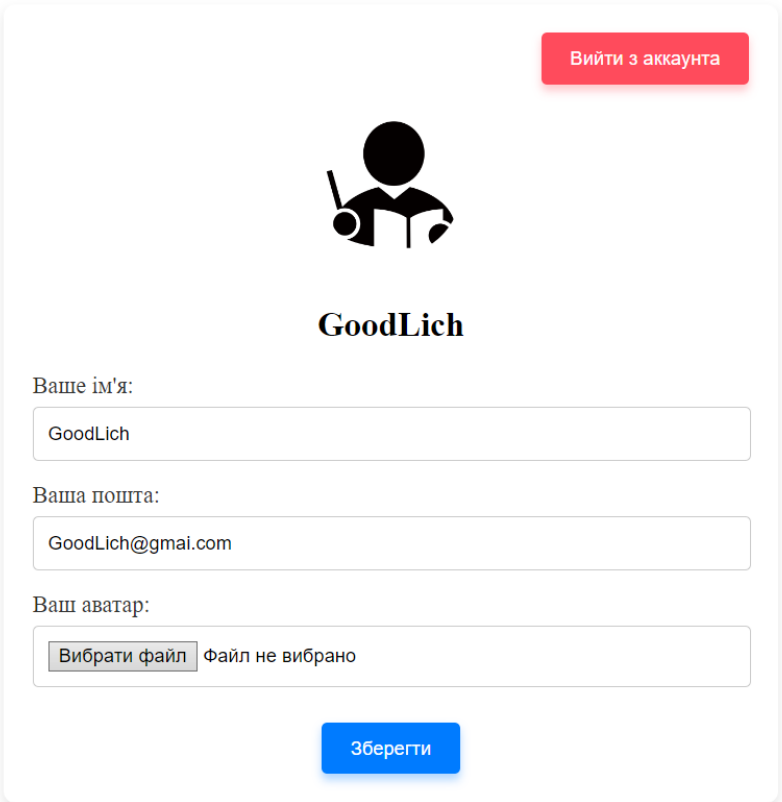


Рис. 4.14 Розділ «Профіль»

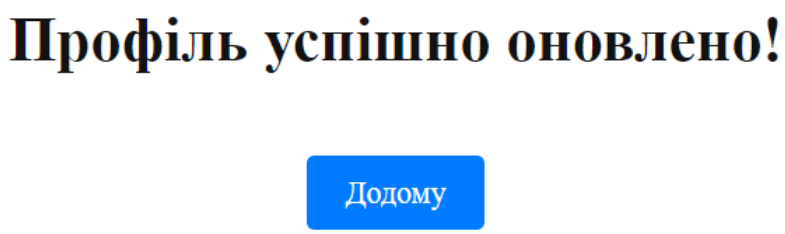


Рис. 4.15 Результат зміни данних

4.2. Впровадження системи в навчальному закладі

Впровадження системи "TeachReport" у навчальному закладі є важливим етапом, що потребує ретельної підготовки та адаптації існуючого проєкту до реальних умов функціонування освітньої установи. Попри те, що початкова розробка системи була виконана в рамках дипломної роботи, і, відповідно, має певні обмеження, потенційні можливості її розвитку та вдосконалення можуть значно підвищити ефективність управління викладацькою діяльністю.

Система "TeachReport" забезпечує базові функції, такі як реєстрація та авторизація користувачів, введення даних про викладацьку діяльність, фільтрація записів за різними параметрами та генерація звітів у форматі Excel. Ці можливості створюють основу для автоматизації процесів збору та обробки інформації про викладацьку діяльність, що може значно полегшити адміністративні задачі та підвищити прозорість і доступність даних.

Однак, для того, щоб система "TeachReport" могла бути успішно впроваджена у реальному навчальному закладі, необхідно врахувати декілька основних факторів:

Треба провести детальний аудит існуючих функцій та виявити їх відповідність реальним потребам закладу. Це включає оцінку зручності інтерфейсу, перевірку надійності та безпеки зберігання даних, а також аналіз можливостей масштабування системи для обробки великого обсягу даних.

Необхідно адаптувати систему до специфічних вимог навчального закладу, що може включати розробку додаткових модулів, таких як модуль для управління розкладом занять, інтеграція з існуючими системами управління навчальним процесом, а також налаштування прав доступу для різних категорій користувачів. Крім того, треба буде забезпечити можливість кастомізації інтерфейсу для відповідності корпоративному стилю закладу.

Необхідно провести всебічне тестування системи в умовах реального навчального процесу, що включатиме проведення пілотних впроваджень у декількох навчальних групах, збір зворотного зв'язку від викладачів та студентів, а також внесення відповідних коректив на основі отриманих даних. Такий підхід дозволить виявити та виправити потенційні проблеми ще до масштабного впровадження системи.

Попри те, що система "TeachReport" була створена в рамках дипломної роботи і має певні обмеження, її подальший розвиток та вдосконалення можуть зробити її дуже актуальним інструментом для управління викладацькою діяльністю у навчальних закладах. Внесення необхідних змін та адаптація системи до реальних умов функціонування дозволять забезпечити високу ефективність, надійність та зручність використання.

ВИСНОВКИ

В розділі 1 було проаналізовано існуючі проблеми в системі звітності та визначено вимоги до нової системи. На нашу думку, аналіз наявних недоліків паперової системи звітності показав, що ця система є неефективною та трудомісткою. Виявлені проблеми включають обмежену доступність інформації, ризики конфіденційності та безпеки, а також екологічні проблеми. Було обґрунтовано необхідність впровадження електронної системи звітності, яка дозволить подолати ці недоліки та підвищити ефективність управління навчальним закладом.

Було визначено, що традиційна паперова система звітності має багато недоліків, серед яких неефективність та трудомісткість процесу, обмежена доступність та прозорість інформації, екологічні проблеми, ризики конфіденційності та безпеки, а також обмежені можливості для співпраці та спільної роботи. Також було зазначено, що паперові документи можуть легко втратити або пошкодити, що може поставити під загрозу конфіденційність даних.

Зроблено та досліджено кращі практики зарубіжного досвіду у сфері електронної звітності. На нашу думку, дослідження таких систем як Blackboard, CELCAT, HIS, Moodle та D2L показало, що впровадження електронних систем звітності значно підвищує ефективність управління навчальними закладами та дозволяє викладачам зосередитися на основних аспектах своєї роботи. Було спроєктовано основні вимоги до електронної системи звітності, яка буде розроблена в рамках даного проєкту.

Розроблено функціональні вимоги до нової системи звітності. Зокрема, було визначено, що система повинна включати модулі реєстрації та входу, особистий кабінет викладача, модуль звітності та інтерфейс користувача. Також було зазначено, що система повинна забезпечувати можливість введення даних, генерування звітів та їх експорт у формат Excel. Було обґрунтовано необхідність використання сучасних технологій веб-розробки для створення зручного та інтуїтивно зрозумілого інтерфейсу користувача.

В розділі 2 було вибрано технології розробки та архітектурне рішення для електронної системи звітності. На нашу думку, обрання мов програмування PHP та JavaScript, а також СКБД MySQL є оптимальним вибором для реалізації даного проєкту. Було обґрунтовано переваги цих технологій, серед яких легкість у використанні, висока швидкість обробки запитів, можливість створення динамічних інтерфейсів та ефективне управління даними.

Далі було проведено аналіз сучасних технологій веб-розробки. Було визначено, що HTML5, PHP та JavaScript є найбільш підходящими для реалізації даного проєкту. Було спроєктовано архітектуру клієнт-серверної системи, яка включає клієнтську частину на основі HTML, CSS та JavaScript, а також серверну частину на основі PHP та MySQL. Було обґрунтовано необхідність використання цих технологій для створення зручного та інтуїтивно зрозумілого інтерфейсу користувача.

