

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук та технологій
(повне найменування факультету)

Кафедра комп'ютерних систем та мереж
(повне найменування кафедри)

Пояснювальна записка
до дипломного проєкту (роботи)
бакалавра
(ступінь вищої освіти (освітній ступінь))

на тему **РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ ОБЛІКУ**
УСПІШНОСТІ УЧНІВ МУЗИЧНИХ ШКІЛ

Виконав: студент 4 курсу, групи **КНТз-511сп**
спеціальності _____

123 «Комп'ютерна інженерія»
(код і найменування спеціальності)

Освітня програма (спеціалізація) _____
«Комп'ютерна інженерія»

КОВАЛЕВСЬКА С. Г.
(ПРИЗВИЩЕ та ініціали)

Керівник **ГОЛУБ Т.В.**
(ПРИЗВИЩЕ та ініціали)

Рецензент **СТЕПАНЕНКО О. О.**
(ПРИЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
 (повне найменування вищого навчального закладу)

Факультет Комп'ютерних наук і технологій
 Кафедра «Комп'ютерні системи та мережі»
 Ступінь вищої освіти бакалаврський
 Спеціальність 123 Комп'ютерна інженерія
 (код і найменування)
 Освітня програма (спеціалізація) Комп'ютерна інженерія
 (назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ
 Завідувач кафедри КУДЕРМЕТОВ Р.К.
 “12” листопада 2024 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА

КОВАЛЕВСЬКОЇ Світлани Геннадіївни
 (ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Розробка інформаційної системи обліку успішності учнів музичних шкіл

керівник проєкту (роботи) к. т. н., ГОЛУБ Тетяна Василівна
 (науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом вищого навчального закладу від “01” квітня 2024 року № 105

2. Строк подання студентом проєкту (роботи) 10 червня 2024 року

3. Вихідні дані до проєкту (роботи) система для автоматичного ведення обліку успішності учнів музичних шкіл, технології проєктування, структура програмного застосування для обліку успішності учнів музичних шкіл

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1) Огляд існуючих систем для обліку успішності учнів та аналіз засобів для їх реалізації, розробка вимог до системи, обґрунтування доцільності створення інформаційної системи обліку успішності учнів музичних шкіл;

2) Розробка структури та алгоритму роботи системи, розробка структури бази даних;

3) Реалізація та дослідження системи.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

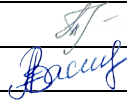
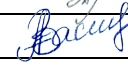
1) Структурна схема реалізації системи;

2) Алгоритм роботи системи;

3) Функціональні можливості програми (Use case diagram);

4) ER-діаграма бази даних.

6. Консультанти розділів проєкту (роботи)

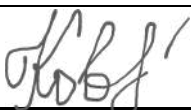
Розділ	ПРИЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4	ГОЛУБ Т.В., доцент		
нормоконтроль	ПОЛЬСЬКА О.В., ст. викл.		

7. Дата видачі завдання 05.03.2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Аналіз літератури за темою дипломного проєкту і технічної документації	15.03.2024 р.	
2	Затвердження плану дипломного проєкту	30.03.2024 р.	
3	Опис загальної предметної області	05.04.2024 р.	
4	Дослідження існуючих систем обліку успішності учнів та доступних засобів їх реалізації	16.04.2024 р.	
5	Розробка вимог до інформаційної системи обліку успішності учнів музичних шкіл	23.04.2024 р.	
6	Аналіз та вибір засобів реалізації системи	02.05.2024 р.	
7	Розробка загальної структури вебзастосунку	06.05.2024 р.	
8	Реалізація інформаційної системи обліку успішності учнів музичних шкіл	20.05.2024 р.	
9	Дослідження інформаційної системи обліку успішності учнів музичних шкіл	25.05.2024 р.	
10	Оформлення отриманих результатів у ПЗ	03.06.2024 р.	
11	Оформлення графічного матеріалу	06.06.2024 р.	
12	Оформлення допоміжного матеріалу	08.06.2024 р.	

Студентка


 (підпис)

Світлана КОВАЛЕВСЬКА
 (Ім'я ПРИЗВИЩЕ)

Керівник проєкту (роботи)


 (підпис)

Тетяна ГОЛУБ
 (Ім'я ПРИЗВИЩЕ)

РЕФЕРАТ

ПЗ: 90 с., 28 рис., 50 табл., 5 додатків, 19 джерел.

ІНФОРМАЦІЙНА СИСТЕМА, КЛІЄНТ, СЕРВЕР, ВЕБЗАСТОСУНОК,
БАЗА ДАНИХ, JAVA SCRIPT, REACT, EXPRESSJS, NODEJS, MYSQL

Об'єкт розробки – інформаційна система обліку успішності учнів музичних шкіл.

Метою роботи є розробка інформаційної системи обліку успішності учнів музичних шкіл.

Проект складається з чотирьох розділів.

Перший розділ присвячено аналізу предметної області, огляду існуючих систем обліку успішності учнів, визначенню загальних вимог до системи та обґрунтуванню доцільності розробки інформаційної системи.

Другий розділ присвячено аналізу та вибору засобів реалізації системи.

Третій розділ присвячено розробці програмного застосування інформаційної системи, де визначено загальну структуру вебзастосунку, розглянуто архітектуру програмного застосування, розроблено структуру бази даних та розроблено інтерфейс взаємодії користувача із системою.

Четвертий розділ присвячено аналізу розробленого програмного застосування, де розглянуто особливості реалізації інформаційної системи та протестовано вебзастосунок.

В результаті виконання дипломного проєкту проведено аналіз, розробку та тестування інформаційної системи обліку успішності учнів музичних шкіл.

ЗМІСТ

Скорочення та умовні позначки	7
Вступ.....	8
1 Огляд існуючих систем для обліку успішності учнів та обґрунтування теми дипломного проєкту	10
1.1 Загальний опис предметної області	10
1.2 Огляд існуючих програмних рішень.....	12
1.3 Визначення загальних вимог до системи.....	18
1.4 Обґрунтування доцільності створення інформаційної системи обліку успішності учнів музичних шкіл	20
1.5 Висновки до розділу	20
2 Аналіз та вибір засобів реалізації системи	21
2.1 Обґрунтування вибору мови програмування	21
2.2 Обґрунтування вибору фреймворку	25
2.3 Обґрунтування вибору бази даних	27
2.4 Висновки до розділу	28
3 Розробка програмного застосунку інформаційної системи.....	29
3.1 Загальна структура програми.....	29
3.2 Архітектура програмного застосування	30
3.3 Структура бази даних	36
3.4 Розробка інтерфейсу взаємодії користувача з системою.....	49
3.5 Висновки до розділу	54
4 Аналіз розробленого програмного застосунку	55
4.1 Особливості реалізації програмного застосунку	55
4.2 Тестування програмного застосунку.....	56
4.3 Висновки до розділу	58
Висновки	59
Перелік джерел посилання	60

Додаток А Специфікація вимог SRS	62
Додаток Б Структура сторінок особистого кабінету адміністратора	67
Додаток В ER-діаграма бази даних	68
Додаток Г Макети сторінок вебзастосунку	69
Додаток Д Фрагменти програмного коду	84

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

ДМШ	— дитяча музична школа
ЗВО	— заклад вищої освіти
ЗЗСО	— заклад загально-середньої освіти
СКБД	— система керування базою даних
API	— Прикладний програмний інтерфейс, Application Programming Interface
CSS	— каскадні таблиці стилів, Cascading Style Sheets
DOM	— об'єктна модель документа, Document Object Model
HTML	— мова розмітки гіпертексту, HyperText Markup Language
SASS	— скриптова метамова, Syntactically Awesome Stylesheets
SQL	— мова структурованих запитів, Structured Query Language
SRS	— специфікація вимог до програмного забезпечення, Software Requirement Specification

ВСТУП

В сучасному світі велику роль відіграє мережа Інтернет, яка дає можливість миттєвого доступу до потрібної інформації із зручного місця розташування, а також взаємодії з іншими користувачами мережі.

З плином часу багато процесів та документів оцифровуються, в тому числі і процес обліку успішності учнів. Усі навчальні заклади вели паперовий документообіг, і тільки з популяризацією комп'ютерних технологій та навичок користування ПК він став переходити в електронний формат. Паперові документи мають свої недоліки: з часом їх стає велика кількість, займають багато місця, потребують архіви для зберігання, з часом можуть псуватись, пошуки потрібної інформації виконуються в ручному режимі та витрачають велику кількість часу.

Для усунення недоліків роботи з паперовими носіями інформації є рішення переходу на електронні. Електронні носії інформації:

- дозволяють зберігати великі об'єми інформації без потреби додаткового простору приміщень;
- дозволяють створювати резервні копії для запобігання втраті інформації;
- забезпечують швидкий доступ до потрібної інформації за допомогою правильної каталогізації;
- надають можливість віддаленого доступу.

Актуальність даної дипломної роботи обумовлена переходом дитячих музичних шкіл з паперового документообігу на електронний, який може забезпечуватись системою, яка спрощуватиме процес обліку успішності учнів.

Учні музичних шкіл мають особливі потреби та вимоги, і облік їх успішності є важливим завданням. Інформаційна система, яка допомагає відстежувати навчальний прогрес та досягнення учнів, може сприяти покращенню якості навчання та розвитку музичної освіти.

Об'єкт дослідження – процес автоматизованого обліку успішності учнів музичних шкіл.

Предмет дослідження – способи організації автоматизованого ведення обліку успішності учнів музичних шкіл.

Мета дослідження – розробка простого у використанні інтерактивного вебзастосунок для автоматизованого обліку успішності учнів музичних шкіл.

Завдання дослідження:

а) познайомитись із роботою електронних систем ведення успішності здобувачів освіти в різних закладах освіти, визначити їх можливості;

б) познайомитися із особливостями ведення обліку успішності учнів музичних шкіл;

в) спроектувати вебзастосунок з наступними функціональними можливостями:

1) внесення, збереження, редагування та видалення інформації про здобувачів освіти ДМШ;

2) внесення, збереження, редагування та видалення інформації про викладачів ДМШ;

3) внесення, збереження, редагування інформації в журналах успішності та відвідування занять учнів окремого викладача;

4) ведення загального журналу успішності учнів по відділах школи.

Методи дослідження:

– теоретичні: аналіз і систематизація науково-теоретичних та методичних джерел;

– емпіричні: аналіз програм-аналогів.

Даний дипломний проєкт розрахований на створення системи обліку успішності учнів однієї музичної школи. В подальшому можливе розширення функціоналу системи та її масштабування і підключення усіх музичних шкіл міста, області або, навіть, всієї країни.

1 ОГЛЯД ІСНУЮЧИХ СИСТЕМ ДЛЯ ОБЛІКУ УСПІШНОСТІ УЧНІВ ТА ОБҐРУНТУВАННЯ ТЕМИ ДИПЛОМНОГО ПРОЄКТУ

1.1 Загальний опис предметної області

Процес навчання в ДМШ спрямований на передачу та засвоєння музичних знань, вмінь і навичок, передбачених навчальним планом. Навчання в ДМШ має на меті оволодіння такими знаннями, вміннями та навичками практичної музично-естетичної діяльності, які б відповідали навчальному плану. Обов'язковими компонентами в цьому процесі є контроль теоретичних знань, умінь та практичних навичок, тобто перевірка його результативності [1].

Процес навчання в ДМШ здійснюється в контексті початкової мистецької освіти, яка поділяється на елементарний, базовий та поглиблений підрівні. Після закінчення кожного підрівня видається свідоцтво з оцінюванням навчальних дисциплін. На даний момент Державним науково-методичним центром змісту культурно-мистецької освіти розроблено Типову навчальну програму тільки для елементарного та базового підрівнів. Учні вже після них мають можливість вступати до мистецьких коледжів.

Етапи контролю знань:

- перевірка – виявлення знань на певному рівні (теоретичних або практичних) для їх подальшого оцінювання;
- оцінка – визначення рівня знань, умінь, компетенцій учня на практиці;
- облік – фіксація рівня знань учня у вигляді оцінок у журналі обліку успішності.

На першому та другому роках навчання елементарного підрівня учні оцінюються не бальною шкалою, а визначається загальний рівень знань, який має свою назву. Назви рівнів: «високий», «достатній», «середній», «недостатній».

На третьому і четвертому роках навчання елементарного підрівня та на наступних підрівнях рівень знань здобувачів освіти оцінюється 12-бальною шкалою.

Контроль отриманих знань, умінь, компетенцій є частиною навчального процесу. Розрізняють поточний, модульний та підсумковий контроль.

Поточний контроль відбувається на уроках (поурочний контроль). Це передбачає систематичне спостереження викладачем за активністю учнів під час занять та їх самостійною працею вдома. Основною метою цього контролю є збір оперативної інформації про рівень засвоєння матеріалу та загальний рівень підготовки учнів. Крім того, цей вид контролю служить для оцінки якості навчального процесу під час занять та ефективного управління навчальним процесом.

Модульний контроль призначений для оцінки рівня теоретичних знань та практичних умінь та компетенцій учнів, які набувались на попередніх заняттях поточного модуля. Його метою є визначення ступеня володіння знаннями та їх відповідність програмі навчання. Цей вид контролю спрямований на перевірку рівня розуміння певної теми учнями. Може здійснюватися у формі контрольних уроків, прослуховувань, заліків, у тому числі технічних.

Підсумковий контроль здійснюється після завершення кожного року навчання. Проводиться в кінці навчального семестру у вигляді прослуховувань або академічних концертів з фаху, хорового класу, додаткового музичного інструменту, ансамблю/оркестру. Та у вигляді контрольної роботи з музично-теоретичних дисциплін.

По закінченню кожного з підрівнів початкової мистецької освіти учні складають комплексний іспит, який включає в себе перевірку знань, умінь та набутих компетенцій з усіх основних навчальних дисциплін.

Особливістю ведення обліку успішності учнів музичних шкіл є те, що навчання відбувається не тільки в групах, а і індивідуально.

Учні певної спеціальності діляться на класи викладачів, які навчають їх грі на основному музичному інструменті. Такі заняття відбуваються в

індивідуальному форматі. Викладач веде журнал відвідування учнями свого класу уроків та індивідуальний журнал моніторингу опанування учнем нормативного змісту освіти.

Також є групові дисципліни, які ведуть облік успішності учнів за аналогією загальної середньої освіти, але мають свої особливості.

Для визначення функціональних можливостей застосунку, який розроблюється в даному дипломному проєкті, було проведено аналіз вже наявних програмних рішень, які пропонують функціонал для ведення електронного журналу успішності учнів ЗЗСО та ЗВО.

1.2 Огляд існуючих програмних рішень

Ідея створення альтернативи паперовому журналу та зберігання даних успішності учнів у комп'ютері не є новою. Запровадження електронного журналу значно спростило доступ до інформації про академічні досягнення учнів, дозволяючи здійснювати різноманітні статистичні запити для подальшого аналізу. Розробка електронних журналів здійснюється як професійними компаніями, так і викладачами та студентами ЗВО.

Для загальної середньої та вищої освіти є ряд існуючих державних та комерційних електронних журналів: «Нові знання», «Electronic Journal», «Єдина Школа», «Атомс», «Google Classroom», «На Урок», «Всеосвіта», «Human», «Moodle» та інші. Проаналізуємо деякі з них.

1.2.1 Google Classroom

Google Classroom – це вебзастосунок, в якому можна створювати власні курси, а також приєднуватись до вже наявних курсів у якості учня. Ця система дозволяє викладачу завантажувати в хмарне сховище матеріали уроків, створювати посилання для онлайн-уроків, створювати домашні завдання та моніторити журнал з оцінками. Особистий кабінет представлено на рис. 1.1.

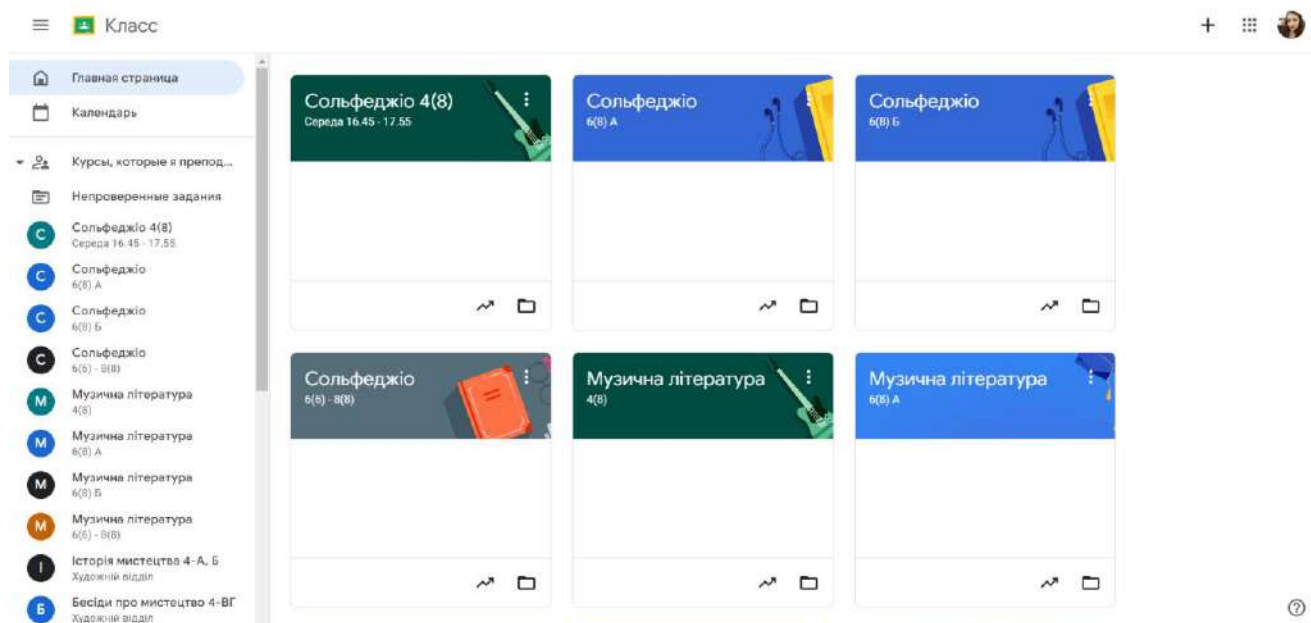


Рисунок 1.1 – Інтерфейс головної сторінки Google Classroom авторизованого користувача

Переваги та недоліки Google Classroom відображено в таблиці 1.1.

Таблиця 1.1 – Переваги та недоліки Google Classroom

Переваги	Недоліки
Існує досить давно, є одним з перших, тому багато педагогічних працівників освоїли функціонал	Універсальний функціонал підходить не під усі види навчальних дисциплін та закладів освіти
Простий та інтуїтивно зрозумілий інтерфейс	Структура журналу не відповідає навчальній документації, немає можливості формувати звітність з успішності
Безкоштовне використання	Доступні корисні функції тільки у версії для навчальних закладів
Є мобільний застосунок	Журнал відображає оцінки тільки за створені завдання, немає можливості ставити оцінки за відповіді на уроці
Учень може бачити свої перевірені завдання	Учень не має повної картини усіх поточних оцінок

1.2.2 Система «Єдина школа»

Система «Єдина школа» рекомендована Міністерством освіти і науки України для використання у ЗЗСО та ЗВО I-II рівнів акредитації. Ця система повноцінно дозволяє організувати освітній процес у загальноосвітній школі або у закладі середньо-спеціальної освіти, містить журнал успішності, журнал розкладу, зведення досягнень учнів, загальні відомості про учнів та інше. Особистий кабінет представлено на рис. 1.2.

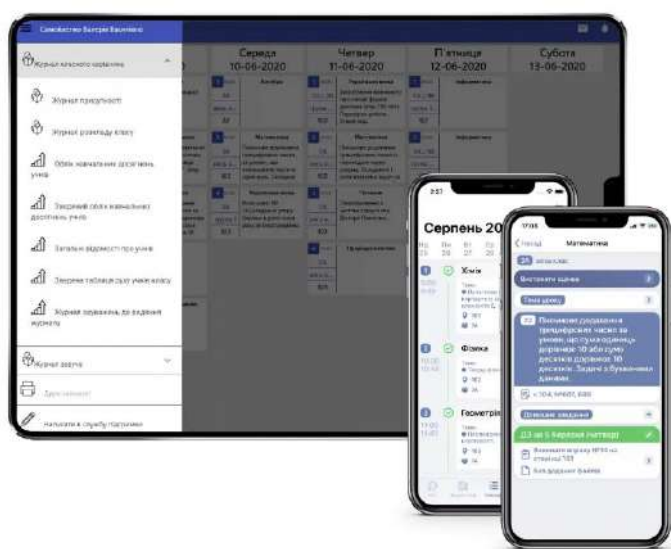


Рисунок 1.2 – Приклад кабінету вчителя в системі «Єдина школа»

Переваги та недоліки системи «Єдина школа» відображено в таблиці 1.2.

Таблиця 1.2 – Переваги та недоліки системи «Єдина школа»

Переваги	Недоліки
Рекомендований МОН	Не підходить для навчальних закладів інших рівнів
Має мобільний застосунок	Немає можливості вираховувати середній бал з дисципліни
Відповідає специфіці ЗЗСО, ЗВО I-II рівня акредитації	Немає можливості розподілу класу на групи
Можна викладати матеріали до уроків та завдання	
Учні та батьки мають доступ до журналу, розкладу та оцінок	
Зрозумілий інтерфейс	

1.2.3 Electronic Journal

Electronic Journal – державна безкоштовна система електронних щоденників та журналів для ЗЗСО. Ця система повноцінно дозволяє організувати освітній процес у загальноосвітній школі, містить журнал успішності, журнал розкладу, щоденник учня, освітню статистику та інше. Сторінка журналу представлена на рис. 1.3.

№	ПІБ	Січень			
		10	17	24	31
1	Андрусенко Марина				
2	Бородея Катерина				
3	Данилюк Олег				
4	Іщенко Світлана				
5	Максименко Олена				
6	Тихенко Олександр				
7	Шахрай Віталій				

Рисунок 1.3 – Журнал оцінок класу з певного предмету в системі «Electronic Journal»

Переваги та недоліки «Electronic Journal» відображено в таблиці 1.3.

Таблиця 1.3 – Переваги та недоліки системи «Electronic Journal»

Переваги	Недоліки
Рекомендований МОН, відповідає нормам законодавства про освіту	Не підходить для навчальних закладів інших рівнів
Простий і зрозумілий інтерфейс	Складність зареєструватись через вимогу надання великої кількості документів
Учні та батьки мають доступ до журналу, розкладу та оцінок	Немає мобільного застосунку

1.2.4 Портал «Нові знання»

Портал «Нові знання» – портал, який повноцінно дозволяє організувати освітній процес у загальноосвітній школі, містить журнал успішності, журнал розкладу, зведення досягнень учнів, загальні відомості про учнів та інше. Вчитель має можливість виставляти матеріали уроків та завдання для учнів. Також цей портал містить онлайн бібліотеку та соціальні функції. Сторінка журналу представлена на рис. 1.4.

Рисунок 1.4 – Журнал оцінок класу з певного предмету на порталі «Нові знання»

Переваги та недоліки порталу «Нові знання» відображено в таблиці 1.4.

Таблиця 1.4 – Переваги та недоліки порталу «Нові знання»

Переваги	Недоліки
Стильний дизайн, зрозумілий інтерфейс	Не підходить для навчальних закладів інших рівнів
доступ учнів до журналу, розкладу та оцінок, можливість провадити дистанційне навчання	За відгуками у PlayMarket: сервери не витримують навантаження в кінці семестру, часті помилки на сторінках
Має онлайн бібліотеку та соціальні функції	
Є мобільний застосунок	
безкоштовний	
В розробці функції онлайн-конференцій, сповіщення в месенджерах та інші	

1.2.5 «Музична школа»

«Музична школа» – електронні щоденники та журнали для музичних шкіл. Нині сайт не функціонує, має статус «Тестова версія. Реєстрація тимчасово закрита». Домен зареєстрований на ФОП. Стартова сторінка представлена на рис. 1.5.

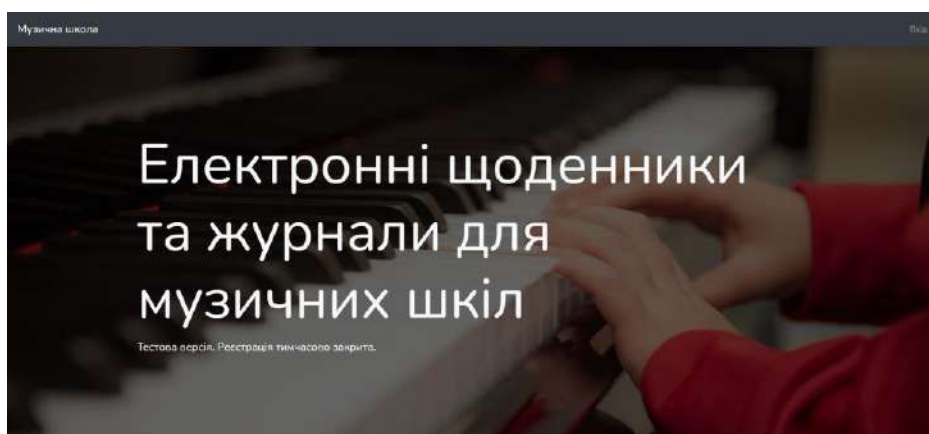


Рисунок 1.5 – Стартова сторінка сайту «Музична школа»

Можна зробити висновок, що ідея створення електронного журналу для музичних шкіл виникала вже не вперше, але донині немає працюючої версії такої інформаційної системи.

1.2.6 Порівняння існуючих програмних рішень

Критерії порівняння:

- підходить для використання в музичних школах (1);
- є можливість ведення успішності учнів (2);
- є можливість збереження інформації про учнів (3);
- є можливість збереження інформації про викладачів (4);
- є можливість формування списку учнів з підсумковими оцінками (5);
- безкоштовний (6).

Результати порівняння розглянутих в підрозділі 1.2.5 існуючих програмних засобів за визначеними критеріями представлені в таблиці 1.5.

