Хмельницький національний університет
Факультет інформаційних технологій
Кафедра інженерії програмного забезпечення

ДИПЛОМНИЙ ПРОЕКТ

Програмне забезпечення для організації, управління та впорядкування навчального процесу у загально-освітніх школах

Назва теми

Рівень вищої освіти Перший (бакалаврський)

Галузь знань 12 «Інформаційні технології»

Спеціальність 121 «Інженерія програмного забезпечення»

Освітня програма Освітньо-професійна програма «Інженерія програмного забезпечення»

Шифр ДПІПЗ.190148.01.03.ПЗ

Виконала студентка ІІІ курсу групи ІПЗс-19-1 _____ К. В. Дрищ
Підпис Ініціали, прізвище

Керівник канд. пед. наук, доцент _____ Н. І. Праворська
Науковий ступінь, звання Підпис Ініціали, прізвище

Нормоконтролер канд. техн. наук, доцент _____ Г. І. Бедратюк
Підпис Ініціали, прізвище

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

1 червня 2022 р.

Хмельницький 2022

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

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

ЗАТВЕРДЖУЮ
Завідувач кафедри ІПЗ
Л. П. Бедратюк
05 02 2022 р.

ЗАВДАННЯ
НА ДИПЛОМНИЙ ПРОЕКТ (РОБОТУ)

Дрищ Катерини Володимирівни
Прізвище, ім'я, по батькові студента

Тема проекту (роботи) Програмне забезпечення для організації, управління та впорядкування навчального процесу у загально-освітніх школах

Керівник проекту (роботи) Граборська Наталія Іванівна канд пед наук доцент
Прізвище, ім'я, по батькові, науковий ступінь, вчене звання

1. Затверджена наказом ректора університету від 05.02.2022 р. № 11
2. Строк подання студентом проекту (роботи) на кафедру 07.06.2022 р.
3. Вихідні дані до проекту (роботи) Матеріали переддипломної практики

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)
Дослідження предметної області (опис предметної області, аналіз існуючих методів та рішень, визначення технічного завдання та вимог до проекту), проектування сайту (проектування архітектури, проектування бази даних, проектування інтерфейсу), реалізація проекту (структура програмних модулів, реалізація бази даних, реалізація сайту, інструкція з експлуатації), тестування, висновки, перелік джерел посилання
5. Перелік графічного матеріалу (із зазначенням обов'язкових креслень)
Діаграма варіантів використання для акторів з різним видом доступу, схема бази даних, діаграма послідовності, діаграма розгортання, лістинг коду, презентаційні матеріали (слайди)

6. Консультанти розділів дипломного проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Бедратюк Т.І., ст. викладач кафедри УІТЗ		
Антиплагіат	Гурман І.В., доцент кафедри УІТЗ		

7. Дата видачі завдання «05» лютого 2022 р.

КАЛЕНДАРНИЙ ПЛАН

Назва етапів (розділів) дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1 Отримання теми дипломного проекту та завдання	*15.11 – 31.12.2021*	
2 Дослідження предметної області	*03.01 – 31.01.2022*	
3 Аналіз існуючих програмних продуктів	*01.02 – 15.02.2022*	
4 Проектування архітектури та інтерфейсу, Проектування БД	*16.02 – 16.03.2022*	
5 Вибір засобів для розробки сайту	*17.03 – 21.03.2022*	
6 Розробка структури проекту	*22.03 – 12.04.2022*	
7 Підключення та розробка БД	*13.04 – 17.04.2022*	
8 Розробка інтерфейсу, Розробка програмних модулів	*18.04 – 9.05.2022*	
9 Тестування проекту	*9.05 – 12.05.2022*	
10 Опрацюання висновку та переліку літератури	*13.05 – 17.05.2022*	
11 Попередній захист ДП	*18.05.2022*	
12 Перевірка ДП на плагіат, нормоконтроль, отримання відгуків та рецензій. Брошурування (зшиття) пояснювальної записки	*19.05 – 3.06.2022*	
13 Підготовка до захисту та захист ДП	*з 06.06.2022*	

Студент _____ Дрищ К.В.
Підпис Ініціали, прізвище

Керівник проекту (роботи) _____ Приворська Н.Ї.
Підпис Ініціали, прізвище

АНОТАЦІЯ

Тема дипломного проекту: «Програмне забезпечення для організації, управління та впорядкування навчального процесу у загально-освітніх школах».

Автор проекту: Дрищ Катерина Володимирівна.

Керівник проекту: Праворська Наталія Іванівна.

Пояснювальна записка: 155 с., 47 рис., 7 табл., 8 дод., 12 джерел.

Графічна частина: 22 презентаційні слайди.

ЕЛЕКТРОННА ШКОЛА, ОРГАНІЗАЦІЯ УЧБОВОГО ПРОЦЕСУ, C#, SQL, ASP Net.CORE.

Об'єктом дослідження є сфера організації учбового процесу в школах.

Мета дипломного проекту: створення сайту, що забезпечив би можливість організації роботи між вчителями та учнями, із наданням всього неохідного функціоналу, що присутній оф-лайн, тобто ведення щоденника, журналу, та ін.

У дипломному проекті проведено аналіз предметної області, визначено функціональні та нефункціональні вимоги до програмного забезпечення, розроблена структура додатку, спроектовано архітектуру та інтерфейс, розроблено базу даних та програмні модулі. Для розробки програмного продукту використано мову C#, MVC, ASP.Net Core, HTML, CSS, JavaScript, JQuery.

30.05
Дата

Підпис

ВІДОМІСТЬ ДОКУМЕНТІВ

№ рядка	Формат	Позначення документа	Найменування документа	К-сть аркушів	№ екз.	Примітка
			Текстові документи			
1	А4	ДПІПЗ.190148.01.03.ПЗ	Пояснювальна записка	73		
2	А4		Завдання на дипломний	1		
			проект			
3	А4		Анотація	1		
			Графічні документи			
4	А4		Презентаційні матеріали	22		

Змн.	Арк.	№ докум.	Підпис	Дата	ДПІПЗ.190148.01.03.ПЗ		
Виконав		Дрищ К.В.		31.05	Програмне забезпечення для організації, управління та впорядкування навчального процесу у загально-освітніх школах Пояснювальна записка	Літ.	Арк. 5 / Аркушів 155
Керівник.		Праворська Н.І.		01.06			
Рецензія.						ХНУ, ІПЗс-19-1	
Н. Контр		Бедратюк Г.І.		01.06			
Затвердив		Бедратюк Л.П.		01.06			

Зміст

ВСТУП

Організація учбового процесу – це та сфера, з якою практично всі досить добре знайомі. Вона завжди буде важливою та необхідною адже певним чином впливає на формування кожної людини.

Якісна організація учбового процесу є дуже важливою, оскільки, це впливає на успішність учнів, їхню задоволеність від навчання та їх батьків. Це дуже важлива сфера, з якою стикався кожен із нас.

Оскільки, розвиток постійно призводить до автоматизації певних процесів, школа, теж не є винятком. Завжди все простіше контролювати, маючи певну систему зі всіма необхідними документами. Це також дозволяє дуже швидко реагувати на всі проблеми чи неточності. Є можливість швидкого отримання та надавання інформації про будь-якого учня школи, вчителів. Передбачена змога слідкувати за шкільними заходами онлайн і не пропускати нічого важливого.

Організація учбового процесу – це завжди актуальне питання, а враховуючи сьогоднішню ситуацію, виникла надважлива потреба у переході на он-лайн платформи. Школи намагаються певним чином організовувати весь учбовий процес он-лайн для забезпечення безпеки учнів. Проте, інколи, такий раптовий перехід є дуже складним або взагалі практично неможливий, оскільки організація учбового процесу в школі є вузькоспеціалізованим напрямком, що потребує відповідного сервісу. Такий сервіс має забезпечувати повноцінну роботу між учнями і вчителями [1].

При потребі переходу на систему навчання он-лайн, часто виникає складність у налаштуванні системи під конкретну школу. Не завжди є весь необхідний функціонал для роботи учнів чи вчителів, що зменшує ефективність роботи чи навчання. І, як наслідок, не вдається повністю змінити стиль навчання і доводиться поєднувати декілька сервісів, он-лайн та оф-лайн форми навчання. Тому постало питання у великому попиті, але при цьому досить обмежених пропозиціях.

Тема роботи: Програмне забезпечення для організації, управління та впорядкування навчального процесу у загальноосвітніх школах» [1].

Мета проекту – розробити систему, у вигляді сайту, що дозволить автоматично здійснювати навчання учнів шкіл он-лайн. Таке програмне забезпечення повинно надати можливість простої та зрозумілої роботи в даному середовищі.

Об’єкт дослідження – процес організації учбового процесу в загальноосвітніх школах.

Предметом дослідження є сфера функціонування шкіл.

Завдання дипломного проекту:

- аналіз та дослідження організації учбового процесу;
- розробка технічного завдання;
- визначення функціональних характеристик;
- визначення вимог та специфікацій;
- визначення типів архітектур та зразків проектування;
- детальне проектування модулів та даних.
- розробка програмнихмодулівтаінтерфейсу;
- тестування розробленого ПЗ.

1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Аналіз предметної області

Навчальний процес у навчальних закладах – це система організаційних і дидактичних заходів, спрямованих на реалізацію змісту освіти на певному освітньому або кваліфікаційному рівні відповідно до державних стандартів освіти.

Система управління навчанням – платформа або програмний додаток, призначений для інтеграції інструментів навчання, а також адміністрування, управління та розповсюдження освітніх та інформаційних матеріалів, формування аналітики та звітності.

Автоматизована організація учбового процесу – це тема, яка є завжди актуальною, особливо в умовах сьогодення. Організація навчання дає можливість забезпечення оптимальної роботи процесу управління навчальною діяльністю. При цьому програмне забезпечення, що надає таку можливість повинно бути постійно доступним. В такому випадку найкращим рішення буде розробка саме сайту, який дозволить виконувати всі необхідні функції, які стосуються організації роботи в певному учбовому закладі.

Сайт – це сукупність веб-сторінок та залежного вмісту, доступних у мережі Інтернет, які об'єднані як за змістом, так і за навігацією під єдиним доменним ім'ям. Популярність сайту обумовлена широкою доступністю. Немає необхідності щось завантажувати, встановлювати, все дуже просто і швидко.

При розробці сайту важливо розуміти, за яким принципом буде працювати майбутнє програмне забезпечення. Тому необхідно правильно підібрати архітектуру, за якою він буде будуватися та програмні рішення, для його реалізації.

Архітектура веб-сайтів – це чітке планування та якісне проектування технічних, візуальних та функціональних компонентів веб-сайту – до його

розробки, розробки та розгортання. Він використовується дизайнерами веб-сайтів та розробниками, як засіб для розробки та розробки веб-сайту.

Архітектура веб-сайтів необхідна при створенні логічного макета веб-сайту відповідно до вимог користувача та/або бізнесу. Такий макет визначає різні компоненти, які становлять веб-сайт, і послуги, які надаватимуться компонентами або сайтом в повному складі.

Доречно вказати фактори, що відносяться до складу архітектури веб-сайтів:

– технічні обмеження, такі як: сервер, сховище, інтерфейси пам'яті та зв'язку;

– функціональні аспекти, такі як: тип послуг або процеси, які надаватиме веб-сайт;

– інтерфейс користувача, візуальний зовнішній вигляд, тобто: кольори, кнопки та інші елементи візуального дизайну;

– параметри безпеки, тобто: забезпечення сайтом безпечного контролю доступу та транзакцій.

Дуже важливим елементом при побудові сайту є дизайн. Дизайн сайту – це оформлення контенту, сукупність усіх графічних елементів на веб-сторінці. Раніше під веб-дизайном розуміли виключно візуальне оформлення, але зараз, в першу, чергу враховується зручність користувача, тому одне із основних завдань веб-дизайнера – це аналітика та грамотне структурування інформації на сайті.

Основне завдання дизайну – ознайомлення користувача зі сторінкою. Він полегшує взаємодію користувача з веб-сторінкою, а отже, позитивно впливає на конверсію та поведінкові фактори. Продуманий дизайн дозволяє ефективно та швидко працювати, застосовувати наданий функціонал.

При правильному поєднанні усіх компонентів та складових сайту із забезпеченням простого та зручного інтерфейсу, не повинно виникнути ніяких проблем в експлуатації.

Також, враховуючи, що система є досить багатофункціональною, є необхідність у розмежуванні доступу. Кожній певній групі користувачів має бути доступний тільки той функціонал, який відповідний для неї, і нічого лишнього.

1.2 Огляд існуючого програмного забезпечення

Перед початком розробки програмного продукту, було прийнято рішення розглянути існуючі версії подібних веб-сайтів, які можуть задовольняти поставленим вимогам, їх переваги та недоліки.

Ayotree

Загальний вигляд сайту Ayotree представлено на рисунку 1.1.

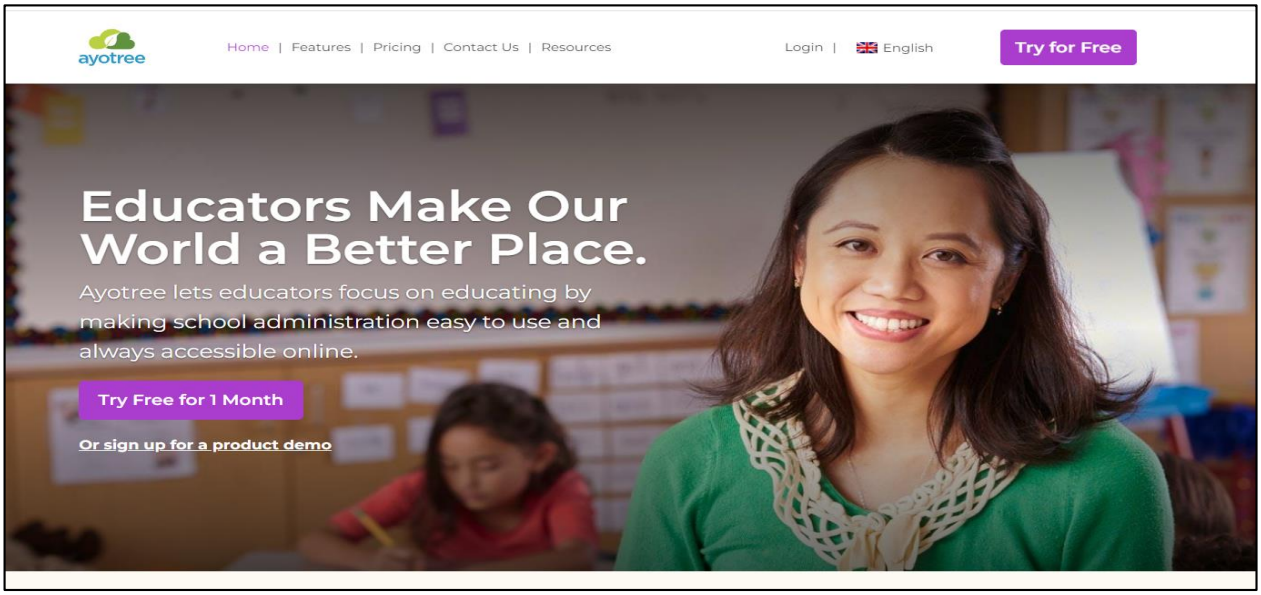


Рисунок 1.1 – Вигляд сайту Ayotree

Дана система для шкільного менеджменту Ayotree допомагає стандартним офф-лайн та он-лайн школам здійснювати керування програмою, розкладами занять, відвідуваннями, оплатою та віртуальними класами за допомогою захищеної системи [2].

Переваги:

– навчальний процес може здійснюватися з будь-якої точки Земної кулі;

– проведення уроків для учнів можливе з іншого міста, країни чи навіть іншого континенту, використовуючи віртуальний клас Aytotree чи будь-яку іншу платформу онлайн-класу, такі як Zoom, ClassIni т.ін. Підтримка Ayotree реалізованана будь-якому девайсі;

– Ayotree працює за щомісячної абонплатою попідписці, відсутні комісії за транзакції. Крім того, перший місяць безкоштовний;

– цілодобова підтримка клієнтів;

– надання допомоги попереходу на Ayotree.

Недоліки.

Система не орієнтована на організацію навчання у школах, тому не містить всього потрібного функціоналу, проте має велику кількість зайвих компонентів

Moodle

Загальний вигляд сайту Moodle представлено на рисунку 1.2.

Рисунок 1.2 – Вигляд стартової сторінки сайту Moodle

Moodle – одна з найбільш популярних систем менеджменту учбового процесу з відкритим вихідним кодом, вона орієнтована на організацію взаємодії між викладачами та учнями. Система може використовуватись як для дистанційного навчання, так і для очного [2].

Moodle є веб-орієнтованою системою, що досить легко масштабується та може бути підлаштована під конкретний учбовий заклад, вона характеризується досить високим рівнем безпеки. В наявності великий набір інструментів для дистанційного навчання. Moodle підтримується групою сертифікованих

партнерів, а також має велику аудиторію активних користувачів спільноти та розробників по всій Землі (близько 130 мільйонів користувачів).

Переваги:

– Moodle має вбудований конструктор курсів, який підтримує синхронне, асинхронне та змішане, мобільне навчання;

– має можливість влаштовувати конференції по відеозв’язку та ін;

– містить велику кількість плагінів;

– відкритий вихідний код.

– недоліки.

Недоліки.

При великій кількості переваг проекту з відкритим кодом, архітектура системи дуже складна і самостійно кастомізувати Moodle можна, але іноді здається, що легше написати потрібну систему з нуля. Також не досить вдалим є інтерфейс користувача. Непрезентабельний дизайн системи нагадує сайти початку 21-го століття.

Canvas LMS

Загальний вигляд сайту Canvas LMS представлено на рисунку 1.3.

Рисунок 1.3 – Стартова сторінка сайту Canvas LMS

Canvas – ще одна платформа для організації навчального процесу, що теж розроблялася як система для навчальних закладів. Це не просто система менеджменту учбового процесу, а й багато іншого цікавого функціоналу[2].

Переваги:

– наявність великої кількості сучасних інструментів для створення курсів;

– відкритий арі, завдяки чому можлива підтримка різноманітних сторонніх розширень, що дозволяє змінювати та вдосконалювати функціонал не орієнтуючись у програмуванні;

– зручність використання вкладок та контекстного меню інтерфейсу користувача;

– досить високий рівень безпеки;

– можливість мобільного навчання.

Недоліки:

– відсутність гейміфікації;

– платна основа користування системою;

– можливий 14-денний пробний доступ до розширених можливостей, але вартість тарифних планів не розголошується у відкритому доступі.

SAP Litmos

Загальний вигляд сайту SAP Litmos представлено на рис 1.4.