Також було обрано мови програмування та СКБД для проєкту. На нашу думку, PHP є оптимальною мовою для серверної частини проєкту завдяки своїй легкості у використанні та відмінній інтеграції з базами даних. JavaScript був обраний для клієнтської частини завдяки своїй високій продуктивності та можливості створення динамічних інтерфейсів. MySQL був обраний як СКБД завдяки своїй високій продуктивності, надійності та можливості інтеграції з різними мовами програмування.

Нами було зроблено архітектуру клієнт-серверної системи. Було спроєктовано структуру бази даних, яка включає таблиці для зберігання інформації про користувачів, звіти та інші необхідні дані. Було визначено, що клієнтська частина системи буде розроблена з використанням HTML, CSS та JavaScript, а серверна частина – з використанням PHP та MySQL. Було обґрунтовано необхідність використання цих технологій для забезпечення високої масштабованості, продуктивності та безпеки системи.

В розділі 3 було розроблено електронну систему звітності. На нашу думку, розробка модуля введення даних є основним етапом для забезпечення зручного та ефективного способу введення та зберігання інформації про звітність викладацької

діяльності. Було спроєктовано форму введення даних, яка включає поля для введення тексту, вибору дат, випадаючі списки та кнопки.

Далі було розроблено модуль введення даних. Було визначено, що форма введення даних буде відправляти дані на сервер за допомогою методу POST. Сервер оброблятиме ці дані за допомогою PHP, що включає отримання даних з форми, валідацію даних та збереження їх у базі даних.

Після чого, було розроблено модуль формування звітів. На нашу думку, модуль генерації звітів є важливим компонентом системи, який дозволяє користувачам вибирати типи звітів, періоди та інші параметри для формування звітів. Було визначено, що звіти будуть формуватися на основі даних, що зберігаються в базі даних, і надаватимуться у зручному форматі для перегляду та експорту.

В розділі 4 було проведено тестування впровадження та супровід системи. Зроблено тестування функціональності та інтерфейсу системи.Була перевірка роботи роботи основних модулів системи, таких як модуль введення даних, модуль генерації звітів та інтерфейс користувача.

Далі було написано теорію щодо впровадження системи в навчальному закладі. На нашу думку, впровадження системи вимагає ретельного планування та налаштування для забезпечення високої доступності та безпеки. Було обґрунтовано необхідність регулярного оновлення програмного забезпечення та адаптації його до мінливих потреб навчального закладу.

Тому:

Впровадження електронної системи звітності викладацької діяльності на основі мов програмування PHP, JavaScript та бази даних MySQL є ефективним рішенням для автоматизації процесів управління навчальним закладом. Така система дозволяє централізовано зберігати та обробляти дані про роботу викладачів, генерувати різноманітні звіти та забезпечувати доступність інформації для всіх зацікавлених сторін. Використання сучасних веб-технологій дозволяє створити зручний та інтуїтивно зрозумілий інтерфейс користувача, що спрощує процес введення та редагування даних.

Розроблена електронна система звітності викладацької діяльності відповідає вимогам безпеки, масштабованості та продуктивності. Вона дозволяє автоматизувати рутинні процеси формування звітності, зменшити ймовірність помилок та підвищити прозорість роботи викладачів. Крім того, система передбачає можливість інтеграції з іншими інформаційними системами навчального закладу, що дозволяє створити єдиний інформаційний простір та оптимізувати управлінські процеси.

Впровадження електронної системи звітності викладацької діяльності вимагає ретельного планування, тестування та відлагодження. Необхідно забезпечити якісну підтримку та супровід системи, регулярно оновлювати програмне забезпечення та адаптувати його до мінливих потреб навчального закладу. Лише за таких умов електронна система звітності зможе ефективно виконувати свої функції та сприяти підвищенню якості освітнього процесу.

Результати досліджень бакалаврської роботи були представлені на Всеукраїнських науково-технічних конференціях:

Ладута К.О., Щербина І.С. Електронна система звітності викладацької діяльності з використанням мов програмування PHP, JavaScript та бази даних MySQL. Всеукраїнська науково-технічна конференція «Застосування програмного забезпечення в ІКТ» 24.квітня.2024 р., ДУІКТ, м. Київ. К.: ДУІКТ – С. 220-222.

ПЕРЕЛІК ПОСИЛАНЬ

1. Іванов В. Г. Інформаційні технології в освіті: навчальний посібник. Київ: Центр учбової літератури, 2020. 400 с.

2. Петров О. М. Веб-технології та їх застосування в освітньому процесі. Науковий вісник Національного університету біоресурсів і природокористування України. Серія: Педагогіка, психологія, філософія. 2019. Вип. 3. С. 10-18.

3. Ковальчук В. В., Бандура В. М. Електронні системи управління навчальним процесом: досвід впровадження та перспективи розвитку. Інформаційні технології і засоби навчання. 2021. Том 81, № 1. С. 40-55.

4. Мельник О. В. Використання мови програмування PHP для розробки веб-додатків. Вісник Національного університету "Львівська політехніка". Серія: Інформаційні системи та мережі. 2020. № 8. С. 75-84.

5. AB M. MySQL Administrator's Guide. Upper Saddle River : Pearson Education, 2005.

6. Benson B. W. JavaScript. *ACM SIGPLAN Notices*. 1999. Vol. 34, no. 4. P. 25–27. URL: https://doi.org/10.1145/312009.312023 (дата звернення: 15.05.2024).

7. Code NodeJS Fishing Log Book. Fishing Log Book Eat Sleep Code NodeJS Repeat Saying Developer Team Saying : Code NodeJS Gifts for Dad: Fishing Log and Trip Record Journal for All Serious Fishermen and Fishing Lovers / ... for Professional Fishermen - Size 6 X9 Inch,Hourly. Independently Published, 2022.

8. Code NodeJS Shooting Log Book. Shooting Log Book Eat Sleep Code NodeJS Repeat Nice Developer Team Nice : Code NodeJS Gifts for Grandpa: Journal to Keep Record Date, Time, Location, Partner, Firearm, Scope Type, Powder, Primer, Brass, ... Diagrams Pages - Gifts for Shooters, Marksman,Wor. Independently Published, 2022.

9. Delisle M. PhpMyAdmin Starter. Packt Publishing, Limited, 2012.

10. JavaScript. *Journal of Information Processing and Management*. 2001. Vol. 44, no. 8. P. 584. URL: https://doi.org/10.1241/johokanri.44.584 (дата звернення: 15.05.2024).

11. Olsson M. Ajax. *JavaScript Quick Syntax Reference*. Berkeley, CA, 2015. P. 59–60. URL: https://doi.org/10.1007/978-1-4302-6494-1_14 (дата звернення: 15.05.2024).

12. PHP. Cooling for Acute Ischemic Brain Damage (COOL AID) - An Open Pilot Study of Induced Hypothermia in Acute Ischemic Stroke. *Journal of Neurosurgical Anesthesiology*. 2002. Vol. 14, no. 1. P. 83–84. URL: https://doi.org/10.1097/00008506-200201000-00020 (дата звернення: 15.05.2024).

13. phpMyAdmin. *The Definitive Guide to MySQL5*. P. 87–116. URL: https://doi.org/10.1007/978-1-4302-0071-0_6 (дата звернення: 15.05.2024).

14. php. PHP Manual, Volume 2. Iuniverse Inc, 2000. 524 p.

15. Queinnec C. Javascript. *Technologies logicielles Architectures des systèmes*. 2017. URL: https://doi.org/10.51257/a-v2-h3120 (дата звернення: 15.05.2024).

16. Rodrigues L., Verissimo P. xAMp: a multi-primitive group communications service. *[1992] 11th Symposium on Reliable Distributed Systems*, Houston, TX, USA. URL: https://doi.org/10.1109/reldis.1992.235136 (дата звернення: 15.05.2024).