Таблиця 1.5 – Порівняльна таблиця існуючих програмних рішень

ПЗ	1	2	3	4	5	6
Google Classroom	-	частково	-	-	-	+
Єдина школа	-	+	+	+	-	+
Electronic Journal	-	+	+	+	-	+
Нові знання	-	+	+	+	-	+
Музична школа	+	немає інформації	немає інформації	немає інформації	немає інформації	немає інформації

Можна зробити висновок, що розробка інформаційної системи обліку успішності учнів музичних шкіл є актуальною, так як усі існуючі системи обліку успішності учнів призначені для ЗЗСО та ЗВО, а працюючих аналогів для музичних шкіл нині не існує на українському ринку.

1.3 Визначення загальних вимог до системи

Електронний журнал має надавати змогу користувачам реалізовувати усі допустимі сценарії роботи з вебзастосунком.

1.3.1 Функціональні вимоги

Даний вебзастосунок має три ролі: адміністратор (директор, завуч), викладач, учень. Кожна роль має набір функціональних можливостей.

Для адміністратора вебзастосунок повинен забезпечувати наступні можливості:

- перегляд, додавання, редагування та видалення відділів школи в системі;
- перегляд, додавання, редагування та видалення спеціальностей на відділах;
- перегляд, додавання, редагування та видалення навчальних дисциплін;
- додавання, перегляд, редагування та архівування даних учнів та викладачів в системі;

- призначення та видалення дисциплін викладачам;
- призначення та видалення класів та учнів викладачам;
- перегляд журналів викладачів та успішності учнів;
- архівування журналів викладачів та створення їм нових журналів;
- створення загального журналу успішності учнів по відділах та архівування загального журналу успішності учнів по відділах за минулий навчальний рік;
- перенесення оцінок за контрольний захід та підсумкових із журналів викладачів у загальний журнал успішності учнів по відділах.

Для викладача вебзастосунок повинен забезпечувати наступні можливості:

- перегляд даних учнів в системі;
- перегляд персональних даних в системі;
- ведення успішності учнів, у яких викладає.

Для учня вебзастосунок повинен забезпечувати наступні можливості:

- перегляд персональних даних у системі;
- перегляд своїх поточних та підсумкових оцінок з різних предметів;
- перегляд контактної інформації своїх викладачів.

Для адміністратора, викладача, учня вебзастосунок повинен забезпечувати наступні можливості:

- авторизація в системі;
- вихід із системи.

1.3.2 Нефункціональні вимоги:

- інтуїтивно-зрозумілий та зручний інтерфейс;
- наявність підказок до функціоналу системи.

Специфікація вимог до програмного забезпечення (SRS) – це детальний опис програмної системи, яка розроблюється на основі функціональних та нефункціональних вимог (див. додаток А).

1.4 Обґрунтування доцільності створення інформаційної системи обліку успішності учнів музичних шкіл

На даний момент в музичних школах м. Запоріжжя електронні журнали ведуться в Google Documents. Такий метод має низку недоліків. Найбільший з яких, це створення таблиць вручну, копіювання їх на всі сторінки журналу і введення інформації вручну на кожній сторінці. Також такий спосіб унеможливорює автоматизацію процесу заповнення журналу та процесу формування списків учнів по відділах з підсумковими оцінками.

В силу власної специфіки навчання та бланків документації для музичних шкіл існуючі системи ведення обліку успішності учнів не підходять, так як вони «заточені» саме під потреби ЗЗСО та ЗВО.

Зважаючи на вищевикладене є актуальною проблема проєктування інформаційної системи електронного обліку успішності учнів музичних шкіл у вигляді електронного вебзастосунку.

1.5 Висновки до розділу

В даному розділі проаналізовано предметну область, де розглянуто процес обліку успішності учнів ДМШ, розглянуто поточний шлях ведення цього процесу.

Також було розглянуто та проаналізовано існуючі системи обліку успішності учнів, шляхом порівняння виявлено їх переваги та недоліки.

На основі проаналізованих даних було сформовано загальні вимоги до системи, розроблено специфікацію вимог SRS та обґрунтовано актуальність теми даного дипломного проєкту.

Даний програмний продукт має «клієнт-серверну» архітектуру. Користувачі поділяються на три ролі: Адміністратор, Викладач, Учень. Кожен користувач має

персональний доступ до системи та ряд функціональних можливостей. Адміністратор додає нових користувачів у систему, організовує функціонал Викладача. Викладач організовує свою роботу. Учень переглядає свою успішність з різних навчальних дисциплін.

З нефункціональних вимог визначено інтуїтивно-зрозумілий та простий інтерфейс та наявність підказок з використання функціоналу системи.

2 АНАЛІЗ ТА ВИБІР ЗАСОБІВ РЕАЛІЗАЦІЇ СИСТЕМИ

Інформаційна система обліку успішності учнів музичних шкіл реалізовуватиметься у форматі вебзастосунку. Це обумовлено доступністю, користуватись таким застосунком зможуть усі користувачі, які мають гаджет з доступом до інтернету та браузер, незалежно від операційної системи без додаткових налаштувань програми. Таким гаджетом може бути смартфон, планшет, ноутбук. Також таке рішення обумовлено спрощенням установки програмного застосування та процесом користування.

Вебзастосунок складається з клієнтської частини, серверної та бази даних. Для розробки вебзастосунку потрібно проаналізувати доступні мови програмування, фреймворки, бібліотеки, обрати базу даних для збереження даних.

2.1 Обґрунтування вибору мови програмування

2.1.1 Клієнтська частина

Клієнтська частина відповідає за взаємодію користувача з системою, виступає в ролі зручного і зрозумілого інтерфейсу. Вона реалізовуватиметься за

допомогою мови розмітки веб-документу HTML, мови опису та зміни зовнішнього вигляду елементів веб-сторінки CSS, використовуватиметься CSS-препроцесор SASS, технологія адаптивної верстки, мови програмування JavaScript.

HTML – це мова розмітки веб-сторінок. HTML відповідає за наповнення веб-сторінки контентом, структуру та зміст. HTML складається з елементів, які називаються тегами [2].

CSS – це мова опису зовнішнього виду елементів веб-сторінки. CSS використовується для оформлення та позиціонування, зовнішній вигляд [3].

JavaScript – це об'єктно-орієнтована мова програмування, що була розроблена Бренданом Айхом у 1995 році. Вона стала стандартом ECMA (European Computer Manufacturers Association) у 1997 році. JavaScript використовується для реалізації інтерактивності веб-сторінок. Вона дозволяє розробникам створювати веб-сторінки, які можуть реагувати на дії користувачів, наприклад, на натискання кнопок або інших елементів сторінки, переміщення миші чи прокручування сторінки [4].

CSS-препроцесор – надбудова, що розширює CSS-синтаксис, що полегшує написання, підтримку та рефакторинг коду. Препроцесори дозволяють використовувати конструкції, які прискорюють і спрощують написання коду. Найпопулярніші препроцесори це SASS, LESS та Stylus. Всі вони схожі і вирішують одне завдання, тому достатньо вибрати та використати один. В цьому проєкті використовуватиметься препроцесор – SASS та його синтаксис SCSS.

2.1.2 Серверна частина

Серверна частина відповідає за звернення клієнтської частини до бази даних, обробляє запити клієнта тощо.

Визначимо мову програмування, за допомогою якої можна побудувати серверну частину. Проаналізуємо Python, PHP, JavaScript та Go.

Python – це високорівнева, інтерпретована, загального призначення мова програмування [5].

Однією з основних характеристик JavaScript є можливість динамічно змінювати вміст сторінки без необхідності перезавантаження. Це досягається шляхом маніпулювання DOM, представленням структури HTML-документа. Мова підтримує асинхронність. Це дозволяє виконувати запити до сервера без блокування інтерфейсу користувача. Це особливо важливо для отримання або відправки даних без очікування повного завантаження сторінки. JavaScript утворює основу для створення вебсайтів, а його використання може розширюватися на розробку серверної частини за допомогою платформи Node.js [6].

Node.js – це відкрите середовище виконання на стороні сервера, що використовує рушій JavaScript Google V8. Дозволяє виконувати JavaScript-код на сервері, що раніше було характерно переважно для виконання на клієнтській стороні [7]. Переваги та недоліки Node.js відображені в таблиці 2.1.

Таблиця 2.1 – Переваги та недоліки Node.js

Переваги	Недоліки
Використання JavaScript на серверній частині, що забезпечує ефективність, так як на клієнтській частині теж використовується JavaScript	Нестабільність API
Пакетний менеджер NPM, який містить велику кількість пакетів	Однопоточне середовище (великі задачі, які потребують інтенсивного використання ресурсів процесору, працюють не ефективно)
Широко використовуване середовище розробки вебзастосунків	Недостатня підтримка бібліотек (використання бібліотек, які не містяться в ядрі Node.js)
Невеликий поріг входу в розробку (простота вивчення)	Підтримка асинхронного коду складніша за підтримку синхронного

PHP – це скриптова мова програмування, призначена для розробки вебзастосунків та динамічних веб-сторінок. Спочатку створювалася як мова для розробки на стороні сервера, що дозволяє генерувати HTML-код на сервері та відправляти його на клієнтський браузер [8].

Go – це мова програмування, також відома як Golang, розроблена компанією Google. Go використовують для розробки високопродуктивних програм, зокрема веб-серверів, мікросервісів та інших застосунків [9].

2.1.3 Порівняння мов програмування

Вибір мови програмування для серверної частини – важливе рішення, яке впливає на продуктивність системи. Порівняння розглянутих мов програмування можна провести на основі їх переваг, недоліків та інших ключових аспектів, наведених у таблиці 2.2.

Таблиця 2.2 – Порівняльна характеристика мов програмування для серверної частини

Критерій	Python	JavaScript	PHP	Go
Простота написання коду	+	+	–	–
Багатофункціональність	+	+	+	+
Паралельне програмування	+	+	+	+
Наявність бібліотек	+	+	+	+
Документація	+	+	+	+
Інтеграція з іншими мовами	+	+	+	+/-
Висока візуалізація даних	+	+	–	–
Обробка операцій з великим обсягом даних	+	+	–	+
Ліцензія	+	+	–	–

З таблиці 2.2 видно, що найбільші переваги мають мови програмування Python та JavaScript. Зважаючи на переваги JavaScript саме у веб-розробці, підтримку багатьох фреймворків, а також предметно-орієнтованих інструментів з відкритим кодом, для розробки серверної частини було обрано мову програмування JavaScript.

2.2 Обґрунтування вибору фреймворку

2.2.1 Серверна частина

Для розроблення серверної частини вебзастосунку потрібно обрати фреймворк середовища JavaScript Node.js. Розглянемо Express.js, Nest.js та Adonis.js.

Express.js – це мінімалістичний та гнучкий веб-фреймворк для програм Node.js, що надає широкий набір функцій для мобільних та вебзастосунків. Він має у своєму розпорядженні безліч службових методів HTTP (HyperText Transfer Protocol) та проміжних обробників, щоб створити надійний API або веб-сайт [10]. Дозволяє використовувати різні шаблонізатори, такі як Pug (Jade), EJS (Embedded JavaScript), Handlebars та інші для генерації динамічних HTML-сторінок.

Adonis.js – повнофункціональний фреймворк, заснований на принципах MVC (Model-view-controller). Надає інструменти для розробки вебзастосунків, в тому числі ORM (Object-relational mapping) для роботи з базою даних, механізм маршрутизації та підтримку автентифікації. Має вбудовану підтримку протоколу WebSocket, що дозволяє створювати застосунки, які забезпечують двосторонній зв'язок між сервером та клієнтом в реальному часі.

Nest.js – потужний фреймворк для розробки масштабованих серверних додатків з використанням мови TypeScript. Він побудований на основі популярного фреймворку Express.js та надає зручні інструменти для створення ефективних та модульних додатків. Головною особливістю Nest.js є його архітектура, яка базується на концепції "Модуль-Контролер-Провайдер" (MCP – Module-Controller-Provider). Це означає, що застосунок розбивається на незалежні модулі, кожен з яких містить свої контролери та провайдери [11].

Беручи до уваги масштаби проєкту та просту архітектуру, найкраще підійде легкий та мінімалістичний фреймворк Express.js.

2.2.2 Клієнтська частина

Розглянемо три фреймворки: React.js, Angular.js та Vue.js [12].

React.js – це JavaScript-бібліотека з відкритим кодом, створена компанією Facebook. Вона дозволяє розробити користувацький інтерфейс у вигляді окремих компонентів та забезпечує їх синхронізацію зі змінами даних.

Vue.js – це прогресивний JavaScript-фреймворк, який також надає зручні засоби для створення інтерфейсів. Vue.js використовує шаблони для опису інтерфейсу, але також підтримує `render`-функції та навіть підтримку JSX (синтаксис React). За допомогою шаблонів або `render`-функцій розробники можуть описати, як має виглядати кожен компонент та як взаємодіяти з даними.

Angular.js – це фреймворк, розроблений для роботи з великими програмами. Він надає широкі можливості для структурування та організації складних застосунків. Однією з важливих особливостей Angular є вбудована підтримка TypeScript, який є обов'язковим у роботі з цим фреймворком.

Для написання даного проєкту було обрано фреймворк React.js завдяки ефективному використанню ресурсів, простоті розробки, розширюваності та широким можливостям.

Особливості React.js:

- віртуальний DOM, що забезпечує ефективну та швидку роботу з інтерфейсом;
- компонентний підхід (кожен компонент – незалежна частина інтерфейсу, компоненти можуть використовуватись повторно);
- односторонній потік даних (дані передаються вниз по ієрархії, вгору дані можуть оновлюватись тільки через передачу `props`);
- дозволяє описувати інтерфейс за допомогою HTML-подібних тегів;
- велике ком'юніті (багато ресурсів, документації, бібліотек і пакетів).

2.3 Обґрунтування вибору бази даних

База даних зберігає дані учнів, викладачів, адміністратора, оцінки та інше. Саме до неї зберігаються дані, внесені користувачем, а також з неї ці дані потім через серверну частину відображаються у браузері.

Розглянемо PostgreSQL, MongoDB та MySQL.

2.3.1 PostgreSQL

PostgreSQL – це об'єктно-реляційна база даних з відкритим вихідним кодом.

Переваги: багато функціональних можливостей, потужна, інтерфейс у вигляді PgAdmin4, підтримує географічні об'єкти, може запускати динамічні вебсайти (стек LAMP – Linux, Apache, MariaDB/MySQL, PHP) [13].

Недоліки: не належить жодній організації, підтримують не всі застосунки з відкритим кодом, може працювати з великими об'ємами даних, підвищене використання ресурсів, багато налаштувань (новачкам може бути складно).

2.3.2 MongoDB

MongoDB – це документоорієнтована база даних з відкритим вихідним кодом, яка зберігає дані у вигляді документів у форматі JSON (JavaScript Object Notation). Не використовує табличну структуру для зберігання даних [14].

Переваги: гнучка у налаштуванні та використанні, може бути розподілена між декількома базами даних, швидка, гарно масштабується.

Недоліки: займає доволі багато пам'яті, розмір одного файлу не має перевищувати 16 мегабайт, максимальна кількість підключень – 20000, підходить для зберігання даних, які не мають багато зв'язків між собою.

2.3.3 MySQL

MySQL – вільна система керування реляційними базами даних. MySQL є рішенням для малих і середніх застосунків [15].

Переваги: підтримка безпеки даних, висока продуктивність, добре масштабується, проста у встановленні та використанні.

Недоліки: не підтримує SQL у повному обсязі, слабкі збережені процедури, вимоги ACID (Atomicity, Consistency, Isolation, Durability) підтримуються лише при використанні з InnoDB та NDB Cluster Storage, складно налагоджувати збережені процедури.

В результаті аналізу декількох СКБД було обрано MySQL, оскільки вона є реляційною, що підходить для чіткого структурування даних та взаємозв'язків між ними, написання індексів для прискорення пошуку даних. Також масштаб версії системи в рамках цього дипломного проєкту не потребує більш потужної СКБД.

2.4 Висновки до розділу

В цьому розділі було проаналізовано засоби реалізації системи обліку успішності учнів музичних шкіл, їх недоліки та переваги. В результаті було обрано найоптимальніші варіанти для розробки системи.

Для серверної частини вебзастосунку було обрано фреймворк Express.js середовища JavaScript Node.js.

Для клієнтської частини було обрано фреймворк React мови JavaScript, так як він використовує ту ж мову, що й клієнтський фреймворк, що є дуже зручним при розробці та відлагодженні.

В ролі СКБД для цього проєкту було обрано MySQL.

3 РОЗРОБКА ПРОГРАМНОГО ЗАСТОСУНКУ ІНФОРМАЦІЙНОЇ СИСТЕМИ

3.1 Загальна структура програми

Розроблюваний вебзастосунок складається з трьох частин: клієнтської, серверної та бази даних.

Клієнтська частина (frontend) – це інтерфейс, за допомогою якого користувачі взаємодіють з базою даних. Тут викладачі ведуть облік успішності учнів музичних шкіл, адміністратор організовує роботу викладачів, реєструє в системі користувачів, учні переглядають свою успішність.

Серверна частина (backend) відповідає за звернення клієнтської частини до бази даних, за обробку цих запитів, авторизацію користувачів тощо.

Маємо три ролі в системі: Адміністратор, Викладач, Учень. Функціональні можливості системи розглядались в попередньому розділі.

Загалом матимемо наступну структуру сторінок вебзастосунку:

- а) в особистому кабінеті Адміністратора (див. додаток Б);
- б) в особистому кабінеті Викладача (див. рис. 3.1);



Рисунок 3.1 – Структура сторінок в особистому кабінеті викладача

в) в особистому кабінеті Учня (див. рис. 3.2);

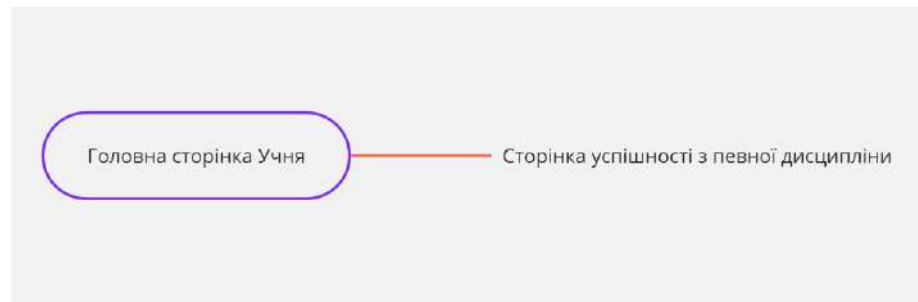


Рисунок 3.2 – Структура сторінок в особистому кабінеті учня

г) інші сторінки:

- 1) головна сторінка вебзастосунку, до якої мають доступ як авторизовані користувачі, так і неавторизовані (має меню з якорями на головній сторінці);
- 2) сторінка «Підключити заклад»;
- 3) сторінка «Не знайдено» в разі помилки.

3.2 Архітектура програмного застосування

Під час роботи користувача з веб-додатком відбувається інтенсивна та постійна взаємодія частин проекту між собою. Інтерфейс користувача взаємодіє з серверною частиною. Серверна частина в свою чергу взаємодіє з базою даних. В такому процесі відбувається майже непереривний обмін інформацією між модульними частинами проекту. На рис. 3.3 представлені інформаційні потоки у веб-додатку.

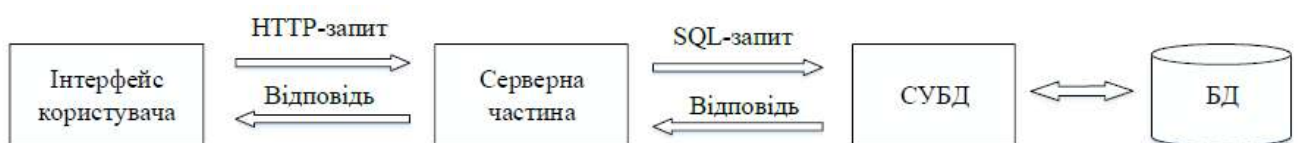


Рисунок 3.3 – Інформаційні потоки у веб-додатку

3.2.1 Структура серверної частини

Створення масштабованих веб-додатків потребує хорошої архітектури коду та файлів. Це робить обслуговування легшим і більш організованим. Структура проєкту представлена на рис. 3.4.

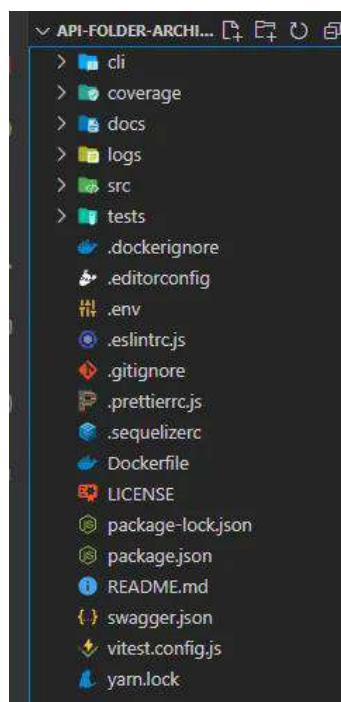


Рисунок 3.4 – Структура проєкту серверної частини

Тека підтримки CLI надає інтерфейс командного рядка (CLI), який можна використовувати для автоматизації створення нових компонентів, також містить команди для інших завдань, таких як виконання модульних тестів, створення програми та запуск сервера розробки. Структура теки відображена на рис. 3.5.

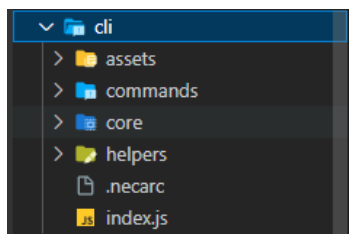


Рисунок 3.5 – Вміст теки cli

Тека coverage в архітектурі використовується для зберігання даних покриття коду. Дані про покриття коду генеруються шляхом виконання модульних та інтеграційних тестів. Він показує, скільки коду програми було виконано тестами. Структура теки відображена на рис. 3.6.

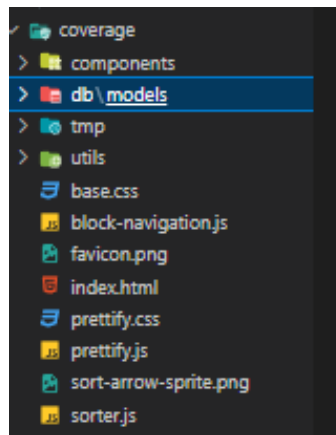


Рисунок 3.6 – Вміст теки coverage

Тека coverage містить два типи файлів, які знаходяться у відповідній тестовій теці:

- файли .json: ці файли містять дані покриття коду у форматі JSON;
- файли .html: ці файли створюються з файлів .json і їх можна відкрити у веб-браузері для перегляду даних покриття коду в графічному форматі.

Тека logs використовується для зберігання журналів програми. Ці журнали можуть містити:

- журнали помилок: ці журнали містять інформацію про помилки, які сталися в програмі;
- інформаційні журнали: ці журнали містять інформацію про дії програми, наприклад оброблені запити та виконані запити до бази даних;
- журнали налагодження: ці журнали містять детальну інформацію про дії програми, наприклад значення змінних і порядок виконання коду.

Тека src у node-express-modular-architecture є основною папкою коду програми. Він містить усі вихідні файли програми (див. рис. 3.7).

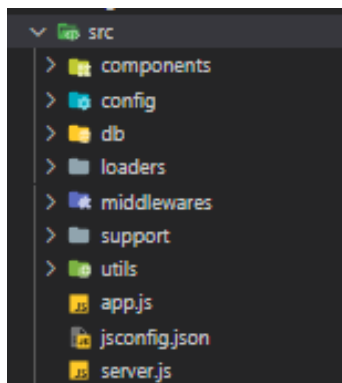


Рисунок 3.7 – Вміст теки src

Підкаталог components використовується для зберігання багаторазового коду, який можна використовувати в кількох частинах програми (див. рис. 3.8).

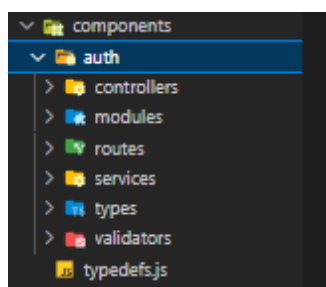


Рисунок 3.8 – Вміст підкаталогу components у теці src

Цей код може містити:

- помічники: помічники — це багаторазові функції, які можна використовувати для виконання типових завдань.
- служби: служби – це багаторазово використовувані об’єкти, які надають доступ до даних або функцій.

Підкаталог components організований у підкаталоги, у кожному з яких міститься код для певного типу компонента. Наприклад, користувацькі компоненти можуть зберігатися у підкаталозі під назвою custom, тоді як допоміжні компоненти можуть зберігатися у підкаталозі під назвою helpers.

Підкаталог config містить змінні середовища та пов’язані з конфігурацією файли.

Підкаталог `loaders`: містить маршрути та конфігурації `Lodash`, також перевіряє конфігурації.

Підкаталог `middlewares` містить спеціальні проміжні програми `Express`. Функції проміжного програмного забезпечення можна використовувати для виконання різноманітних завдань, наприклад:

- автентифікація: функції проміжного програмного забезпечення можна використовувати для автентифікації користувачів, перш ніж їм буде дозволено доступ до програми;
- ведення журналу: функції проміжного програмного забезпечення можна використовувати для реєстрації запитів і відповідей;
- кешування: функції проміжного програмного забезпечення можна використовувати для кешування відповідей для підвищення продуктивності;
- обробка помилок: функції проміжного програмного забезпечення можна використовувати для обробки помилок і запобігання збою програми.

Підкаталог `support` містить обгортку навколо використаних пакетів, що дозволяє легко замінити пакет.

Підкаталог `utils` містить службові класи та функції.

`app.js` : містить програму `Express`.

`server.js` : містить точку входу для програми.

Тека `tests` використовується для зберігання модульних тестів та інтеграційних тестів. Модульні тести – це тести, які перевіряють окремі одиниці коду, наприклад функції або класи. Інтеграційні тести – це тести, які перевіряють, як різні одиниці коду взаємодіють один з одним [16].

3.2.2 Структура клієнтської частини

Потрібно створювати проєкт у структурованій формі, щоб він був гарним і легким, а також щоб в ньому могли розібратись інші розробники.

За замовчанням `React`-застосунок має визначену структуру (рис. 3.9).