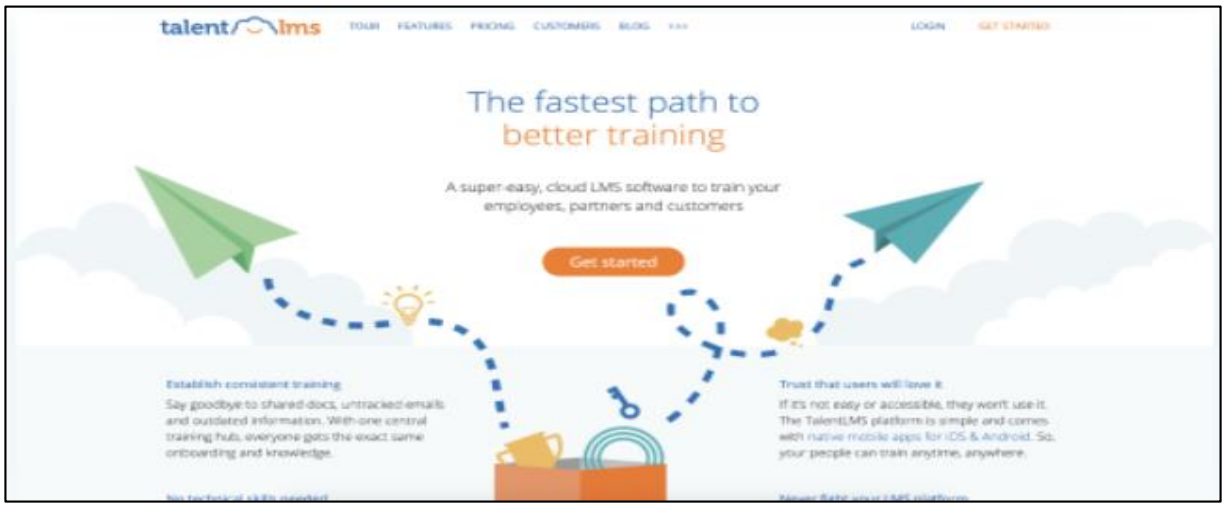


Рисунок 1.4 – Вигляд сайту SAP Litmos

Litmos – популярна система менеджменту учбового процесу, що відзначена нагородою G2 як одна з найкращих корпоративних багатофункціональних платформ 2019 року для цифрового навчання [2].

Переваги:

– основними особливостями системи є наявність вбудованої бібліотеки готових курсів, гнучка та досить детальна система звітності, управління користувачами;

– платформу можна легко масштабувати завдяки потужному API з можливістю інтегрувати з багатьма іншими додатками;

– система підтримує асинхронне, а також змішане навчання, можливість налаштування платного доступу до курсів, стандарти SCORM, синхронне навчання, наявність можливості проведення конференцій за допомогою відеозв’язку та багато іншого;

– локалізована на 24 мови.

Недоліки:

– підтримка системи відбувається через систему онлайн-тікетів, тобто навіть після дзвінка створюється тікет від вашого імені, що збільшує час очікування, та приносить складнощі з об'єднанням курсів;

– недоліки інтерфейсу в режимі для учнів.

Вартість передплати залежить від числа користувачів та обраного попередньо тарифного плану. Надається 14-денний безкоштовний доступ.

1.3 Визначення вимог до ПЗ та постановка задачі

Після повноцінного аналізу предметної області та оцінки наявних аналогів було визначено набір функціональних вимог до програмного забезпечення. Дана система буде містити досить значну кількість функціоналу, при цьому передбачається різнопланова робота за ролями. Основними з них будуть учні, батьки, класні керівники та вчителі. З іншого боку всю роботу по заповненню

сайту буде здійснювати адміністратор, що відповідатиме за налаштування сайту для різних користувачів. Його робота буде полягати в заповненні розкладу, формуванні груп, журналів. У викладачів основною діяльністю буде заповнення оцінок та розсилка домашнього завдання. Учні ж матимуть лише можливість перегляду та роботи із чек-лістом домашнього завдання.

На відміну від наявних продуктів, дана система буде вузькоспеціалізованою, орієнтований на організацію учбового процесу в школі. Проте завдяки цьому він надаватиме ряд функцій, що є нетиповими для іншого ПЗ у даній предметній області. Така система дозволить перенести звичний процес навчання у школах у он-лайн режим.

Отже було визначено такі функціональні вимоги:

– реєстрація/авторизація в системі;

– розмежування доступу користувачів за ролями;

– можливість роботи із групами, а саме додавання/редагування;

– можливість реєстрації користувачів із додаванням їх у базу даних;

– можливість додавання/редагування розкладу;

– робота із щоденником, що передбачає виведення предметів із врахуванням домашнього завдання та оцінок учня;

– чек-ліст із домашніми завданнями учня, враховуючи ступінб завершенності завдань;

– формування табелів;

– виставлення оцінок вчителями та розсилка домашнього завдання;

– додавання та редагування вчителів із вибором предметів, при цьому передбачається, що кожен вчитель може викладати декілька дисциплін;

– в системі повинна бути можливість додавання і виведення шкільних заходів для всіх категорій користувачів;

– передбачається можливість роботи у даній системі і для батьків, а саме перегляд щоденника дітей, перегляд шкільних заходів, отримання повідомлень від куратора чи вчителів.

Також було визначено список вимог до інтерфейсу, зокрема:

– інтерфейс повинен бути достатньо простим та зрозумілим, оскільки система призначена і для роботи з дітьми;

– сайт складатиметься із багатьох сторінок, що не містять великої кількості інформації;

– при розробці дизайну сайту, повинні переважати світлі тони;

– основна меню має містити всі необхідні пункти в залежності від ролі користувача в системі;

– при редагування розкладу повинні виводитись списки із дисциплінами і в залежності від вибору дисципліни, є мобливість обрати вчителя, який її викладає, це не дозволяє ввід невірної інформації;

– при додаванні/редагуванні групи повинен відображатися список із користувачами, які можуть бути кураторами, тобто повинна здійснюватись фільтрація при виводі таких даних;

– повинна бути присутня валідація, при цьому повідомлення про невідповідність даних повинні відображатися зрозуміло, щоб у користувачів не виникали додаткові питання;

– вся інформація, представлена на сайті, повинна бути зрозумілою та читабельною.

Отже, постановка задачі полягає в тому, щоб розробити якісну і в той же час просту та зрозумілу систему організації учбового процесу. Забезпечувати той набір функціоналу, що дозволить здійснювати процес навчання за допопмогою даної системи.

Потрібно передбачити декілька можливих користувачів системи, а саме:

– учень;

– вчитель;

– батьки;

– адміністратор;

– куратор.

Відповідно кожній групі користувачів доступний різний функціонал. Для того, щоб здійснювати діяльність за певною роллю, необхідно авторизуватися.

Функціонал, доступний кожному з користувачів, і те як вони можуть взаємодіяти із системою зображено за допомогою діаграми варіантів використання, яка знаходиться у додатку Б.

Суть діаграми варіантів використання полягає в тому, що проектована система подається у вигляді множини сутностей чи акторів, які взаємодіють із системою за допомогою, так званих, варіантів використання.

Варіант використання призначений для опису послуг, які система надає актору. Іншими словами, кожен варіант використання визначає деякий набір дій, який виконує система під час діалогу з актором. При цьому нічого не говориться про те, яким чином буде реалізовано взаємодію акторів із системою.

Оскільки акторів декілька, було спроектовано діаграму варіантів використання для кожного окремо. Так можна найбільш вдало показати взаємодію із системою користувачів однієї ролі.

Коротко описано акторів та їх дії у таблиці 1.1.

Таблиця 1.1 - Взаємодія акторів із системою

Актор	Опис
Учень	Може взаємодіяти із системою, переглядаючи ту інформацію, що доступна для нього, а саме: щоденник, табелі, чек-ліст із домашніми завданнями, шкільні заходи. При цьому редагування недоступне. Є можливість робити тільки помітки у чек-лісті.
Вчитель	Може взаємодіяти з системою, працюючи із журналом групи, оцінками учнів, також є змога додавати домашнє завдання. Також як і всі інші користувачі, має змогу переглядати шкільні заходи.

Таблиця 1.1 - Продовження

Актор	Опис
Батьки	Батьки мають схожі функції в системі, що і учні. Їм доступна інформація про їхніх дітей.
Адміністратор	Взаємодіє із системою шляхом виконання різноманітних функцій, зокрема: редагування даних про шкільні заходи; редагування, видалення та додавання груп, та студентів до них. Також відповідає за розклад, тобто є можливість внесення змін чи додавання у разі відсутності
Класний керівник	Взаємодіє із системою, працюючи із інформацією про власну групу. Може формувати табель, додавати батьків учнів і т.д.

Враховуючи вище-перераховані вимоги та набір функціоналу було сформовано технічне завдання у додатку А.

В ході першого розділу було досліджено та проаналізовано предметну область. Визначено основні переваги та недоліки сфери інтернет-обслуговування. Проведено аналіз існуючих подібних розробок, що мають як переваги, так і недоліки. На основі вивчених даних було сформовано технічне завдання, та визначено ряд вимог: фунціональні, нефункціональні, до документації, надійності, до апаратного забезпечення.

2 ПРОЕКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Аналіз та проектування архітектури системи

Розглянемо існуючі архітектурні рішення, за допомогою яких буде проводитись розробка сайту.

Першим типом є «клієнт-серверна» архітектура. Станом на сьогодні ця архітектура є найбільш поширеною, і вона використовується, в більшості працюючих інформаційних систем [3].

Переважну більшість інших архітектур можна інтерпретувати, як одну з варіацій «клієнт-сервера».

В основному розумінні система, що розроблена за типом архітектури «клієнт-сервер», є сукупністю взаємодіючих компонент двох типів, тобто – клієнтів і серверів. Стандартним також являється рознесення цих компонент по двох типів вузлах – відповідно: вузлів-клієнтів і вузлів-серверів. Клієнт надсилає запити до сервера, сервер їх обробляє та надає відповідь.

«Модель сервісу» – така архітектура, в якій на один запит відповідає декілька серверів, що в розумінні клієнта є єдиним цілим. Тобто, така модель реалізовується так само, як і «клієнт-сервером», проте в ній компонент сервер має складну структуру, тобто не монолітну. Така модель широко використовується у сервісах, що є критично важливими, для яких зупинка обслуговування недопустима ні на секунду [3].

Прикладом системи, розробленої за архітектурою «підключення через proxy», є стандартна система з браузером, proxy-сервера і веб-сервера. В даному випадку відмінність від інших архітектур полягає в тому, що клієнт з'єднується не напряму з сервером, а з певним проміжним елементом. Цей компонент вже передає запит серверу від імені клієнта.

У архітектурі «сервер ініціює з'єднання» – з’єднання ініціює саме сервер, «пропихаючи» інформацію до клієнта. В такій системі участь клієнта полягає в реакції (перегляду) на повідомлення від сервера. Прикладом є взаємодія «по

підписці». Тобто, сервер отримує певний набір повідомлень із зовнішнього джерела. Повідомлення мають певний тип. Клієнт повідомляє серверу, що в нього є зацікавленість в отриманні повідомлення конкретного типу і повідомляє про цю зацікавленість серверу. Після того як сервер отримає певні події, він передає їх всім зацікавленим клієнтам. Даний алгоритм є спрощеним описом роботи часто застосовуваної служби подій.

В даному випадку було обрано архітектуру «клієнт-сервер», оскільки такий варіант забезпечить найбільш доцільне використання функціоналу, що надає дана система організації учбового процесу.

Сервером виступить комп'ютер, який містить бази даних, СКБД та пов'язане з ними програмне забезпечення, і налаштований на надання користувачам інформаційної системи доступу до бази даних. Клієнти, які працюють із даними (вони можуть бути розташовані на різних комп'ютерах мережі), надсилають певні запити на сервер. В свою чергу сервер їх отримує, опрацьовує, та надсилає відповідь клієнту на його запит. Дану архітектуру зображено на діаграмі розгортання у додатку В.

Доцільно будувати проект за архітектурою MVC. Вона дозволить поділити код програми на 3 частини: Модель (Model), Вид або Представлення (View) і Контролер (Controller) [4].

Поділ на частини надає можливість значно спростити великий за обсягом код. Якщо код писати одним довгим скриптом, в ньому завжди буде важко розібратися, і важко змінювати. В такому випадку досить легко допустити помилки.

Модель – це та сутність, яка описує всю логіку додатку, вона дозволяє зберігати та обробляти інформацію, при цьому не взаємодіючи з користувачем напряму (звернутися до Моделі можна тільки з коду, викликаючи її функції).

До прикладу можна взяти збереження інформації в БД, перевірка правильності введених даних – це те, чим займається Модель, а вже отримання цих даних від користувача або відображення інформації на екрані чи, наприклад, обробка

натискання на кнопку – ні. Можна сказати, що моделі – це своєрідні класи [4].

У даному проекті передбачається створення таких основних моделей:

– групи;

– вчителя;

– журналу;

– розкладу;

– учня;

– шкільного заходу;

– щоденника;

– батьків;

– повідомлення;

– домашнього завдання;

– оцінки;

– табеля та ін.

Проте, не завжди можна обійтися тільки тими моделями, що відповідають таблицям у базі даних. Тому, крім вищеперерахованих та пов’язаних із ними моделями, передбачається також додавання допоміжних моделей, які призначені для коректної передачі необхідних даних у представлення.

Представлення дозволяє відобразити дані на сторінці, які йому передали. У веб-додатку воно зазвичай складається з HTML-сторінок. Представлення – це код, який відповідає за виведення інформації на екран, зображення всіх елементів інтерфейсу.

Контролер приймає запити, що надійшли від користувача, після чого обробляє їх та повертає результат. У веб-додатку зазвичай контролер розбирає параметри HTTP-запиту з $ _POST / $ _ GET, звертається до моделі, щоб отримати або змінити якісь дані, і в кінці викликає представлення, щоб відобразити результат виконання запиту. Тобто, контролер встановлює зв’язок між моделлю та представленням [4].

Передбачається створення декількох контролерів, що в свою чергу описують

певний модуль програми – Diary, Schedule, Account, Activity, Journal, Group, Parent, Timesheet, Subject, Teacher, Student, Notification.

Контролер Group відповідає за роботу, що пов’язана із управлінням групами учнів. Всі дії з групами доступні тільки зареєстрованим користувачам, при цьому редагування, видалення студентів із групи дозволено тільки адміністраторам.

Контролер Account відповідає за особисті дані користувачів. Його основні методи – це реєстрація, авторизація та вихід з сайту. Авторизація та реєстрація можлива тільки для не авторизованого користувача.

Контролер Student, Teacher, Parent – це контролери, які відповідають за додавання, видалення чи редагування учнів, вчителів та батьків відповідно. Всі ці дії доступні тільки для адміністратора сайту.

Контролер Schedule відповідає за дії, що пов’язані з розкладом. Для кожної групи він є індивідуальним. За його заповнення чи редагування відповідає адміністратор сайту.

Контролер Diary включає в себе дії, що пов’язані із щоденником, при цьому щоденник формується динамічно на основі розкладу групи. Перегляд щоденника доступний для учня, його батьків та вчителів.

Контролер Journal відповідає за журнали груп. Кожен журнал формується динамічно. При цьому він відповідає також за додавання оцінок та домашнього завдання. Доступний він є для вчителів та класних керівників групи.

Контролер Timesheet дозволяє формувати табель учнів. Після цього він є доступний для учня. Формувати табель може класний керівник.

Контролер Activity відповідає за роботу, що пов’язана із шкільними заходами. Додавати їх може адміністратор, а переглядати – всі користувачі.

Контролер Notification надає можливість для створення повідомлень та надсилання їх певним групам користувачів. Надсилати повідомлення можуть вчителі учням та батькам.

Послідовність взаємодії різних користувачів з модулями зображено на діаграмі послідовності у додатку Г.

2.2 Аналіз та вибір типу бази даних, проектування структури бази даних

При розробці даного програмного забезпечення постало питання у виборі бази даних, де буде зберігатися всі інформація, що необхідна для коректної роботи системи.

База даних – це інтегрована сукупність структурованих і взаємозалежних даних, організована за певними правилами, які передбачають загальні принципи опису, зберігання і обробки даних.

При цьому бази даних можуть бути двох типів, а саме реляційна та нереляційна. Зважаючи на принцип організації даних в системі було прийнято рішення, що кращим варіантом все ж буде реляційна база даних.

Реляційна база даних – це тип бази даних, що зберігає інформацію в електронних таблицях і здійснює пошук даних в одній таблиці на підставі визначених ключових полів іншої таблиці [5].

Було обрано СКБД – SQL Server.

При проектуванні логічної моделі потрібно спроектувати можливі таблиці та зв’язки між ними. Тому передбачаються такі основні таблиці:

– таблиця користувача – де міститься інформація, що записується при реєстрації користувача в системі;

– таблиця учнів – де розміщена вся основна інформація про учнів класів;

– таблиця вчителів – де міститься інформація про вчителів;

– таблиця розкладу – це сполучна таблиця між групою та набором днів розкладу;

– таблиця дня розкладу – сполучна таблиця між днем та набором записів розкладу;

– таблиця запису розкладу – містить інформацію про записи в розкладі;

– таблиця шкільних заходів – де розміщено інформацію про заходи школи (різні шкільна події) із вказанням місця, часу, опису та інших деталей;

– таблиця ролей – зберігається інформація про ролі користувачів, за якими визначається доступ до окремих функцій;

– таблиця батьків – зберігає інформацію батьків, а також їх дітей;

– таблиця повідомлень – містяться повідомлення відправлені вчителями іншим користувачам;

– таблиця оцінок – де розміщуються всі оцінки учнів, що були додані;

– таблиця стану виконання домашніх робіт – вказується інформація про те, які домашні роботи були виконані конкретним студентом;

– таблиця домашніх робіт – містить домашні роботи груп.

При побудові таблиць бази даних та зв’язків між ними необхідно дотримуватись принципів нормалізації [5]:

– в одній таблиці не повинно бути повторюваних полів;

– необхідно, щоб кожна таблиця містила унікальний ідентифікатор, тобто первинний ключ;

– кожен запис в таблиці повинен містити достатньо інформації про тип суті;

– зміна значень в полях таблиці не має впливати на значення в інших полях (крім змін у полях ключа).

Також необхідно спроектувати зв’язки між таблицями. Є наступні види зв’язків:

– один-до-одного – кожному запису з однієї таблиці відповідає один запис у іншій таблиці;

– один-до-багатьох – кожному запису з однієї таблиці відповідає кілька записів у іншій таблиці;

– багато-до-одного – безлічі записів з однієї таблиці відповідає один запис у іншій таблиці;

– багато-до-багатьох – безлічі записів з однієї таблиці відповідає кілька записів в іншій таблиці.

Зв’язки між деякими таблицями бази даних зображено у таблиці 2.1.

Таблиця 2.1 - Зв'язки між таблицями бази даних

Назва таблиці	Назва таблиці	Зв’язок
Користувач	Вчитель	один-до-багатьох
Користувач	Учень	один-до-багатьох
Користувач	Батьки	один-до-багатьох
Батьки	Учні	багато-до-багатьох
Група	Учні	один-до-багатьох
Група	Вчитель(куратор)	один-до-одного
Група	Розклад	один-до-одного
Оцінка	Учень	багато-до-одного

Після побудови логічної моделі бази даних, можна приступати до побудови фізичної моделі бази даних.

Фізична модель визначає розміщення даних у зовнішній пам’яті. Вона ще називається внутрішньою моделлю системи і форма її представлення залежить від обраної СКБД. Так як, була обрана СКБД, яка підтримує реляційну модель даних, то треба таблиці разом з атрибутами і зв’язки між таблицями перенести в середовище СКБД з врахуванням вимог до відповідних об’єктів БД [6].

Тому було дотримано таких пунктів:

– ідентифікатори таблиць і полів мають задовольняти вимогам СКБД;

– типи даних, розміри полів, обмеження мають бути приведені у відповідність до прийнятих у даній СКБД;

– визначено стратегії індексування, а також взаємозв’язки між таблицями, первинні та зовнішні ключі.

На етапі фізичного проектування слід приділити особливу увагу забезпеченню цілісності БД. У СКБД цілісність даних забезпечується обмеженнями цілісності, тобто набором правил, що встановлюють допустимість даних та зв’язків між ними. Схему бази даних дивитись у додатку Д.