17. Russel C. SCO OpenServer: The Windows Network Solution. Prentice Hall PTR, 1996. 400 p.

18. Rutherford-Johnson T. Jonah Haven - Jonah Haven, gasser. Wolftone, Ensemble Proton Bern, Mayrhofer, Ensemble Recherche, Trio Catch, Duo XAMP. Wergo WER64412. *Tempo*. 2024. Vol. 78, no. 308. P. 94–95. URL: https://doi.org/10.1017/s0040298223001067 (дата звернення: 15.05.2024).

19. Wang Y., Solihin Y. XAMP: An eXtensible Analytical Model Platform. *2013 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, Austin, TX, USA, 21–23 April 2013. 2013. URL: https://doi.org/10.1109/ispass.2013.6557142 (дата звернення: 15.05.2024).

20. Winkler J. Ajax = JavaScript + XML. Franzis Verlag GmbH, 2007.

21. Блейклі, Р., Рід, А., Харріс, Л. 2016. Практичний посібник для адміністраторів. Кембридж: Університет Кембриджу.

22. Єременко, О. М. 2018. Електронна система. Наукові праці: Інститут інформаційних технологій. 5(2),

23. Іваненко, В. М. 2019. Використання сучасних технологій веб-розробки. Вісник університету. 12(1), 28-36.

24. Kaplan, R. S., Norton, D. P. 2001. The Balanced Scorecard: Translating Strategy into Action. Boston: Harvard Business School Press.

25. Коваль, М. О. 2020. Порівняльний аналіз систем керування базами даних для навчальних закладів. Вісник Національного університету. 15(4), 87-93.

26. Кравчук, Л. В. 2018. Особливості впровадження інформаційних систем в освітніх установах. Інформаційні технології та суспільство. 3(2), 15-22.

27. Moodle. 2021. About Moodle. Retrieved from https://moodle.org/about/

28. MySQL. 2020. MySQL 8.0 Reference Manual. Oracle Corporation. Retrieved from https://dev.mysql.com/doc/refman/8.0/en/ (дата звернення: 15.05.2024).

29. Петренко, С. О. 2019. Використання мови програмування PHP для створення динамічних веб-додатків. Вісник ІТ. 7(3), 56-63.

30. Smith, M., Brown, J., Johnson, P. 2017. Implementing Electronic Reporting Systems in Higher Education Institutions. London: Oxford University Press.

31. Thomas, K. W., Kilmann, R. H. 1974. Thomas-Kilmann Conflict Mode Instrument. Tuxedo, NY: Xicom Inc.

32. WordPress. 2021. About WordPress. Retrieved from https://wordpress.org/about/

33. Zhang, L., Yu, W., Chen, Y. 2019. Modern Web Technologies for Educational Reporting Systems. Journal of Educational Technology. 22(1), 78-85.

ДОДАТОК А. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО -КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО -НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Розробка електронної системи звітності викладацької діяльності мовами PHP, JavaScript та MySQL

Виконав студент 4 курсу

Групи ПД-43
Ладута Кіріл Олексійович
Керівник роботи

К.т.н., доц. доцент кафедри ІПЗ Щербина Ірина Сергіївна

Київ – 2024

МЕТА, ОБ’ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

Мета роботи: підтримка ведення звітності викладачів початкової школи про результати своєї діяльності за рахунок використання електронної системи.

Об’єкт дослідження: звітність викладачів початкової школи про результати їхньої діяльності.

Предмет дослідження: електронна система для ведення звітності викладачів початкової школи за результатами їхньої діяльності.

ЗАДАЧІ ДИПЛОМНОЇ РОБОТИ

1. Провести аналіз предметної галузі зі звітності викладацької діяльності.
2. Провести аналіз існуючих рішень для звітності викладацької діяльності.
3. Визначити необхідні функціональні та нефункціональні вимоги до електронної системи звітності викладацької діяльності.
4. Провести аналіз інструментів для реалізації електронної системи звітності викладацької діяльності.
5. Розробити прототип системи електронної звітності.
6. Розробити електронну систему звітності викладацької діяльності з використанням PHP, JavaScript та MySQL.
7. Провести тестування електронної системи звітності викладацької діяльності.

3

АНАЛІЗ АНАЛОГІВ

Характеристика	Campus Solutions	Banner by Ellucian	Canvas LMS	EduReport
Користувацький інтерфейс	Складний, потребує навчання	Складний, потребує навчання	Простий у використанні	Інтуїтивно зрозумілий, простий у використанні
Експорт звітів у формати Excel	-	-	-	+
Легкість налаштування	-	-	+	+
Вартість	Висока	Висока	Середня	Безкоштовно
Формування звітів	-	+	-	+
Підтримка української мови	-	-	-	+

4

ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Функціональні вимоги:

1. Модуль реєстрації та входу.
2. Перегляд персональної інформації.
3. Особистий кабінет викладача.
4. Можливість редагування профілю.
5. Доступ до особистих звітів та статистики.
6. Введення даних про проведені заняття (дата, час, предмет, кількість учнів).
7. Введення даних про адміністративну діяльність (наради, консультації).
8. Формування щомісячних, квартальних та річних звітів.
9. Експорт звітів у формати Excel.
10. Фільтрація та сортування даних за різними критеріями (дата, тип діяльності, предмет).

Нефункціональні вимоги:

1. Можливість редагування профілю повинна бути доступною через простий та зрозумілий інтерфейс.
2. Час відгуку на запити у модулі реєстрації та входу не повинен перевищувати 2 секунд.
3. Система повинна забезпечувати можливість фільтрації та сортування даних за різними критеріями з часом відгуку не більше 3 секунд.