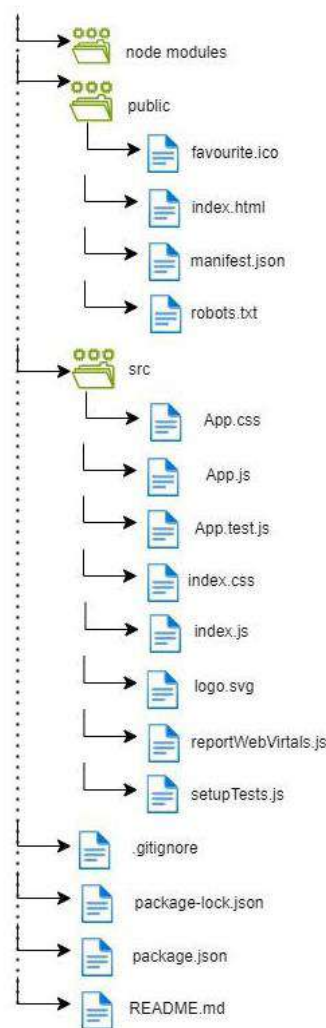


Рисунок 3.9 – Структура React-застосунку

- index.html: цей файл містить базову структуру нашої веб-програми. Необхідно відобразити DOM нашої програми.
- index.js : цей файл є першим запущеним файлом і містить у ньому файл App.js, тоді всі компоненти відображатимуться після нього.
- App.js : містить функцію App, яка може мати всі компоненти для рендерингу один за іншим.
- index.css і App.css : ці обидва файли призначені для оформлення веб-сторінки. index.css має стиль для index.css, а App.css має стиль для нашої функції програми та її елементів. Стиль може відрізнятися залежно від класів та ідентифікаторів таблиці стилів.
- package.json : містить усі залежності, доступні в нашій програмі, і наш сценарій для запуску нашого сервера.

Але ця структура не є повною, в процесі розробки її потрібно доповнити, а саме доповнити теку src підкаталогами: assets, components, views, service, utils, hooks, store.

Тека assets містить статичні файли, такі як логотипи, шрифти та піктограми.

Тека components містить користувацькі компоненти. Користувальницькі компоненти — це багаторазово використовувані блоки, які можна використовувати для створення програми. Для кожного такого компоненту потрібно створити підкаталог в середині теки components, який має назву компоненту, який там описується. Підкаталог компоненту міститиме 3 файли:

- ComponentName.js — цей файл зазвичай містить усі функціональні можливості та логіку, необхідні для компонента;
- ComponentName.css — цей файл містить правила, пов'язані зі стилями компоненту;
- ComponentName.test.js — цей файл містить тестові приклади.

Тека views містить усі веб-сторінки вашої програми React.

Тека service містить файли з кодом для зв'язку із зовнішніми API.

Тека utils містить багаторазові фрагменти функцій для виконання швидких завдань, таких як скорочення тексту або реєстр.

Тека hooks містить файли з кодом, який можна повторно використовувати в кількох компонентах.

Тека store зберігає файли керування станом, наприклад Redux, які використовуються, щоб зробити певні функції та змінні доступними у програмі [17].

3.3 Структура бази даних

В цьому проєкті не буде використовуватись класичний MySQL, а її форк MariaDB.

Для побудови ER-діаграми використовуватимемо MySQL Workbench, а для роботи, безпосередньо, з базою даних DBeaver.

Сутності бази даних (табл. 3.1 – 3.21):

Таблиця 3.1 – Тип сутності departments

Ключ	Тип значення
departmentId	primary key, uuid, not null
name	text, not null

Таблиця 3.2 – Тип сутності specialities

Ключ	Тип значення
specialitiesId	primary key, uuid, not null
name	text, not null
departmentId	foreign key, uuid, not null

Таблиця 3.3 – Тип сутності typesOfSubject

Ключ	Тип значення
typeOfSubjectId	primary key, uuid, not null
name	text, not null

Таблиця 3.4 – Тип сутності subjects

Ключ	Тип значення
subjectId	primary key, uuid, not null
name	text, not null
departmentId	foreign key, uuid, not null
typeOfSubjectId	foreign key, uuid, not null

Таблиця 3.5 – Тип сутності sublevelsOfEducation

Ключ	Тип значення
sublevelOfEducationId	primary key, uuid, not null
name	text, not null

Таблиця 3.6 – Тип сутності termsOfEducation

Ключ	Тип значення
termOfEducationId	primary key, uuid, not null
sublevelOfEducationId	foreign key, uuid, not null
numberOfYears	integer

Таблиця 3.7 – Тип сутності yearsOfStudy

Ключ	Тип значення
yearOfStudyId	primary key, uuid, not null
sublevelOfEducationId	foreign key, uuid, not null
termOfEducationId	foreign key, uuid, not null
yearOfStudy	integer

Таблиця 3.8 – Тип сутності teachers

Ключ	Тип значення
teacherId	primary key, uuid, not null
surname	text, not null
name	text, not null
middleName	text, not null
departmentId	foreign key, uuid, not null
phone	text, not null
education	text, not null
authorizationDataId	foreign key, uuid, not null

Таблиця 3.9 – Тип сутності students

Ключ	Тип значення
studentId	primary key, uuid, not null
surname	text, not null
name	text, not null
middleName	text, not null
specialityId	foreign key, uuid, not null
yearOfAdmission	integer
yearOfStudyId	foreign key, uuid, not null
nameOfMother	text, not null
nameOfFather	text, not null
motherPhone	text, not null
fatherPhone	text, not null
authorizationDataId	foreign key, uuid, not null

Таблиця 3.10 – Тип сутності authorizationData

Ключ	Тип значення
authorizationDataId	primary key, uuid, not null
login	text, not null
password	text, not null
status	text, not null

Таблиця 3.11 – Тип сутності nameOfGroups

Ключ	Тип значення
nameOfGroupId	primary key, uuid, not null
name	text, not null

Таблиця 3.12 – Тип сутності theoryTeachersSubjectsYearOfStudy

Ключ	Тип значення
theoryTeachersSubjectsYearOfStudyId	primary key, uuid, not null
subjectId	foreign key, uuid, not null
yearOfStudyId	foreign key, uuid, not null
teacherId	foreign key, uuid, not null

Таблиця 3.13 – Тип сутності theoryGroups

Ключ	Тип значення
theoryGroupId	primary key, uuid, not null
nameOfGroupId	foreign key, uuid, not null
theoryTeachersSubjectsYearOfStudyId	foreign key, uuid, not null
studentId	foreign key, uuid, not null

Таблиця 3.14 – Тип сутності teacherStudent

Ключ	Тип значення
teacherStudentId	primary key, uuid, not null
subjectId	foreign key, uuid, not null
teacherId	foreign key, uuid, not null
studentId	foreign key, uuid, not null

Таблиця 3.15 – Тип сутності currentGrades

Ключ	Тип значення
currentGradeId	primary key, uuid, not null
subjectId	foreign key, uuid, not null
studentId	foreign key, uuid, not null
date	text, not null
grade	text, not null

Таблиця 3.16 – Тип сутності typesOfFinalGrade

Ключ	Тип значення
typeOfFinalGradeId	primary key, uuid, not null
name	text, not null

Таблиця 3.17 – Тип сутності finalGrades

Ключ	Тип значення
finalGradeId	primary key, uuid, not null
subjectId	foreign key, uuid, not null
studentId	foreign key, uuid, not null
typeOfFinalGradeId	foreign key, uuid, not null
grade	text, not null

Таблиця 3.18 – Тип сутності typesOfJournal

Ключ	Тип значення
typeOfJournalId	primary key, uuid, not null
name	text, not null

Таблиця 3.19 – Тип сутності typeOfJournalTeacher

Ключ	Тип значення
typeOfJournalTeacherId	primary key, uuid, not null
typeOfJournalId	foreign key, uuid, not null
teacherId	foreign key, uuid, not null

Таблиця 3.20 – Тип сутності concertmasterTeacherStudentMarks

Ключ	Тип значення
concertmasterTeacherStudentMarkId	primary key, uuid, not null
teacherStudentId	foreign key, uuid, not null
date	text, not null
teacherId	foreign key, uuid, not null
mark	text, not null

Таблиця 3.21 – Тип сутності concertmasterTheoryGroupMarks

Ключ	Тип значення
concertmasterTheoryGroupMarkId	primary key, uuid, not null
theoryGroupId	foreign key, uuid, not null
date	text, not null
teacherId	foreign key, uuid, not null
mark	text, not null

Зв'язки між сутностями відображено на ER-діаграмі (див. додаток В).

Розглянемо код створення бази даних та таблиці у DBeaver на прикладі сутності «teachers» (лістинги 3.1 – 3.2).

Лістинг 3.1 – Код для створення бази даних

```
CREATE DATABASE music_journal
  CHARACTER SET = 'utf8'
  COLLATE = 'utf8_general_ci';
```

Лістинг 3.2 – Код для створення таблиці

```
CREATE TABLE IF NOT EXISTS teachers (
  teacherId INT UNSIGNED AUTO_INCREMENT NOT NULL COMMENT
  'Ідентифікатор викладача',
  name VARCHAR(50) NOT NULL COMMENT 'Ім'я',
  surname VARCHAR(50) NOT NULL COMMENT 'Прізвище',
  middleName VARCHAR(50) NOT NULL COMMENT 'По батькові',
  departmentId INT UNSIGNED NOT NULL,
  phone VARCHAR(50) NOT NULL COMMENT 'Телефон',
  education VARCHAR(50) NOT NULL COMMENT 'Освіта',
  authorizationDataId INT UNSIGNED NOT NULL,
  CONSTRAINT teachers_PK PRIMARY KEY (teacherId)
  CONSTRAINT departments_departmentId_fk FOREIGN KEY
  (departmentId) REFERENCES departments(departmentId) ON DELETE
  CASCADE
)
ENGINE=InnoDB
DEFAULT CHARSET=utf8
COLLATE=utf8_general_ci;
```

IF NOT EXISTS – створити, якщо немає таблиці, це необов'язковий параметр.

AUTO_INCREMENT – означає, що при додаванні запису число буде збільшуватися на 1, якщо останній запис був 2, то наступний автоматично стане 3 і так далі.

ENGINE – тип сховища, за замовчуванням InnoDB.

COLLATE – також є сенс задавати порівняння рядків, якщо відрізняється від налаштувань бази.

id INT UNSIGNED – це тип. UNSIGNED означає беззнаковий (немає від'ємного числа).

NOT NULL – Обов'язково не порожній.

PRIMARY KEY (id) – каже, яка колонка або які колонки будуть ключовими, можна вписувати скільки завгодно ключів через кому.

ON DELETE CASCADE – означає, що якщо в parents видалити запис, то і в children видаляться всі пов'язані записи.

Опис сутностей у вебзастосунку відображено в таблицях 3.22 – 3.42:

Таблиця 3.22 – Опис сутності departments

Ключ	Значення
departmentId	Ідентифікатор відділу
name	Назва відділу

Сутність departments відповідає за зберігання назв відділів школи.

Таблиця 3.23 – Тип сутності specialities

Ключ	Значення
specialitiesId	Ідентифікатор спеціальності
name	Назва спеціальності
departmentId	Ключ, який поєднує спеціальність з відділом

Сутність specialities відповідає за зберігання назв спеціальностей за відділами.

Таблиця 3.24 – Тип сутності typesOfSubject

Ключ	Значення
typeOfSubjectId	Ідентифікатор типу дисципліни
name	Назва типу дисципліни

Сутність typesOfSubject відповідає за зберігання назв типів дисциплін. За освітньою програмою кожен учень має 5 типів дисциплін, але на кожен тип є декілька дисциплін, які в різних учнів можуть відрізнятись.

Таблиця 3.25 – Тип сутності subjects

Ключ	Значення
subjectId	Ідентифікатор дисципліни
name	Назва дисципліни

Продовження таблиці 3.25

Ключ	Значення
departmentId	Ключ, який поєднує дисципліну з відділом, до якого відноситься
typeOfSubjectId	Ключ, який поєднує дисципліну з її типом

Сутність subjects відповідає за зберігання назв дисциплін за відділами та має інформацію про тип дисципліни.

Таблиця 3.26 – Тип сутності sublevelsOfEducation

Ключ	Значення
sublevelOfEducationId	Ідентифікатор підрівня освіти
name	Назва підрівня освіти

Сутність sublevelsOfEducation відповідає за зберігання назв підрівнів освіти.

Таблиця 3.27 – Тип сутності termsOfEducation

Ключ	Значення
termOfEducationId	Ідентифікатор терміну навчання
sublevelOfEducationId	Ключ, який поєднує термін навчання з підрівнем освіти
numberOfYears	Кількість років навчання за певним терміном навчання

Сутність termsOfEducation відповідає за збереження термінів навчання підрівнів освіти.

Таблиця 3.28 – Тип сутності yearsOfStudy

Ключ	Значення
yearOfStudyId	Ідентифікатор року навчання
termOfEducationId	Ключ, який поєднує рік навчання з терміном навчання підрівня освіти
yearOfStudy	Рік навчання

Сутність yearsOfStudy відповідає за збереження років навчання на певних термінах навчання різних підрівнів освіти.

Таблиця 3.29 – Тип сутності teachers

Ключ	Значення
teacherId	Ідентифікатор викладача
surname	Прізвище
name	Ім'я
middleName	По батькові
departmentId	Ключ, який поєднує викладача та відділ, до якого він відноситься
phone	Номер телефону
education	Освіта
authorizationDataId	Ключ, який поєднує викладача з його даними для авторизації в системі

Сутність teachers відповідає за збереження даних викладачів.

Таблиця 3.30 – Тип сутності students

Ключ	Значення
studentId	Ідентифікатор студента
surname	Прізвище
name	Ім'я
middleName	По батькові
specialityId	Ключ, який поєднує учня зі спеціальністю певного відділу
yearOfAdmission	Рік вступу
yearOfStudyId	Ключ, який поєднує учня з поточним роком навчання на певному терміні навчання певного підрівня освіти
nameOfMother	ПІБ матері
nameOfFather	ПІБ батька
motherPhone	Телефон матері
fatherPhone	Телефон батька
authorizationDataId	Ключ, який поєднує викладача з його даними для авторизації в системі

Сутність students відповідає за збереження даних учнів.

Таблиця 3.31 – Тип сутності authorizationData

Ключ	Значення
authorizationDataId	Ідентифікатор запису авторизаційних даних
login	Логін
password	Пароль
status	Статус доступу

Сутність students відповідає за збереження даних учнів.

Таблиця 3.32 – Тип сутності nameOfGroups

Ключ	Значення
nameOfGroupId	Ідентифікатор групи
name	Назва групи

Сутність nameOfGroups відповідає за збереження назв груп.

Таблиця 3.33 – Тип сутності theoryTeachersSubjectsYearOfStudy

Ключ	Значення
theoryTeachersSubjectsYearOfStudyId	Ідентифікатор призначення вчителя до дисципліни на певному році навчання
subjectId	Ключ, який поєднує призначення з дисципліною
yearOfStudyId	Ключ, який поєднує призначення з роком навчання
teacherId	Ключ, який поєднує призначення з викладачем

Сутність theoryTeachersSubjectsYearOfStudy відповідає за призначення викладача до дисципліни на певному році навчання.

Таблиця 3.34 – Тип сутності theoryGroups

Ключ	Значення
theoryGroupId	Ідентифікатор призначення групи до дисципліни певного року навчання, призначену певному викладачу

Продовження таблиці 3.34

Ключ	Значення
nameOfGroupId	Ключ, який поєднує назву групи з призначенням
theoryTeachersSubjectsYearOfStudyId	Ключ, який поєднує призначення вчителя з дисципліною і роком навчання та групу
studentId	Ключ, який поєднує групу з учнем

Сутність theoryGroups відповідає за створення груп учнів при вивченні музично-теоретичних дисциплін у одного вчителя, який викладає певну дисципліну на певному році навчання, так як кількість дітей може перевищувати допустиму наповнюваність груп за навчальною програмою.

Таблиця 3.35 – Тип сутності teacherStudent

Ключ	Значення
teacherStudentId	Ідентифікатор призначення викладачам учнів з певних дисциплін
subjectId	Ключ, який додає до призначення дисципліну
teacherId	Ключ, який додає до призначення викладача
studentId	Ключ, який додає до призначення учня

Сутність teacherStudent відповідає за призначення викладачам з усіх інших дисциплін, окрім музично-теоретичних, учнів.

Таблиця 3.36 – Тип сутності currentGrades

Ключ	Значення
currentGradeId	Ідентифікатор поточної оцінки
subjectId	Ключ, який поєднує оцінку з дисципліною
studentId	Ключ, який поєднує оцінку з учнем
date	Дата уроку
grade	Оцінка

Сутність currentGrades відповідає за збереження поточних оцінок учнів з різних дисциплін.

Таблиця 3.37 – Тип сутності typesOfFinalGrade

Ключ	Значення
typeOfFinalGradeId	Ідентифікатор типу підсумкової оцінки
name	Назва типу підсумкової оцінки

Сутність typesOfFinalGrade відповідає за збереження назв типів підсумкових оцінок.

Таблиця 3.38 – Тип сутності finalGrades

Ключ	Значення
finalGradeId	Ідентифікатор підсумкової оцінки
subjectId	Ключ, який поєднує оцінку з дисципліною
studentId	Ключ, який поєднує оцінку з учнем
typeOfFinalGradeId	Ключ, який поєднує підсумкову оцінку з її типом
grade	Оцінка

Сутність finalGrades відповідає за збереження різних типів підсумкових оцінок учнів з різних дисциплін.

Таблиця 3.39 – Тип сутності typesOfJournal

Ключ	Значення
typeOfJournalId	Ідентифікатор типу журналу
name	Назва типу журналу

Сутність typesOfJournal відповідає за збереження назв типів журналів.

Таблиця 3.40 – Тип сутності typeOfJournalTeacher

Ключ	Значення
typeOfJournalTeacherId	Ідентифікатор призначення журналу певного типу викладачу

Продовження таблиці 3.40

Ключ	Значення
typeOfJournalId	Ключ, який поєднує призначення з типом журналу
teacherId	Ключ, який поєднує призначення з викладачем

Сутність typeOfJournalTeacher відповідає за призначення викладачам журналів певного типу.

Таблиця 3.41 – Тип сутності concertmasterTeacherStudentMarks

Ключ	Значення
concertmasterTeacherStudentMarkId	Ідентифікатор відмітки про присутність концертмейстера
teacherStudentId	Ключ, який призначає концертмейстера на уроки до певного викладача з певної дисципліни у певного учня
date	Дата уроку
teacherId	Ключ, який поєднує призначення з викладачем, який займає посаду концертмейстера
mark	Відмітка про присутність концертмейстера на уроці

Сутність concertmasterTeacherStudentMarks відповідає за ведення журналу концертмейстера. Концертмейстер може бути призначений на індивідуальні заняття з фаху чи ансамблі до певних викладачів та їх учнів. Концертмейстер в журналі позначає свою присутність чи відсутність на уроці.

Таблиця 3.42 – Тип сутності concertmasterTheoryGroupMarks

Ключ	Значення
concertmasterTheoryGroupMarkId	Ідентифікатор відмітки про присутність концертмейстера
theoryGroupId	Ключ, який призначає концертмейстера на уроки музично-теоретичних дисциплін до певних груп з певних дисциплін у певних викладачів
date	Дата уроку

Продовження таблиці 3.42

Ключ	Значення
teacherId	Ключ, який поєднує призначення з викладачем, який займає посаду концертмейстера
mark	Відмітка про присутність концертмейстера на уроці

Сутність `concertmasterTheoryGroupMarks` також відповідає за ведення журналу концертмейстера. Концертмейстер може бути призначений на групові заняття з музично-теоретичних дисциплін до певних викладачів певних груп з певних дисциплін. Концертмейстер в журналі позначає свою присутність чи відсутність на уроці.

3.4 Розробка інтерфейсу взаємодії користувача з системою

У нефункціональних вимогах до системи було визначено інтуїтивно-зрозумілий та зручний інтерфейс, який розроблювався у десктопній версії застосунку Figma.

Figma — векторний онлайн-сервіс розробки інтерфейсів та прототипування з можливістю організації спільної роботи, що розробляється однойменною компанією. Працює у двох форматах: у браузері та як клієнтський застосунок на десктопі користувача. Зберігає онлайн-версії файлів, з якими працював користувач [18].

На початку визначимо кольорову палітру (див. рис. 3.10) та шрифти (див. рис. 3.11), які використовуватимуться у проєкті.

Головні кольори	Допоміжні кольори
 FFFFFFFF	 757575
 C43012	 FAFafa
 010101	 424242
 252629	
 F5F5F5	

Рисунок 3.10 – Кольорова палітра інтерфейсу користувача інформаційної системи

Text		Text		Text	
Montserrat		Montserrat		Montserrat	
Bold ▾	14	Bold ▾	16	Regular ▾	16

Рисунок 3.11 – Налаштування шрифтів інтерфейсу користувача інформаційної системи

Вебзастосунок має головну сторінку, до якої мають доступ як авторизовані користувачі, так і неавторизовані (див. рис. 3.12).

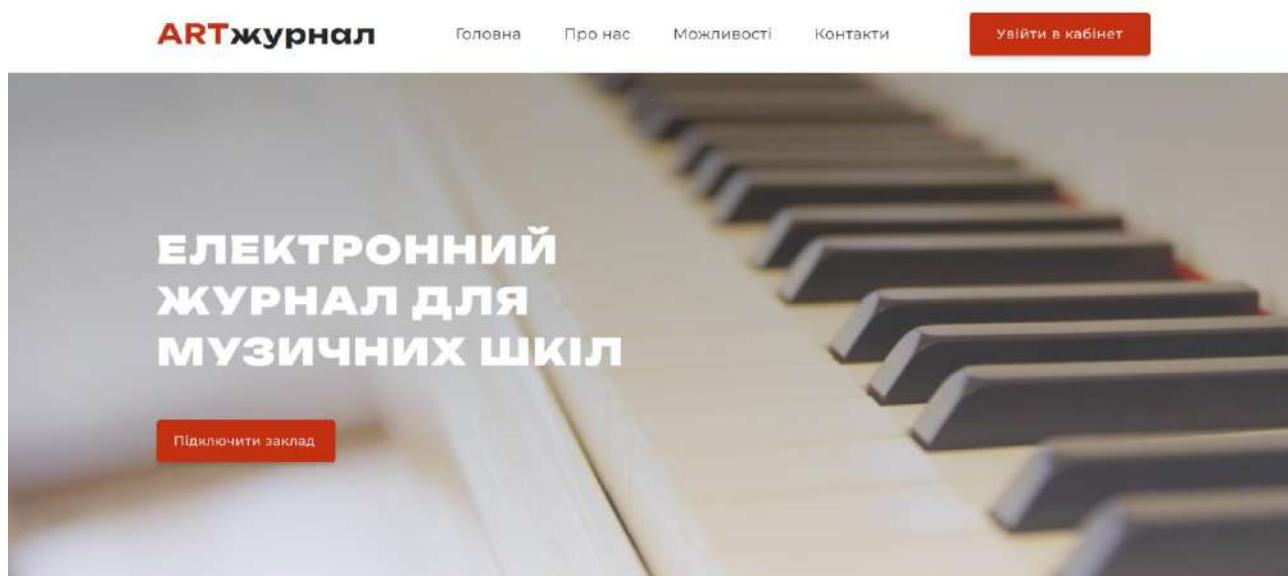


Рисунок 3.12 – Головна сторінка системи

Також вебзастосунок має сторінку «Підключити заклад» (див. рис. 3.13).

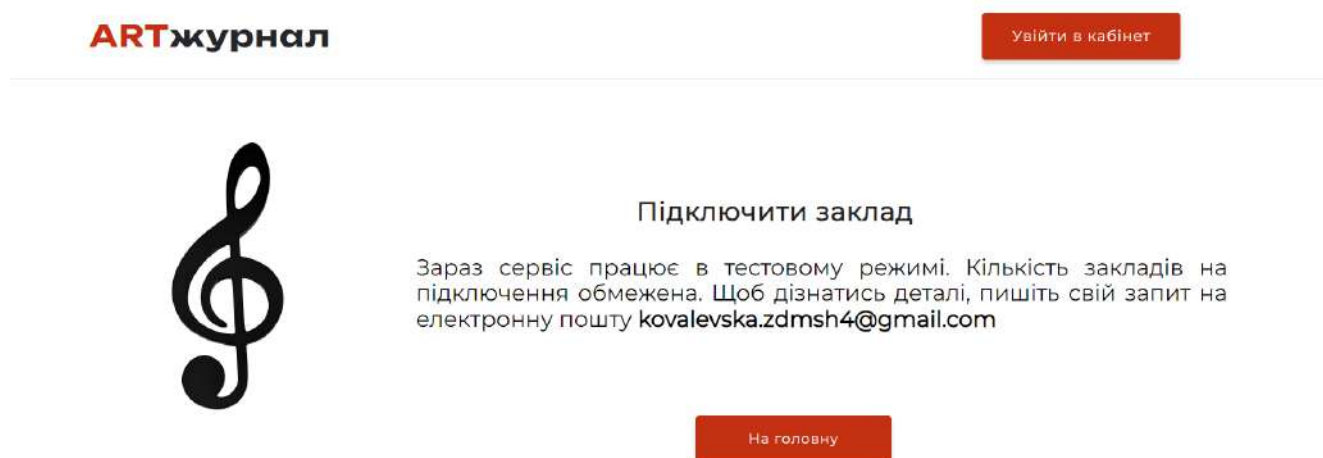


Рисунок 3.13 – Сторінка «Підключити заклад»

У разі виникнення помилки присутня сторінка «Не знайдено» (див. рис. 3.14).