2.3 Проектування інтерфейсу користувача

Користувацький інтерфейс – це своєрідний комунікаційний канал, по якому здійснюється взаємодія між користувачем і комп’ютером.

Користувацький інтерфейс повинен бути таким, щоб користувач не приділяв занадто багато уваги, він практично не повинен відчувати його. Користувач не повинен розмірковувати, яку кнопку натиснути, де знайти певний функціонал або де клацнути мишею. Такий інтерфейс є прозорим – користувач ніби дивиться крізь нього на свою роботу. Саме такий принцип при побудові інтерфейсу забезпечить найефективнішу роботу користувача на сайті, тому саме цей механізм є провідним при розробці даного програмного забезпечення .

Кожна сторінка повинна відповідати певному шаблону, будуватися за певними правилами. Це дозволить користувачу не витрачати час на пошук певних елементів, адже все буде інтуїтивно зозуміло, зважаючи на попередні сторінки. Контент слід розмістити за певними правилами. Ці правила можна зобразити за допомогою макету, який зображено на рисунку 2.1.

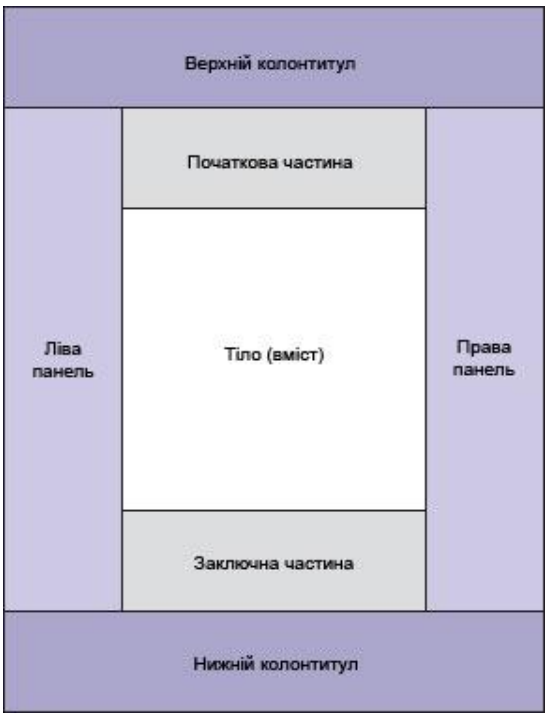


Рисунок 2.1 – Шаблон побудови сторінок у веб-застосунках

Весь контент сторінки знаходиться у тілі сторінки, що показано на вище-продемонстрованому макеті. Тіло кожної сторінки є динамічною частиною, на відміну від колонтитулів, початкової та заключної частини.

Для зручного переходу між сторінками має бути меню, що є на кожній сторінці, яке знаходиться у верхньому колонтитулі і воно є сталим.

Основні сторінки які складатимуть сайт це:

– сторінка розкладу;
– сторінка журналу;
– сторінка переліку груп;
– сторінка щоденника;
– сторінка переліку шкільних заходів;
– сторінка із табелем учня;
– сторінка чек-ліст із домашніми завданнями учня;
– сторінка із повідомленнями та ін.

Макет розкладу класу зображено на рисунку 2.2.

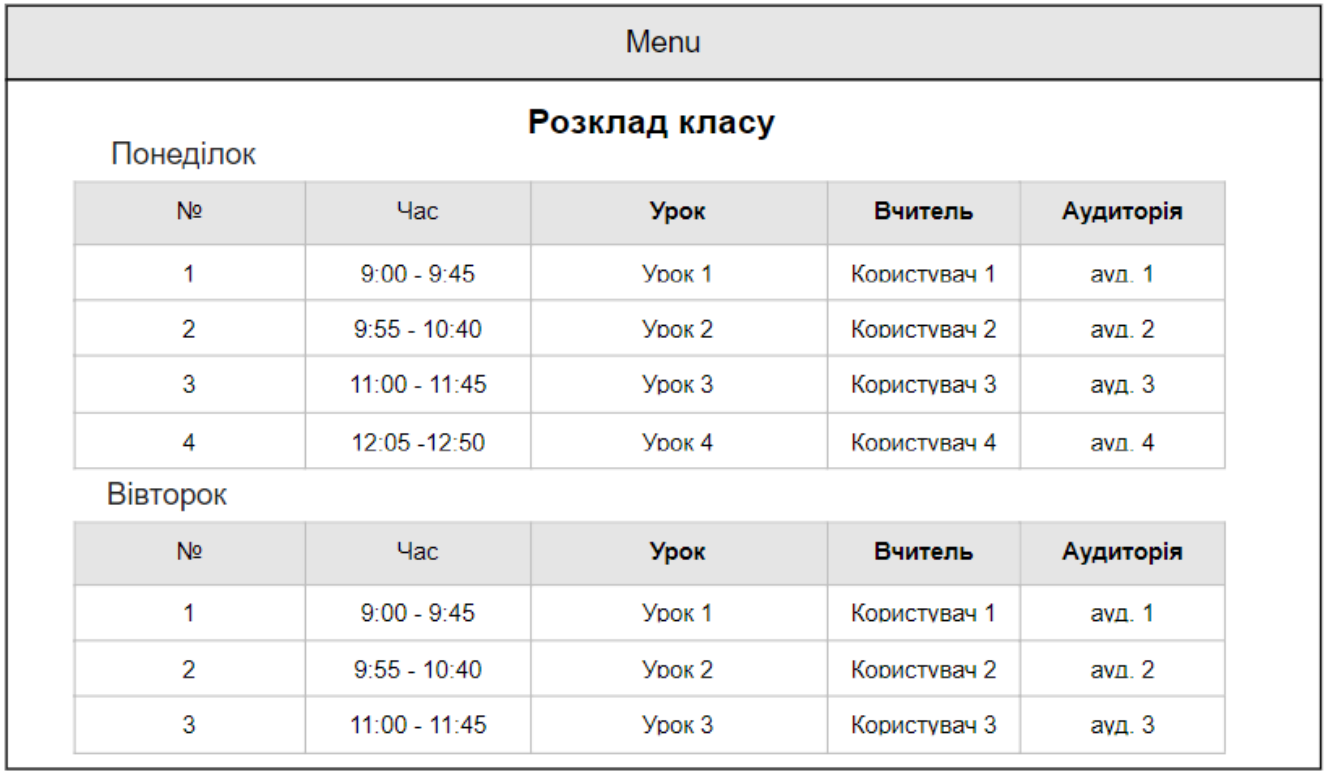


Рисунок 2.2 – Макет сторінки "Розклад класу"

Передбачено, що на сайті буде виводитись інформація про шкільні заходи. Макет цієї сторінки зображено на рисунку 2.3.

Рисунок 1.3 – Макет сторінки "Шкільні заходи"

Для учнів система передбачає формування щоденника із вказанням домашніх завдань та оцінок. Макет сторінки зі щоденником зображено на рисунку 2.4.

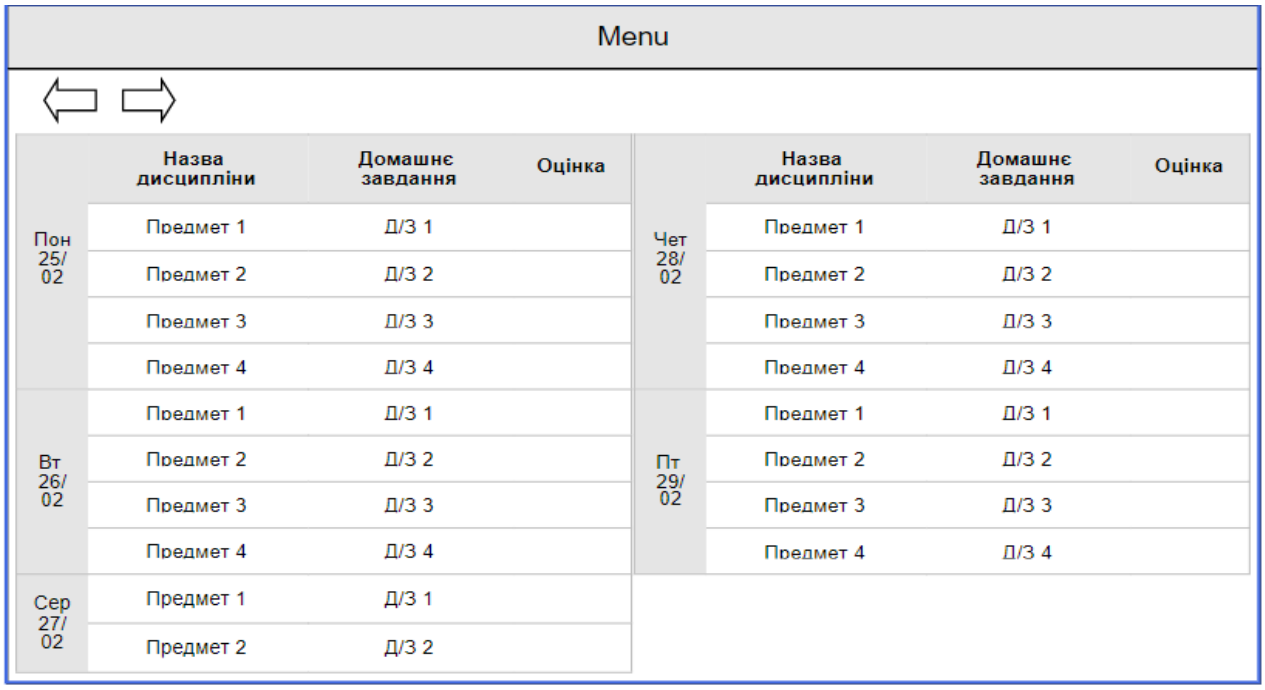


Рисунок 2.4 – Макет сторінки щоденника учня

Для вчителів буде створена сторінка із журналом, де також опрацьовуються оцінки та домашнє завдання. Макет цієї сторінки зображено на рисунку 2.5.

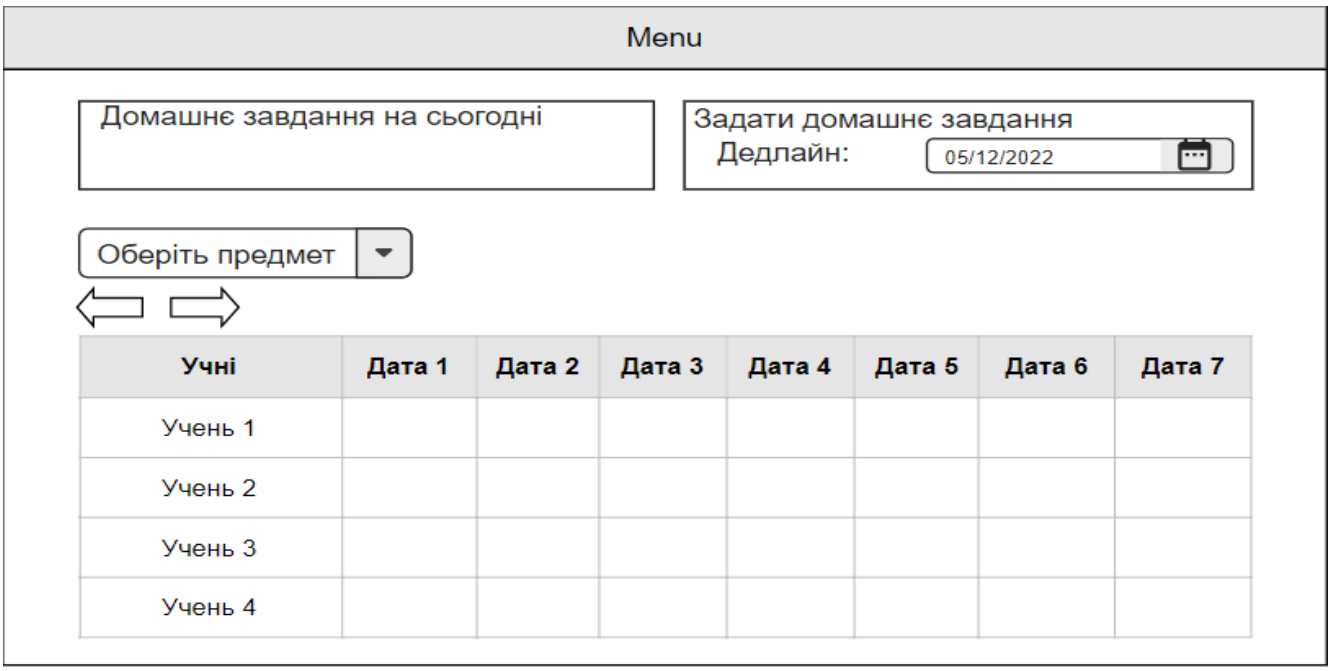


Рисунок 2.5 – Макет сторінки журналу

Для роботи із класами буде створено сторінку з виведенням всіх класів, з можливістю переглядати список учнів, журналу та розкладу. Макет сторінки зображено на рисунку 2.6.

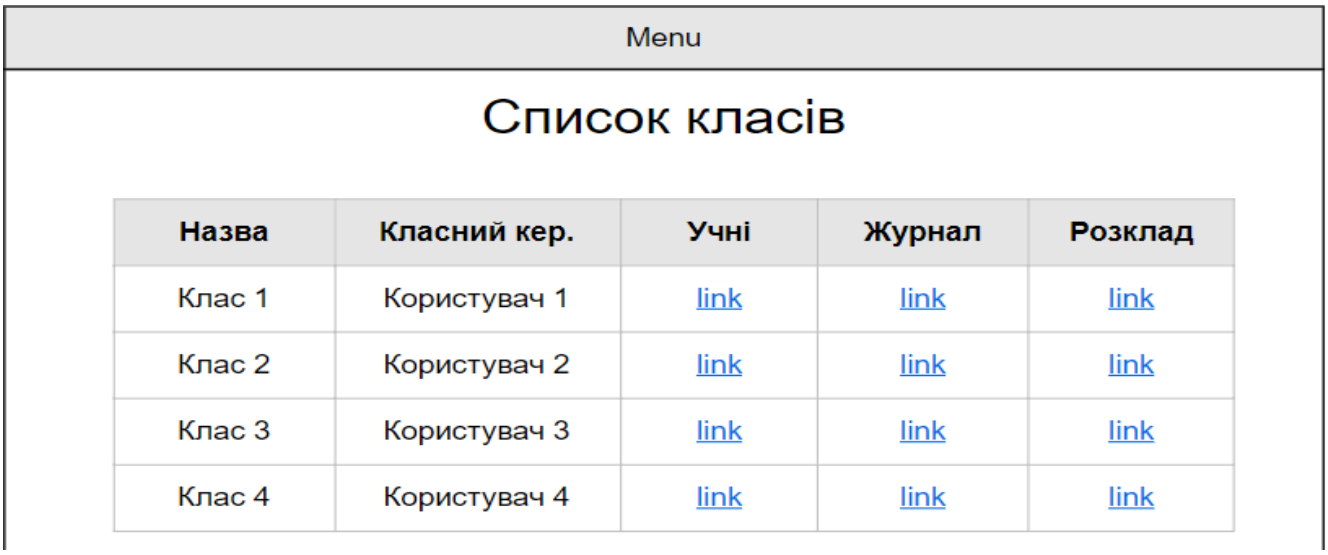


Рисунок 2.6 – Макет сторінки зі списком класів

Учню також буде доступна сторінка із його табелями. Макет сторінки із табелем зображено на рисунку 2.7.

Рисунок 2.7 – Макет сторінки із табелем

Також учню доступний функціонал із переглядом домашніх завдань та можливість відмічати як виконані, так і не виконані роботи. Макет цієї сторінки зображено на рис. 2.8.

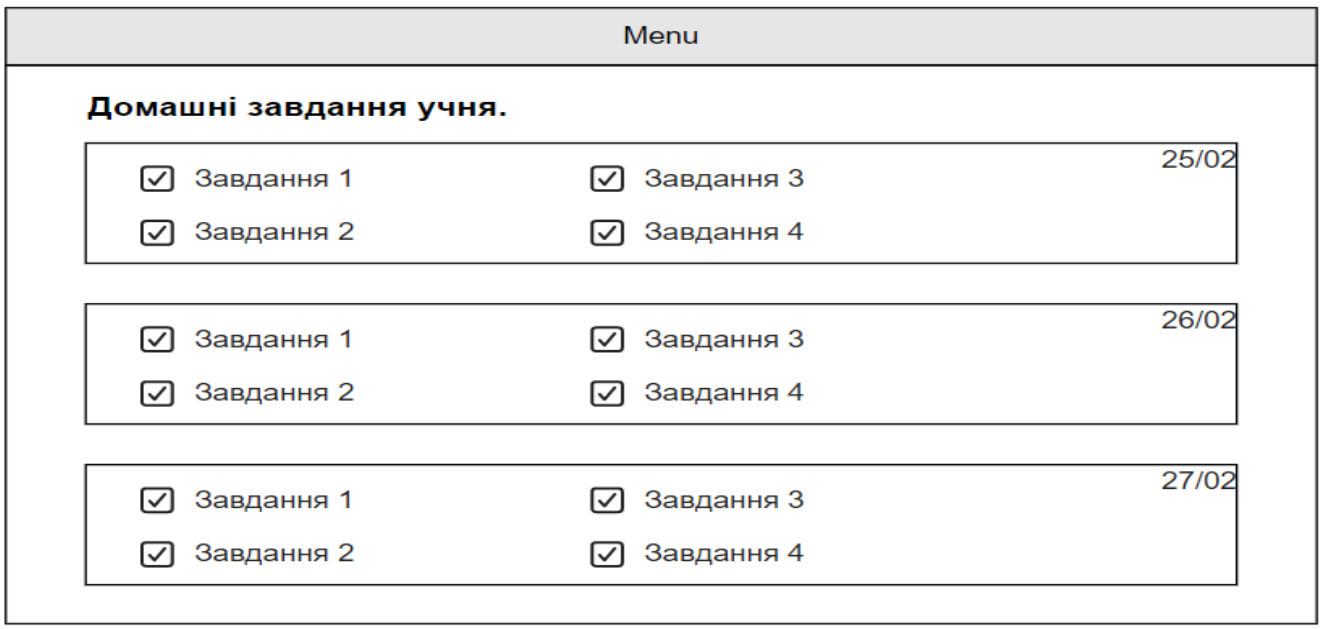


Рисунок 2.8 – Макет сторінки із чек-лістом домашніх завдань

Користувач може переглядати повідомлення. Макет сторінки із повідомленнями зображено на рисунку 2.9.