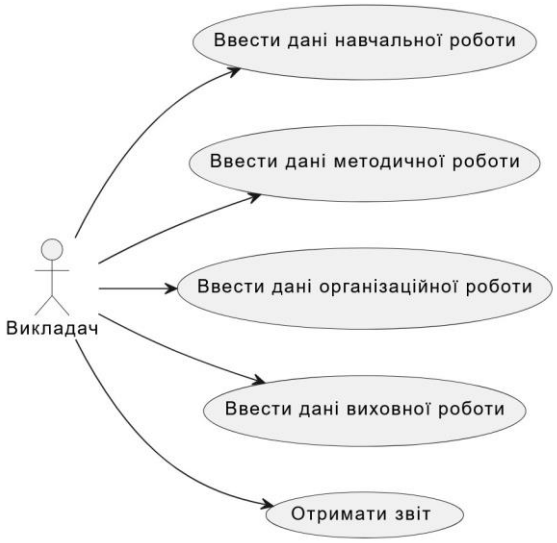
5

ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ

PHPStorm

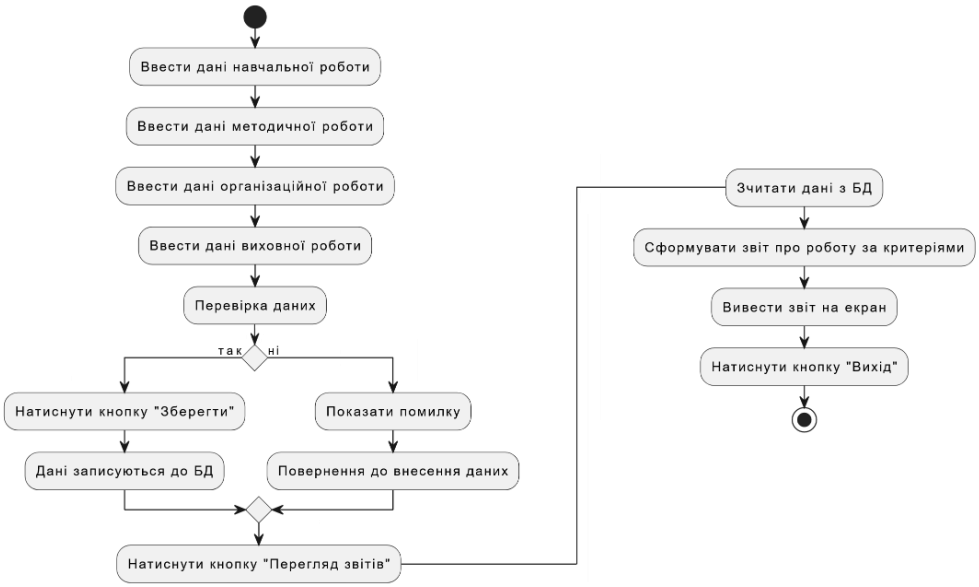
6

USE CASE ДІАГРАМА



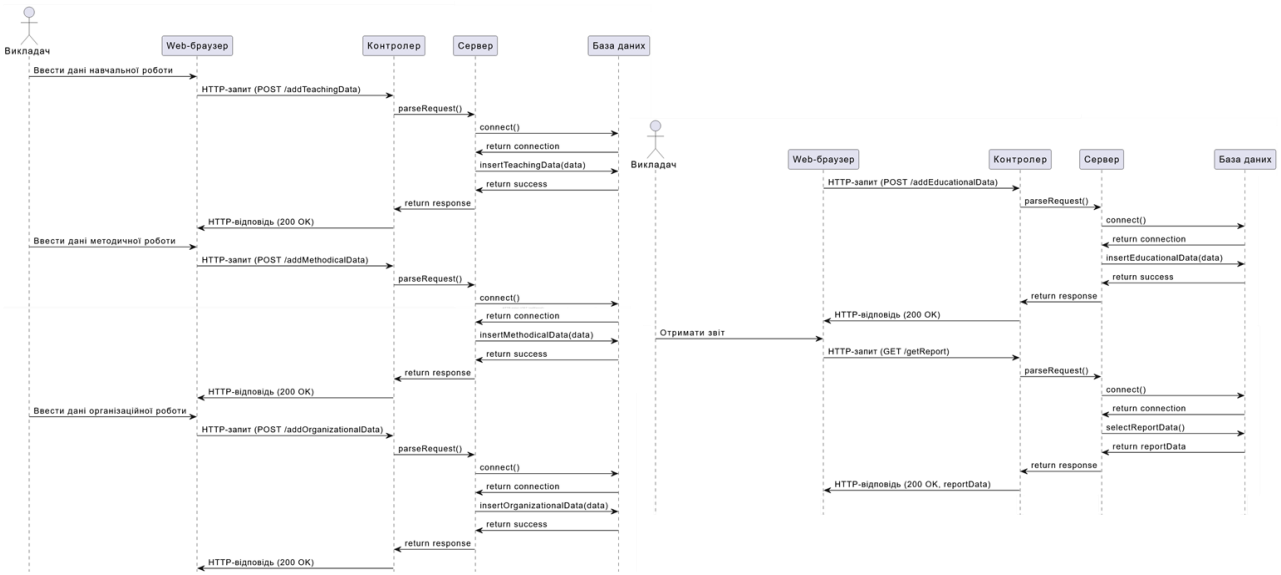
7

ДІАГРАМА ДІЯЛЬНОСТІ БАЗОВОГО СЦЕНАРІЮ РОБОТИ ПРОГРАММИ



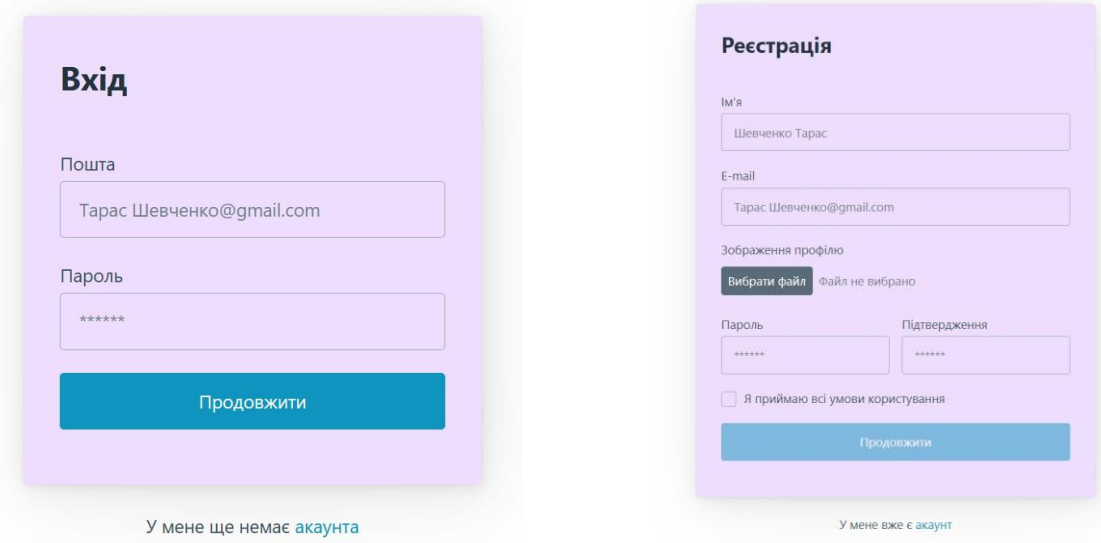
8

ДІАГРАМА ПОСЛІДОВНОСТІ



9

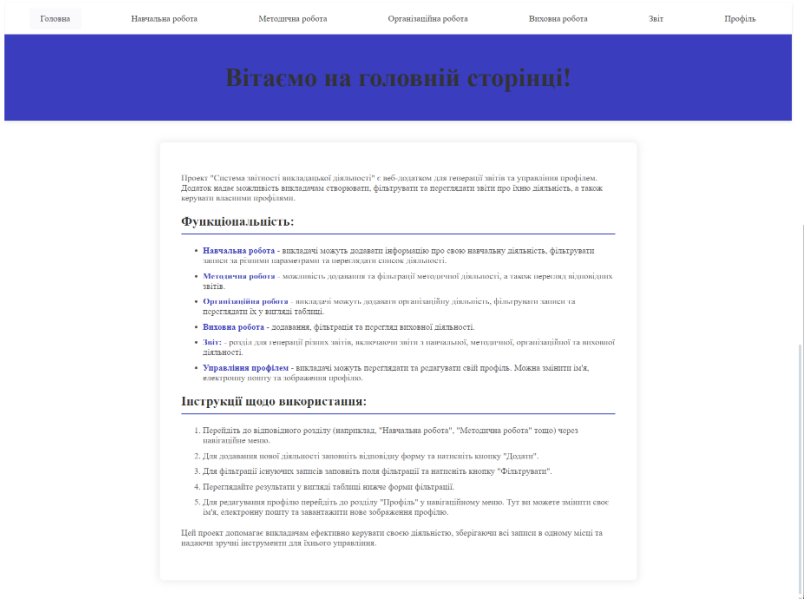
ЕКРАННІ ФОРМИ



Реєстрація та вхід користувача

10

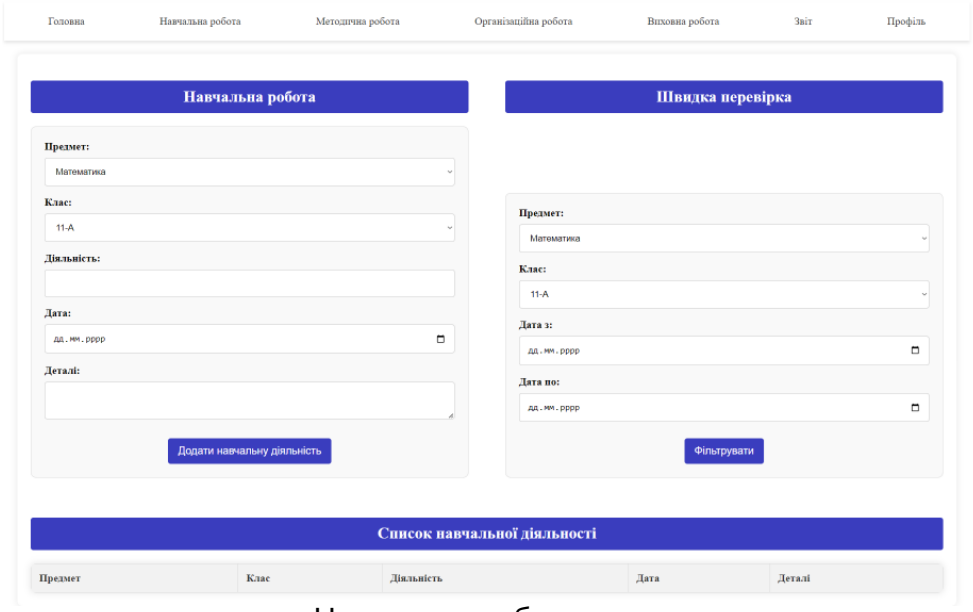
ЕКРАННІ ФОРМИ



Головна сторінка

11

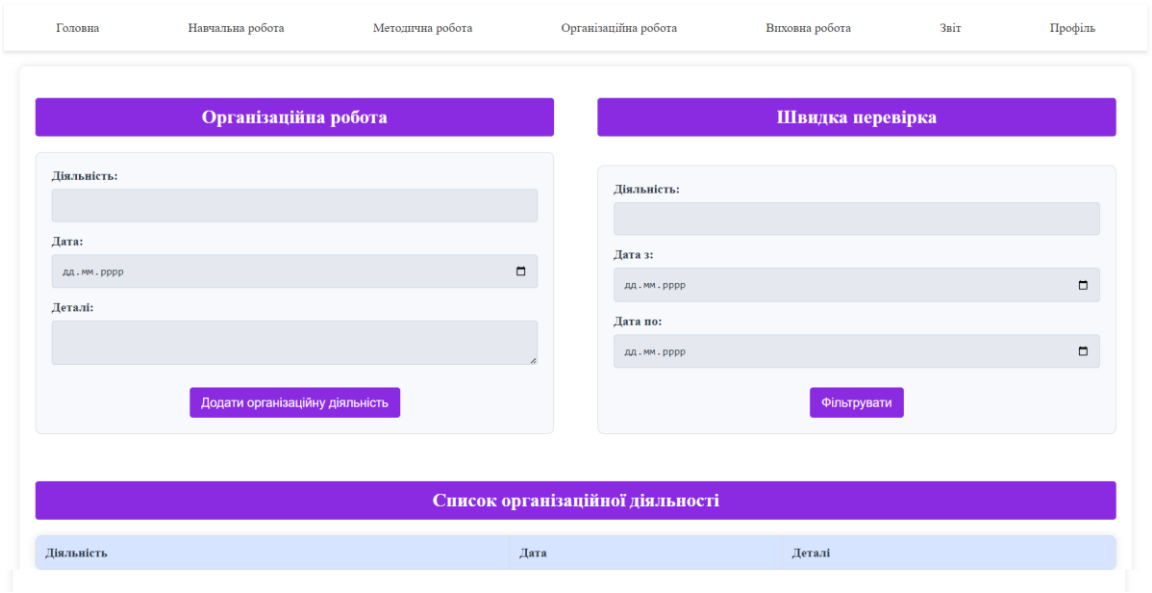
ЕКРАННІ ФОРМИ



Навчальна робота

12

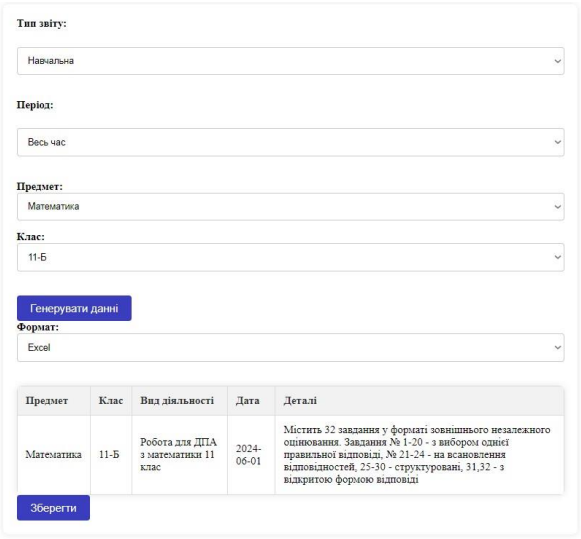
ЕКРАННІ ФОРМИ



Організаційна робота

13

ЕКРАННІ ФОРМИ



Результат заповнення форми звіту

14

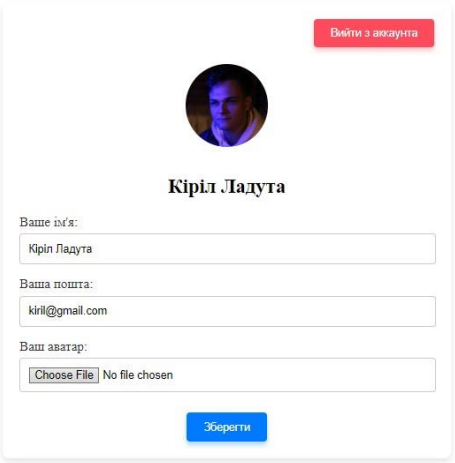
ЕКРАННІ ФОРМИ

	A	B	C	D	E
1	Предмет	Клас	Вид діяльності	Дата	Деталі
2	Математика	11-Б	Робота для ДПА з математики 11 клас	2024-06-01	Містить 32 завдання у форматі зовнішнього незалежного оцінювання. Завдання № 1-20 - з вибором однієї правильної відповіді, № 21-24 - на всановлення відповідностей, 25-30 - структуровані, 31,32 - з відкритою формою відповіді
3					
4					

Exel-форма

15

ЕКРАННІ ФОРМИ



Панель користувача

16

АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

Ладута К.О., Щербина І.С. Електронна система звітності викладацької діяльності з використанням мов програмування PHP, JavaScript та бази даних MySQL є ефективним рішенням для автоматизації процесів управління навчальним закладом. : Всеукраїнська науково-технічна конференція «Застосування програмного забезпечення в ІКТ» 24.04.2024, ДУІКТ, м. Київ. К.: ДУІКТ – ст. 220.

17

ВИСНОВКИ

1. Проаналізовано предметну галузь звітності викладацької діяльності.
2. Проведено аналіз існуючих рішень для звітності викладацької діяльності.
3. Визначено основні функціональні та нефункціональні вимоги до програмного забезпечення.
4. Обрано інструменти розробки для реалізації електронної системи звітності. Код програми писався мовами PHP і JavaScript з використанням системи управління базами даних phpMyAdmin для керування базами даних MySQL.
5. Розроблено електронну систему звітності викладацької діяльності з використанням PHP, JavaScript та MySQL. Система забезпечує зручне введення, зберігання та обробку даних про викладацьку діяльність, автоматичне генерування звітів та їх експорт у форматі Excel.
6. Протестовано електронну систему звітності на відповідність вимогам та правильність роботи програми.