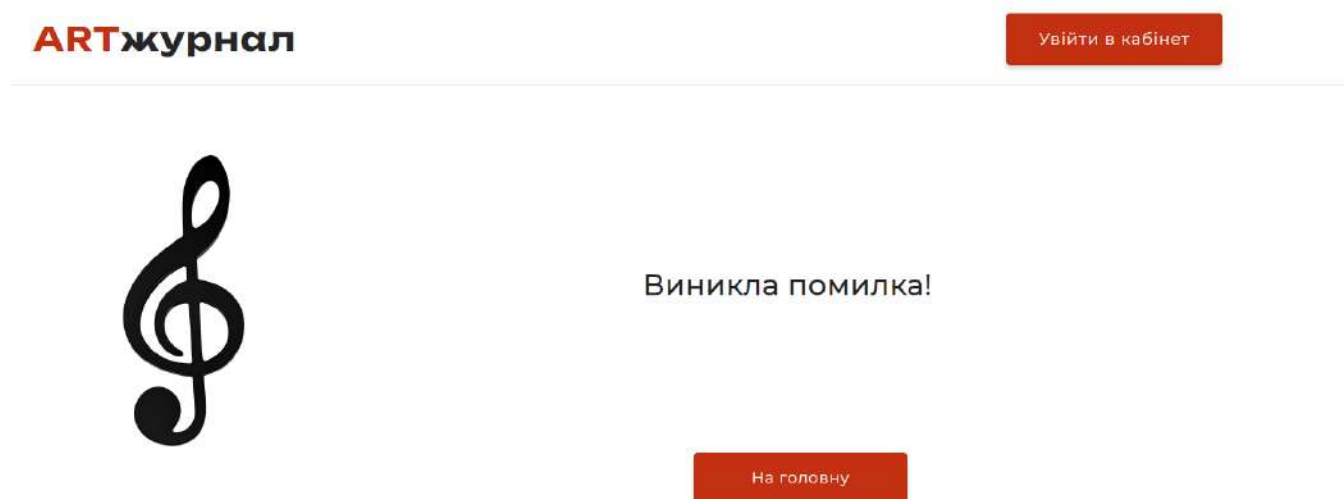


Рисунок 3.14 – Сторінка «Не знайдено»

Сторінка авторизації в системі представлена на рис. 3.15.

Вхід на сайт

E-mail

Пароль

☐ Запам'ятати мене

Увійти до кабінету

Рисунок 3.15 – Сторінка авторизації

Головні сторінки користувачів Адміністратор, Викладач, Учень представлено на рис. 3.16 - 3.18.

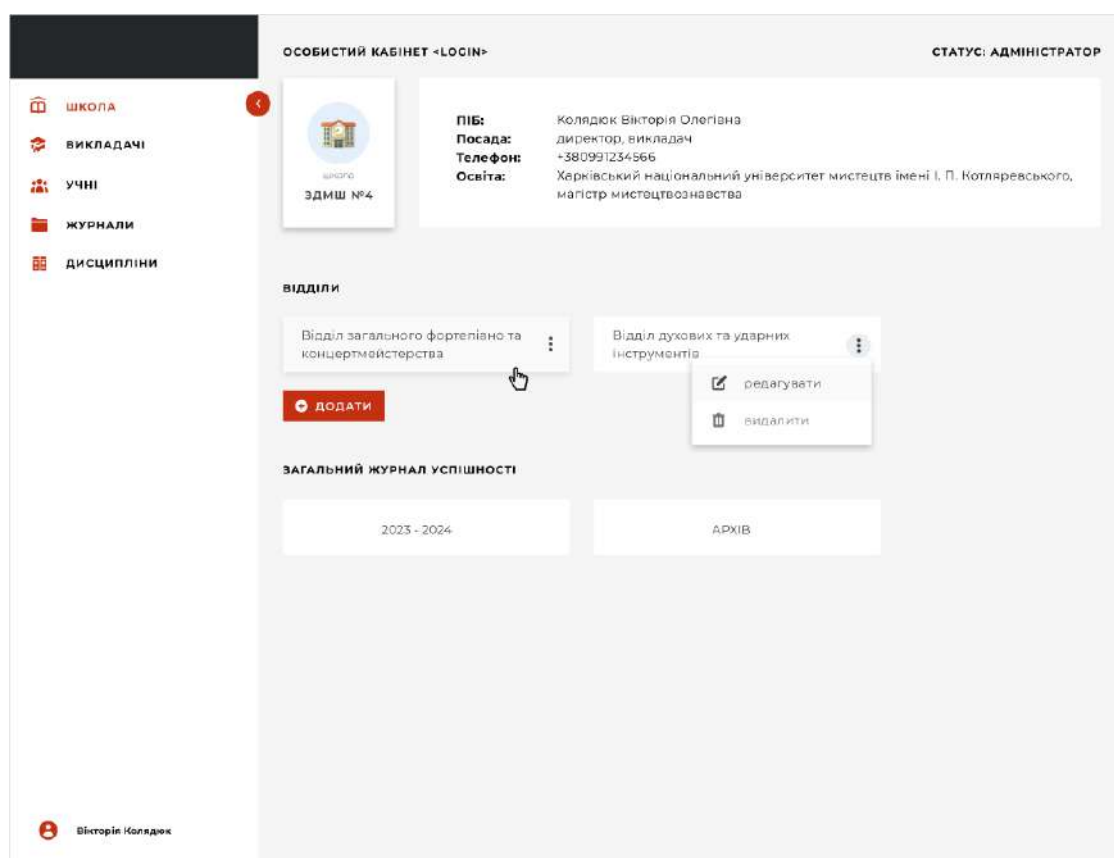


Рисунок 3.16 – Макет головної сторінки адміністратора

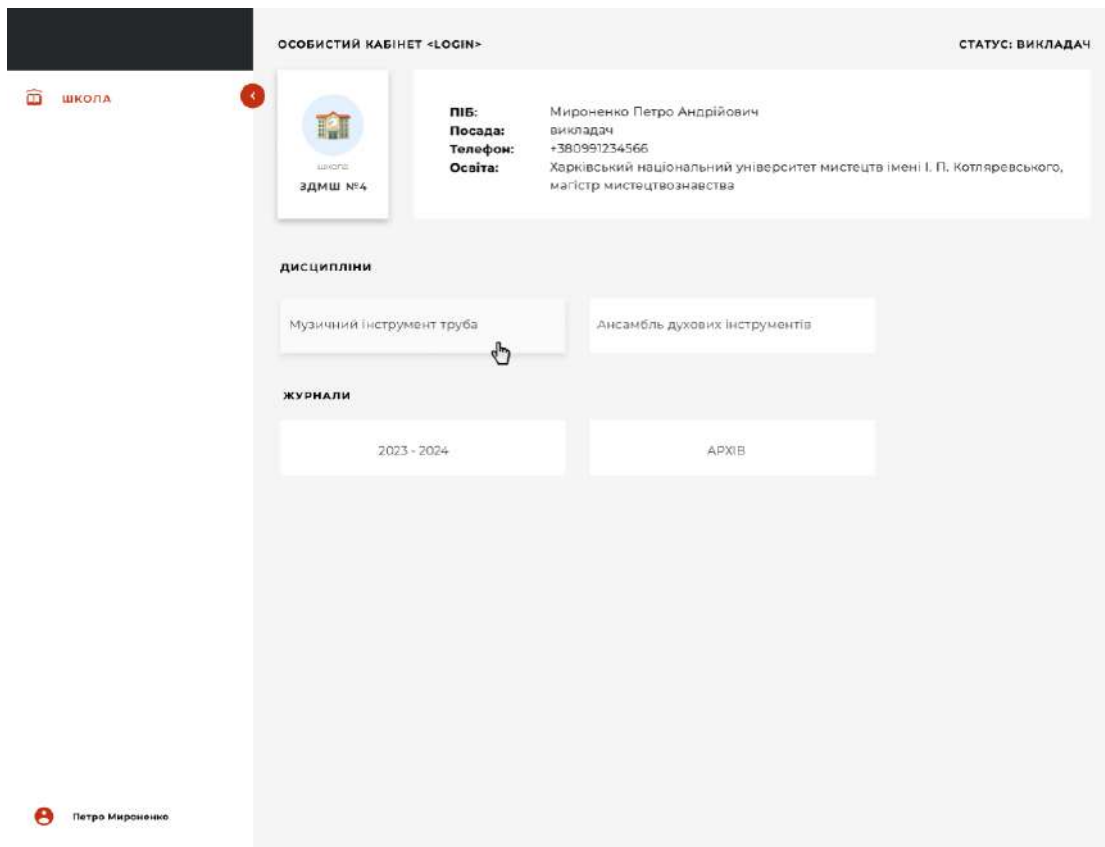


Рисунок 3.17 – Макет головної сторінки викладача

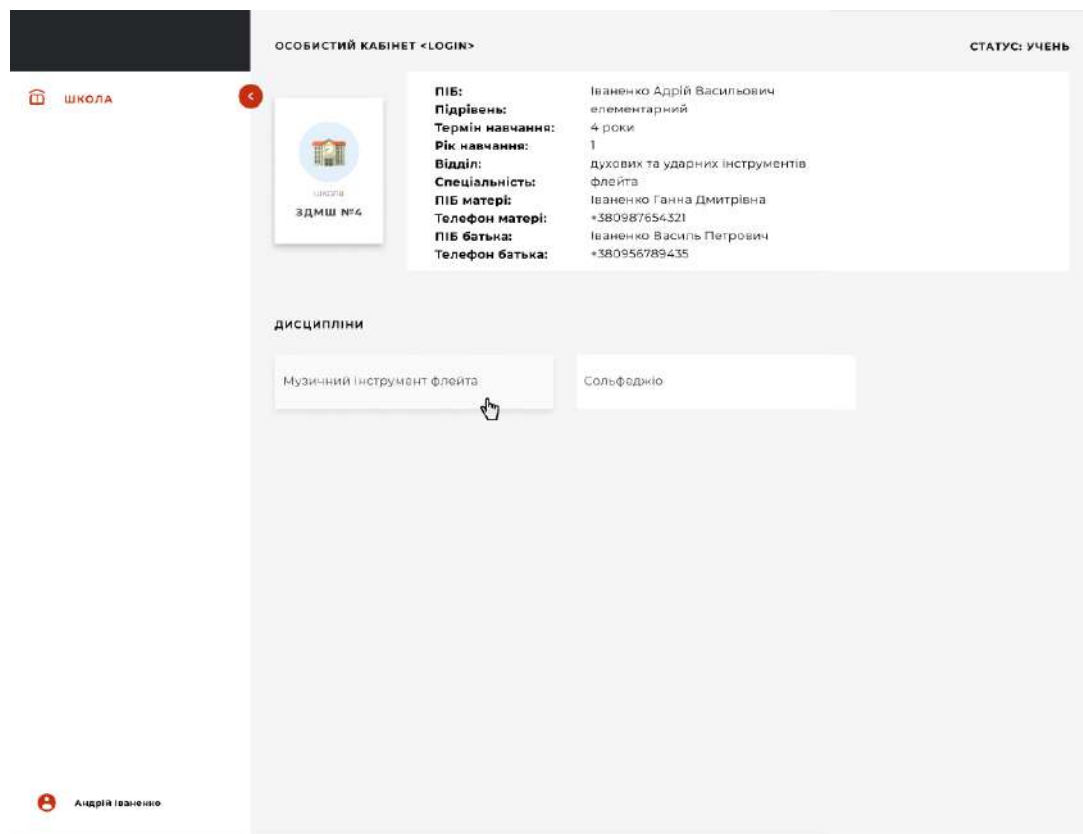


Рисунок 3.18 – Макет головної сторінки учня

На рис. 3.19 представлена форма для реєстрації учня.

Рисунок 3.19 – Макет головної сторінки учня

Інші сторінки інтерфейсу наведено у додатках (див. додаток Г).

3.5 Висновки до розділу

В цьому розділі було розглянуто архітектуру вебзастосунку, а саме структуру сторінок. Було наведено mind-map структури сторінок особистого кабінету користувачів різних рівнів доступу: адміністратор, викладач, учень.

Також було розглянуто як побудовані серверна та клієнтська частини, їх структуру. Спроектовано базу даних, в ній визначено типи значень сутностей, їх роль в системі, наведено в додатках ER-діаграму зв'язків сутностей.

Наприкінці розділу наведено приклад інтерфейсу користувача для взаємодії із системою.

4 АНАЛІЗ РОЗРОБЛЕНОГО ПРОГРАМНОГО ЗАСТОСУНКУ

4.1 Особливості реалізації програмного застосунку

Вебзастосунок обліку успішності учнів музичних шкіл реалізовано за допомогою клієнт-серверної архітектури. Клієнтська частина написана з використанням фреймворку React мови JavaScript. Серверна частина за допомогою фреймворку Express.js середовища Node.js мови JavaScript. База даних реалізована за допомогою СКБД MySql. Код основних файлів представлений в додатках (див. додаток Д). Розроблений вебзастосунок має зручний функціонал, простий та інтуїтивно-зрозумілий інтерфейс.

Згідно з функціональними вимогами, визначеними в першому розділі були розроблені:

- головна сторінка, доступна як авторизованим, так і неавторизованим користувачам, для знайомства з системою;
- сторінка з інформацією про підключення закладу до системи;
- сторінка авторизації, яка на основі введених даних, або дає доступ до системи згідно зі статусом користувача, або не дає доступ до системи;
- особистий кабінет Адміністратора, де є меню, за допомогою якого адміністратор зможе здійснювати навігацію системою;
- особистий кабінет Викладача, де є меню, за допомогою якого викладач зможе здійснювати доступ до своїх журналів та учнів;

- особистий кабінет Учня, де є меню, за допомогою якого учень зможе здійснювати доступ до своїх оцінок;

- інші сторінки, зазначені в третьому розділі.

Згідно з нефункціональними вимогами, було розроблено:

- зручний та зрозумілий користувацький інтерфейс;
- впроваджено в систему підказки, щодо користування нею.

4.2 Тестування програмного застосунку

Були протестовані інтерфейс, функціональні можливості та робота застосунку в різних браузерах. Тестування проводилось вручну. Результати цього тестування відображені в таблиці 4.1.

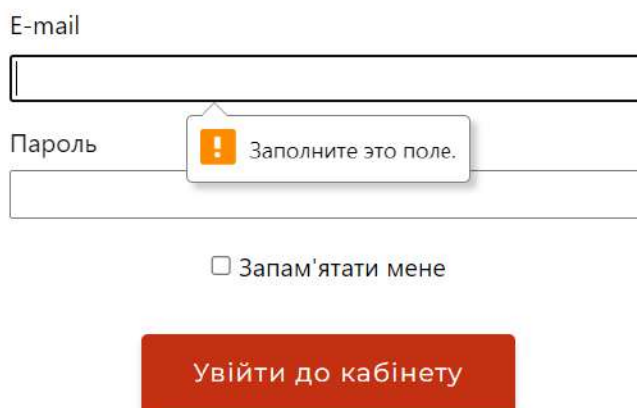
Таблиця 4.1 – Результати тестування розробленого вебзастосунку

Модуль	Функціональні можливості	Результат
Авторизація	Авторизація в системі, надання доступу відповідно до статусу користувача	Авторизація успішна, доступ видається коректно
Сторінки	Відображення даних, модальні вікна	Коректне відображення даних та робота модальних вікон
Перехід між сторінками	Перехід на іншу сторінку з передачею даних	Успішний перехід між сторінками без втрати даних
Запити до сервера	Отримання даних з сервера	Успішне та коректне отримання даних та валідація

Далі розглянемо на прикладах результати тестування.

Застосунок не дозволяє авторизуватись з пустими полями логіна та пароля (див. рис. 4.1).

Вхід на сайт



E-mail

Пароль

 Заполните это поле.

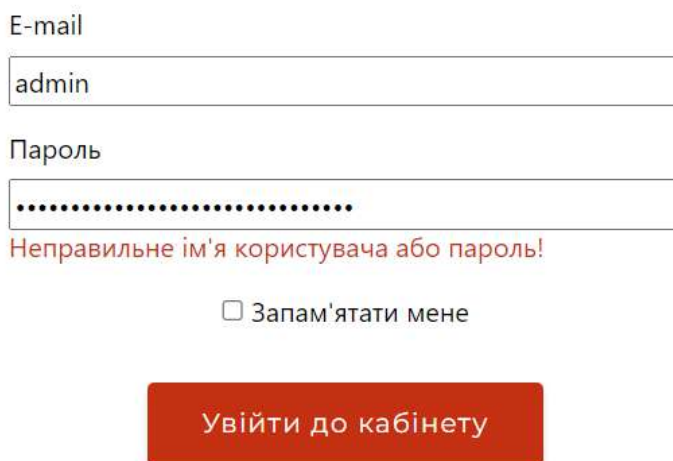
☐ Запам'ятати мене

Увійти до кабінету

Рисунок 4.1 – Валідація пустих полів

Авторизація не відбувається за допомогою невалідних даних (див. рис. 4.2).

Вхід на сайт



E-mail

Пароль

Неправильне ім'я користувача або пароль!

☐ Запам'ятати мене

Увійти до кабінету

Рисунок 4.2 – Авторизація за допомогою невалідних даних

При видаленні відділів, спеціальностей та ін. з'являються модальні вікна з попередженням (див. рис. 4.3).

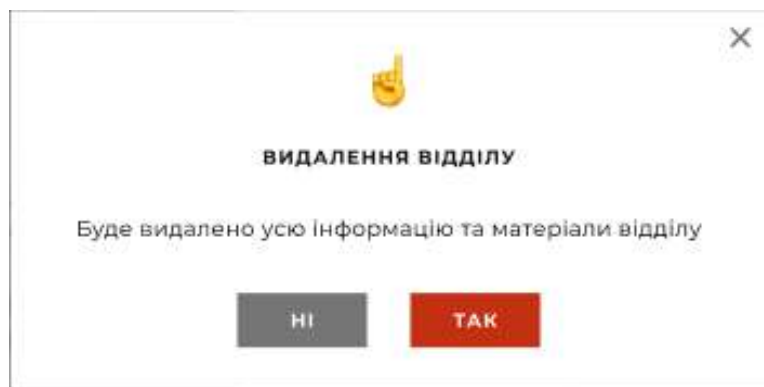


Рисунок 4.3 – Попередження при видаленні об’єкту

При створенні відділів, спеціальностей та інших об’єктів не активується кнопка створення, доки поле не буде заповнене, тобто система не дозволяє створювати порожні об’єкти (див. рис. 4.4).



Рисунок 4.4 – Неактивна кнопка «Додати» для порожнього поля

4.3 Висновки до розділу

В останньому розділі було проаналізовано реалізацію розробленого вебзастосунку, а саме яким чином були виконані функціональні та нефункціональні вимоги.

Також було проведено тестування в ручному режимі, яке мало на меті перевірити функціональність вебзастосунку та підтвердити його працездатність.

ВИСНОВКИ

Дипломний проєкт мав на меті розробку інформаційної системи обліку успішності учнів музичних шкіл. В ході виконання даної роботи було проведено огляд та аналіз існуючих програмних рішень обліку успішності учнів, визначені загальні вимоги до системи.

В ході огляду аналогів на ринку було виявлено, що існуючі рішення підходять для ЗЗСО та коледжів, а в силу іншої специфіки навчання в музичних школах ці аналоги не є ефективними в обліку успішності учнів музичних шкіл. Тобто, фактично, аналогів прямого застосування на ринку немає. Саме це і стало поштовхом для створення цього вебзастосунку.

Перед розробкою вебзастосунку було розглянуто і проаналізовано різні засоби реалізації системи, це мови програмування, фреймворки, СКБД. Враховуючи переваги і недоліки різних засобів було обрано саме ті, що найкраще підходять для втілення проєкту.

В ході реалізації було визначено загальну структуру вебзастосунку, архітектуру серверної та клієнтської частин, розроблено структуру бази даних. Також було створено макети користувацького інтерфейсу вебзастосунку.

Після розробки було проведено аналіз програмного застосування та тестування функціональних можливостей. В результаті аналізу та тестування було виявлено, що функціональні на нефункціональні вимоги до програми було виконано.

Отже, результатом цього дипломного проєкту є виконання його мети, а саме отримання кінцевого працюючого програмного продукту, інформаційної системи обліку успішності учнів музичних шкіл.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Музична педагогіка. *Bikinedia*. URL: https://uk.wikipedia.org/wiki/Музична_педагогіка (дата звернення: 18.04.2024)
2. HTML Introduction. *W3Schools*. URL: https://www.w3schools.com/html/html_intro.asp (дата звернення: 25.04.2024)
3. CSS Introduction. *W3Schools*. URL: https://www.w3schools.com/css/css_intro.asp (дата звернення: 25.04.2024)
4. JavaScript Introduction. *W3Schools*. URL: https://www.w3schools.com/js/js_intro.asp (дата звернення: 25.04.2024)
5. Python. *Python*. URL: <https://www.python.org/> (дата звернення: 25.04.2024)
6. JavaScript. *JavaScript*. URL: <https://www.javascript.com/> (дата звернення: 25.04.2024)
7. The Pros and Cons of Node.js Web App Development: A Detailed Look URL: <https://javascript.plainenglish.io/the-pros-and-cons-of-node-jsweb-app-development-a-detailed-look-c91a22f013c> (дата звернення: 25.04.2024)
8. Advantages and Disadvantages of PHP. *Medium*. URL: <https://www.geeksforgeeks.org/advantages-and-disadvantages-of-php/> (дата звернення: 30.04.2024)
9. Go. *Go.dev*. URL: <https://go.dev/> (дата звернення: 30.04.2024)
10. Express.js. *Express*. URL: <https://expressjs.com/uk/> (дата звернення: 30.04.2024)
11. Найкращі фреймворки Node.js: розбираємо їх особливості. *Wezom*. URL: <https://wezom.com.ua/ua/blog/luchshie-freymvorki-nodejs> (дата звернення: 30.04.2024)
12. Стаття про порівняння JavaScript фреймворків Vue.js, React і Angular (2019). *Mkdev*. URL: <https://mkdev.me/ru/posts/sravnenie-javascript-freymvorkov-vue-js-reacti-angular-2019> (дата звернення: 30.04.2024)

13. What is PostgreSQL? Introduction, Advantages & Disadvantages. *Guru99*. URL: <https://www.guru99.com/introduction-postgresql.html> (дата звернення: 05.05.2024)

14. What is MongoDB? Simple Pros and Cons. *Aloa*. URL: <https://aloe.co/blog/what-is-mongodb-simple-pros-and-cons> (дата звернення: 05.05.2024)

15. Битва титанів: що краще — PostgreSQL чи MySQL? *Highload*. URL: <https://highload.today/uk/postgresql-vs-mysql/> (дата звернення: 05.05.2024)

16. Best Express js Architecture Folder for biggest project| Best Praticce. *Medium*. URL: <https://medium.com/@ramsesndame34/best-express-js-architecture-folder-for-biggest-project-best-praticce-8131401b1c80> (дата звернення: 15.05.2024)

17. React Architecture Pattern and Best Practices in 2024. *GeeksforGeeks*. URL: <https://www.geeksforgeeks.org/react-architecture-pattern-and-best-practices/> (дата звернення: 15.05.2024)

18. Figma. *Вікіпедія*. URL: <https://uk.wikipedia.org/wiki/Figma> (Дата звернення: 20.05.2024)

19. Коцовський В. М. Крос-платформне програмування. Лекції для студентів ЗФН. *Електронний репозитарій ДВНЗ "УжНУ"*. URL: <https://dspace.uzhnu.edu.ua/jspui/handle/lib/21365> (дата звернення: 20.05.2024)

ДОДАТОК А Специфікація вимог SRS

1 ВВЕДЕННЯ

Даний документ проєктується для опису вебзастосунку «Електронний журнал для ДМШ», а також для опису системних, функціональних та нефункціональних вимог до продукту.

1.1 Призначення

Призначення даного документу:

- проєктування для узагальнення знань бакалаврського циклу освіти спеціальності «Комп’ютерна інженерія»;
- фіксація всіх вимог до продукту;
- опис поведінки майбутнього продукту.

1.2 Область дії

Назва програмного продукту – «Електронний журнал для ДМШ». Ведення обліку успішності учнів ДМШ. Продукт буде корисним ДМШ та музичним відділенням мистецьких шкіл.

1.3 Визначення, акроніми та скорочення

- ДМШ – дитяча музична школа;
- СКБД – система керування базами даних;
- Багатоплатформність – властивість програмного забезпечення працювати більш ніж на одній програмній (в тому числі — операційній системі) або апаратній платформі [19].

1.4 Публікації

Не містить.

1.5 Короткий огляд

Цей документ містить загальний опис програмного продукту, контекст його використання, описані функції програмного продукту, характеристики користувачів, опис вимог: функціональних, нефункціональних та ін.

2 ЗАГАЛЬНИЙ ОПИС

Призначення продукту – автоматизація процесу ведення обліку успішності учнів ДМШ, спрощення ведення документації.

Програмний продукт не прив'язується до конкретної операційної системи. Вимагається доступ до мережі Інтернет.

2.1 Контекст програмного продукту

Програмний продукт є незалежним і повністю автономним. На різних пристроях використовується адаптивна версія застосунку.

2.2 Функції програмного продукту

Даний вебзастосунок має три ролі: Адміністратор (директор, завуч), Викладач, Учень. Кожна роль має набір функціональних можливостей:

а) Адміністратор:

1) має можливість переглядати, додавати, редагувати та видаляти відділи школи в системі;

2) має можливість переглядати, додавати, редагувати та видаляти спеціальності на відділах;

3) має можливість переглядати, додавати, редагувати та видаляти навчальні дисципліни;

4) має можливість додавати, переглядати, редагувати та архівувати дані учнів в системі;

5) має можливість додавати, переглядати, редагувати та архівувати дані викладачів у системі;

6) має можливість призначати та видаляти дисципліни викладачам;

7) має можливість призначати та видаляти класи та учнів викладачам;

8) має можливість переглядати журнали викладачів та успішність учнів;

9) має можливість архівувати журнали викладачів та створювати їм нові журнали;

10) має можливість створювати загальний журнал успішності учнів по відділах та архівувати загальний журнал учнів по відділах за минулий навчальний рік;

11) має можливість автоматично переносити оцінки за контрольний захід та підсумкові із журналу викладача у загальний журнал учнів по відділах;

б) Викладач:

- 1) має можливість переглядати дані учнів в системі;
- 2) має можливість переглядати персональні дані в системі;
- 3) має можливість вводити та редагувати успішність учнів, у яких викладає;

в) Учень:

- 1) має можливість переглядати персональні дані у системі;
- 2) має можливість переглядати свої поточні і підсумкові оцінки з різних предметів;
- 3) має можливість переглядати ПІБ та контактну інформацію своїх викладачів.