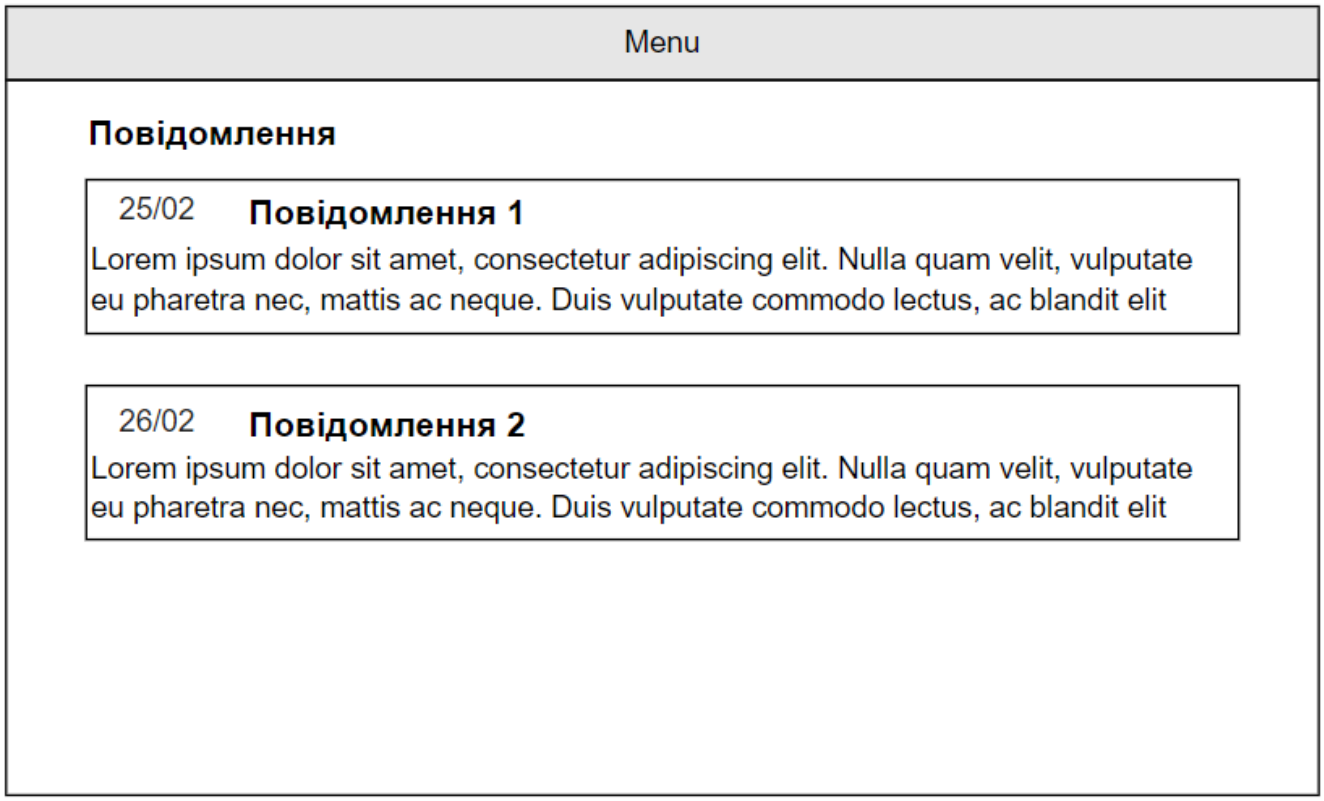


Рисунок 2.9 – Макет сторінки із повідомленнями

2.4 Аналіз та вибір технологій і методів реалізації системи

Після етапу проектування архітектури, в даному проекті це архітектура «клієнт-сервер», наступний крок при побудові сайту для інтернет-магазину є вибір мови програмування та фреймворку, за допомогою яких буде розроблятися програмне забезпечення.

Так як передбачається створення веб-ресурсу, потрібно вибрати мову програмування, яка є зручною і пристосованою для побудови сайтів. Є безліч різних мов, які дозволяють це зробити, а саме Java, Python, Ruby, PHP, C#.

Java – одна з провідних мов програмування для кросплатформенних-програм. Java-програмісти можуть розробляти застосунок на комп'ютері, а потім працювати із ним на цільовій платформі – телефоні, сервері тощо. Також тут є

доступ до великої кількості різноманітних платформ.

Пайтон (Python) – це досить проста при розробці мова програмування, і нею легко оволодіти. Використання цієї мови програмування дозволить розробляти прикладні програми у багатьох галузях на більшості платформ.

Ruby – це інтерпретована, повністю об'єктно-орієнтована мова програмування, що має чітку динамічну типізацію. Мова має високу ефективність при розробці програм. Багатоплатформова реалізація інтерпретатора мови Ruby розповсюджується як вільне програмне забезпечення.

PHP – скриптова мова програмування, що була створена для генерації HTML-сторінок на стороні веб-сервера. PHP досить часто застосовується при розробці веб-застосунків [7].

C# – це об’єктно-орієнтована мова програмування, яка характеризується постійним розвитком та вдосконаленням. Оскільки дана мова програмування створена і постійно супроводжується компанією Microsoft, вона постійно оновлюється, розширюється відповідно до сучасних технологій і з впевненістю можна сказати, що її розвиток продовжиться і в майбутньому. Мова є строготипізована і краще опановується початківцями. C# має досить простий і зрозумілий синтаксис, при цьому вона є гнучкою, і дуже часто використовується для створення масштабних проектів. За допомогою цієї мови програмування написано багато бібліотек, які застосовується при розробках іншими мовами, інтерпретатори багатьох сучасніших мов програмування, що свідчить про широкі можливості цієї мови [7].

Після аналізу найбільш поширених мов програмування для веб-розробок, було прийнято рішення про написання коду на мові програмування C#.

При розробці сайту досить важливим інструментом є фреймворк – це комплекс програмних рішень, що готовий до застосування, включаючи дизайн, логіку та базову функціональність системи або підсистеми. Фреймворк – це, можна сказати, каркас, фундамент або набір групи інструментів для розробки різнопланових сайтів і web-застосунків. Існує безліч фреймворків, які дозволяють швидше

розробити сайт і правильно оформити його. Після того, як було обрано мову програмування, кількість фреймворків зменшилась, оскільки потрібно обирати той, який працює із C#.

Для створення веб-застунків можливо використовувати фреймворк Asp.Net MVC або Asp.Net Core. Ці платформи мають ряд певних відмінностей, можна сказати, що Asp.Net Core має певні перевги над Asp.Net MVC.

ASP.Net Core – це платформа, з допомогою якої, можлива розробка будь-яких додатків для будь-яких операційних систем, таких як Windows, Linux, macOS і т.д., тобто даний фреймворк забезпечує кросплатформенність.

Порівняно з Asp.Net MVC у Asp.Net Core була змінена структура проекту. Було проведено зміни файлів конфігурації, а також додано папку з іменем wwwroot. В ній знаходяться усі статичні файли, які необхідно відправити у браузер, наприклад – css, html, images, JavaScripts тощо.

У Asp.Net Core дуже зручно взаємодіяти з базою даних за допомогою Entity Framework.

Даний фреймворк заснований на компонентній структурі і реалізує парадигму MVC (Model View Controller). Архітектура MVC – схема поділу даних програми, інтерфейсу і керуючої логіки на три окремих компоненти: модель, представлення і контролер – таким чином, що модифікація кожного компонента може здійснюватися незалежно [4].

Модель (Model) надає дані і реагує на команди контролера, змінюючи свій стан .

Представлення (View) відповідає за відображення даних моделі користувачеві, реагуючи на зміни моделі .

Контролер (Controller) інтерпретує дії користувача, сповіщаючи модель про необхідність змін [4] .

Razor Pages – це важливий і гнучкий елемент ASP.NET Core, який дозволяє зробити програмні сценарії, засновані на веб-сторінках, більш продуктивними. З технічної точки зору, Razor Pages – це модель кодування, заснована на веб-

сторінках, яка значно спростовує створення веб-інтерфейсу. З допомогою цього інструменту спрощується взаємодія з представленнями, оскільки в будь-якому місці html-сторінки можна застосувати код C#.

Ще одна цікава особливість Asp.net Core – інтеграція jQuery, що дозволяє зробити інтерфейс значно динамічнішим та гнучкішим.

Інтерфейс сайту є однією з основних складових зацікавленості користувача в роботі із системою. Він повинен бути простим, зручним та адаптивним. Для цього доцільно використовувати CSS та Bootstrap. Перший дозволить зробити хороший дизайн сайту разом з використанням мови Javascript, яка надасть кращий візуальний ефект, з додаванням анімації, слайдерів і т.п.

CSS (Cascading Style Sheets – каскадні таблиці стилів) – одна з базових технологій, що застосовується при розробці сайтів. Вона має місце майже у всіх веб-застосунках. CSS-код – це список інструкцій для браузера, – яким чином і де відображати елементи веб-сторінки. CSS використовується авторами та відвідувачами веб-сторінок, щоб визначити кольори, шрифти, верстку та інші аспекти вигляду сторінки. Одна з головних переваг — можливість розділити зміст сторінки (або контент, наповнення, зазвичай HTML, XML або подібна мова розмітки) від вигляду документу (що описується в CSS).

Bootstrap – вільний набір інструментів для створення сайтів і веб-додатків . Включає в себе HTML – і CSS -шаблони оформлення для веб-форм, кнопок, міток, блоків навігації та інших компонентів веб-інтерфейсу. Він дозволить розробити такий сайт, що буде адаптивним для різних екранів.

Отже після розгляду існуючих методів рішень було визначено, що сайт буде написаний на мові програмування C# з використанням Asp.Net Core фреймворку.

При побудові інтерфейсу сайту буде використовуватись CSS, для оформлення сторінки, розміщення інформації, задання різних стилів. Для адаптивності сторінки будуть застосовуватись класи Bootstrap [8].

Для динамічного функціонування HTML сторінки передбачається робота з javascript та бібліотекою jquery. JavaScript – це мова програмування, яка

застосовується до HTML документу, і може забезпечити динамічну інтерактивність на веб-сервісах. Вона є досить універсальною і гнучкою.

Попри основні засоби мови JavaScript є величезна кількість додаткових функцій. Багато з них буде застосовувано у даній системі, а саме програмні інтерфейси додатка (API), вбудовані в браузери, що забезпечують різноманітні функціональні можливості, такі як динамічне створення HTML і встановлення CSS стилів, а також додаткові бібліотеки, які дозволяють пришвидшити створення сайту, і додати більше функціональних можливостей.

Одна з функцій, яку JavaScript повинна забезпечувати на даному сайті – це зміна поведінки при натисканні кнопки, так як функціонал мови C# при натиснені кнопки надсилає дані на сервер і перезапускає сторінку, що інколи є проблемним і потребує зміни поведінки. Проте, якщо необхідно дані надіслати без перезавантаження сторінки, існує так званий AJAX-запит – це підхід до побудови інтерактивних користувацьких інтерфейсів веб-додатків, що полягає в «фоновому» обміні даними браузера з веб-сервером . В результаті, при оновленні даних веб-сторінка не перезапускається повністю, можлива заміна певної частини сторінки, і сайт стає швидшим і зручнішим.

Для більшої зрозумілості можна порівняти дії, які виконуються у класичній моделі веб додатків та при використанні таких запитів [9].

У класичній моделі:

– користувач заходить на веб-сторінку і натискає на який-небудь її елемент;

– браузер формує і відправляє запит на сервер;

– у відповідь сервер генерує абсолютно нову веб-сторінку і відправляє її браузеру , після чого браузер повністю перезавантажує всю сторінку.

При використанні AJAX:

– користувач заходить на веб-сторінку і натискає на який-небудь її елемент;

– скрипт (на мові JavaScript) дозволяє визначити, яка інформація необхідна для оновлення сторінки;

– браузер відправляє відповідний запит на сервер ;

– сервер повертає лише ту частину документа, на яку прийшов запит;

– скрипт вносить заміни з урахуванням отриманої інформації (без повного перезавантаження сторінки).

Отже даний інструмент є досить зручним і важливим.

У другому розділі було спроектовано всі основні компоненти системи, включаючи базу даних, архітектуру, інтерфейс. Першим кроком стало визначення архітектури, оскільки наступні етапи будуються за цими принципами. Даний веб-ресурс спроектовано за клієнт-серверною архітектурою з використанням MVC, що призведе до розділення проекту на 3 компоненти, і це значно полегшить розробку та збільшить читабельність коду. Щодо інтерфейсу, то було спроектовано макети основних сторінок, які далі стануть основою їх розробки. Також на етапі прийнято рішення користуватися мовою програмування C# та фреймворком ASP.Net Core.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Структура програмного забезпечення

Дане програмне забезпечення для організації учбового процесу у школах побудовано за шаблоном MVC, що означає поділ програми не тільки на моделі, контролери та представлення, а й на певні блоки. В системі є декілька контролерів, кожен з них відповідає за певну область розробки, що в свою чергу виводиться користувачу певною кількістю представлень. Тому доцільно виділити та охарактеризувати контролери кожен з яких відповідає за свій набір функціоналу.

AccountController – відповідає за реєстрацію, авторизацію користувача в системі. Ці операції здійснюються за допомогою бібліотеки ASP.NET Core Identity.

ASP.NET Core Identity – це API від Microsoft, який призначений для реалізації логіки управління користувачами в веб-додатку на Core.

RoleController призначений для управління ролями у системі. Як вже було сказано вище, ролі потрібні для розмежування дій та обов'язків між різними користувачами. Даний модуль також функціонує на основі ASP.NET Core Identity. Саме даний механізм забезпечує різноплановий доступ до дій різних контроллерів та розмежовує адміністративну та користувацьку частини.

ScheduleController відповідає за формування та обробку розкладу системи. Кожній групі відповідає свій розклад.

Для додавання розкладу було створенно дві дії: GET, яка відповідає за вивід представлення із вікном додавання розкладу, та Post, що отримує дані з представлення та записує дані у базу даних. Оскільки модель розкладу є складною, воно складається з днів розкладу, а ті у свою чергу з записів розкладу, то і додавання даних в базу є дещо ускладненим. В даному випадку відбувається додавання запису наявного розкладу. Якщо це перший запис, то додається і сам розклад.

DiaryController відповідає за роботу із щоденниками учнів. В даному випадку немає потреби робити POST дії, оскільки щоденник не додається користувачами.

Він формується автоматично на основі створеного раніше розкладу, оцінок та домашніх робіт. Формується спеціальна модель виведення, у якій формуються всі ці дані, після чого вони у формі таблиці щоденника виводяться у представленні. Дії контролера доступні користувачам системи із роллю батьків або учня.

JournalController відповідає за всі дії, що пов’язані із журналами груп. Журнал також не додається окремо адміністратором. Він формується на основі розкладу. Тому тут присутні тільки GET запити. Було додано модель представлення для коректного виведення журналу у такому вигляді, який ми звикли бачити не в електронному вигляді. Також в цьому ж контролері відбувається додавання оцінок в базу даних, оскільки функціоналу пов'язаного з ними практично немає і недоцільно створювати новий контролер.

ParentController, TeacherController та StudentController – всі вони призначені для додавання, редагування, та видалення батьків, вчителів та студентів відповідно. Тут присутні як і GET дії, що виводять представлення із певним набором даних, так і POST, що роблять зміни в базі даних.

TimesheetController – відповідає за створення та додавання табелів учнів. Створення відбувається також динамічно, запитом із бази даних дістаються всі предмети групи та всі оцінки учнів цієї групи, і для кожного учня обчислюється середня оцінка, що і записується в таблицю табелю. За допомогою GET дії, здійснюється запит у базу даних, дістаються дані про всі табелі учнів та виводяться у представлення.

ActivityController призначений для управління шкільними заходами. Також відбувається їх додавання адміністратором, після чого відбувається виведення. Це єдинна дія в системі, що доступна всім користувачам.

HomeworkController відповідальний за створення домашніх завдань з предметів, виведення у журнал та виведення чек-ліста для учня.

Взаємодію різних користувачів із системою зображено на діаграмі активностей у додатку Е.

3.2 Реалізація БД

Під час проектування системи було вирішено, що база даних побудується за принципом CodeFirst, оскільки передбачається використання EntityFramework. Це є дуже зручно оскільки не потрібно кожного разу створювати нову таблицю в базі даних, при цьому порівнювати її із класом, що за неї відповідає, щоб вони були ідентичні, і не виникало проблем із додаванням нових даних.

Для створення бази даних, необхідно створити класи, оскільки використовується MVC, то ці класи доцільно називати моделями. Кожна модель повинна містити ключ та набір даних, що найбільш точно описують сутність.

Наприклад модель користувача:

```
public class User
    {
        [Key]
        public int Id { get; set; }
        public string Email { get; set; }
        public string Password { get; set; }
        public string FirstName { get; set; }
        public string LastName { get; set; }
        public string Surname { get; set; }
        public List<Role> Role { get; set; }
        public string Phone { get; set; }
    }
```

Після того як модель було додано, необхідно вказати її в класі що наслідується від DBContext:

```
public class AppDBContext : DbContext
    {
     public DbSet<User> Users { get; set; }

    }
```

Цей клас як раз і відповідає за те, які таблиці повинні бути в базі даних. Шлях до бази даних необхідно вказати у конфігураційному файлі.

Після того, як все це було вказано необхідно створити базу даних. Для цього було вирішено використовувати такий інструмент, як міграції баз даних. Суть міграцій полягає в тому, що всі зміни, що були додані раніше зберігаються в базі

даних. І при редагуванні якихось моделей, видалення чи створення нових, при створенні нової міграції, порівнюються зміни із попередніми міграціями, якщо це щось нове, то база даних оновлюється. Це дуже зручно, оскільки знову ж таки не потрібно змінювати саму базу даних, достатньо лише прописати зміни в моделях, це значно пришвидшує розробку продукту.

В результаті таблиця User зображена на рисунку 3.1.

	Имя столбца	Тип данных	Разрешить значения NULL
🔑	Id	int	☐
	Email	nvarchar(MAX)	☐
	Password	nvarchar(MAX)	☐
	FirstName	nvarchar(MAX)	☐
	LastName	nvarchar(MAX)	☐
	Surname	nvarchar(MAX)	☐
	Phone	nvarchar(MAX)	☐

Рисунок 3.1 – Вигляд таблиці Users в базі даних

Таблиця класу зображена на рисунку 3.2.

	Имя столбца	Тип данных	Разрешить значения NULL
🔑	Id	int	☐
	Name	nvarchar(MAX)	☐
	CreatedDateTime	datetime2(7)	☐
	TeacherId	int	☐

Рисунок 3.2 – Вигляд таблиці Groups в базі даних

Таблиця записів розкладу зображена на рисунку 3.3.

	Имя столбца	Тип данных	Разрешить значения NULL
🔑	Id	int	☐
	TeacherId	int	☐
	SubjectId	int	☐
	LessonTimeFrameId	int	☐
	ScheduleDayId	int	☐
	Classroom	nvarchar(MAX)	☐

Рисунок 3.3 – Вигляд таблиці SchedulNotes в базі даних

Таблиця днів розкладу зображена на рисунку 3.4.

Имя столбца	Тип данных	Разрешить значения NULL
Id	int	☐
Day	nvarchar(MAX)	☐
ScheduleId	int	☐

Рисунок 3.4 – Вигляд таблиці ScheduleDays в базі даних

Таблиця шкільних заходів зображена на рисунку 3.5.

Имя столбца	Тип данных	Разрешить значения NULL
Id	int	☐
Topic	nvarchar(MAX)	☐
Description	nvarchar(MAX)	☑
Photo	varbinary(MAX)	☑
Date	datetime2(7)	☑
PupilsCategory	nvarchar(MAX)	☑
OtherDetails	nvarchar(MAX)	☑
Place	nvarchar(MAX)	☑
ShortDescription	nvarchar(MAX)	☑

Рисунок 3.5 – Вигляд таблиці SchoolActivities в базі даних

Таблиця оцінок зображена на рисунку 3.6.

Имя столбца	Тип данных	Разрешить значения NULL
Id	int	☐
Point	int	☐
SubjectId	int	☐
CreatedDateTime	datetime2(7)	☐
StudentId	int	☐

Рисунок 3.6 – Вигляд таблиці Marks в базі даних

Таблиця домашніх завдань зображена на рисунку 3.7.