18

ДОДАТОК Б. ЛІСТИНГИ ОСНОВНИХ МОДУЛІВ

Конфіг для підключення к БД:

```
<?php

const DB_HOST = 'localhost';
const DB_PORT = '3306';
const DB_NAME = 'suite';
const DB_USERNAME = 'root';
const DB_PASSWORD = '';
```

Вхід та реєстрація:

```
<?php
require_once __DIR__ . '/src/helpers.php';

checkGuest();
?>

<!DOCTYPE html>
<html lang="uk" data-theme="light">
<?php include_once __DIR__ .
'/components/head.php'?>
<body>

<form class="card" action="src/actions/login.php"
method="post">
    <h2>Вхід</h2>

    <?php if(hasMessage('error')): ?>
        <div class="notice error"><?php echo
getMessage('error') ?></div>
    <?php endif; ?>

    <label for="email">
        Пошта
        <input
            type="text"
            id="email"
            name="email"
            placeholder="Тарас Шевченко@gmail.com"
            value="<?php echo old('email') ?>"
            <?php echo validationErrorAttr('email'); ?>
        >
        <?php if(hasValidationError('email')): ?>
            <small><?php echo
validationErrorMessage('email'); ?></small>
        <?php endif; ?>
    </label>

    <label for="password">
        Пароль
        <input
            type="password"
            id="password"
            name="password"
            placeholder="*******"
        >
    </label>

    <button
        type="submit"
        id="submit"
    >Продовжити</button>
</form>

<p>У мене ще немає <a
href="/register.php">акаунта</a></p>
</body>
</html>
<?php
require_once __DIR__ . '/src/helpers.php';
checkGuest();
?>
<?php

require_once __DIR__ . '/../helpers.php';

$email = $_POST['email'] ?? null;
$password = $_POST['password'] ?? null;

if (empty($email) || !filter_var($email,
FILTER_VALIDATE_EMAIL)) {
    setOldValue('email', $email);
    setMessage('error', 'Невірний формат електронної
пошти');
    redirect('/');
}

$user = findUser($email);

if (!$user) {
    setMessage('error', "Користувач $email не
знайдений");
    redirect('/');
}

if (!password_verify($password, $user['password'])) {
    setMessage('error', 'Невірний пароль');
    redirect('/');
}

$_SESSION['user']['id'] = $user['id'];

redirect('/home.php');

<!DOCTYPE html>
<html lang="uk" data-theme="light">
<?php include_once __DIR__ .
'/components/head.php'?>
<body>

<form class="card" action="src/actions/register.php"
method="post" enctype="multipart/form-data">
    <h2>Реєстрація</h2>

    <label for="name">
```

```
        Ім'я
        <input
            type="text"
            id="name"
            name="name"
            placeholder="Шевченко Тарас"
            value="<?php echo old('name') ?>"
          <?php echo validationErrorAttr('name'); ?>
        >
        <?php if(hasValidationError('name')): ?>
          <small><?php echo
validationErrorMessage('name'); ?></small>
        <?php endif; ?>
      </label>

      <label for="email">
        E-mail
        <input
            type="text"
            id="email"
            name="email"
            placeholder="Тарас Шевченко@gmail.com"
            value="<?php echo old('email') ?>"
          <?php echo validationErrorAttr('email'); ?>
        >
        <?php if(hasValidationError('email')): ?>
          <small><?php echo
validationErrorMessage('email'); ?></small>
        <?php endif; ?>
      </label>

      <label for="avatar">Зображення профілю
        <input
            type="file"
            id="avatar"
            name="avatar"
          <?php echo validationErrorAttr('avatar'); ?>
        >
        <?php if(hasValidationError('avatar')): ?>
          <small><?php echo
validationErrorMessage('avatar'); ?></small>
        <?php endif; ?>
      </label>

      <div class="grid">
        <label for="password">
          Пароль
          <input
              type="password"
              id="password"
              name="password"
              placeholder="*******"
            <?php echo validationErrorAttr('password'); ?>
          >
          <?php if(hasValidationError('password')): ?>
            <small><?php echo
validationErrorMessage('password'); ?></small>
          <?php endif; ?>
        </label>

        <label for="password_confirmation">
          Підтвердження
          <input
              type="password"
              id="password_confirmation"
              name="password_confirmation"
              placeholder="******"
          >
        </label>
      </div>

      <fieldset>
        <label for="terms">
          <input
              type="checkbox"
              id="terms"
              name="terms"
          >
          Я приймаю всі умови користування
        </label>
      </fieldset>

      <button
          type="submit"
          id="submit"
          disabled
      >Продовжити</button>
    </form>

    <p>У мене вже є <a href="/">акаунт</a></p>

    <?php include_once __DIR__ . '/components/scripts.php'
    ?>
    </body>
    </html>

    <?php

    require_once __DIR__ . '/../helpers.php';

    $avatarPath = null;
    $name = $_POST['name'] ?? null;
    $email = $_POST['email'] ?? null;
    $password = $_POST['password'] ?? null;
    $passwordConfirmation =
    $_POST['password_confirmation'] ?? null;
    $avatar = $_FILES['avatar'] ?? null;

    if (empty($name)) {
      setValidationError('name', 'Невірне ім\'я');
    }

    if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
      setValidationError('email', 'Вказано неправильну
    пошту');
    }

    if (empty($password)) {
      setValidationError('password', 'Пароль порожній');
    }

    if ($password !== $passwordConfirmation) {
      setValidationError('password', 'Паролі не
    співпадають');
```

```
}

if (!empty($avatar)) {
    $types = ['image/jpeg', 'image/png'];

    if (!in_array($avatar['type'], $types)) {
        setValidationError('avatar', 'Зображення профілю
має невірний тип');
    }

    if (($avatar['size'] / 1000000) >= 1) {
        setValidationError('avatar', 'Зображення повинно
бути менше 1 Мб');
    }
}

if (!empty($_SESSION['validation'])) {
    setOldValue('name', $name);
    setOldValue('email', $email);
    redirect('/register.php');
}

if (!empty($avatar)) {
    $avatarPath = uploadFile($avatar, 'avatar');
}

$pdo = getPDO();

$query = "INSERT INTO users (name, email, avatar,
password) VALUES (:name, :email, :avatar,
:password)";

$params = [
    'name' => $name,
    'email' => $email,
    'avatar' => $avatarPath,
    'password' => password_hash($password,
PASSWORD_DEFAULT)
];

$stmt = $pdo->prepare($query);

try {
    $stmt->execute($params);
} catch (\Exception $e) {
    die($e->getMessage());
}

redirect('/');
```