г) Адміністратор, Викладач, Учень:

- 1) має можливість авторизуватись в системі;
- 2) має можливість вийти із системи.

2.3 Характеристики користувача

Користувачі: адміністратор (директор/завуч), викладач, учень. Усім користувачам потрібно на базовому рівні володіти навичками роботи з комп'ютером.

2.4 Обмеження

Немає.

2.5 Допущення і залежності

Немає.

2.6 Розподілення вимог

Система являє собою веб-орієнтований застосунок з архітектурою «клієнт-сервер», використовує СКБД. Продукт багатоплатформний, тому не прив'язується до конкретної операційної системи. Вимагається доступ до мережі Інтернет.

3 СПЕЦИФІЧНІ ВИМОГИ

3.1 Функціональні вимоги

1. Вебзастосунок повинен забезпечувати роботу трьом категоріям користувачів: адміністратори (директор, завуч), редактори (викладачі), користувачі (учні).

1.1 Ідентифікація користувачів в залежності з категорією повинна виконуватись за допомогою логіна і пароля, які вводяться на стартовій сторінці вебзастосунку.

1.2 При коректному вводі логіна і пароля повинен здійснюватися перехід на робочу сторінку користувача, редактора чи адміністратора відповідно до введених даних.

1.2.1 При невірному вводі логіна та/чи пароля з'являється повідомлення з проханням перевірити правильність вводу логіна і пароля.

2. Адміністратор має змогу вводити, переглядати, редагувати та видаляти дані учасників освітнього процесу (учнів, викладачів).

3. Викладач має змогу вводити, переглядати та редагувати дані про відвідуваність та оцінювання учнів.

3.1 Викладач має змогу переглядати персональні дані своїх учнів.

4. Учень має змогу переглядати свої персональні дані та успішність за всіма предметами.

3.2 Нефункціональні вимоги

Зручність використання:

Система повинна бути простою в експлуатації для усіх користувачів. Після однієї години навчання роботі з системою користувач повинен бути ознайомлений з усіма функціями системи.

Організаційні вимоги:

– проектування та розробка системи здійснюються відповідно до розроблених специфікацій;

– процес розробки здійснюється з урахуванням усіх частин життєвого циклу розробки.

Інтерфейсні вимоги:

Вебзастосунок повинен мати простий та інтуїтивно-зрозумілий інтерфейс.

Сторінки:

- стартова сторінка авторизації;
- робоча сторінка користувача;
- інші сторінки, якщо такі призначені в кабінеті користувача.

ДОДАТОК Б Структура сторінок особистого кабінету адміністратора

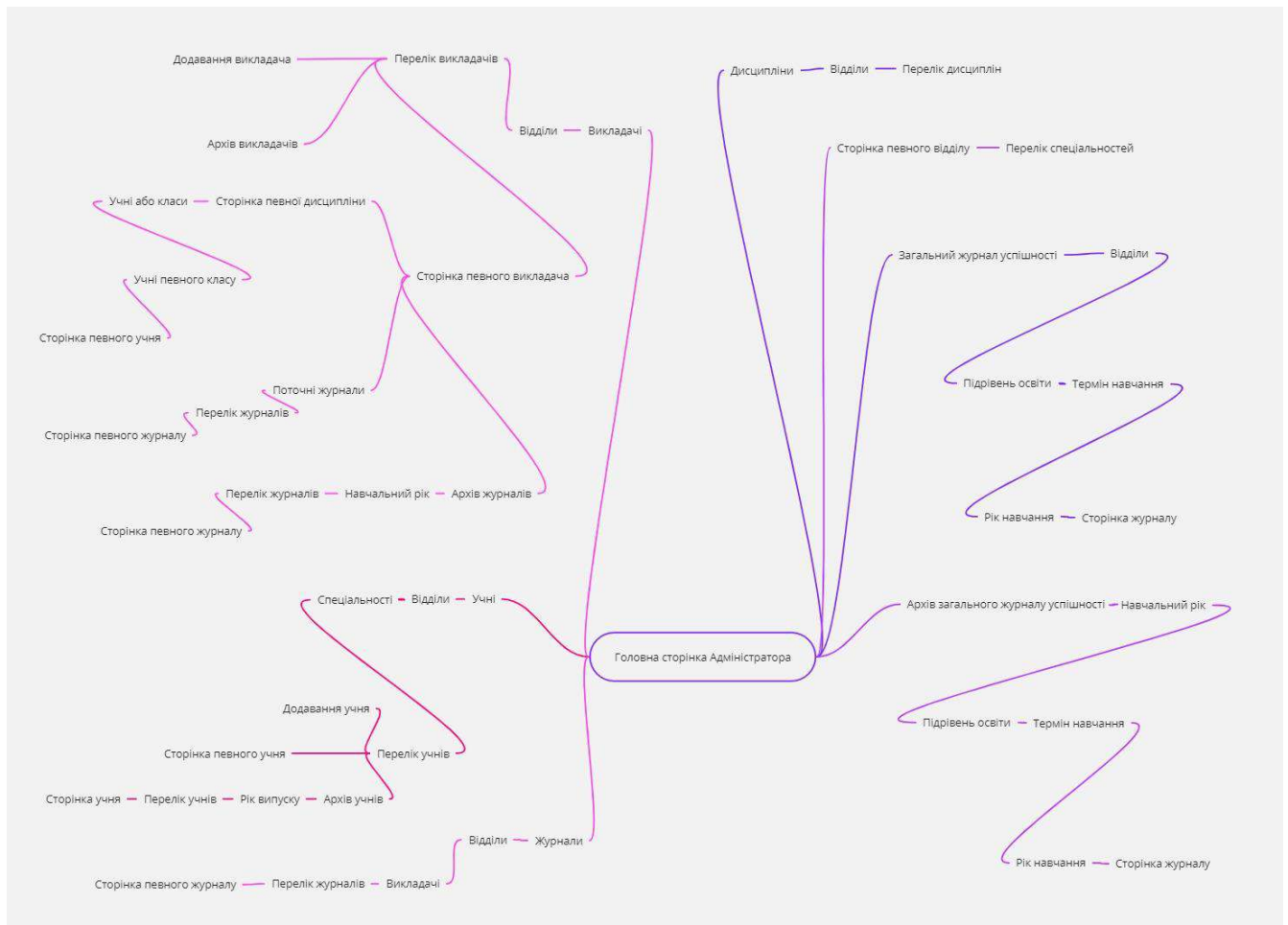


Рисунок Б.1 – Mind-Мар структури сторінок особистого кабінету Адміністратора

ДОДАТОК В ER-діаграма бази даних

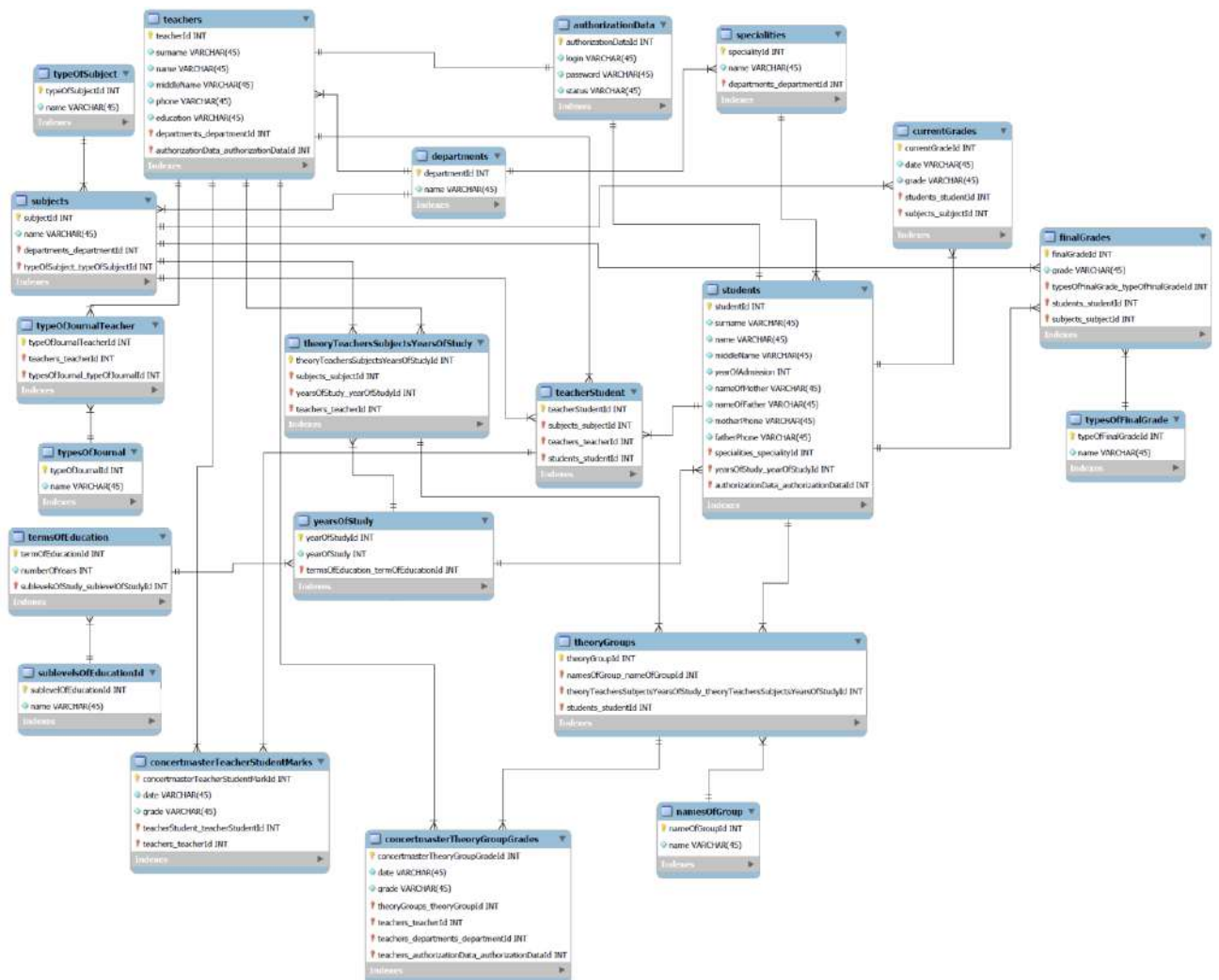


Рисунок В.1 – ER-діаграма бази даних

ДОДАТОК Г Макети сторінок вебзастосунку

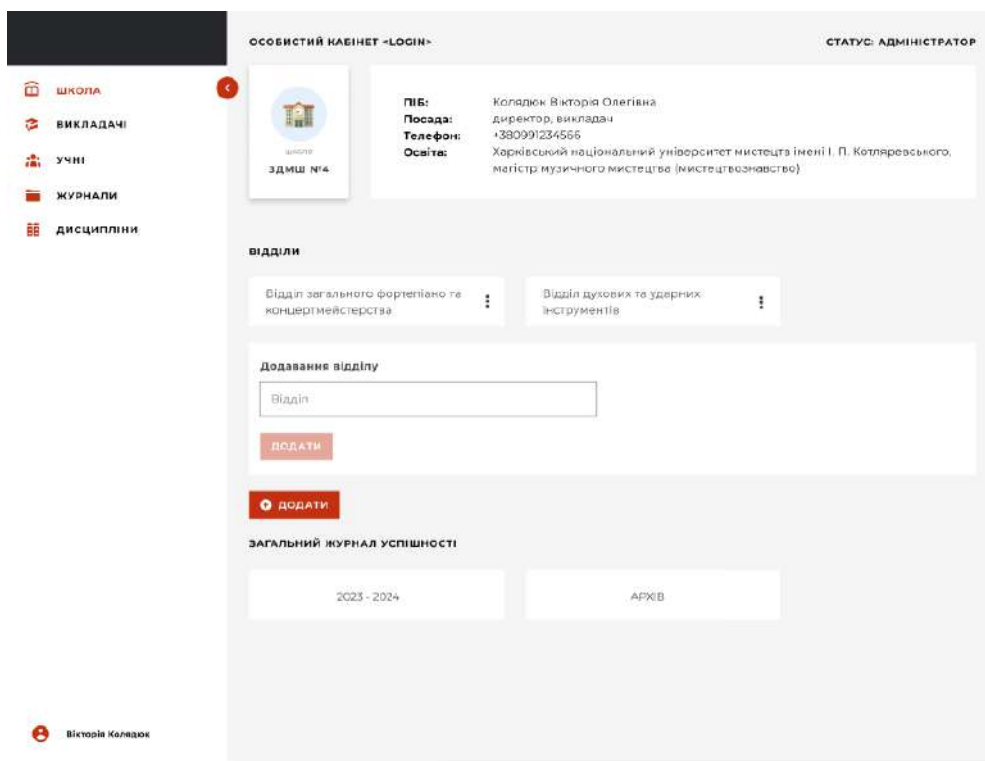


Рисунок Г.1 – Функція додавання відділу на головній сторінці Адміністратора

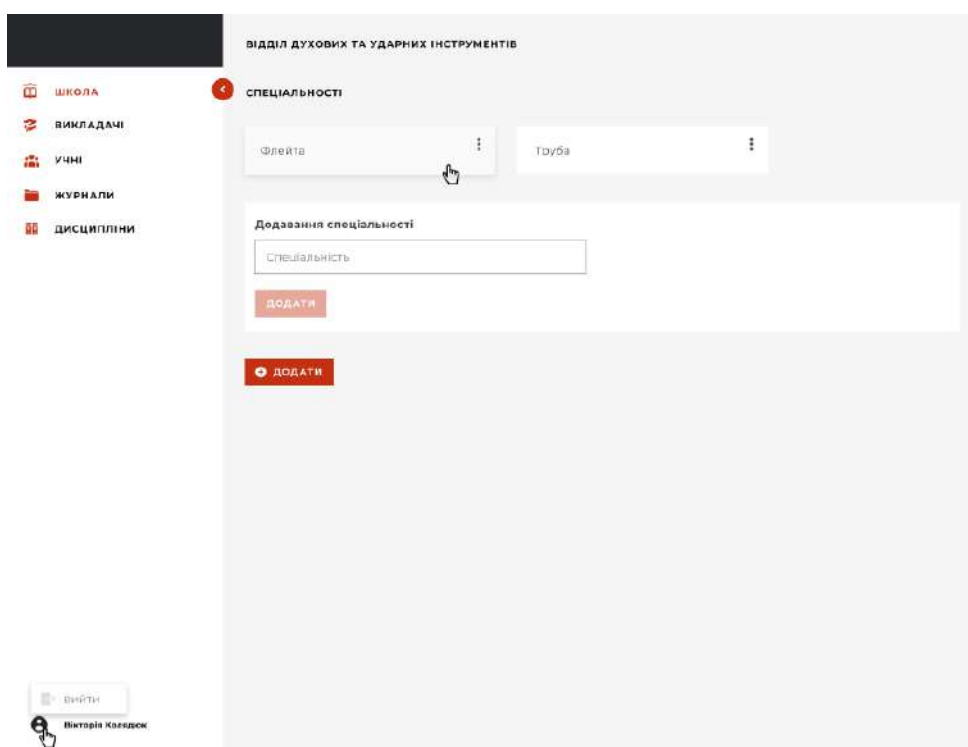


Рисунок Г.2 – Додавання спеціальності на сторінці відділу

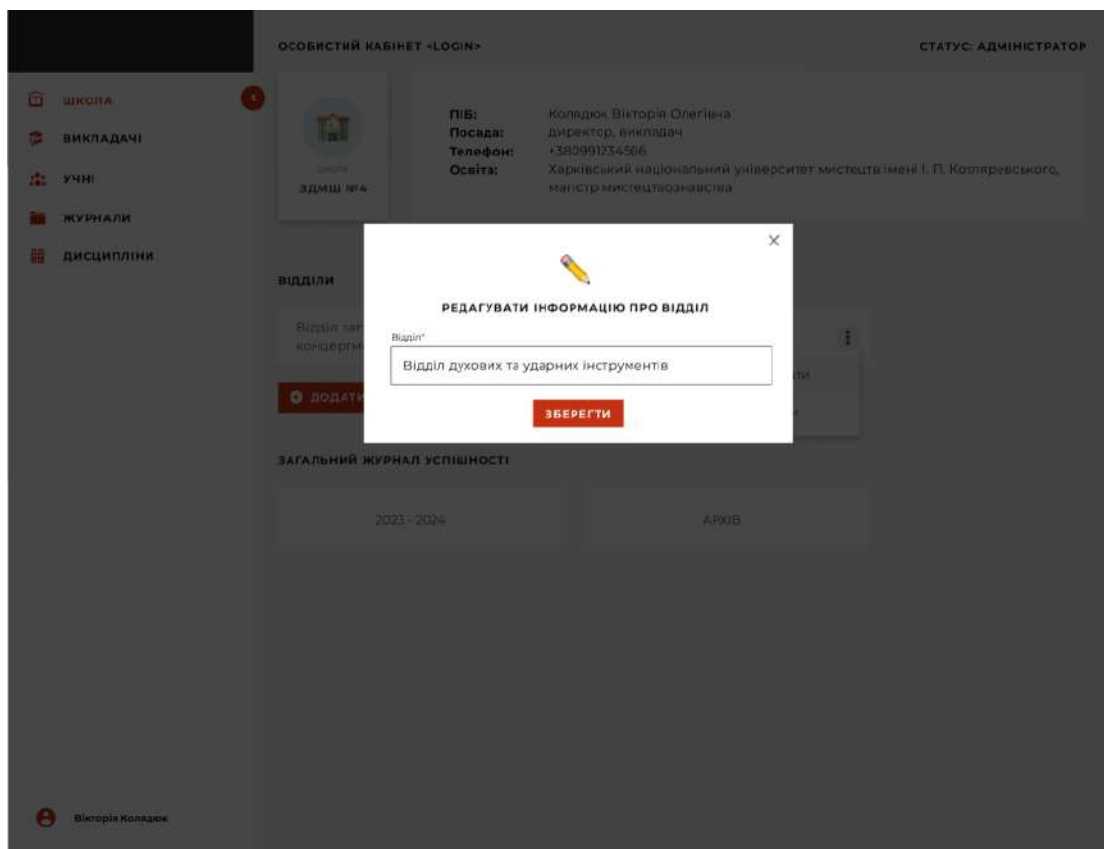


Рисунок Г.3 – Редагування назви відділу

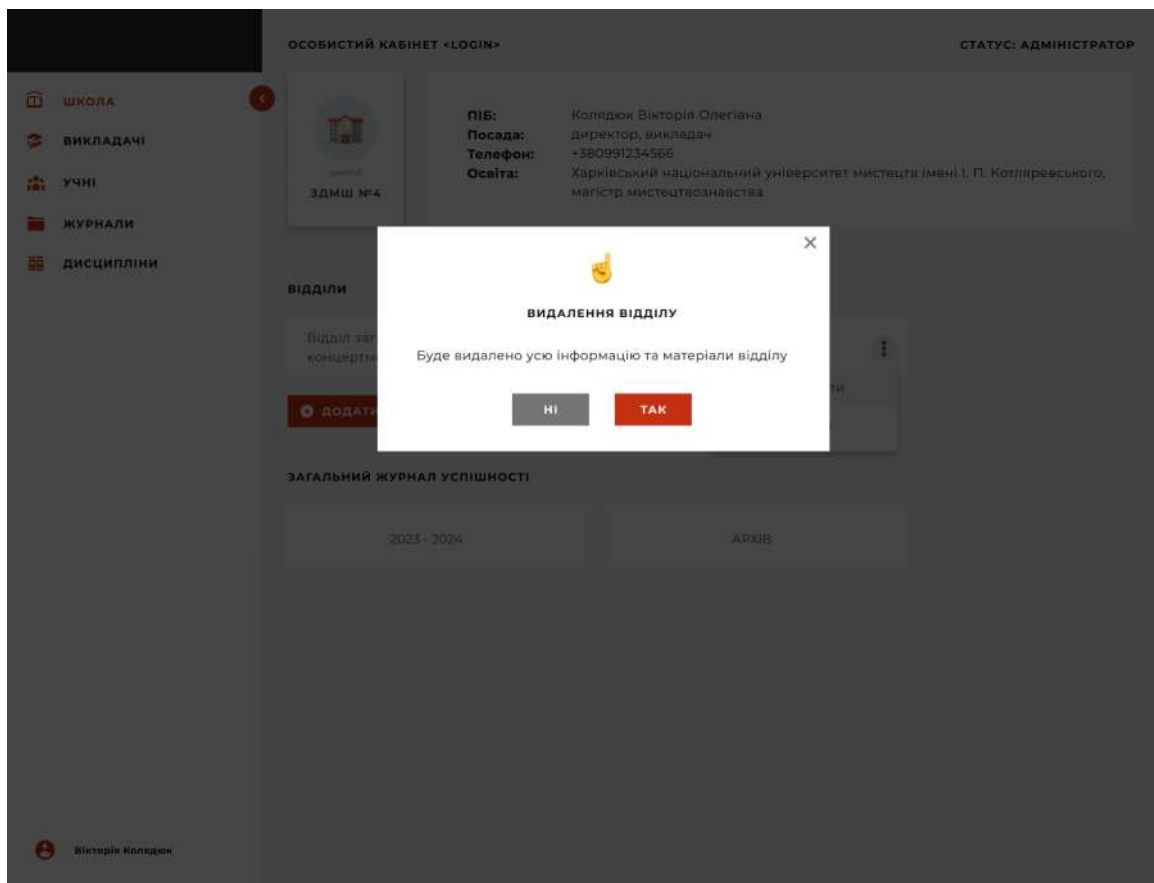


Рисунок Г.4 – Видалення відділу

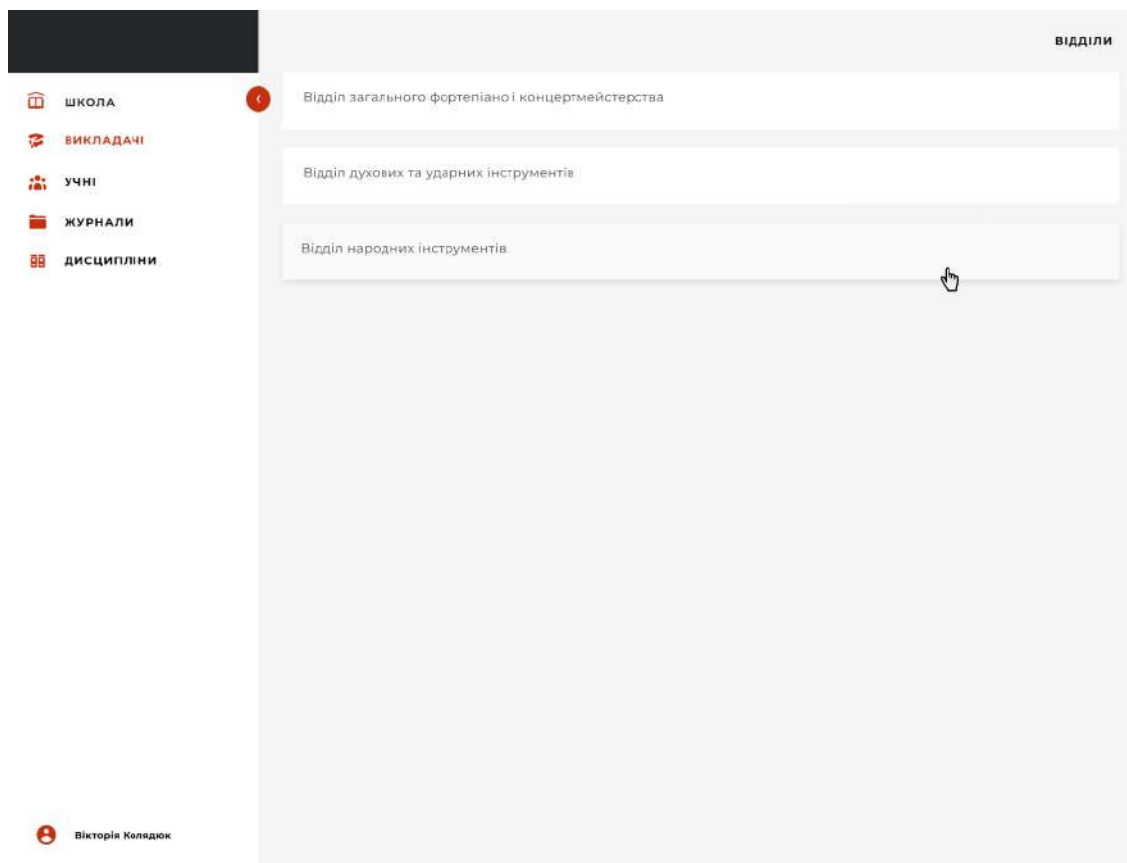


Рисунок Г.5 – Сторінка вибору відділу у вкладці Викладачі

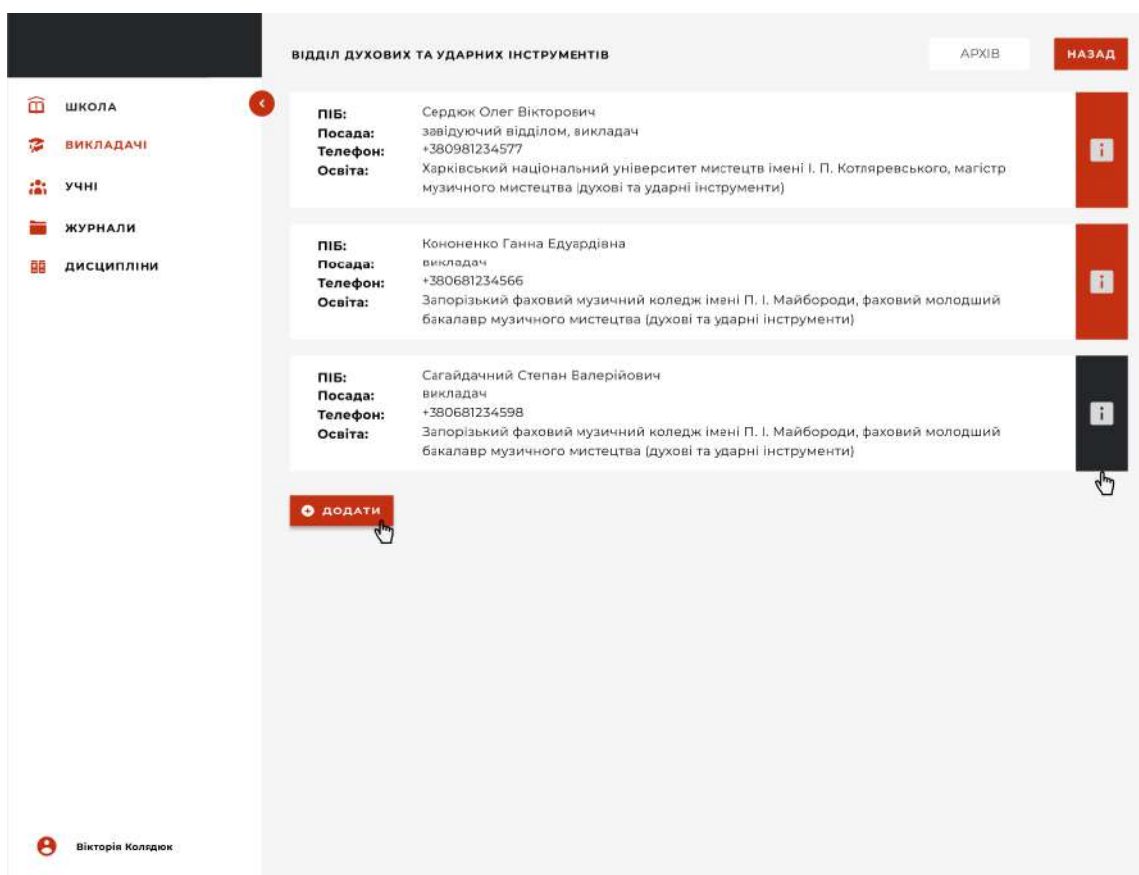


Рисунок Г.6 – Список викладачів

ШКОЛА

ВИКЛАДАЧІ

УЧНІ

ЖУРНАЛИ

ДИСЦИПЛІНИ

ВІДДІЛ ДУХОВИХ ТА УДАРНИХ ІНСТРУМЕНТІВ

НАЗАД

Додавання викладача

Прізвище*

Ім'я*

По батькові*

Телефон*

Email*

Освіта*

ДОДАТИ

Вікторія Коледж

Рисунок Г.7 – Сторінка реєстрації викладача

ШКОЛА

ВИКЛАДАЧІ

УЧНІ

ЖУРНАЛИ

ДИСЦИПЛІНИ

ВІДДІЛ ДУХОВИХ ТА УДАРНИХ ІНСТРУМЕНТІВ

НАЗАД

ПІБ:

Посада:

Телефон:

Освіта:

Сердюк Олег Вікторович

завідуючий відділом, викладач

+380981234577

Харківський національний університет мистецтв імені І. П. Котляревського

музичного мистецтва (духові та ударні інструменти)

редагувати

ДИСЦИПЛІНИ

Музичний інструмент труба

Ансамбль духових інструментів

призначити

видалити

ЖУРНАЛИ

2023 - 2024

АРХІВ

АРХІВУВАТИ ВИКЛАДАЧА

Рисунок Г.8 – Сторінка інформації про викладача

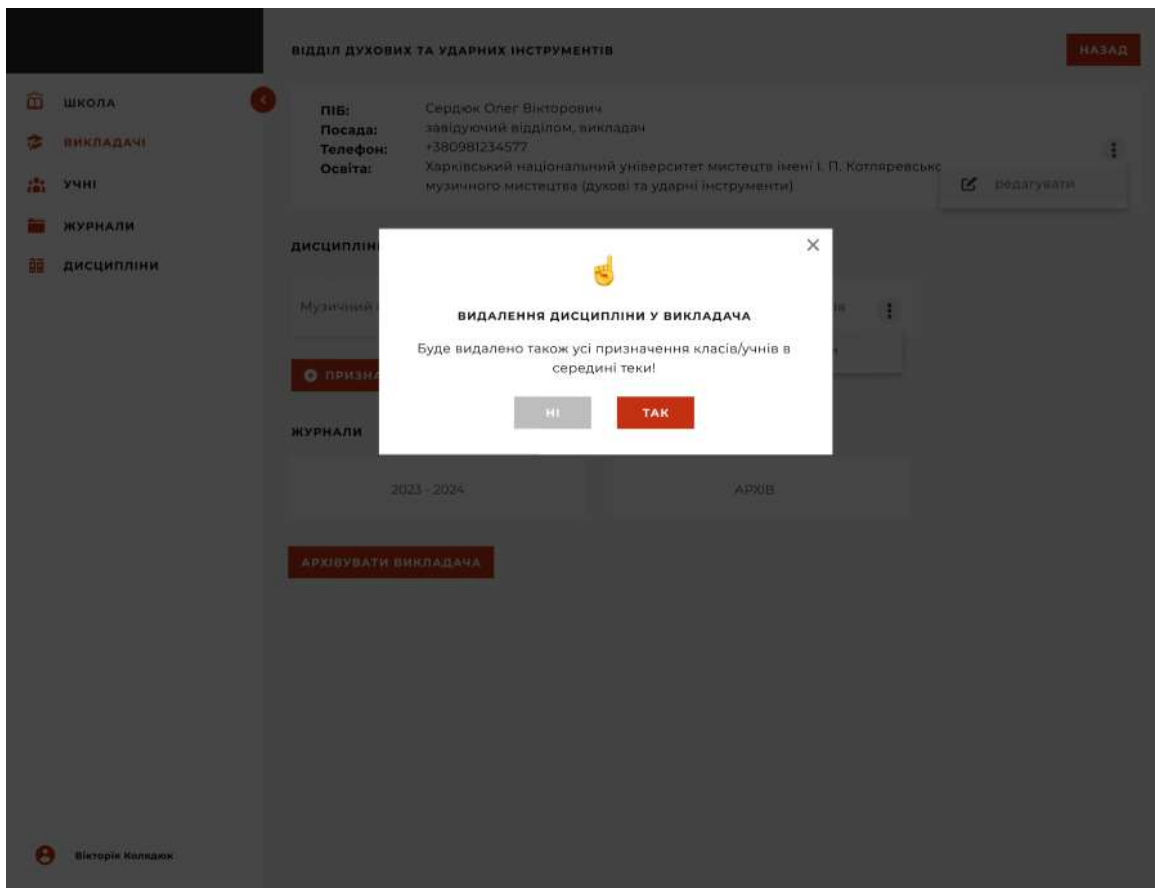


Рисунок Г.9 – Видалення дисципліни у викладача

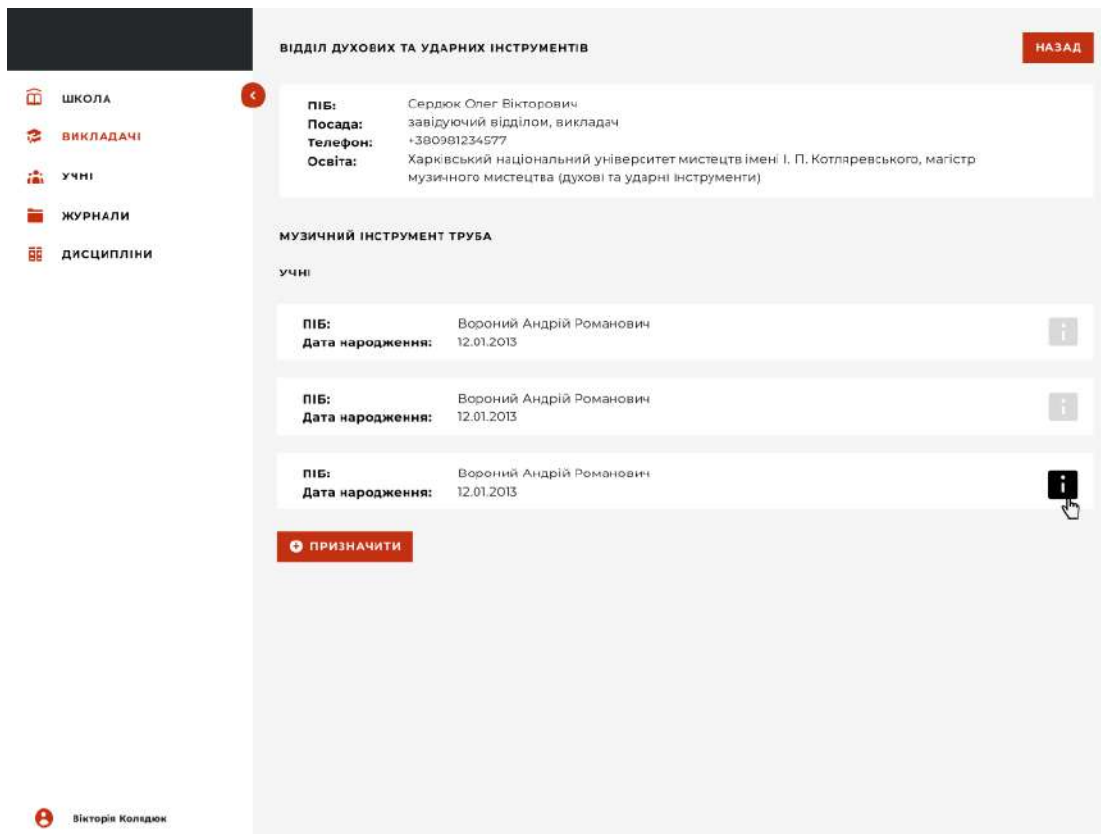


Рисунок Г.10 – Сторінка учнів з певної дисципліни викладача

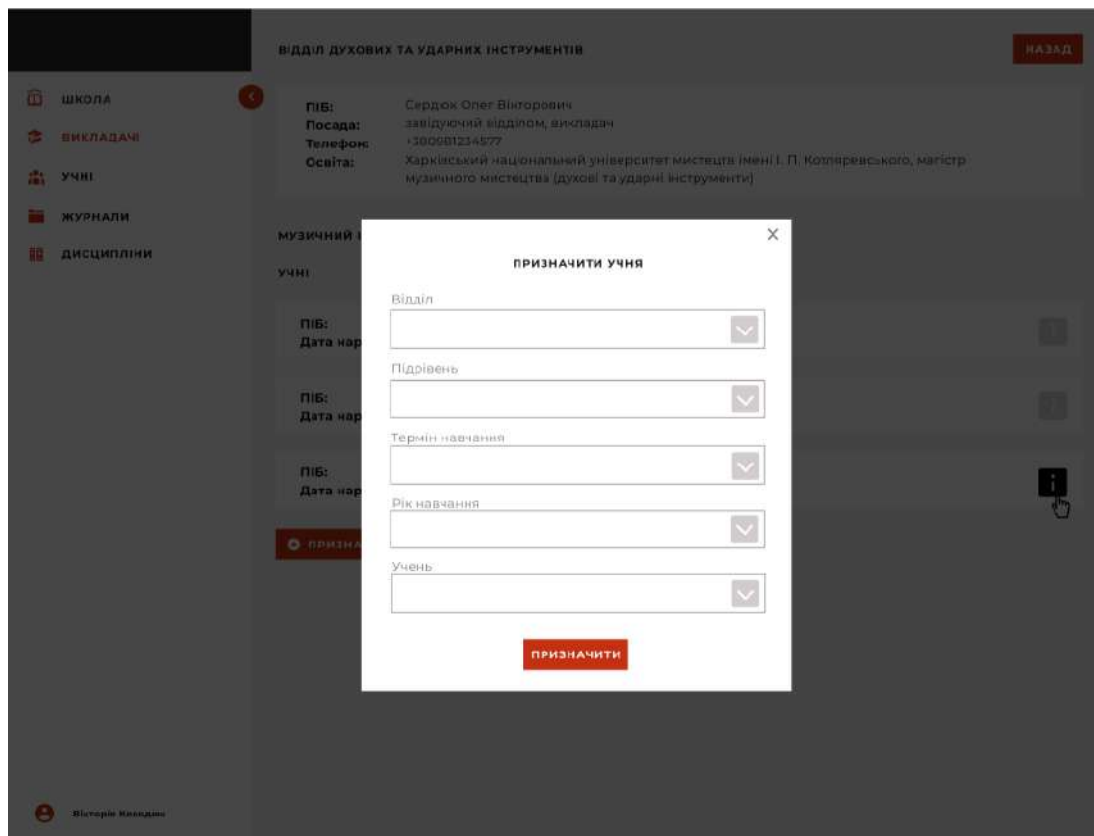


Рисунок Г.11 – Призначення учня викладачу з дисципліни

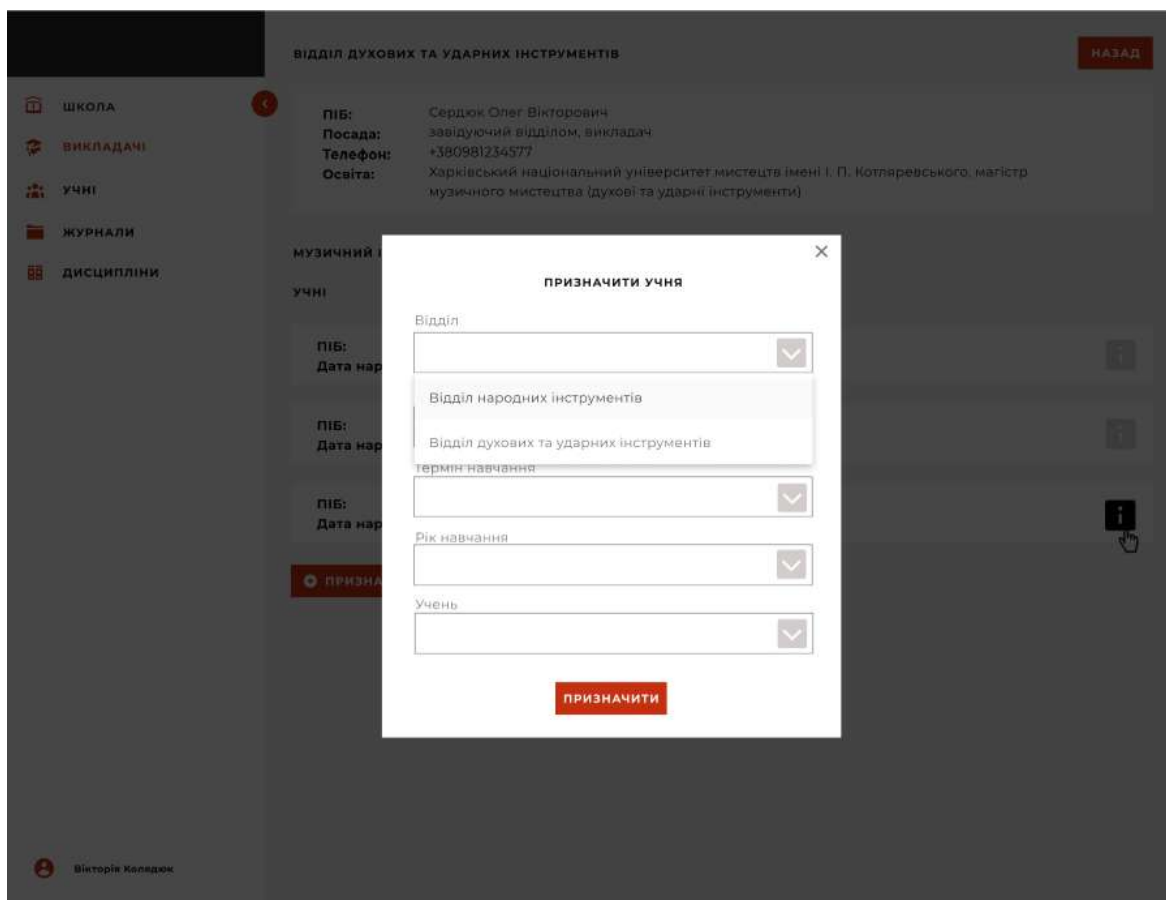


Рисунок Г.12 – Випадний список у призначенні учня

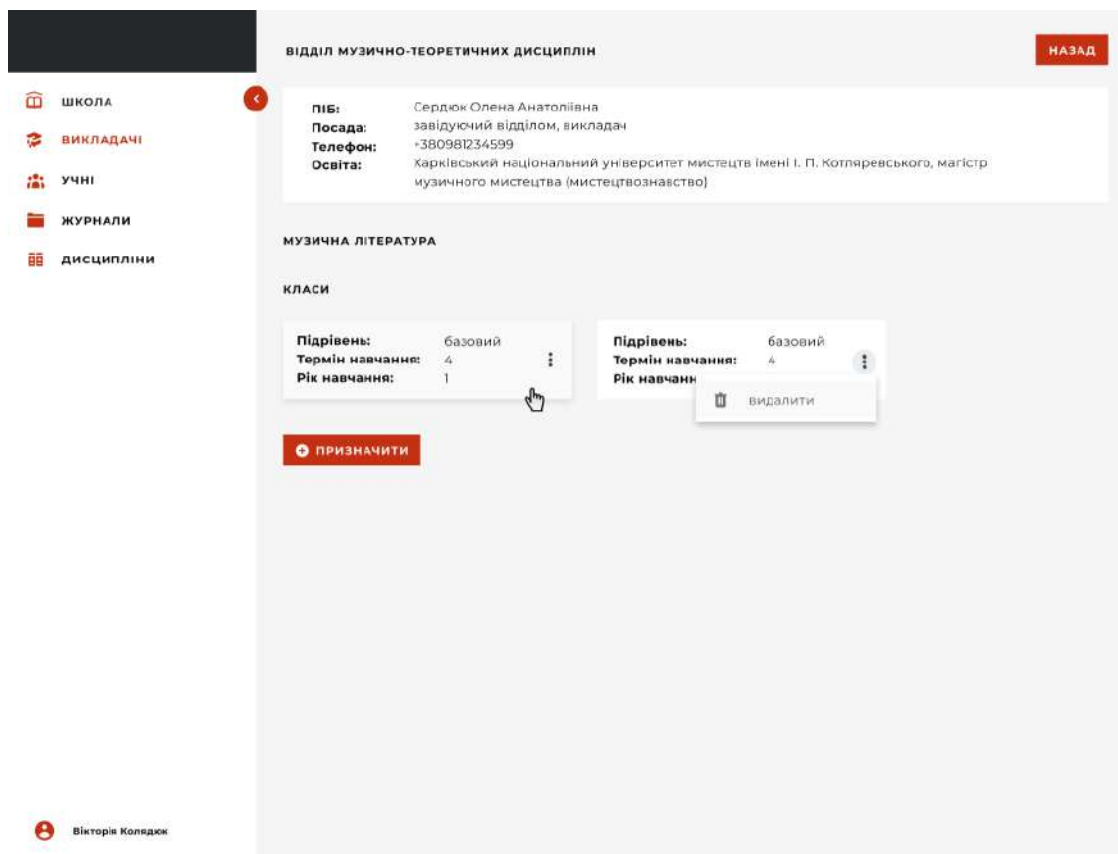


Рисунок Г.13 – Список класів з музично-теоретичної дисципліни викладача

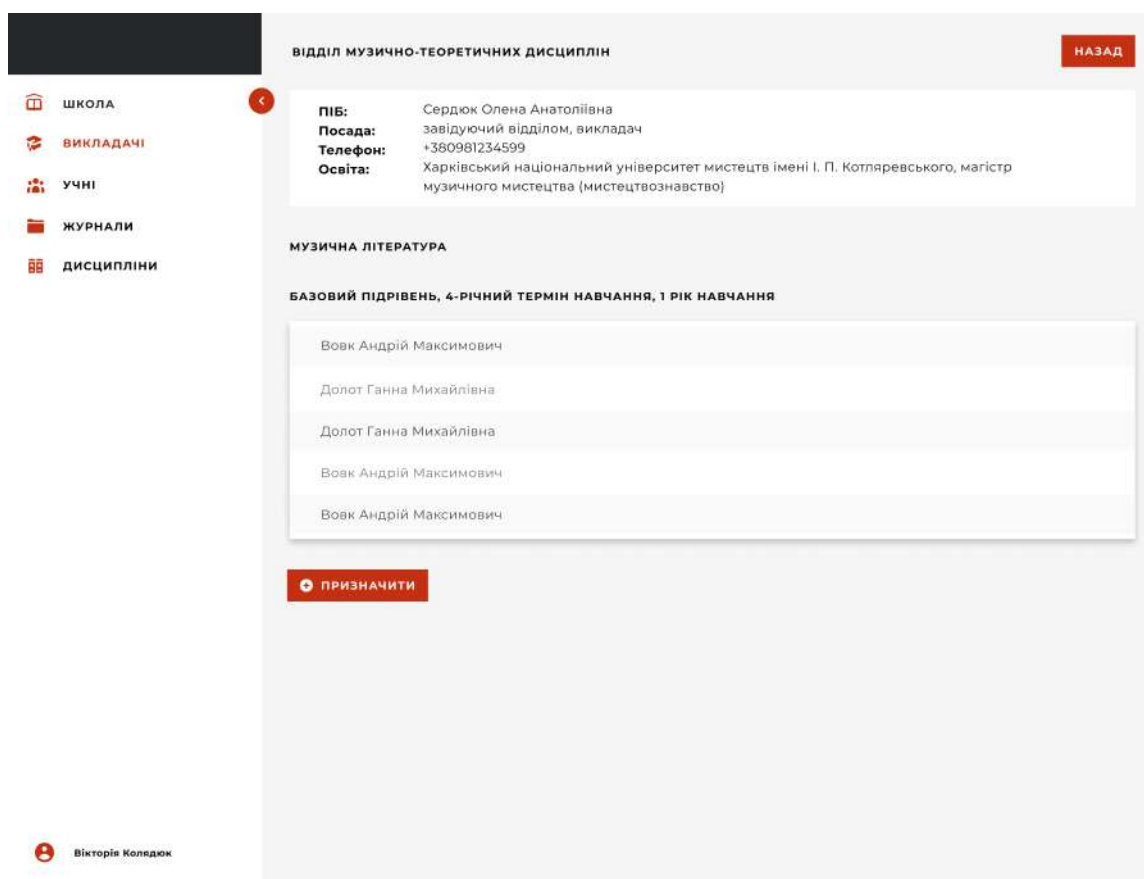


Рисунок Г.14 – Сторінка учнів класу з музично-теоретичної дисципліни викладача

ВІДДІЛ МУЗИЧНО-ТЕОРЕТИЧНИХ ДИСЦИПЛІН НАЗАД

ПІБ: Сердюк Олена Анатоліївна
 Посада: завідувачий відділом, викладач
 Телефон: +3809812345999
 Освіта: Харківський національний університет мистецтв імені Л. П. Котляревського, магістр музичного мистецтва (мистецтвознавство)

МУЗИЧНА ЛІТЕРАТУРА

БАЗОВИЙ ПІДРІВЕНЬ, 4-РІЧНИЙ ТЕРМІН НАВЧАННЯ, 1 РІК НАВЧАННЯ

Вовк Андрій Максимович	☒
Долот Ганна Михайлівна	☒
Долот Ганна Михайлівна	☒
Вовк Андрій Максимович	☒
Вовк Андрій Максимович	☒
Долот Ганна Михайлівна	☒
Вовк Андрій Максимович	☐
Долот Ганна Михайлівна	☒

ПРИЗНАЧИТИ

Вікторія Колядок

Рисунок Г.15 – Призначення учнів з музично-теоретичної дисципліни (чорні галочки – призначені учні, біле поле -непризначені, сіра галочка – призначені учні іншому викладачу)