Имя столбца	Тип данных	Разрешить значения NULL
Id	int	☐
Task	nvarchar(MAX)	☐
CreatedDateTime	datetime2(7)	☐
DeadlineDateTime	datetime2(7)	☐
GroupId	int	☐
SubjectId	int	☐

Рисунок 3.7 – Вигляд таблиці Homeworks в базі даних

Після того, як таблиці сформовано, можна з ними починати взаємодію. Як вже було сказано вище, працювати із базами даних дозволить EntityFramework. Він дозволяє формувати запити до бази даних, за допомогою раніше створених моделей. Наприклад:

```
_db.Teachers.Include(m=>m.User).ToList();
```

Такий запит дозволить дістати всі записи про вчителів, включаючи користувача, що відповідає даному вчителю. Тобто, ми зможемо напряму звертатися до користувача і виводити інформацію з таблиці про нього.

Приклад ще одного запиту:

```
_db.Groups.Include(m=>m.Students).ThenInclude(u=>u.User).FirstOrDefault(c => c.Id == id);
```

Такий запит дозволить витягнути запис з бази даних про групу за певним ідентифікатором, включаючи класного керівника, та користувача, який пов’язаний із цим викладачем.

Даний фреймворк значно спрощує роботу із базою даних. Всі запити здійснюються на основі моделей та їх зв’язків.

3.3 Реалізація сайту

Після етапу проектування модулів, вже чітко можна охарактеризувати, які основні функції будуть в системі. Перешим кроком є створення можливості для додавання користувачів в систему. В даному випадку була розроблена форма реєстрації, за допомогою якої додаються нові користувачі в систему. Оскільки, робота сайту відрізняється для користувачів різних ролей, при реєстрації також вказується і роль, за якою користувач реєструється на сайті.

Після цього було додано можливість створення груп та додавання учнів до них. Це виконується за допомогою:

```
    var teacher = _db.Teachers.FirstOrDefault(t => t.Id==group.TeacherId);
    var user = _db.Users.Include(r=>r.Role).FirstOrDefault(u =>
u.Id==teacher.UserId);
    user.Role.Add(_db.Roles.FirstOrDefault(c=>c.Name=="Curator"));
    _db.Users.Update(user);
    _db.Groups.Add(group);
    _db.SaveChanges();
```

При додаванні групи, їй встановлюється класний керівник. Враховуючи це вчителю, який ним стає необхідно надати нову роль. За допомогою цієї ролі він зможе переглядати записи про власну групу.

Наступним кроком є створення розкладу для групи. Для зручності роботи з ним, було створено 3 моделі, які пов'язані між собою.

Модель розкладу містить дані про дні розкладу (дні тижня без прив'язки до дати), в свою чергу кожен день містить записи розкладу. Кожен запис містить інформацію про викладача, предмет та аудиторію.

Далі було розроблено журнал, що має досить цікаву побудову. На сторінці із журналом є випадаючий список, за допомогою якого користувач обирає предмет. В залежності від вибраного предмету завантажується сама таблиця розкладу. Все це відбувається за допомогою наступного коду:

```
$('#subject').change(function()
```

```
                {
                    var subjectId = $(this).val();
                    var groupId = $("#groupId").val();
                    $('#subjectId').val(subjectId);

                    $.ajax({
                        type: 'GET',
                        url: '@Url.Action("GetSubjectTable")',
                        data: { "groupId": groupId, "subjectId":subjectId,
"addMonth":-1 },
                        success: function (data) {

                            $('#subjectTable').replaceWith(data);

                        }
                    });
                    $.ajax({
                        type: 'GET',
                        url: '@Url.Action("GetDays")',
                        data: { "groupId": groupId, "subjectId":subjectId },
                        success: function (data) {
                            arr=data;
                            $("#addTaskFieldset").prop('disabled', false);
                        }
                    });
                     $.ajax({
                        type: 'GET',
                        url: '@Url.Action("GetHomework")',
                        data: { "groupId": groupId, "subjectId":subjectId },
                        success: function (data) {
                            $('#getHomework').val(data);
                        }
                    });
            });
```

При зміні дисципліни, одразу спрацьовує дана функція. В тілі даної функції вибирається Id предмету, що обрав користувач і формується ajax-запит. Він викликає певну дію контролеру та передає у неї параметр - зазначений Id. Після цього у контролері відбувається формування таблиці предмету та повертається часткове представлення із цією таблицею. Далі ajax-запит отримує цю відповідь та заміняє блок з певним ідентифікатором на ті дані, що було отримано. Таким чином, відбувається заміна таблиці без перезавантаження сторінки. Також, все це можливе, за допомогою часткового представлення, що не є повноцінною сторінкою, як основне представлення, а лише містить якусь частину, завдяки якій відбувається заміна [9].

Сама таблиця формується на основі розкладу. Було розроблено функцію, що знаходить дні тижня, коли в розкладі є даний предмет, що обрано зі списку. Далі знаходяться дати цих днів протягом місяця. На основі дат, учнів та предмету

дістються оцінки. Було додано модель, що відповідає за виведення даних, вона не співвідноситься із базою даних. Ця модель дозволяє вивести таблицю зі всіма оцінками та учнями.

Також на сторінці із журналом є функціонал, що відповідає за додавання та виведення домашнього завдання. Виведення також здійснюється на основі ajax-запиту, який прописаний в тілі функції, що реагує на зміну предмету [10].

Щодо можливості додавання домашнього завдання, тут реалізація є дещо складнішою, оскільки спершу необхідно обрати дату дедлайну. Проте календар, де вона обирається містить певні обмеження – можна обрати тільки день, коли є цей урок. При зміні предмету, за допомогою знову ж таки ajax-запиту, формується масив із днями тижня, що повинні бути активними. Далі встановлюється формат виведення для календара:

```
    function unavailable(date) {
           var day = date.getDay();
           var dayVisibility = false;
           for(var i=0; i<arr.length;i++){
                 if(arr[i]==day){
                       dayVisibility = true;
                       break;
                 }
           }
           return [dayVisibility,  ''];
    }

    $("#datepicker").datepicker({
    minDate: 0,
    beforeShowDay: unavailable
    });
```

Після того як користувач обирає певну дату, спрацьовує функція:

```
    $("#datepicker").change(function()
                  {
                  var deadline = $(this).val();
                      var subjectId = $("#subjectId").val();
                      var groupId = $("#groupId").val();
                      $('#subjectId').val(subjectId);
                      $.ajax({
                          type: 'GET',
                          url: '@Url.Action("AddHomework")',
                          data: { "groupId": groupId, "subjectId":subjectId, "deadline"
: deadline},
                          success: function (data) {
```

```
$('#dialogContent').html(data);
$('#modDialog').modal('show'); }});});
```

В результаті повертається модальне вікно, що потрібно заповнити для додавання домашнього завдання.

Ще одна функція, що забезпечується ajax-запитами, це переключання між сторінками журналу. Це також необхідно робити без перезавантаження сторінки, тому це найбільш раціональний спосіб [9].

Формування щоденника є також динамічним, тобто немає певної таблиці в базі даних де все прописано. Знову ж таки, основою є розклад, з якого витягуються оцінки та домашнє завдання.

3.4 Інструкція з експлуатації

3.4.1 Інструкція для неавторизованого користувача

При вході на сайт користувачу виводиться інформація про події, які відбуваються в школі. Це сторінка, яка доступна всім користувачам (рис. 3.8)

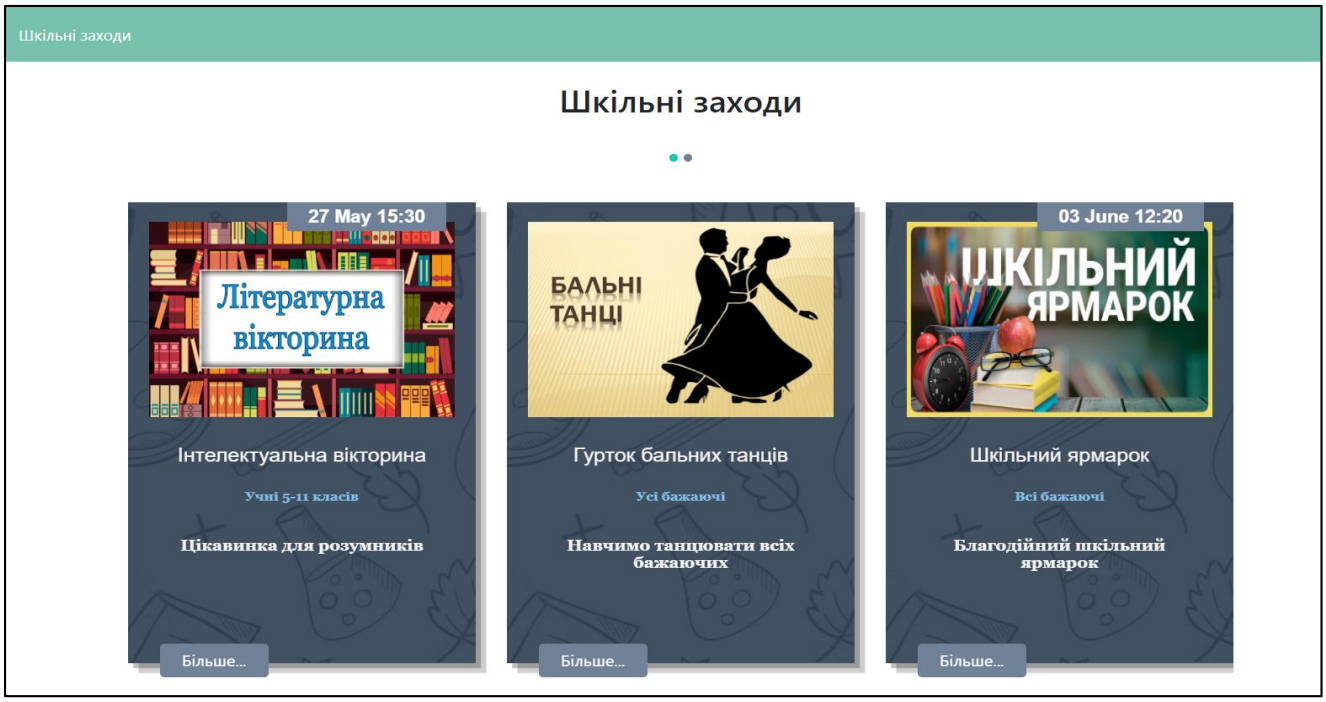


Рисунок 3.8 – Сторінка із шкільними заходами

Щоб продовжити користуватись сайтом необхідно авторизуватися. Форма авторизації зображена на рисунки 3.9.

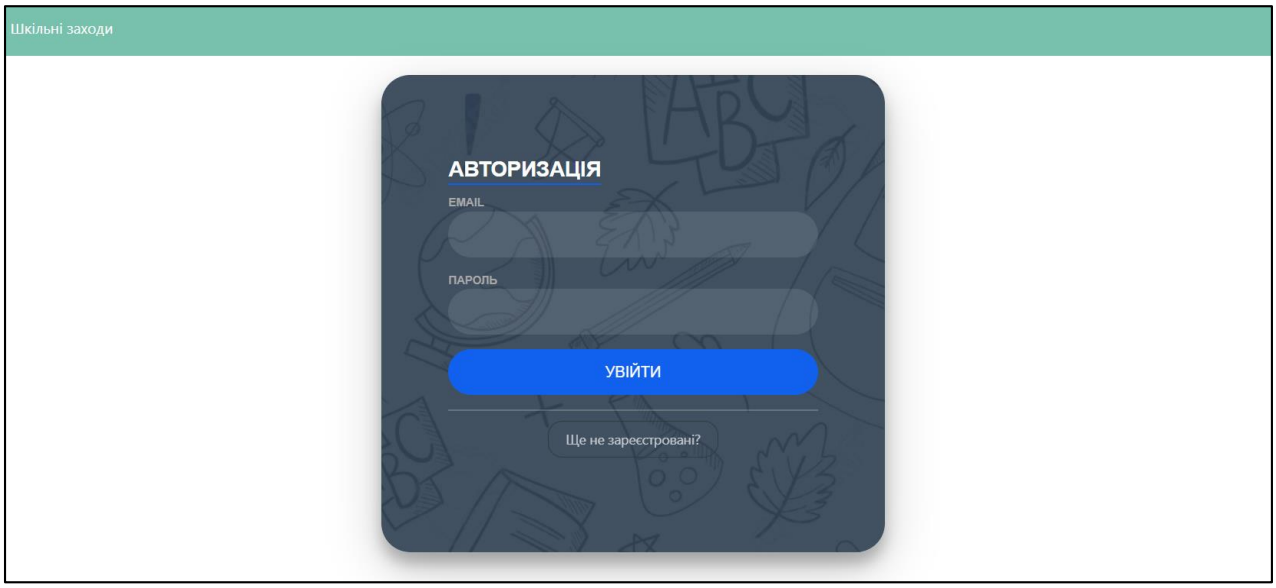


Рисунок 3.9 – Форма авторизації

Якщо користувач ще не зареєстрований, він може перейти на форму реєстрації. Вікно реєстрації зображено на рисунку 3.10.

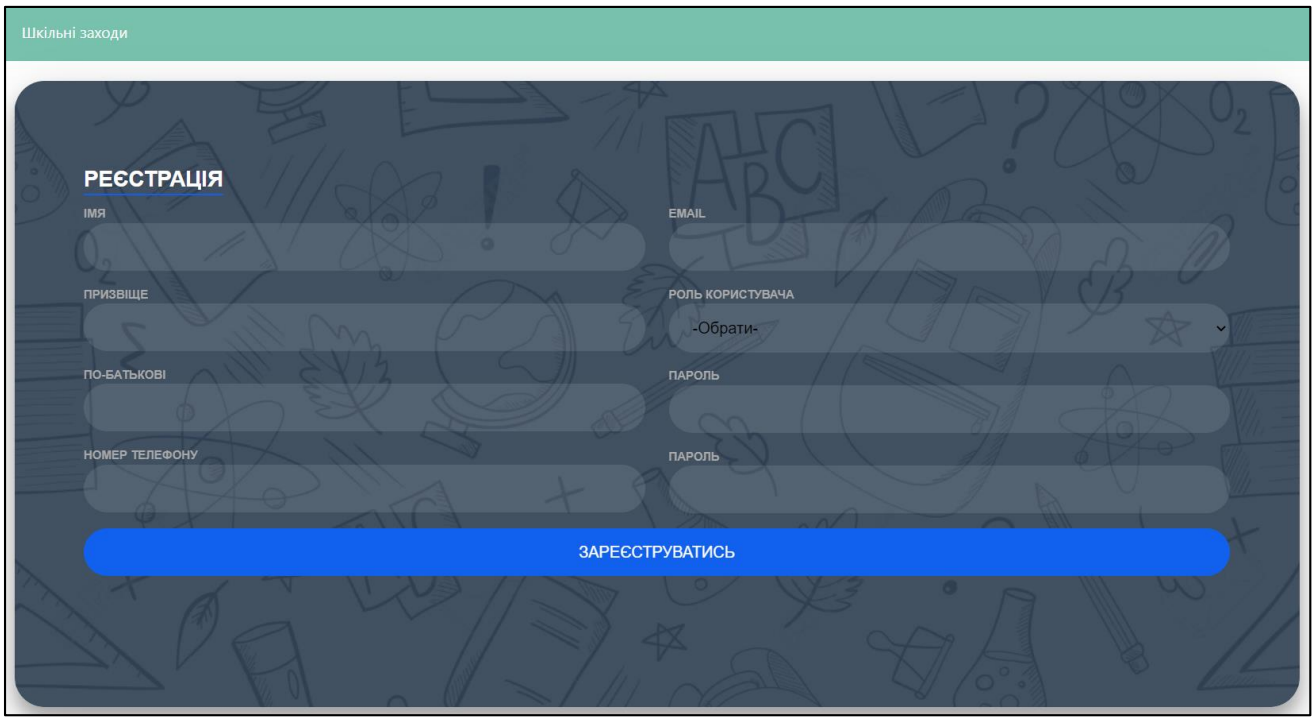


Рисунок 3.10 – Форма реєстрації

3.4.2 Інструкція для адміністратора

Адміністратор робить всю організаційну роботу на сайті – додає, редагує, видаляє певні сутності.

Для адміністратора можливості зареєструватися немає, тільки авторизація. Він може здійснювати додавання шкільних заходів (рис. 3.11).

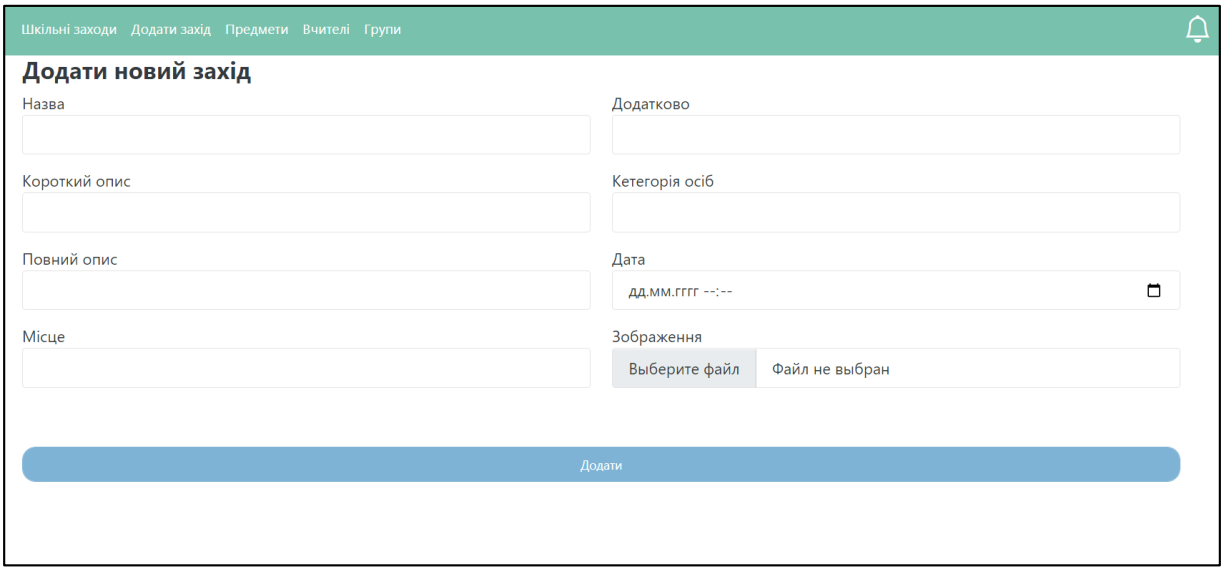


Рисунок 3.11 – Вікно додавання нового шкільного заходу

Адміністратору доступний перелік всіх предмет із посиланнями на редагування та видалення (рис. 3.12)