Звіти:

```
<?php
ini_set('display_errors', 1);
ini_set('display_startup_errors', 1);
error_reporting(E_ALL);

require_once '../../src/config.php';

header('Content-Type: application/json');

try {
    session_start();
    if (!isset($_SESSION['user_id'])) {
        throw new Exception('User ID not found in
session');
    }

    $user_id = $_SESSION['user_id'];
    $category = $_POST['category'] ?? null;
    $period = $_POST['period'] ?? null;
    $start_date = $_POST['start_date'] ?? null;
    $end_date = $_POST['end_date'] ?? null;
    $selected_activities =
json_decode($_POST['selected_activities'] ?? '[]', true);
    $subject_id = $_POST['subject'] ?? null;
    $class_id = $_POST['class'] ?? null;

    if (!$category || !$period) {
        throw new Exception('Category or period is
missing');
    }

    $dsn = 'mysql:host=' . DB_HOST . ';port=' . DB_PORT
. ';dbname=' . DB_NAME;
    $pdo = new PDO($dsn, DB_USERNAME,
DB_PASSWORD);
    $pdo->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);

    $sql = '';
    $params = [':user_id' => $user_id];

    switch ($category) {
        case 'teaching':
            $sql = 'SELECT s.name AS subject, c.name AS
class, t.activity, t.date, t.details
                FROM TeachingActivities t
                JOIN Subjects s ON t.subject_id = s.id
                JOIN Classes c ON t.class_id = c.id
                WHERE t.user_id = :user_id';
            if (!empty($subject_id)) {
                $sql .= ' AND t.subject_id = :subject_id';
                $params[':subject_id'] = $subject_id;
            }
            if (!empty($class_id)) {
                $sql .= ' AND t.class_id = :class_id';
                $params[':class_id'] = $class_id;
            }
            break;
        case 'methodical':
            $sql = 'SELECT NULL AS subject, NULL AS
class, activity, date, details
                FROM MethodicalActivities
                WHERE user_id = :user_id';
            break;
        case 'organizational':
            $sql = 'SELECT NULL AS subject, NULL AS
class, activity, date, details
                FROM OrganizationalActivities
                WHERE user_id = :user_id';
            break;
```

```
        case 'educational':
            $sql = 'SELECT NULL AS subject, NULL AS
class, activity, date, details
                    FROM EducationalActivities
                    WHERE user_id = :user_id';
            break;
        default:
            throw new Exception('Invalid category');
    }

    if ($period === 'custom' && !empty($start_date) &&
!empty($end_date)) {
        $sql .= ' AND date BETWEEN :start_date AND
:end_date';
        $params[':start_date'] = $start_date;
        $params[':end_date'] = $end_date;
    } elseif ($period === 'week') {
        $sql .= ' AND date >= CURDATE() - INTERVAL
7 DAY';
    } elseif ($period === 'month') {
        $sql .= ' AND date >= CURDATE() - INTERVAL
1 MONTH';
    }

    if (!empty($selected_activities) &&
count($selected_activities) > 0) {
        $activityParams = [];
        foreach ($selected_activities as $index => $activity)
{
            $key = ':activity_' . $index;
            $activityParams[$key] = $activity;
        }
        $placeholders = implode(',',
array_keys($activityParams));
        $sql .= ' AND id IN (' . $placeholders . ')';
        $params = array_merge($params, $activityParams);
    }

    error_log('SQL Query: ' . $sql);
    error_log('Params: ' . json_encode($params));

    $stmt = $pdo->prepare($sql);
    foreach ($params as $key => $value) {
        $stmt->bindValue($key, $value);
    }
    $stmt->execute();
    $reportData = $stmt-
>fetchAll(PDO::FETCH_ASSOC);

    echo json_encode($reportData);
} catch (PDOException $e) {
    error_log('Database error: ' . $e->getMessage());
    echo json_encode(['error' => 'Database error: ' . $e-
>getMessage()]);
} catch (Exception $e) {
    error_log('General error: ' . $e->getMessage());
    echo json_encode(['error' => 'General error: ' . $e-
>getMessage()]);
}
?>
```

```
<!DOCTYPE html>
<html lang="uk" data-theme="light">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-
width, initial-scale=1.0">
    <link rel="stylesheet" href="../css/report.css">
    <script
src="https://cdnjs.cloudflare.com/ajax/libs/xlsx/0.17.0/xl
sx.full.min.js"></script>
    <title>Звіти</title>
</head>
<body>
<div class="container">
    <form id="reportForm" method="POST">
        <label for="category">Тип звіту:</label>
        <select name="category" id="category" required>
            <option value="">--Виберіть категорію--
</option>
            <option value="teaching">Навчальна</option>
            <option
value="methodical">Методична</option>
            <option
value="organizational">Організаційна</option>
            <option value="educational">Виховна</option>
        </select>

        <label for="period" class="report-
period">Період:</label>
        <select name="period" id="period" required>
            <option value="">--Виберіть період--</option>
            <option value="week">Тиждень</option>
            <option value="month">Місяць</option>
            <option value="all_time">Весь час</option>
            <option value="custom">Обрати дату
самостійно</option>
        </select>

        <div id="customDates" style="display:none;">
            <label for="start_date">Початкова дата:</label>
            <input type="date" name="start_date"
id="start_date">
            <label for="end_date">Кінцева дата:</label>
            <input type="date" name="end_date"
id="end_date">
        </div>

        <div id="subjectClassContainer"
style="display:none;">
            <label for="subject">Предмет:</label>
            <select name="subject" id="subject">
                <option value="">--Всі предмети--</option>
            </select>
            <label for="class">Клас:</label>
            <select name="class" id="class"></select>
        </div>

        <div id="activityContainer" style="display:none;">
            <label for="activity">Діяльність:</label>
            <select name="activity" id="activity">
                <option value="all">Всі</option>
                <option value="some">Деякі</option>
```

```
            </select>
        </div>

        <div id="activityList" style="display:none;
overflow-y:auto; max-height:200px;">
        </div>

        <button type="button" id="generateButton"
disabled>Генерувати данні</button>
    </form>

    <div id="formatContainer" style="display:none;">
        <label for="format">Формат:</label>
        <select name="format" id="format" required>
            <option value="excel">Excel</option>
        </select>
    </div>

    <div id="reportResult" style="display:none;">
        <table id="reportTable">
            <thead id="reportTableHeader"></thead>
            <tbody id="reportTableBody"></tbody>
        </table>
    </div>
        <input type="hidden" name="category"
id="save_category">
        <input type="hidden" name="period"
id="save_period">
        <input type="hidden" name="format"
id="save_format">
        <input type="hidden" name="start_date"
id="save_start_date">
        <input type="hidden" name="end_date"
id="save_end_date">
        <input type="hidden" name="selected_activities"
id="save_selected_activities">
        <button type="submit" id="saveButton"
style="display:none;">Зберегти</button>
</div>

<script>
    document.addEventListener("DOMContentLoaded",
function () {
        const categorySelect =
document.getElementById("category");
        const periodSelect =
document.getElementById("period");
        const subjectSelect =
document.getElementById("subject");
        const classSelect =
document.getElementById("class");
        const activitySelect =
document.getElementById("activity");
        const customDates =
document.getElementById("customDates");
        const subjectClassContainer =
document.getElementById("subjectClassContainer");
        const activityContainer =
document.getElementById("activityContainer");
        const activityList =
document.getElementById("activityList");
        const generateButton =
document.getElementById("generateButton");
        const reportTableHeader =
document.getElementById("reportTableHeader");
        const reportTableBody =
document.getElementById("reportTableBody");
        const saveButton =
document.getElementById("saveButton");

        function log(message) {
            console.log(`[LOG]: ${message}`);
        }

        function resetForm() {
            log("Очищення полів форми");
            subjectSelect.value = '';
            classSelect.value = '';
            activitySelect.value = 'all';
            activityList.innerHTML = '';
            customDates.style.display = 'none';
            document.getElementById("start_date").value =
"";
            document.getElementById("end_date").value =
"";

document.getElementById('reportResult').style.display =
'none';

document.getElementById('formatContainer').style.displ
ay = 'none';
            saveButton.style.display = 'none';
        }

        function loadActivities(category) {
            log(`Завантаження діяльностей для категорії:
${category}`);

fetch(`../assets/report/get_activities.php?category=${cate
gory}`)
                .then(response => {
                    if (!response.ok) {
                        throw new Error('Мережева відповідь не
є успішною');
                    }
                    return response.json();
                })
                .then(data => {
                    log(`Діяльності отримано:
${JSON.stringify(data)}`);
                    if (data.error) {
                        console.error("Помилка при отриманні
діяльностей:", data.error);
                        alert("Помилка при отриманні
діяльностей: " + data.error);
                        return;
                    }
                    activityList.innerHTML = '';
                    data.forEach(activity => {
                        const checkbox =
document.createElement('input');
                        checkbox.type = 'checkbox';
                        checkbox.name = 'activities[]';
                        checkbox.value = activity.id;
```

```
                checkbox.id = 'activity_' + activity.id;

                const label =
document.createElement('label');
                label.htmlFor = 'activity_' + activity.id;
                label.textContent = activity.name;

                activityList.appendChild(checkbox);
                activityList.appendChild(label);

activityList.appendChild(document.createElement('br'));
            });
            validateForm();
        })
        .catch(error => {
            console.error("Помилка при отриманні