ВІДДІЛИ НАЗАД

Відділ загального фортепіано і концертмейстерства

Відділ духових та ударних інструментів

Відділ народних інструментів

ВІДІЛИТИ

Вікторія Колядок

Рисунок Г.16 – Сторінка вибору відділу у вкладці Дисципліни

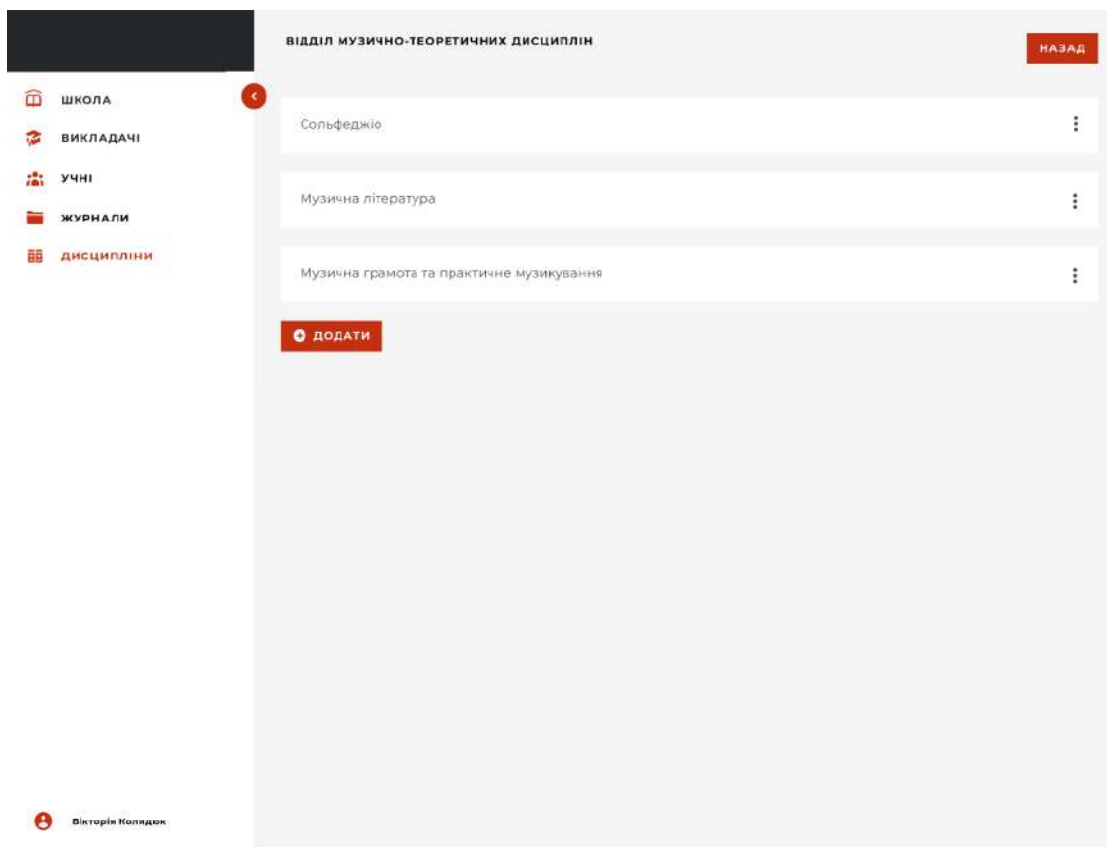


Рисунок Г.17 – Сторінка дисциплін відділу

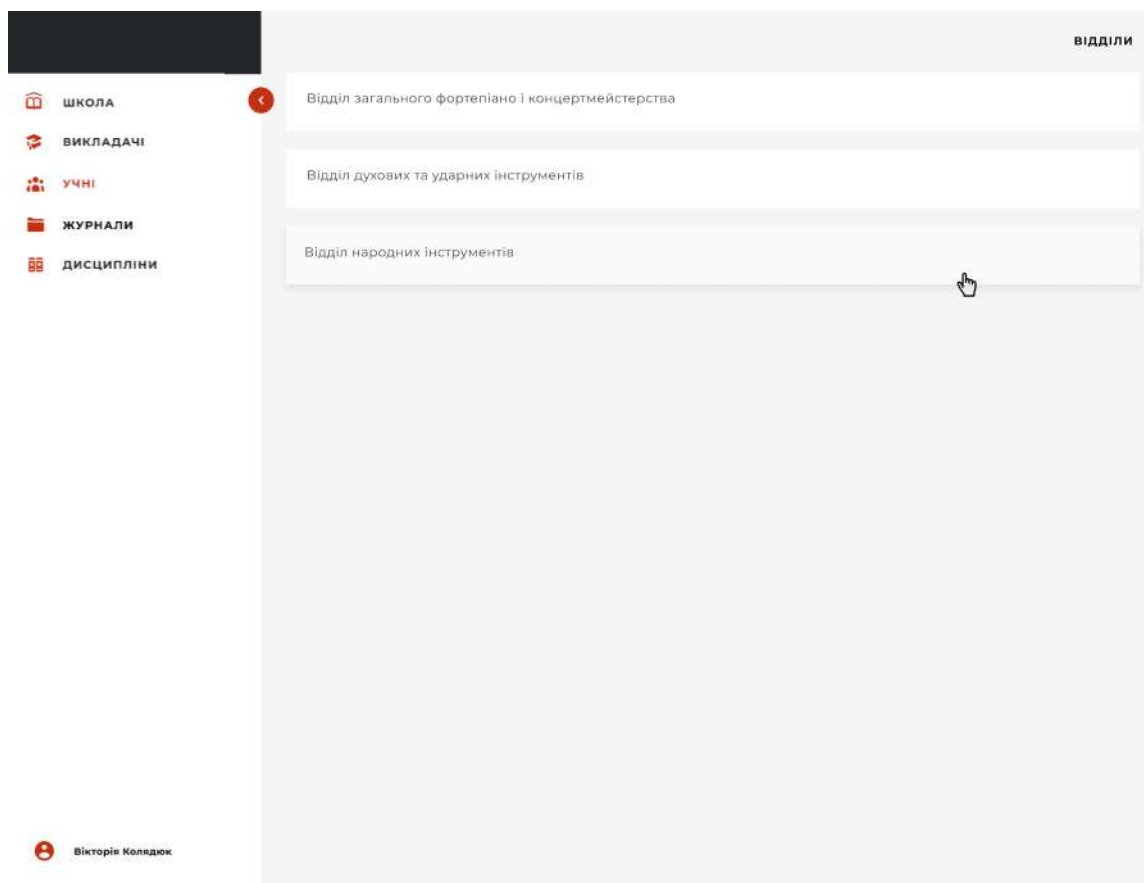


Рисунок Г.18 – Сторінка вибору відділу у вкладці учні

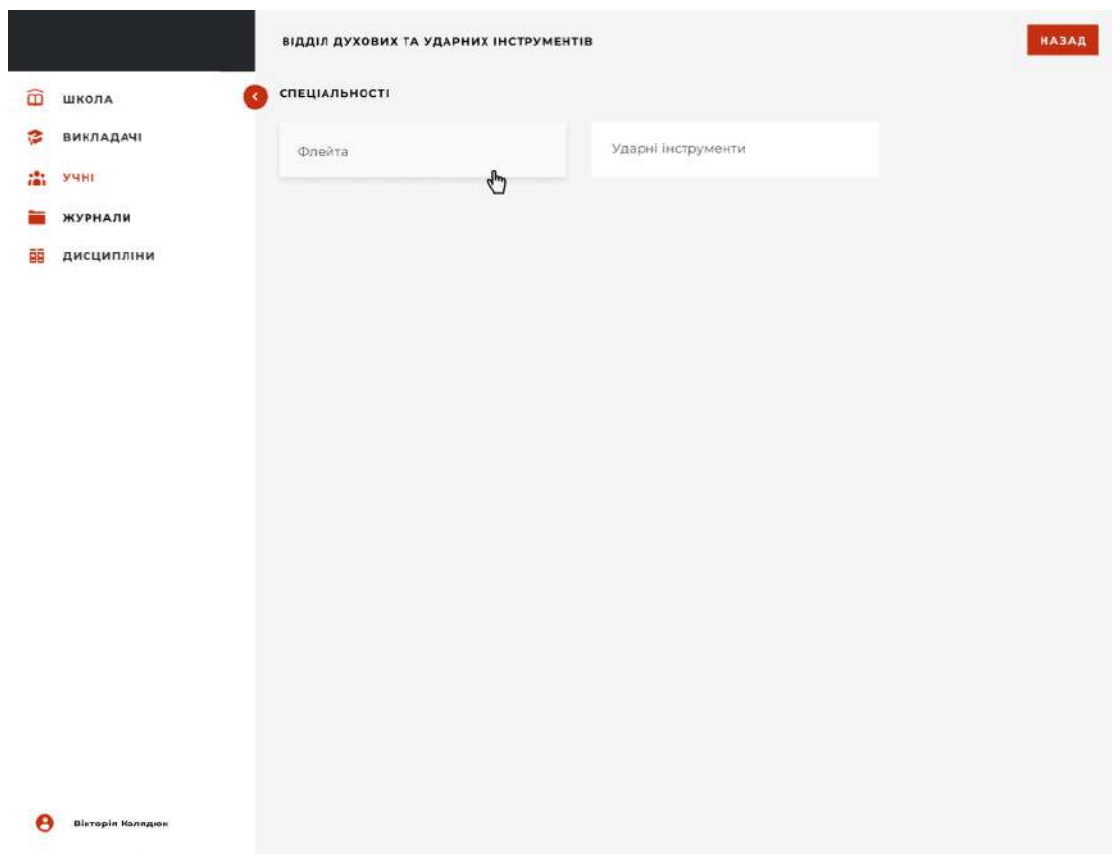


Рисунок Г.19 – Сторінка вибору спеціальності відділу

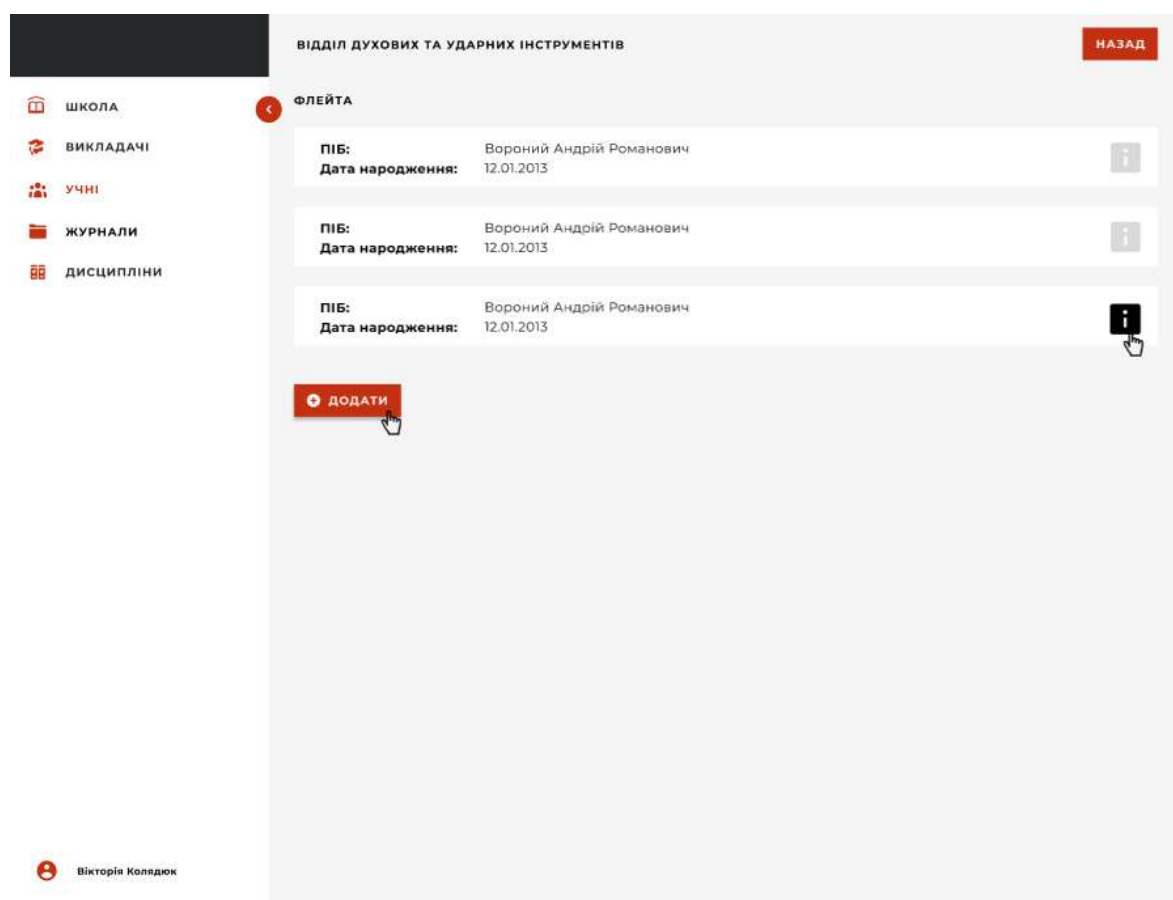


Рисунок Г.20 – Сторінка зі списком учнів певної спеціальності

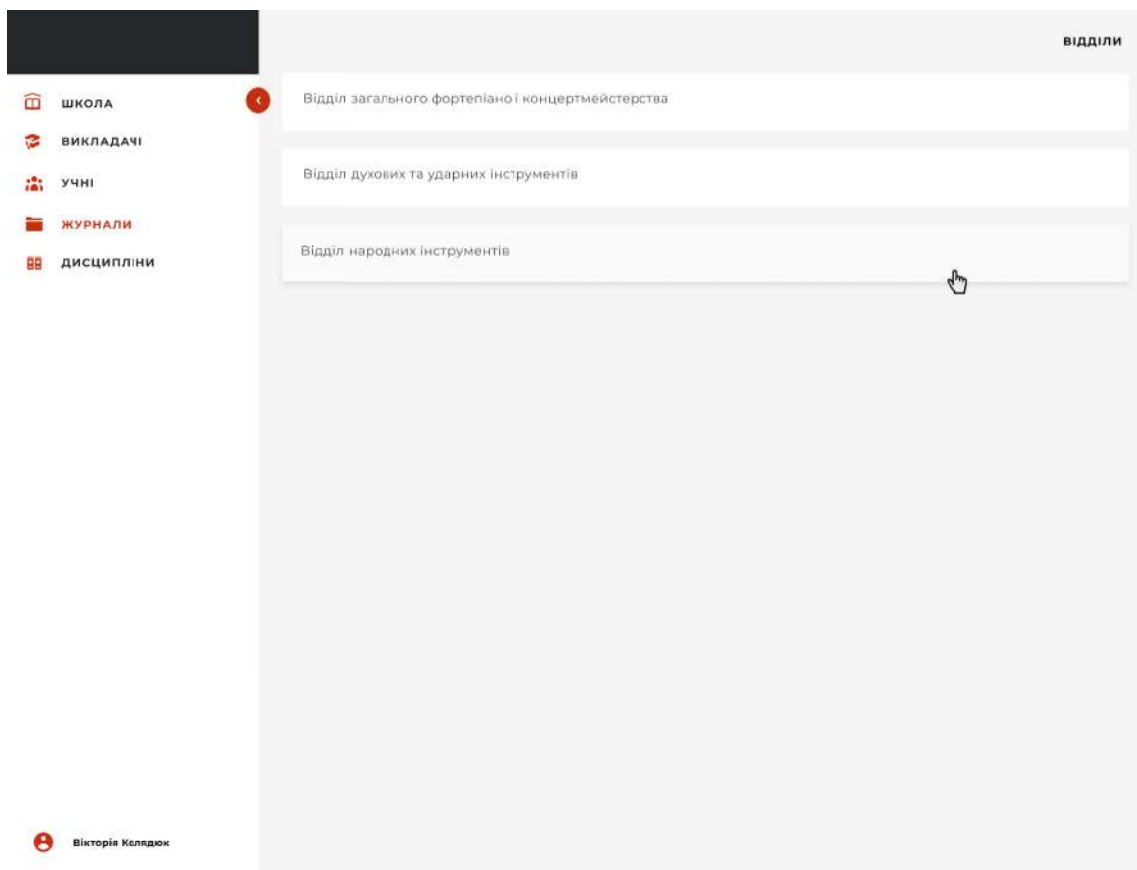


Рисунок Г.21 – Сторінка вибору відділу у вкладці журнали

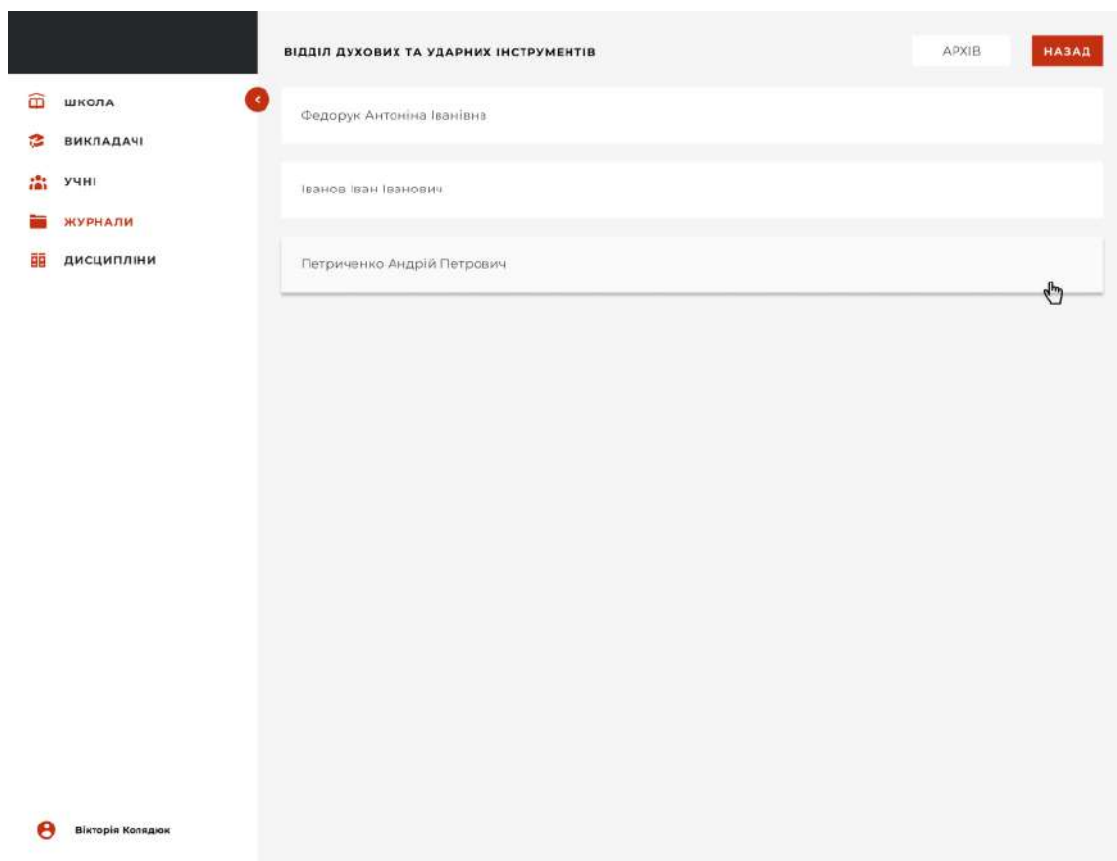


Рисунок Г.22 – Список викладачів відділу

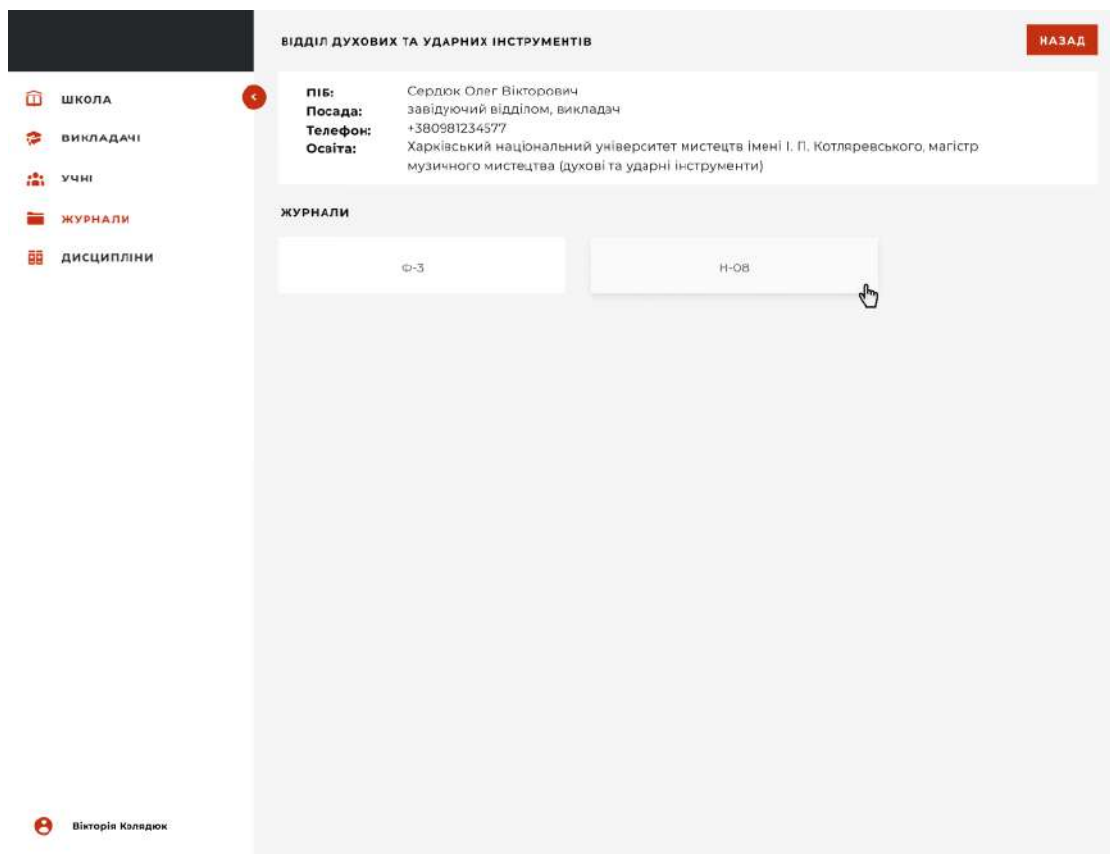


Рисунок Г.23 – Список журналів викладача

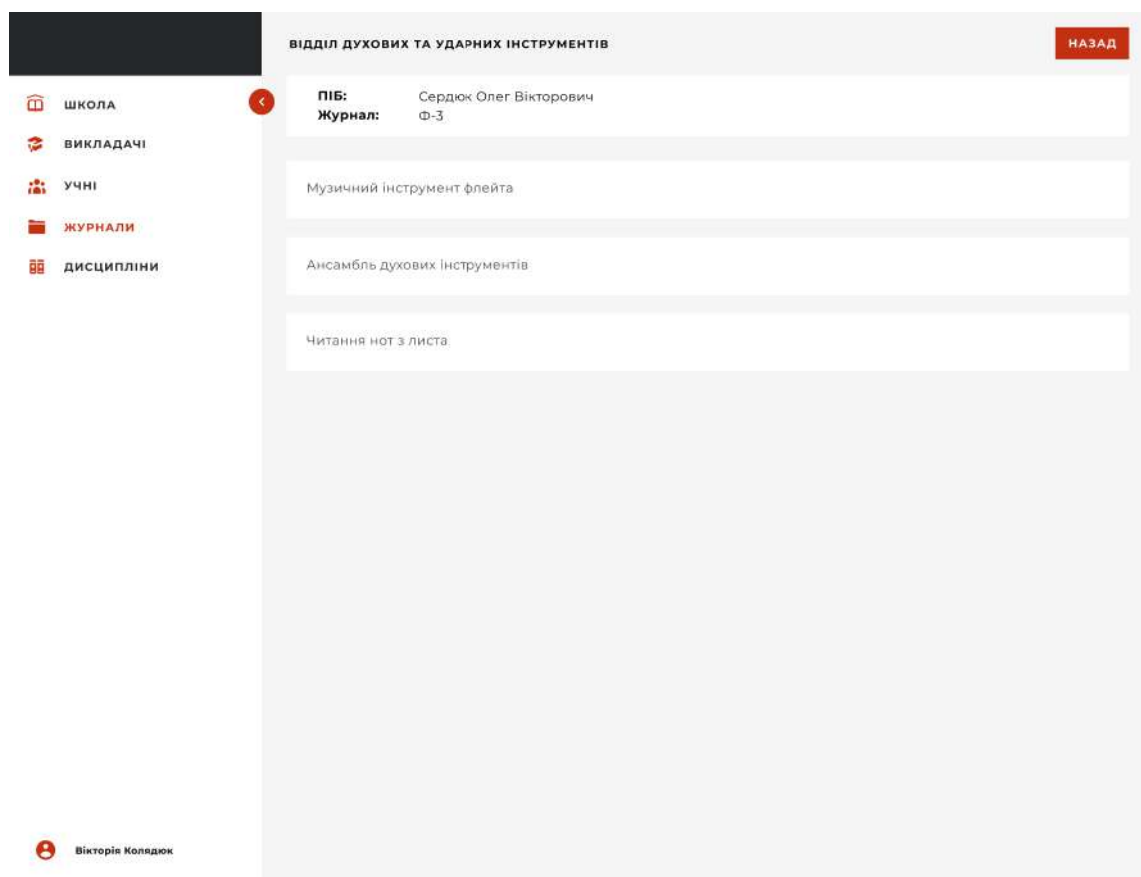


Рисунок Г.24 – Список дисциплін журналу викладача



Рисунок Г.25 – Сторінка журналу з певної дисципліни



Рисунок Г.26 – Додавання графі в журналі для обліку успішності

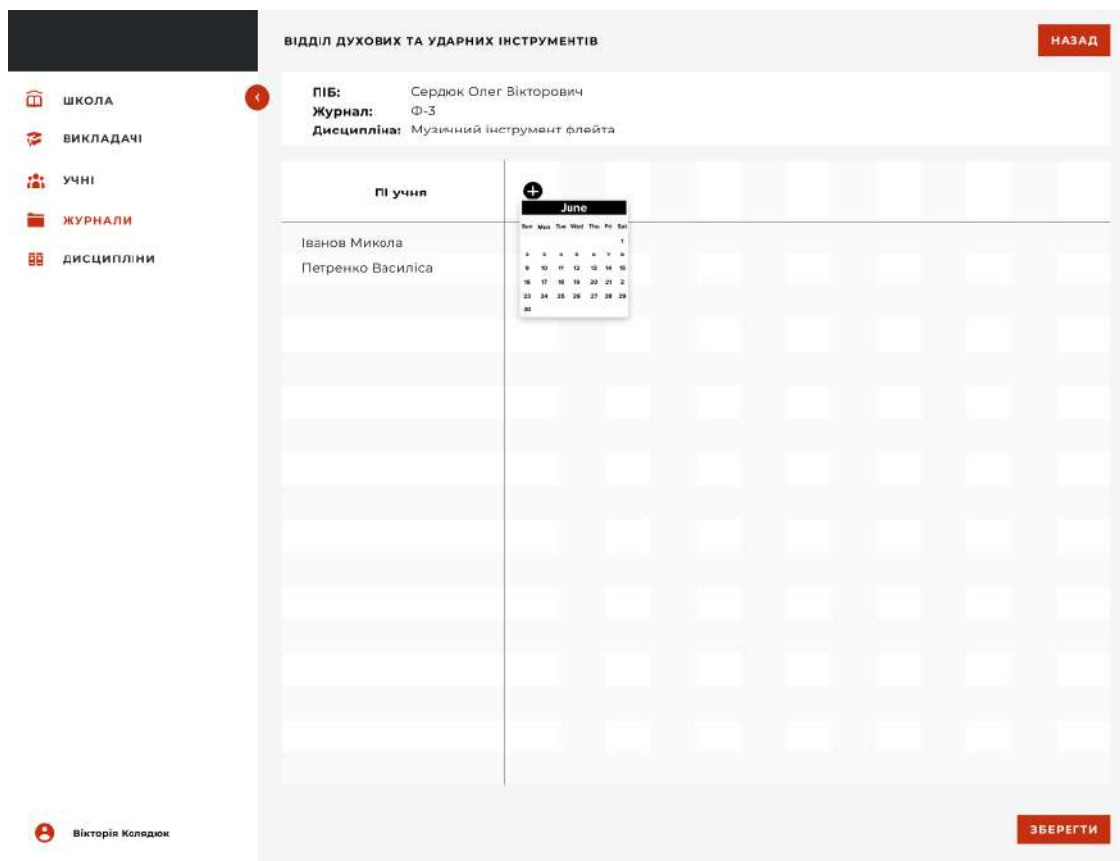


Рисунок Г.27 – Вибір дати для поточної оцінки

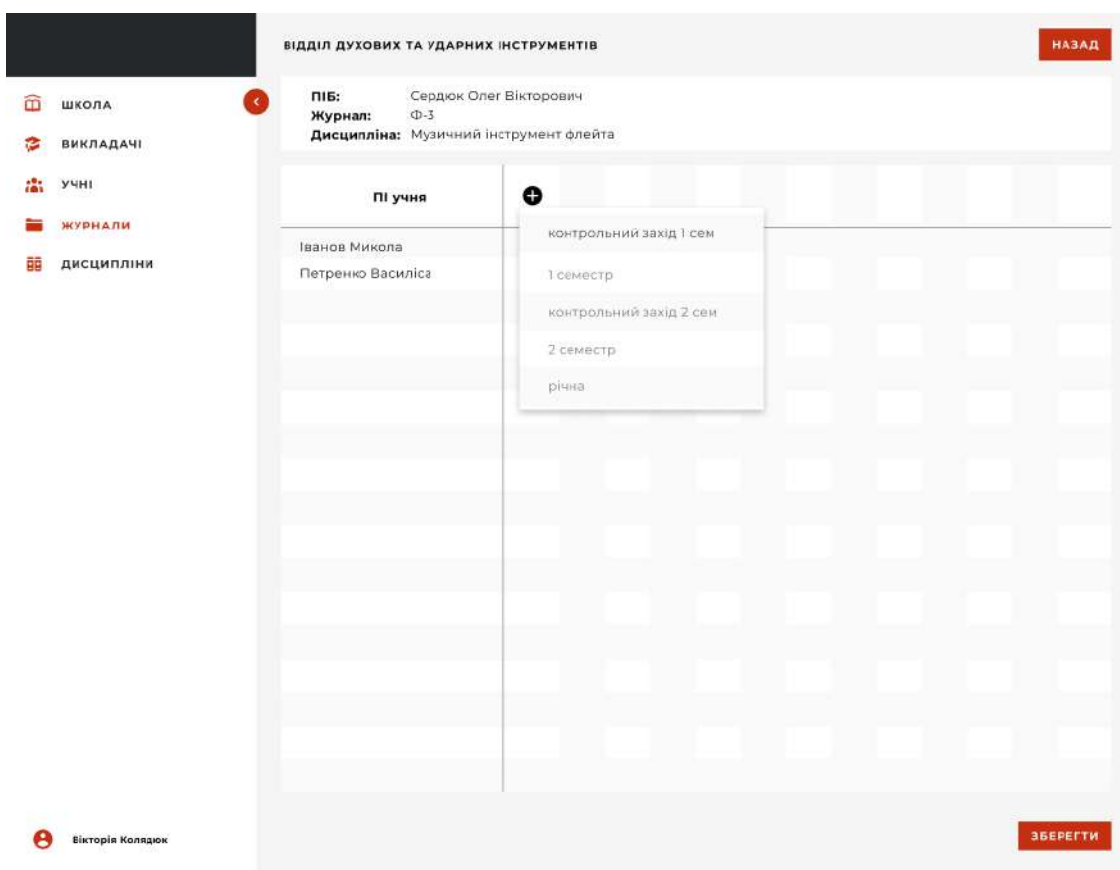


Рисунок Г.28 – Вибір підсумкової оцінки

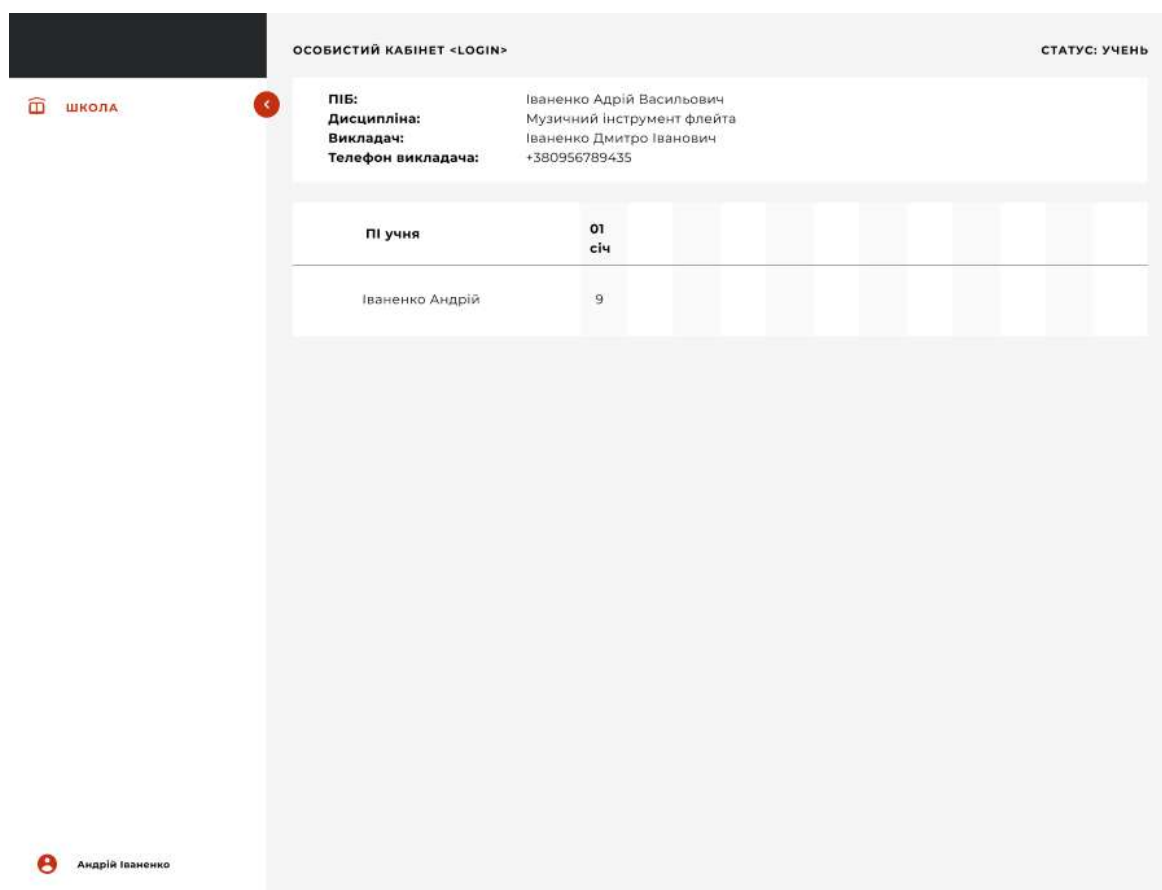


Рисунок Г.29 – Сторінка успішності учня з певної дисципліни

ДОДАТОК Д Фрагменти програмного коду

Клієнт:

Лістинг Д.1 – Код файлу index.html:

```

<!DOCTYPE html>
<html lang="ua">
  <head>
    <meta charset="utf-8" />
    <link rel="shortcut icon" href="%PUBLIC_URL%/favicon.ico"
/>
    <meta
      name="viewport"
      content="width=device-width, initial-scale=1, shrink-to-
fit=no"
    />
    <meta name="theme-color" content="#000000" />
    <!--
      manifest.json provides metadata used when your web app is
installed on a
      user's mobile device or desktop. See
https://developers.google.com/web/fundamentals/web-app-manifest/
    -->
    <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />
    <!--
      Notice the use of %PUBLIC_URL% in the tags above.
      It will be replaced with the URL of the `public` folder
during the build.
      Only files inside the `public` folder can be referenced
from the HTML.

      Unlike "/favicon.ico" or "favicon.ico",
"%PUBLIC_URL%/favicon.ico" will
      work correctly both with client-side routing and a non-
root public URL.
      Learn how to configure a non-root public URL by running
`npm run build`.
    -->
    <title>React App</title>
  </head>
  <body>
    <noscript>You need to enable JavaScript to run this
app.</noscript>
    <div id="root"></div>
    <!--
      This HTML file is a template.
      If you open it directly in the browser, you will see an
empty page.

```

You can add webfonts, meta tags, or analytics to this file.

The build step will place the bundled scripts into the `<body>` tag.

To begin the development, run ``npm start`` or ``yarn start``.

To create a production bundle, use ``npm run build`` or ``yarn build``.

```
-->
</body>
</html>
```

Лістинг Д.2 – Код файлу App.js

```
import React, { Component } from "react";
import { Router, Route } from "react-router-dom";
import { connect } from "react-redux";

import PrivateRoute from
"../../../../../shared/components/PrivateRoute";
import { history } from "../../../../../redux/helpers";
import Layout from "../../Layout";
import AuthForm from "../../Auth";
import "../App.css";

class App extends Component {
  render() {
    return (
      <Router history={history}>
        <PrivateRoute path="/layout" component={Layout} />
        <Route path="/auth" component={AuthForm} />
      </Router>
    );
  }
}

export default connect()(App);
```

Лістинг Д.3 – Код файлу App.css

```
@import '~antd/dist/antd.css';
```

Лістинг Д.4 – Код файлу Auth.js

```
import React from "react";
import { Form, Icon, Input, Button, Spin, Alert } from "antd";
import { connect } from "react-redux";

import { userActions } from "../../../../../redux/actions";
```

```

import { openNotification, notificationConstants } from
"../../utils";
import {
  StyledCentererWrapper,
  StyledAuthWrapper,
  StyledCentererLabel,
  StyledRightAnchor
} from "../styled";

const loginError = mess => (
  <>
    <Alert message={mess} type="error" showIcon />
    <hr />
  </>
);

const NormalLoginForm = ({
  form,
  dispatch,
  loggingIn,
  loginFail,
  errorMessage,
  location
}) => {
  const handleSubmit = e => {
    e.preventDefault();
    form.validateFields((err, values) => {
      if (!err) {
        dispatch(userActions.login(values));
      }
    });
  };

  const forgotPasswordNotification = () => {
    openNotification(...notificationConstants.FORGOT_PASSWORD);
  };

  const isSubmittable = func => func("email") &&
func("password");

  const { getFieldDecorator, getFieldValue } = form;

  return (
    <StyledCentererWrapper>
      <StyledAuthWrapper>
        {loggingIn ? (
          <Spin tip="Входим в аккаунт..." />
        ) : (
          <>
            <StyledCentererLabel>Війти на
сайт</StyledCentererLabel>
            {loginFail ? loginError(errorMessage) : null}
          </>
        )}
      </StyledAuthWrapper>
    </StyledCentererWrapper>
  );
}

```

```

    <Form onSubmit={handleSubmit} className="login-
form">
      <Form.Item>
        {getFieldDecorator("email", {
          rules: [
            {
              type: "email",
              message: "Невірно введена поштова
адреса!"
            },
            {
              required: true,
              message: "Будь ласка, введіть вірну
поштову адресу!"
            }
          ]
        }) (
          <Input
            prefix={