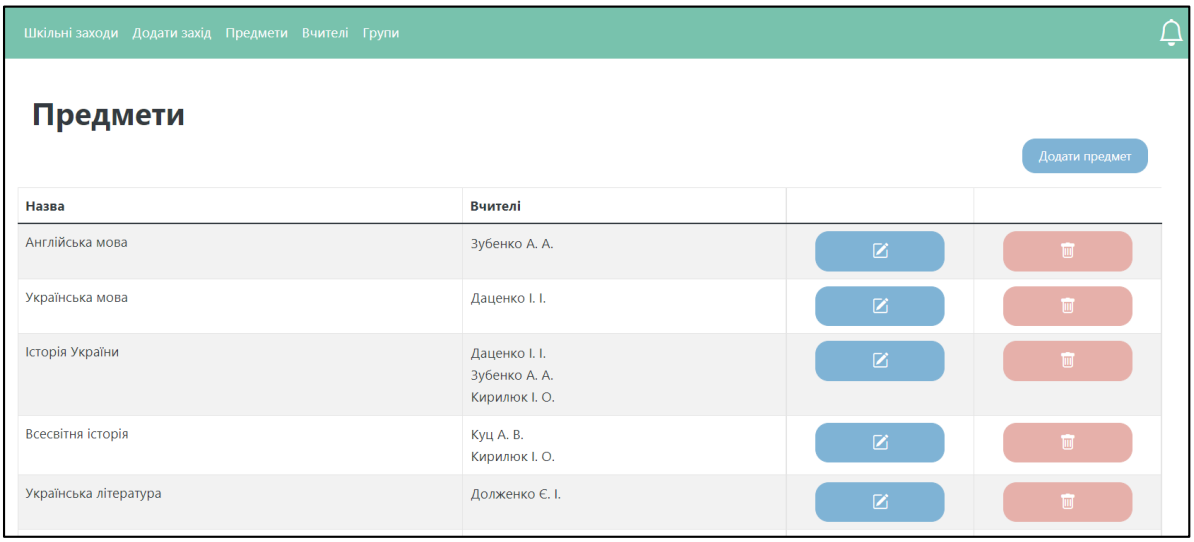


Рисунок 3.2 – Сторінка перегляду наявних предметів

Після натиснення кнопки «Додати предмет» користувачу відкриється форма, де він зможе вказати назву дисципліни. Також адміністратору доступні списки всіх викладачів (рис. 3.13)

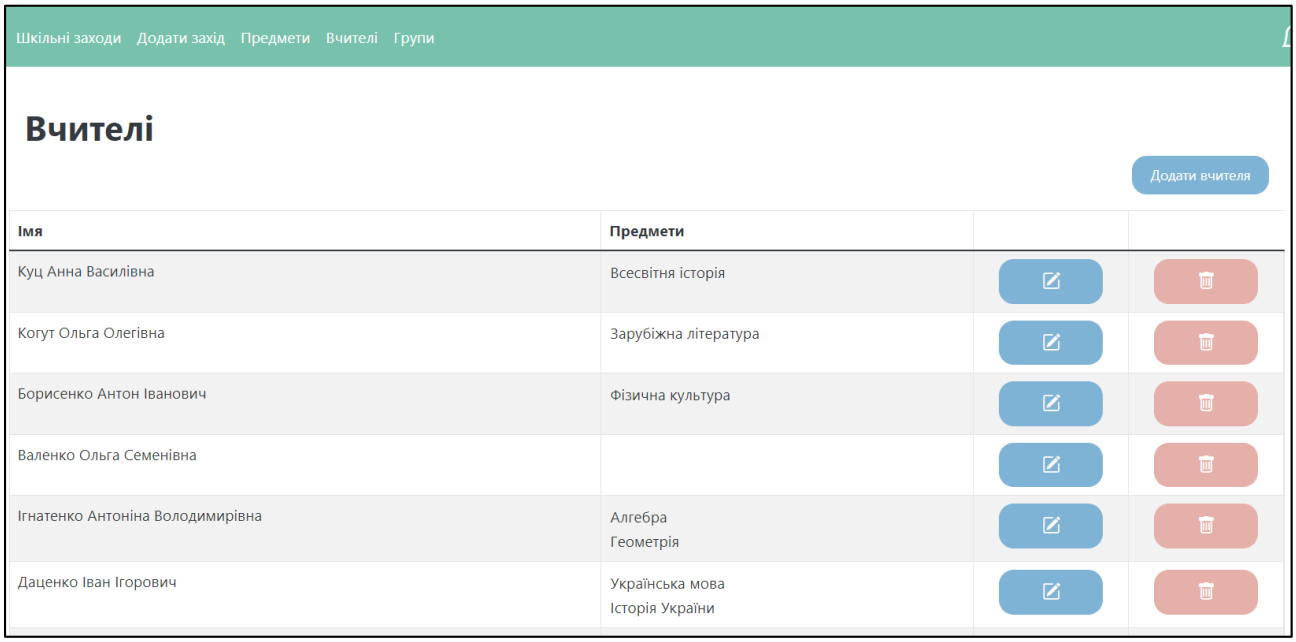


Рисунок 3.13 – Список вчителів

Для того, щоб додати вчителя, необхідно перейти за посиланням на форму, обрати зі списку користувача та один або декілька предметів (рис 3.14)

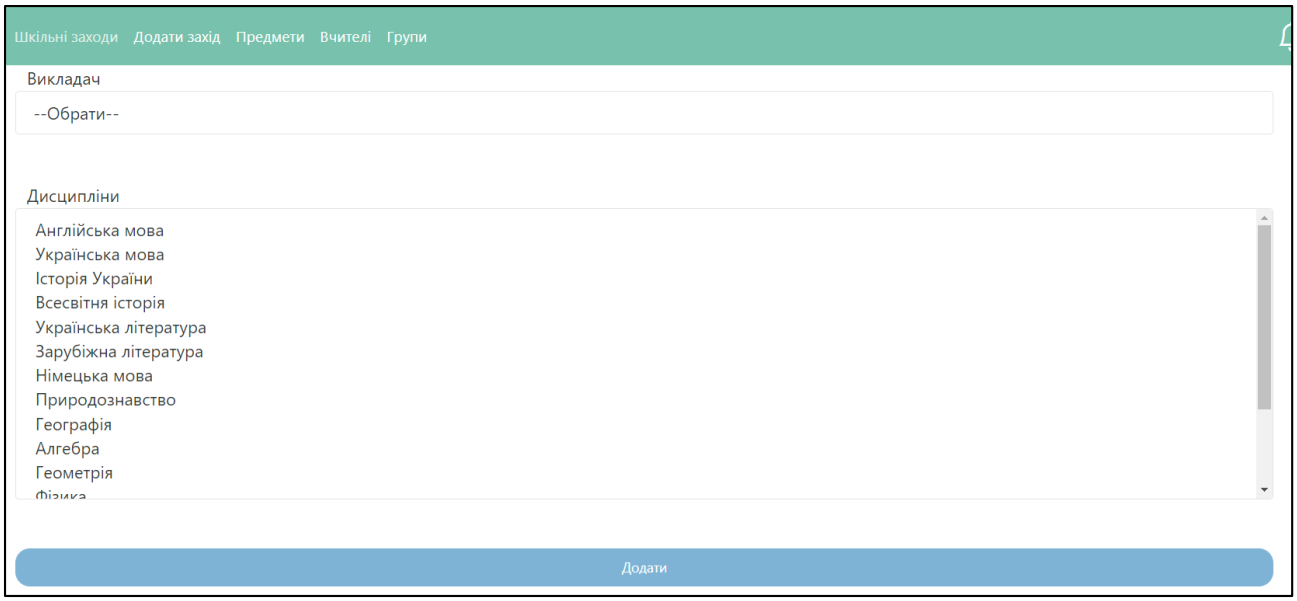


Рисунок 3.14 – Додавання вчителя

Адміністратору доступний список класів з можливістю їх редагування і видалення (рис. 3.15).

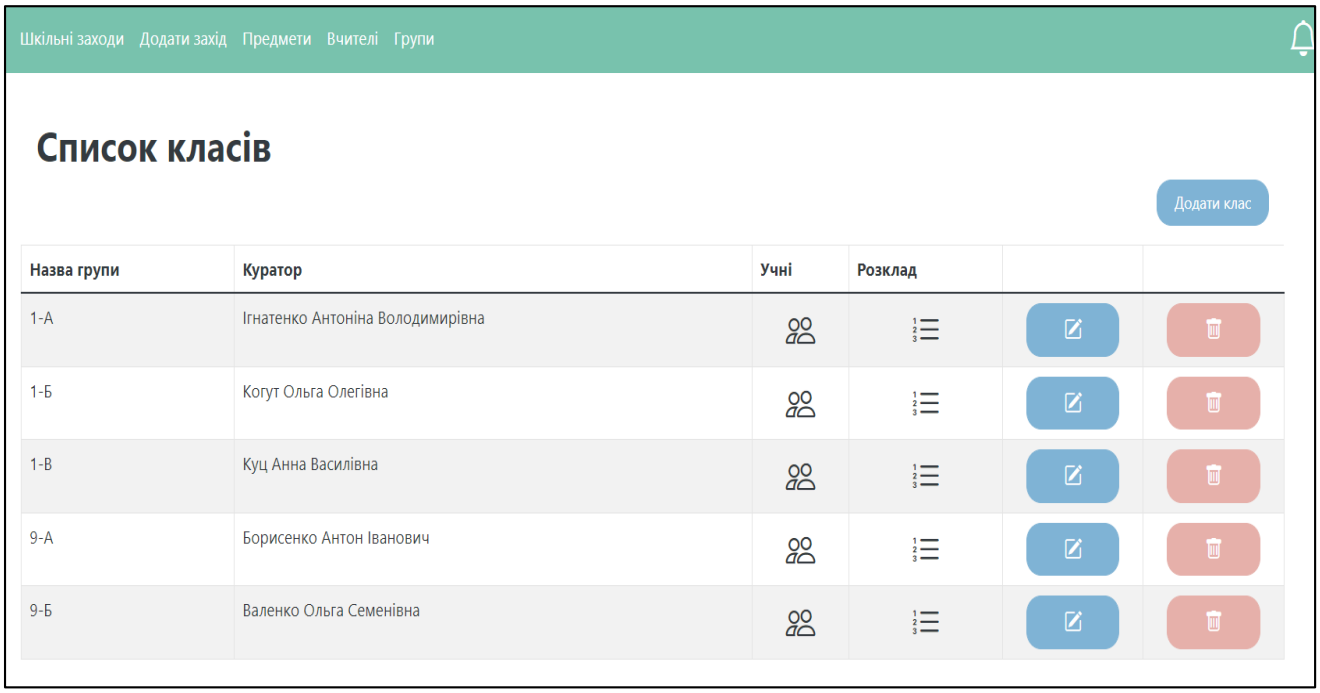


Рисунок 3.15 – Список класів

Вигляд сторінки зі списком учнів зображена на рисунку 3.16.

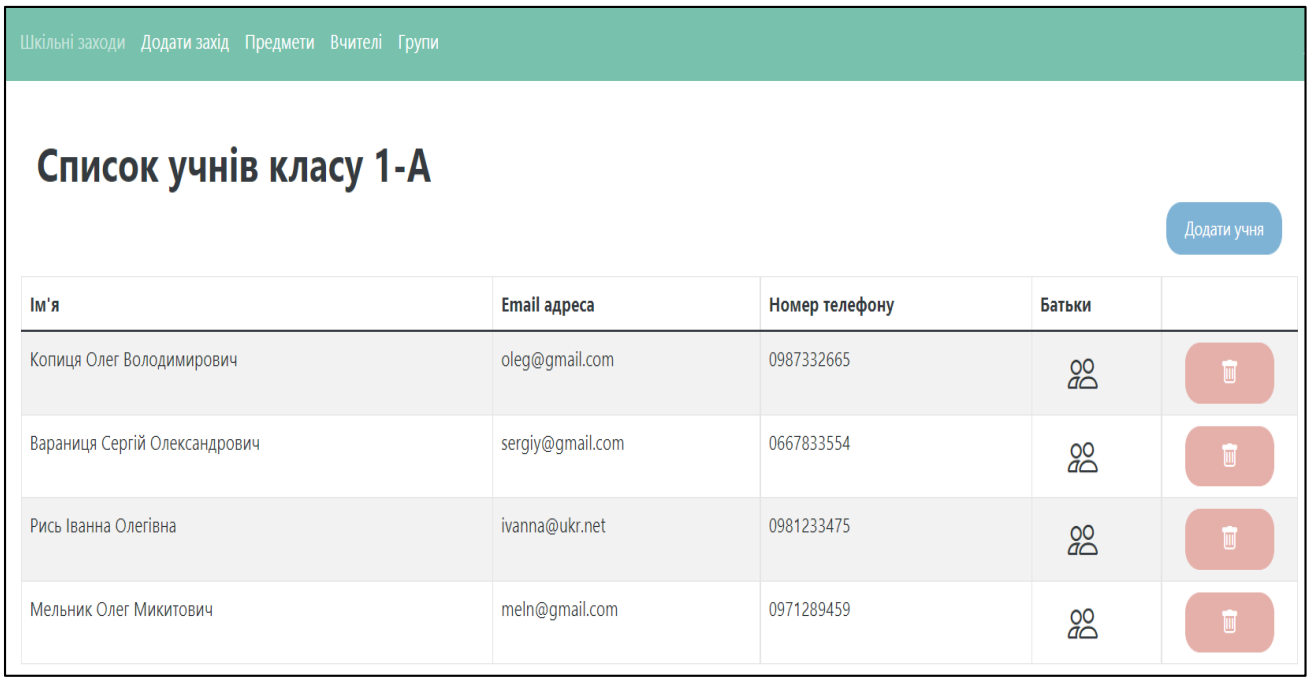


Рисунок 3.16 – Список учнів

Додавання учнів у клас зображено на рисунку 3.17.

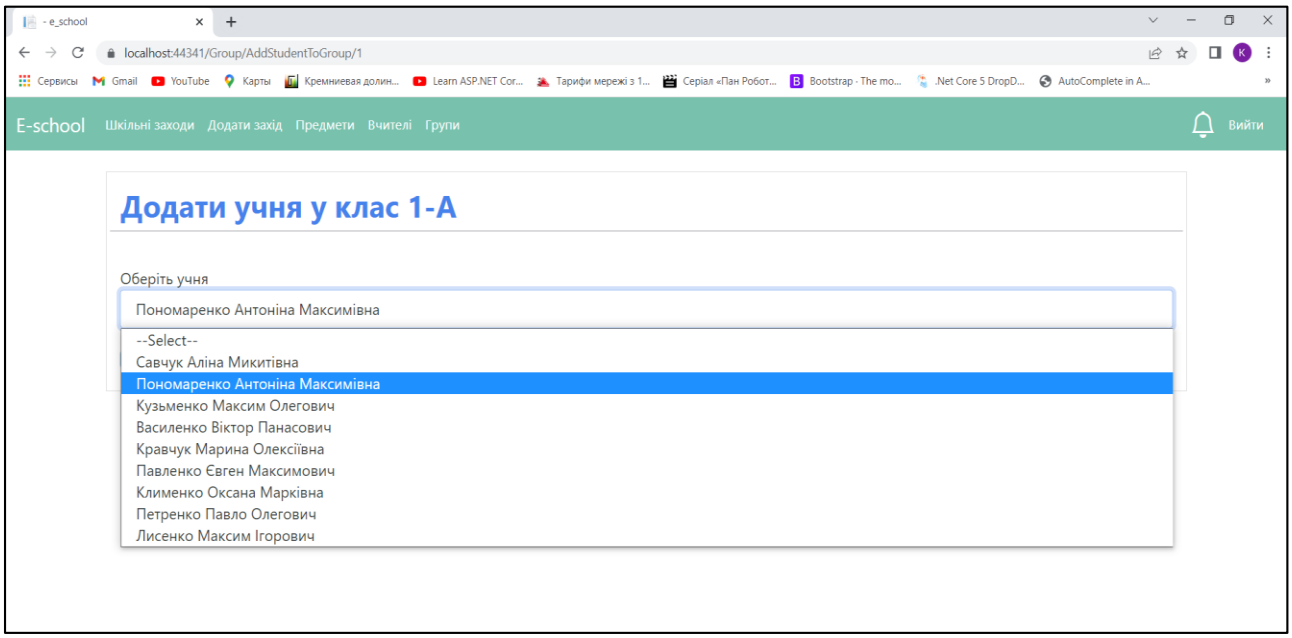


Рисунок 3.17 – Форма додавання учня до класу

Вигляд розкладу групи зображено на рисунку 3.18.

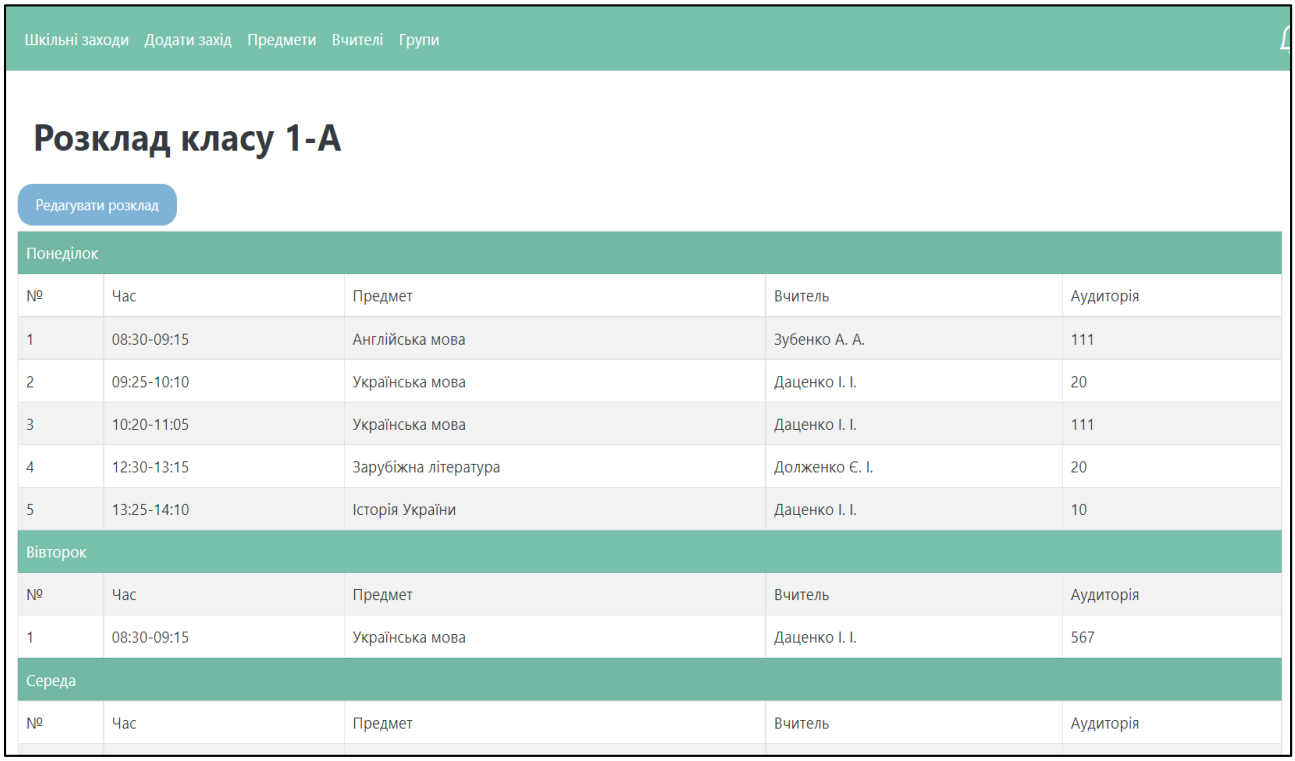


Рисунок 3.18 – Сторінка із розкладом групи

При натисненні кнопки «Редагувати розклад», користувач переходить на форму із забезпеченням можливості редагування пунктів розкладу (рис. 3.19)

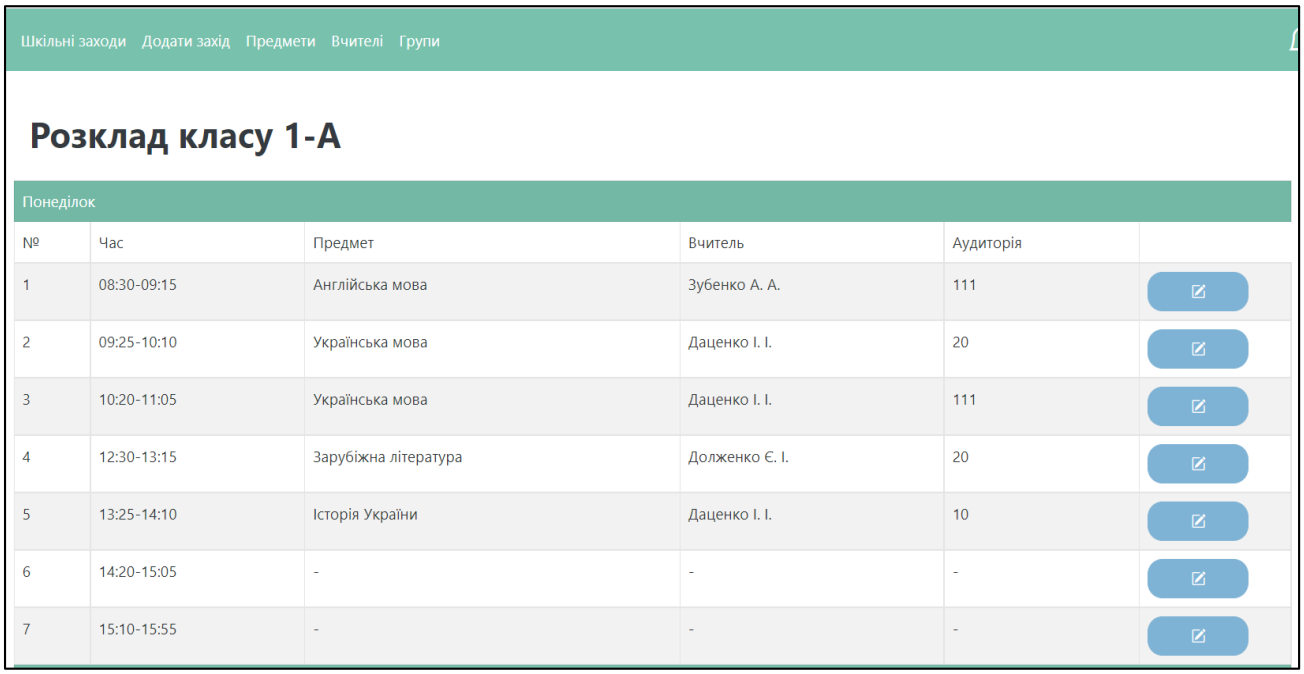


Рисунок 3.19 – Вигляд сторінки для редагування розкладу

Якщо натиснути на кнопку редагування певного пункту, користувачу відкриється форма із додаваннням/редагуванням пункту розкладу (рис. 3.20).