діяльностей:", error);
            alert("Помилка при отриманні
діяльностей: " + error.message);
        });
    }

    function loadSubjectsAndClasses() {
        log("Завантаження предметів та класів");
        fetch(`../assets/report/get_subjects_classes.php`)
            .then(response => {
                if (!response.ok) {
                    throw new Error('Мережева відповідь не
є успішною');
                }
                return response.json();
            })
            .then(data => {
                log(`Предмети та класи отримано:
${JSON.stringify(data)}`);
                if (data.error) {
                    console.error("Помилка при отриманні
предметів та класів:", data.error);
                    alert("Помилка при отриманні
предметів та класів: " + data.error);
                    return;
                }
                subjectSelect.innerHTML = '<option
value="">--Всі предмети--</option>';
                classSelect.innerHTML = '';
                data.subjects.forEach(subject => {
                    const option =
document.createElement('option');
                    option.value = subject.id;
                    option.textContent = subject.name;
                    subjectSelect.appendChild(option);
                });
                data.classes.forEach(cls => {
                    const option =
document.createElement('option');
                    option.value = cls.id;
                    option.textContent = cls.name;
                    classSelect.appendChild(option);
                });
                validateForm();
            })
            .catch(error => {
                console.error("Помилка при отриманні
предметів та класів:", error);
                alert("Помилка при отриманні предметів
та класів: " + error.message);
            });
    }

    categorySelect.addEventListener("change", function
() {
        resetForm();
        const category = categorySelect.value;
        log(`Зміна категорії: ${category}`);
        if (category === 'teaching') {
            subjectClassContainer.style.display = 'block';
            activityContainer.style.display = 'none';
            loadSubjectsAndClasses();
        } else if (category) {
            subjectClassContainer.style.display = 'none';
            activityContainer.style.display = 'block';
            loadActivities(category);
        } else {
            subjectClassContainer.style.display = 'none';
            activityContainer.style.display = 'none';
        }
        validateForm();
    });

    periodSelect.addEventListener("change", function
() {
        const period = periodSelect.value;
        log(`Зміна періоду: ${period}`);
        if (period === "custom") {
            customDates.style.display = "block";
        } else {
            customDates.style.display = "none";
            document.getElementById("start_date").value
= "";
            document.getElementById("end_date").value
= "";
        }
        validateForm();
    });

    activitySelect.addEventListener("change", function
() {
        const activity = activitySelect.value;
        log(`Зміна діяльності: ${activity}`);
        if (activity === 'some') {
            activityList.style.display = 'block';
        } else {
            activityList.style.display = 'none';
            activityList.innerHTML = '';
        }
        validateForm();
    });

    subjectSelect.addEventListener("change",
validateForm);
    classSelect.addEventListener("change",
validateForm);
```

```
document.getElementById('generateButton').addEventLi
stener('click', function (event) {
        event.preventDefault();
        generateReport();
    });


document.getElementById('saveButton').addEventListen
er('click', function (event) {
        event.preventDefault();
        saveReportAsExcel();
    });

    function generateReport() {
        log("Генерація звіту");
        const formData = new
FormData(document.getElementById('reportForm'));
        formData.append('user_id', 'user_id');

        const selectedActivities =
Array.from(document.querySelectorAll('#activityList
input[type="checkbox"]:checked'))
            .map(cb => cb.value);
        formData.append('selected_activities',
JSON.stringify(selectedActivities));

        fetch('../assets/report/generate_report.php', {
            method: 'POST',
            body: formData
        })
            .then(response => {
                if (!response.ok) {
                    throw new Error('Мережева відповідь не
є успішною');
                }
                return response.json();
            })
            .then(data => {
                if (data.error) {
                    console.error("Помилка від сервера: ",
data.error);
                    alert("Помилка при генерації звіту: " +
data.error);
                    return;
                }

                log(`Дані звіту отримано:
${JSON.stringify(data)}`);
                reportTableBody.innerHTML = '';

                if (data.length === 0) {

reportTableBody.insertRow().insertCell(0).innerText =
"Дані для обраних параметрів відсутні";
                    return;
                }

                if (categorySelect.value === 'teaching') {
                    reportTableHeader.innerHTML = `
                    <tr>
                        <th>Предмет</th>
                        <th>Клас</th>
                        <th>Вид діяльності</th>
                        <th>Дата</th>
                        <th>Деталі</th>
                    </tr>
                `;
                    data.forEach(function (activity) {
                        const row =
reportTableBody.insertRow();
                        row.insertCell(0).innerText =
activity.subject || '';
                        row.insertCell(1).innerText =
activity.class || '';
                        row.insertCell(2).innerText =
activity.activity;
                        row.insertCell(3).innerText =
activity.date;
                        row.insertCell(4).innerText =
activity.details;
                    });
                } else {
                    reportTableHeader.innerHTML = `
                    <tr>
                        <th>Вид діяльності</th>
                        <th>Дата</th>
                        <th>Деталі</th>
                    </tr>
                `;
                    data.forEach(function (activity) {
                        const row =
reportTableBody.insertRow();
                        row.insertCell(0).innerText =
activity.activity;
                        row.insertCell(1).innerText =
activity.date;
                        row.insertCell(2).innerText =
activity.details;
                    });
                }


document.getElementById('reportResult').style.display =
'block';

document.getElementById('formatContainer').style.displ
ay = 'block';
                saveButton.style.display = 'block';


document.getElementById('save_category').value =
formData.get('category');

document.getElementById('save_period').value =
formData.get('period');

document.getElementById('save_format').value =
formData.get('format');

document.getElementById('save_start_date').value =
formData.get('start_date');

document.getElementById('save_end_date').value =
formData.get('end_date');
```

```
document.getElementById('save_selected_activities').value = JSON.stringify(selectedActivities);
            })
            .catch(error => {
                console.error("Помилка при генерації звіту:", error);
                alert("Помилка при генерації звіту: " + error.message);
            });
        }

        function saveReportAsExcel() {
            log("Збереження звіту у форматі Excel");

            const wb = XLSX.utils.book_new();
            const ws_data = [];
            const category = document.getElementById('category').value;

            let headers;
            if (category === 'teaching') {
                headers = ['Предмет', 'Клас', 'Вид діяльності', 'Дата', 'Деталі'];
            } else {
                headers = ['Вид діяльності', 'Дата', 'Деталі'];
            }

            ws_data.push(headers);

            const rows = Array.from(reportTableBody.rows);
            rows.forEach(row => {
                let rowData;
                if (category === 'teaching') {
                    rowData = Array.from(row.cells).map(cell => cell.innerText);
                } else {
                    rowData = [
                        row.cells[0].innerText,
                        row.cells[1].innerText,
                        row.cells[2].innerText
                    ];
                }
                ws_data.push(rowData);
            });

            const ws = XLSX.utils.aoa_to_sheet(ws_data);
            XLSX.utils.book_append_sheet(wb, ws, 'Звіт');

            const wbout = XLSX.write(wb, { bookType: 'xlsx', type: 'binary' });

            function s2ab(s) {
                const buf = new ArrayBuffer(s.length);
                const view = new Uint8Array(buf);
                for (let i = 0; i < s.length; i++) {
                    view[i] = s.charCodeAt(i) & 0xFF;
                }
                return buf;
            }

            const blob = new Blob([s2ab(wbout)], { type: "application/octet-stream" });
            const url = window.URL.createObjectURL(blob);
            const a = document.createElement('a');
            a.style.display = 'none';
            a.href = url;
            a.download = 'звіт.xlsx';
            document.body.appendChild(a);
            a.click();
            window.URL.revokeObjectURL(url);
        }


        function validateForm() {
            const category = categorySelect.value;
            const period = periodSelect.value;
            const activity = activitySelect.value;

            log(`Перевірка форми: категорія = ${category}, період = ${period}, діяльність = ${activity}`);
            let valid = category && period;

            if (category === 'teaching') {
                valid = valid && classSelect.value;
            } else if (activity === 'some') {
                valid = valid && activityList.querySelectorAll('input[type="checkbox"]:checked').length > 0;
            }

            generateButton.disabled = !valid;
        }

        categorySelect.addEventListener('change', validateForm);
        periodSelect.addEventListener('change', validateForm);
        activitySelect.addEventListener('change', validateForm);
        activityList.addEventListener('change', validateForm);
        subjectSelect.addEventListener('change', validateForm);
        classSelect.addEventListener('change', validateForm);
        validateForm();
    });

</script>
</body>
</html>
```