              <Icon type="user" style={{ color:
"rgba(0,0,0,.25)" }} />
            }
            placeholder="e-mail адреса"
          />
        )}
      </Form.Item>
      <Form.Item>
        {getFieldDecorator("password", {
          rules: [
            { required: true, message: "Будь ласка,
введіть пароль!" }
          ]
        }) (
          <Input
            prefix={
              <Icon type="lock" style={{ color:
"rgba(0,0,0,.25)" }} />
            }
            type="password"
            placeholder="Пароль"
          />
        )}
      </Form.Item>
      <Form.Item>
        <Button
          block
          type="primary"
          htmlType="submit"
          className="login-form-button"
          disabled={!isSubmittable(getFieldValue)}
        >
          Увійти в кабінет
        </Button>

```

```

        </Form.Item>
      </Form>
    </>
  )}
</StyledAuthWrapper>
</StyledCentererWrapper>
);
};

const mapStateToProps = state => {
  const { loggingIn, loginFail, errorMessage } =
state.authentication;
  return {
    loggingIn,
    loginFail,
    errorMessage
  };
};

const connectedLoginForm = connect(mapStateToProps)(
  Form.create({ name: "normal_login" })(NormalLoginForm)
);

export default connectedLoginForm;

```

Сервер:

Лістинг Д.5 – Код файлу srs/app.js

```

const express = require("express");
const cors = require("cors");
const helmet = require("helmet");
const { createErrorResponse } = require("../response");

const logger = require("../logger");
const pino = require("pino-http")({
  logger,
});

const app = express();

// Middleware: logger, headers, cors
app.use(pino);
app.use(helmet());
app.use(cors());

app.use("/", require("../routes"));

// 404-handling middleware
app.use((req, res) => {
  res.status(404).json(createErrorResponse(404, "not
found"));
});

```

```
// Error-handling middleware
// eslint-disable-next-line no-unused-vars
app.use((err, req, res, next) => {
  const status = err.status || 500;
  const message = err.message || "unable to process request";

  if (status > 499) {
    logger.error({ err }, `Error processing request`);
  }

  res.status(status).json(createErrorResponse(status,
message));
});

module.exports = app;
```

Лістинг Д.6 – Код файлу srs/index.js

```
require("dotenv").config({
  path: process.env.ENV_FILE ? process.env.ENV_FILE : ".env",
});

const logger = require("./logger");
const db = require("./models");

process.on("uncaughtException", (err, origin) => {
  logger.fatal({ err, origin }, "uncaughtException");
  throw err;
});

process.on("unhandledRejection", (reason, promise) => {
  logger.fatal({ reason, promise }, "unhandledRejection");
  throw reason;
});

db.init()
  .then(() => {
    require("./server");
  })
  .catch((err) => {
    logger.fatal(err, "Unable to connect to DB");
  });
```

Лістинг Д.7 – Код файлу srs/config.js

```
module.exports = {
  dbConfig: {
    database: process.env.DB_DATABASE,
    user: process.env.DB_USER,
    password: process.env.DB_PASSWORD,
    host: process.env.DB_HOST,
```

```

    dialect: process.env.DB_DIALECT || "mysql",
    alterTables: process.env.DB_ALTER_TABLES || false,
  },
  port: parseInt(process.env.PORT || 8080, 10),
  maxImageSize: 2000000,
};

```

Лістинг Д.8 – Код файлу srs/response.js

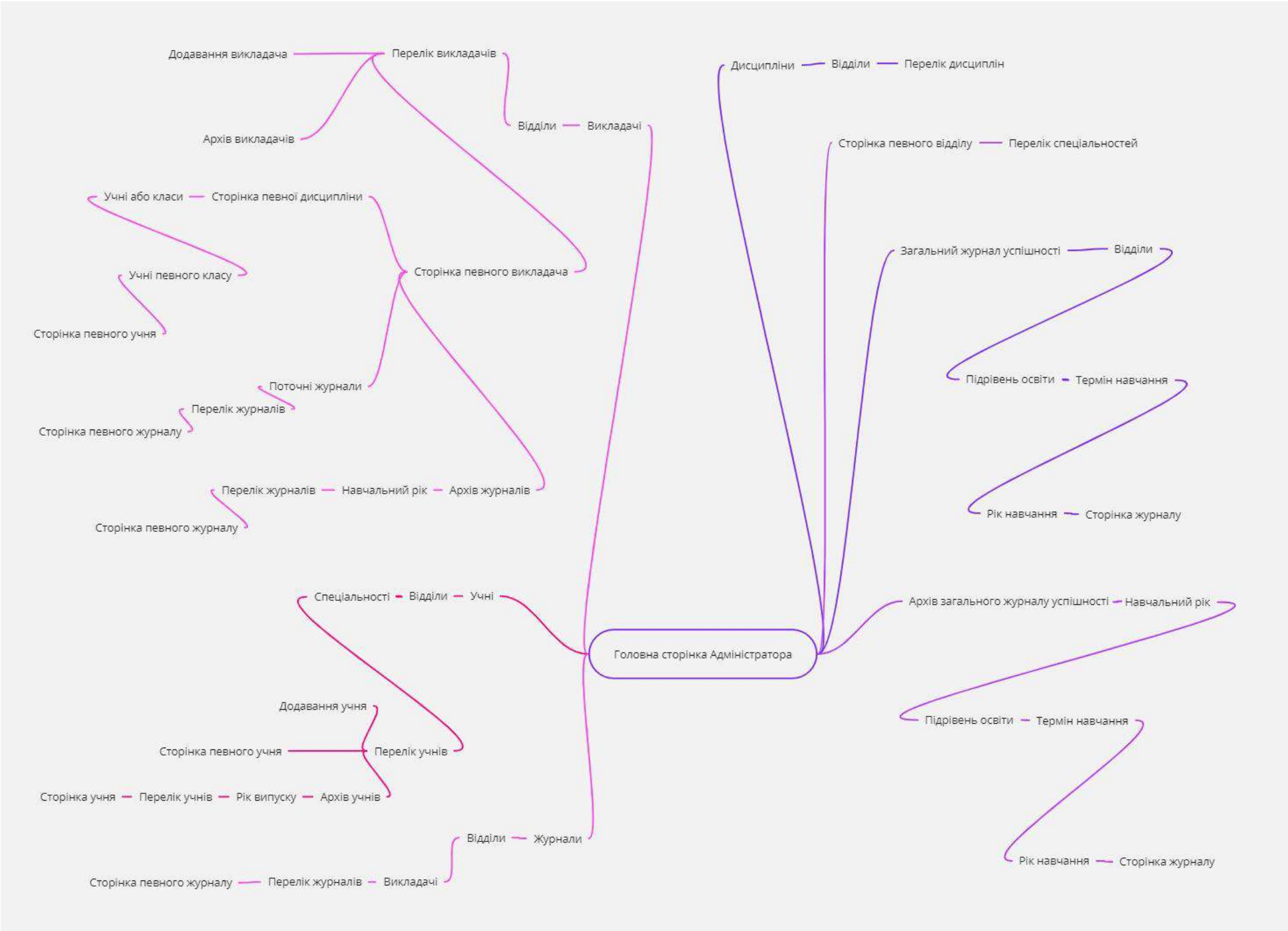
```

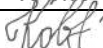
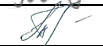
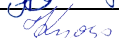
module.exports.createSuccessResponse = function (data) {
  return {
    status: "ok",
    ...data,
  };
};

module.exports.createErrorResponse = function (code, message) {
  return {
    status: "error",
    error: {
      code: code,
      message: message,
    },
  };
};

```

Mind-Map структури сторінок особистого кабінету адміністратора

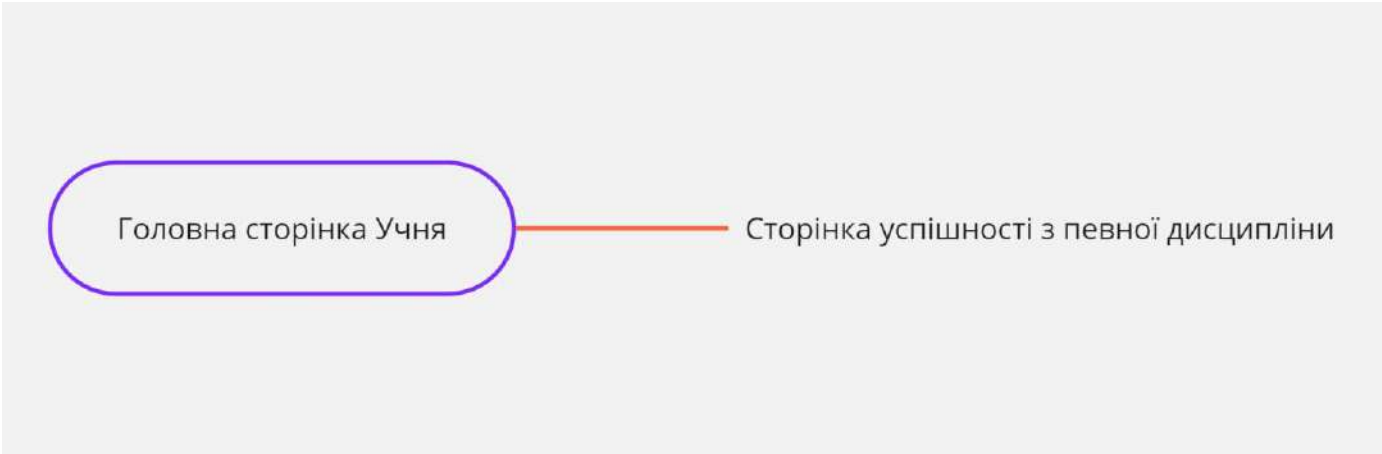


					13.02070849.00063 ПЛ1						
					Інформаційна система обліку успішності учнів музичних шкіл Mind-Map структури сторінок особистого кабінету адміністратора	Лит.			Маса	Масшт.	
Зам.	Лист	№ документа	Підпис	Дата							
Розроб.		Ковалевська С.Г.									
Перевір.		Голуб Т.В.									
Т.контр.											
Н.контр.											
Затвердж.		Кудерметов Р.К.									

Mind-Map структури сторінок особистих кабінетів викладача та учня



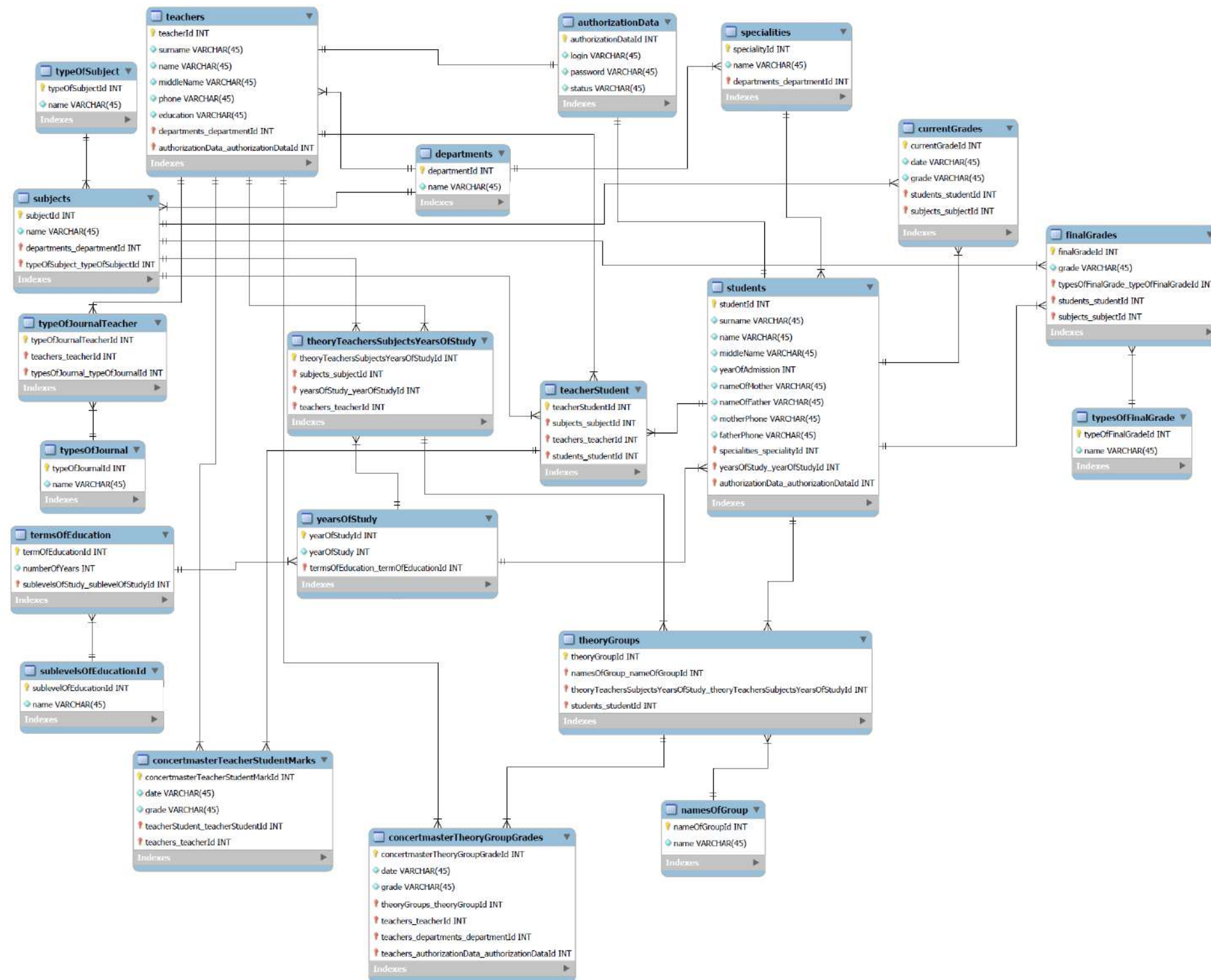
Mind-Map структури сторінок особистого кабінету викладача

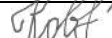


Mind-Map структури сторінок особистого кабінету учня

					13.02070849.00063 ПЛ2						
					Інформаційна система обліку успішності учнів музичних шкіл Mind-Map структури сторінок особистих кабінетів викладача та учня	Лит.			Маса	Масшт.	
Зам.	Лист	№ документа	Підпис	Дата							
Розроб.		Ковалевська С.Г.									
Перевір.		Голуб Т.В.									
Т.контр.											
Н.контр.											
Затвердж.		Кудерметов Р.К.									
						Лист. 2 Листів 4					
						НУ «Запорізька політехніка» КНТз-511сп					

ER-діаграма бази даних



					13.02070849.00063 ПЛЗ								
					Інформаційна система обліку успішності учнів музичних шкіл ER-діаграма бази даних				Лит.		Маса	Масшт.	
Зам.	Лист	№ документа	Підпис	Дата									
Розроб.		Ковалевська С.Г.											
Перевір.		Голуб Т.В.											
Т.контр.													
									Лист. 3		Листів 4		
Н.контр.									НУ «Запорізька політехніка» КНТз-511сп				
Затвердж.		Кудерметов Р.К.											

Use Case Diagram електронного журналу



					13.02070849.00063 ПЛ4									
					Інформаційна система обліку успішності учнів музичних шкіл Use Case Diagram електронного журналу	Лит.			Маса		Масшт.			
Зам.	Лист	№ документа	Підпис	Дата										
						Лист. 4			Листів 4					
						НУ «Запорізька політехніка» КНТз-511сп								
								</						