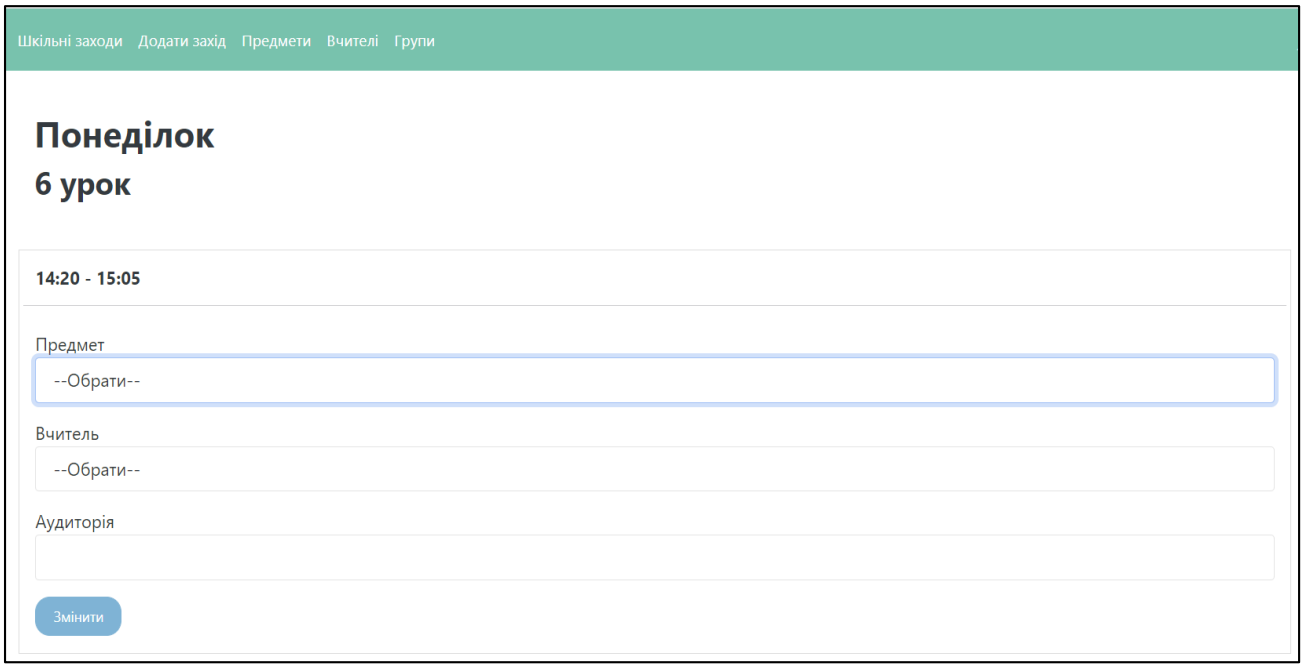


Рисунок 3.20 – Вигляд форми для додання пункту розкладу

3.4.3 Інструкція для вчителя та класного керівника

При переході на список класів для вчителя доступний ще один розділ для кожного класу – журнал (рисунок 3.21)

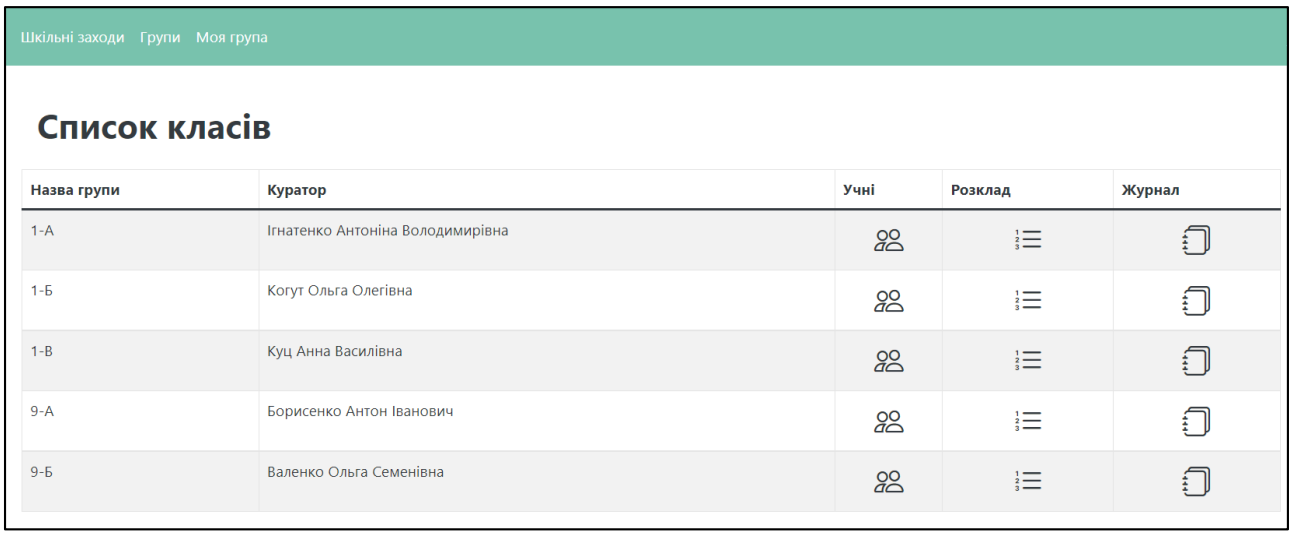


Рисунок 3.21 – Вигляд списку класів для вчителя/класного керівника

При натисненні іконки журналу навпроти будь-якого з класів, відкриється сторінка журналу для обраного класу. Далі, необхідно обрати дисципліну із списку. Після цього вигляд журналу зміниться (рисунок 3.22).

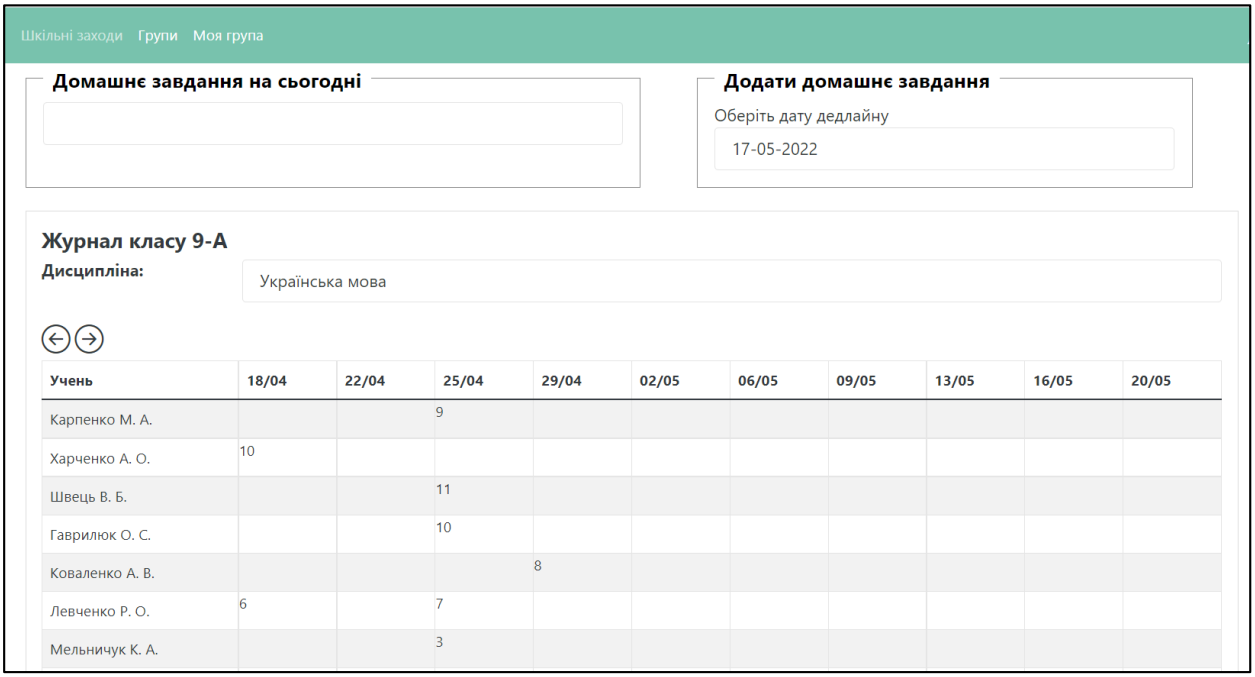


Рисунок 3.22 – Сторінка із журналом класу

Вчитель може додати домашнє завдання по обраному зі списку предмету. Для цього необхідно обрати дату дедлайну. При натиснені на поле, відкриється календар із доступними датами, а саме доступні тільки ті дні, у які по розкладу будуть уроки (рисунок 3.23).

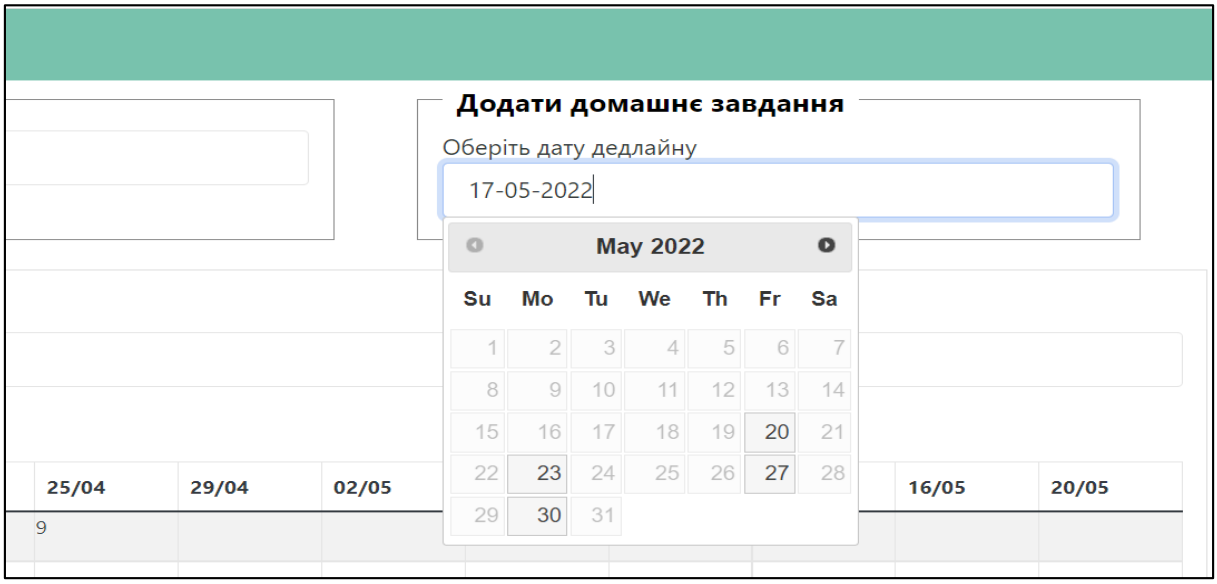


Рисунок 3.23 – Календар дедлайну домашнього завдання

Після вибору дати відкриється модальне вікно, де потрібно додати завдання (рисунок 3.24)

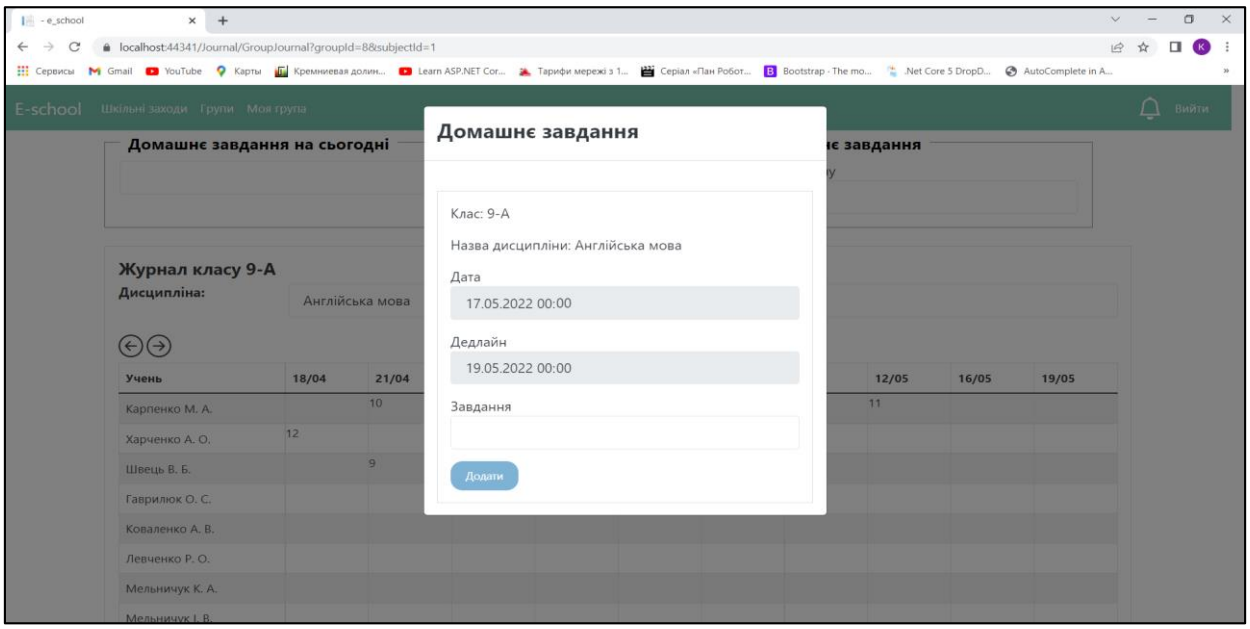


Рисунок 3.24 – Модальне вікно для введення домашнього завдання

Вчитель може додати оцінки до журналу. Для цього потрібно натиснути на будь-якій клітинці журналу. Відкриється модальне вікно із параметрами, що автоматично зчитуються при натисканні на комірку, потрібно тільки ввести бал від 1 до 12. Якщо в цій клітинці оцінка вже була, то модальне вікно відкриється для редагування (рисунок 3.25).

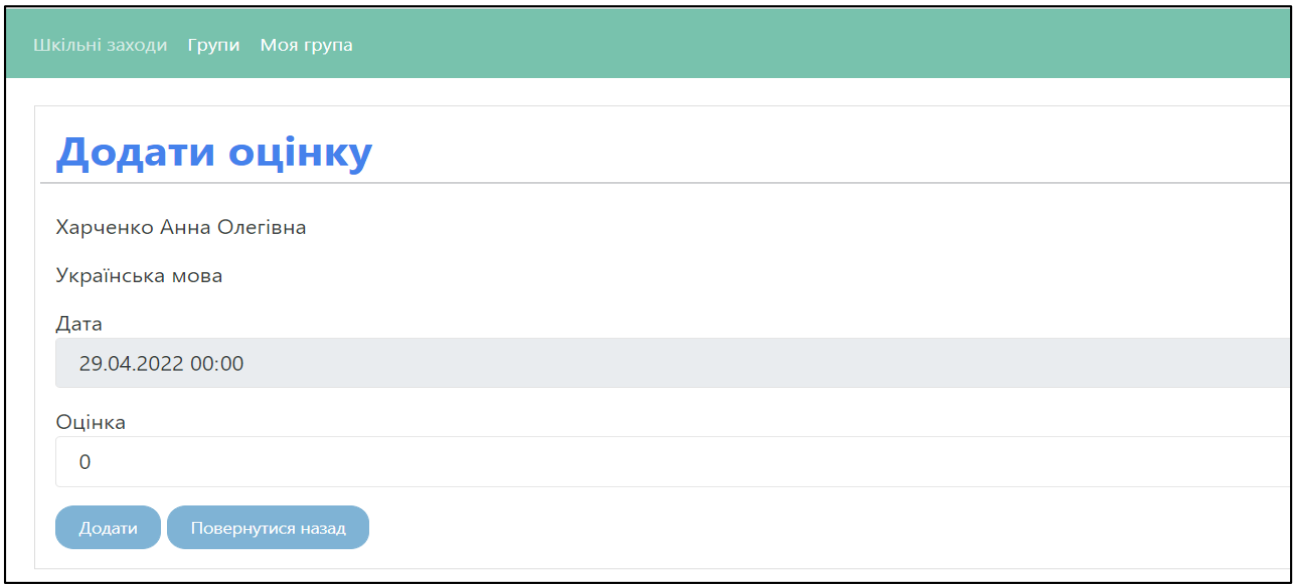


Рисунок 3.25 – Вигляд сторінки із додаванням оцінки учня

Якщо користувачу присвоєно роль класного керівника, то йому доступний пункт меню «Моя група». При переході на цю сторінку, йому відображаються розділи для його класу (рисунок 3.26)

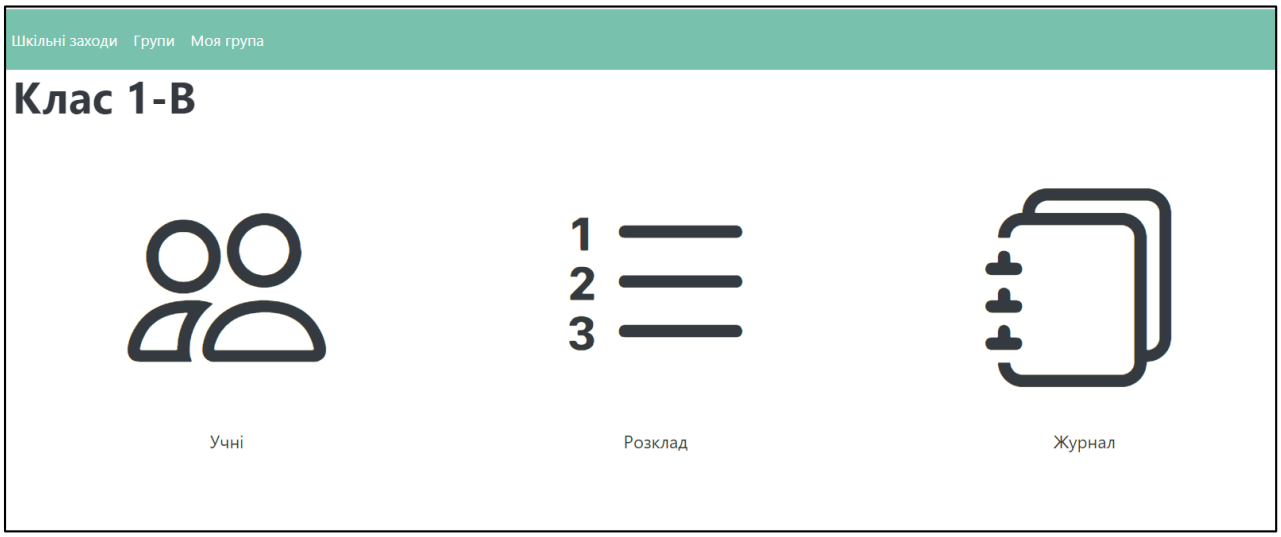


Рисунок 3.26 – Сторінка із категоріями певного класу

Також для куратора доступна кнопка «Сформувати табель» на сторінці із журналом. Після її натиснення з'явиться модальне вікно, де потрібно заповнити необхідну інформацію для формування табелю (рис 3.27)

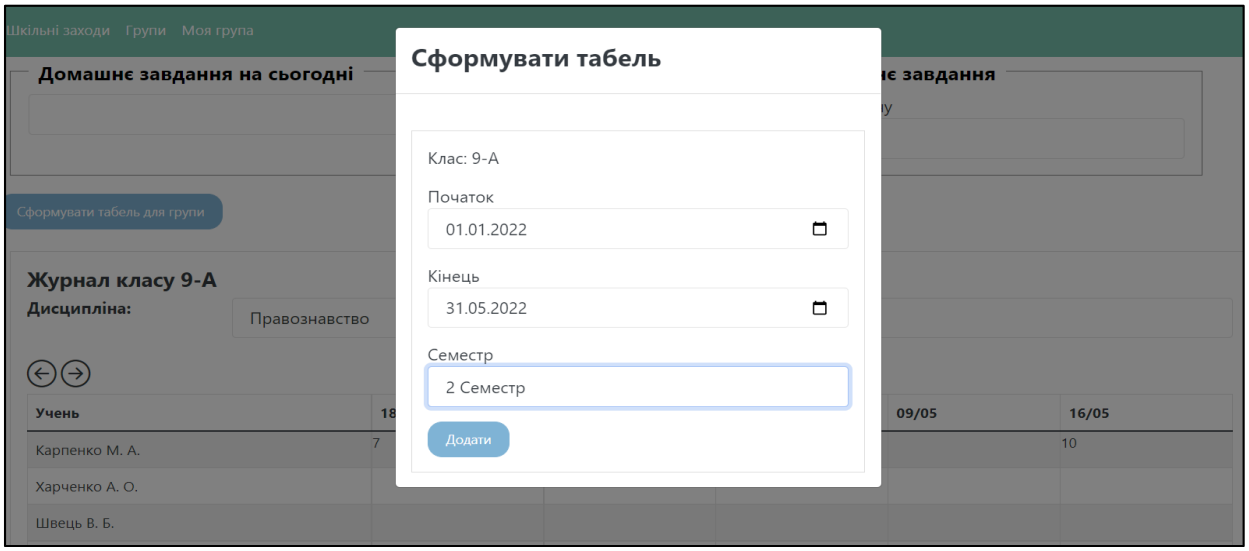


Рисунок 3.27 – Модальне вікно для формування табелів учнів класу

3.4.4 Інструкція для учня та батьків

Для учня та його батьків доступний щоденник учня (рисунок 3.28).

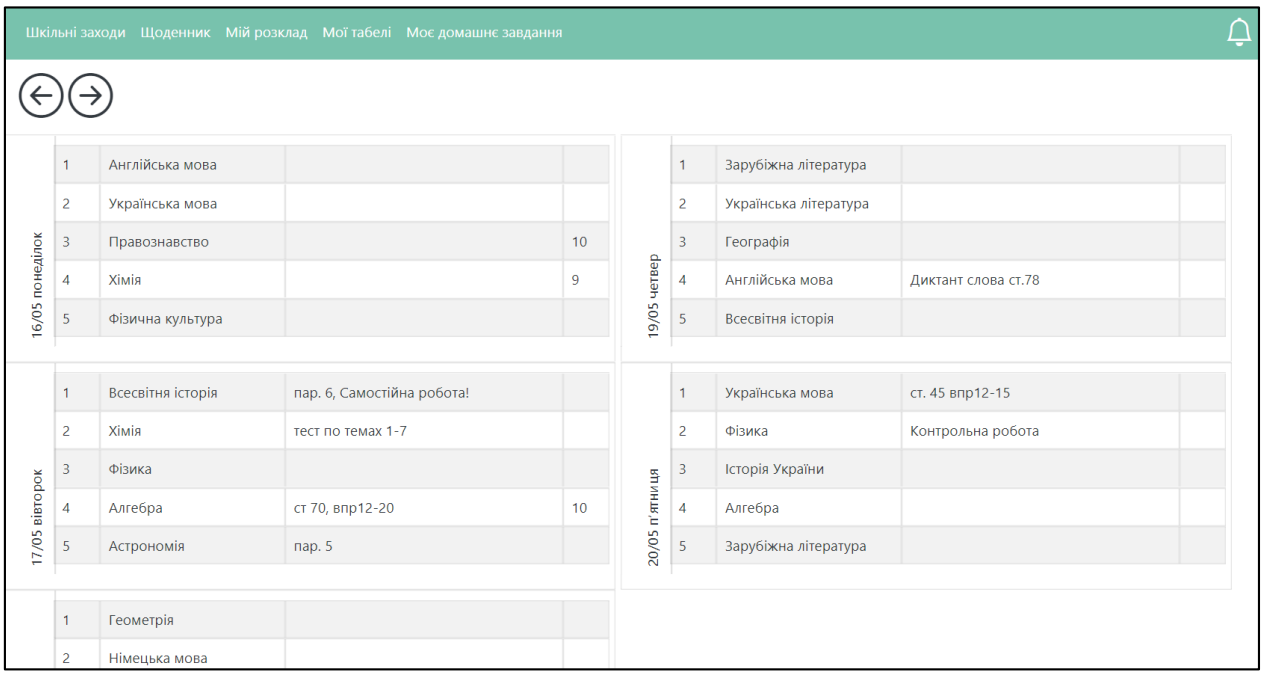


Рисунок 3.28 – Вигляд сторінки із щоденником учня

При переході на сторінку «Моє домашнє завдання» учню відкривається список його домашніх завдань із ступенем готовності. Учень може ставити відмітку у разі виконання завдання або забирати її. Чек-лист із домашніми завданнями зображено на рисунку 3.29.

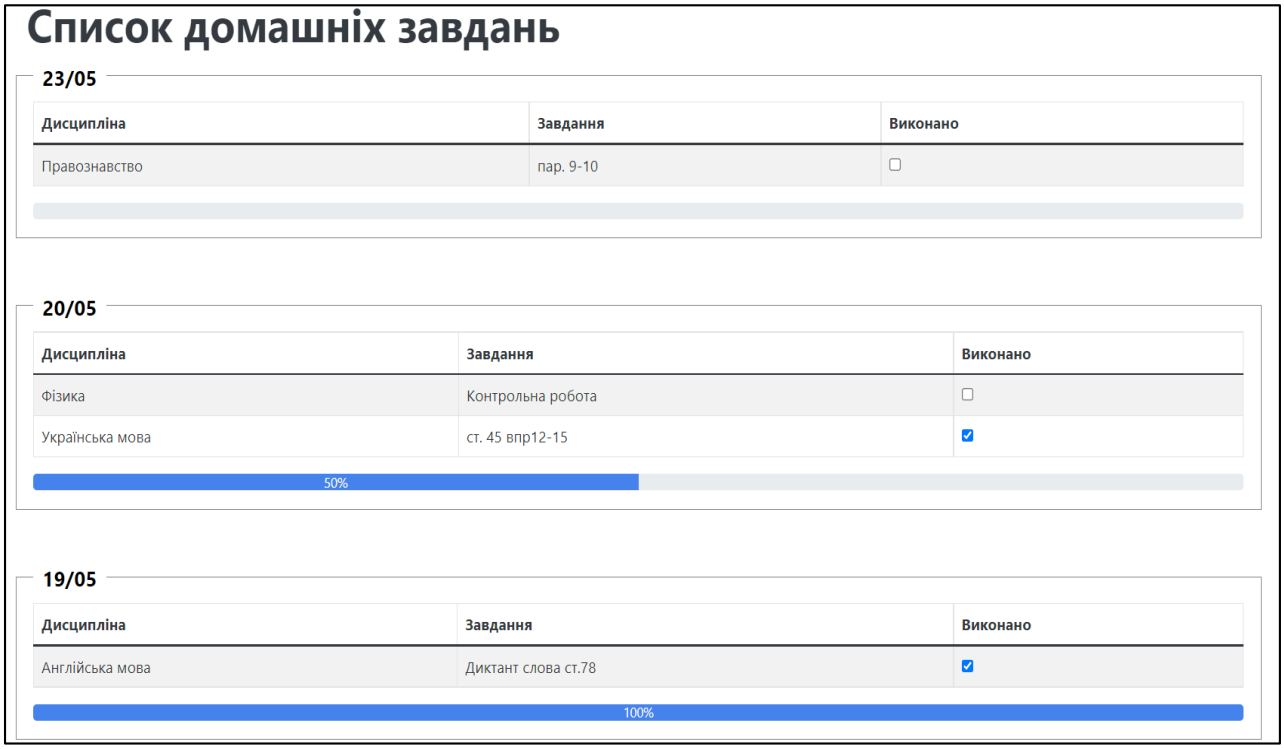


Рисунок 3.29 – Список домашніх завдань учня

Також учень може переглянути свої табелі, обравши відповідний пункт меню на навігаційному меню. Йому відобразиться сторінка із списком всіх сформованих табелів (рисунок 3.30)

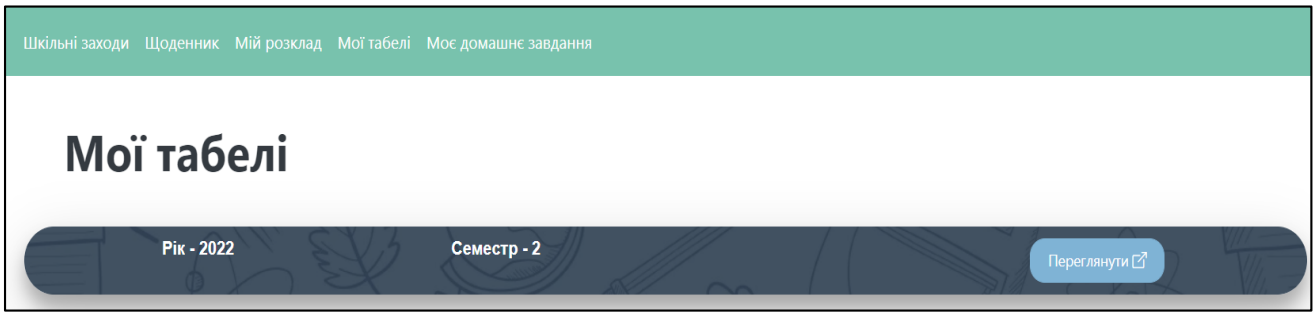


Рисунок 3.30 – Сторінка із списком наявних табелів учня

При натисненні на один із табелів відкриється сторінка обраного табелю. Табель відображує список предметів та підсумкову оцінку (рисунок 3.31)

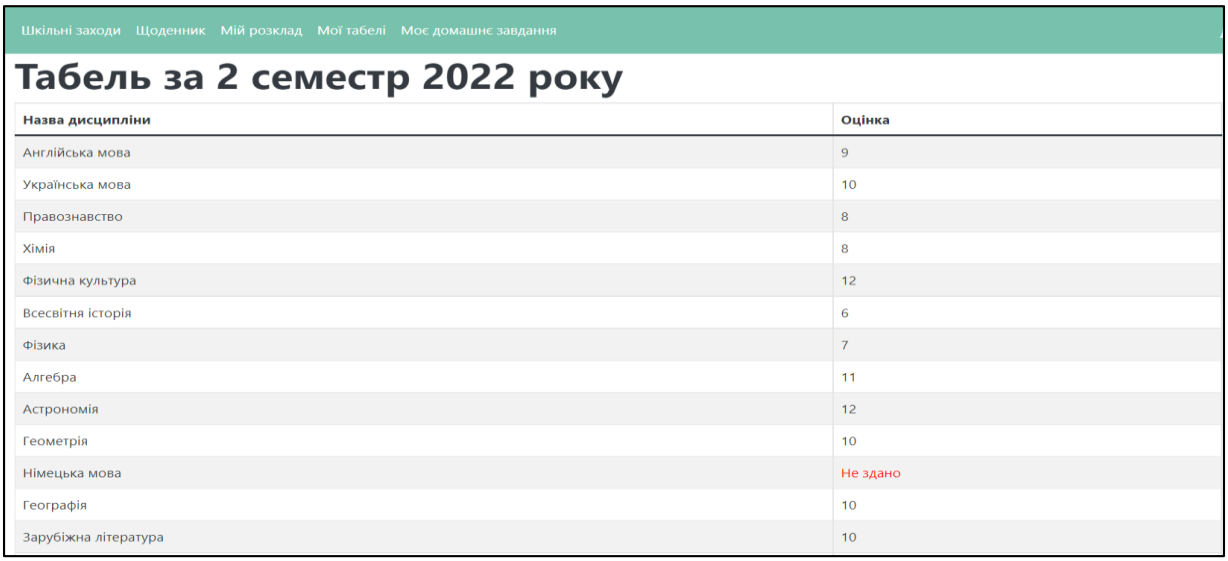


Рисунок 3.31 – Сторінка із табелем

Також учню доступний перегляд повідомлень, що були надіслані вчителями. Для переходу на цю сторінку, необхідно обрати відповідний пункт навігаційного меню (рисунок 3.32)

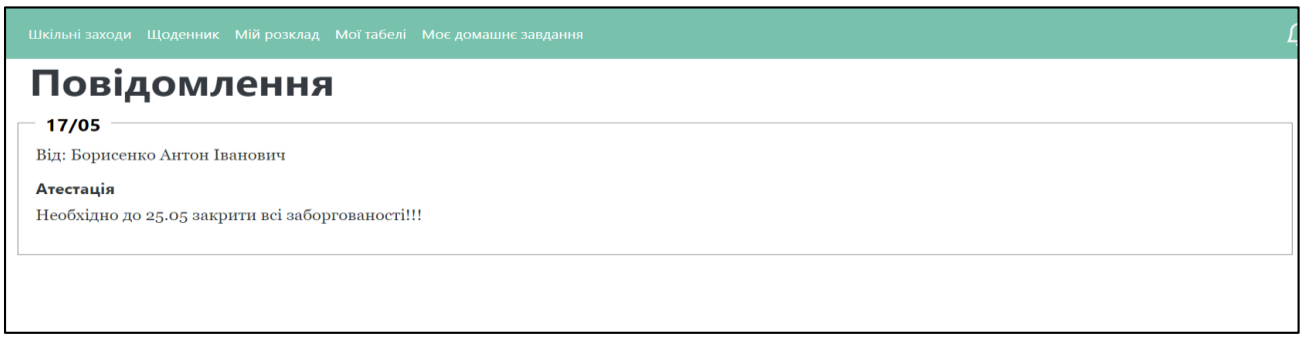


Рисунок 3.32 – Сторінка із списком повідомлень

У 3 розділі було визначено структуру програмних модулів. Наступним кроком стало створення моделей, а на їх основі реалізовано базу даних, із встановленням зв’язків між її таблицями. Після цього було запрограмовано всю логіку додатку MVC, включаючи контролери та представлення для їх методів.

4 ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Методика тестування програмного забезпечення

Даний етап проводиться для перевірки якості продукту відносно контексту, в якому він має використовуватися. Тестування проводиться з метою виявлення різнопланових помилок чи інших дефектів. Програмний продукт перевіряється в спеціально-створених штучних умовах.

Прийнято рішення, що дане програмне забезпечення буде тестуватися двома способами, а саме: перевірка на коректність введених даних, та модульне тестування.

Модульне тестування проводиться для кожного модуля окремо. Модулем може бути будь-який метод, функція, тобто це найменша частинка програми, яка може бути протестована. Даний вид тестування полягає в написанні окремих блоків коду, які перевіряють модулі основної програми. Зазвичай тестування виконується, щоб перевірити чи кожен модуль має очікувану поведінку [11].

Метою модульного тестування є ізоляція кожної частини програми та впевненість у тому, що кожна окрема частина є коректною. Модульний тест забезпечує жорсткий «контракт», за яким має працювати тестований код. Як результат, це надає деякі переваги. Модульне тестування допомагає знайти помилки раніше в циклі розробки ПЗ, що робить розробку дешевшою та швидшою.

Оскільки дане тестування застовується до кожного модуля окремо, то перевірку можна здійснювати під час написання самої програми. Це зручно оскільки інколи необхідно перевірити попередній модуль для написання наступного [11].

Щодо ручного тестування, то воно необхідне для перевірки валідації даних, тобто наскільки користувач проінформований про помилки при введені полів. І чи програмне забезпечення не допускає невірного введення даних.

4.2 Розробка тестових сценаріїв

Для перевірки всіх функцій системи було розроблено ряд тест-кейсів, що дозволять оцінити роботу кожного користувача системи. Тестові сценарії для неавторизованого користувача наведено у таблиці 4.1.

Таблиця 4.1 – Тестові сценарії для неаторизованого користувача

	Модуль	Чек-лист	Очікуваний результат
Неавторизований користувач	**Стартова сторінка**	1. Перейти на сайт	1 Відкривається стартова сторінка "Шкільні заходи"
		2 На одному з заходів натиснути кнопку "Більше"	2 Відкривається модальне вікно з детальним описом вибраної події
		3 Натиснути будь-де за межами модального вікна	3 Модальне вікно закриється
	Форма Авторизації	1 Знаходячись на стартовій сторінці натиснути кнопку "Увійти" на навігаційному меню	1 Відкривається сторінка авторизації
		2 Ввести валідні дані в поле Емейл	2 Дані в полі Емейл відображаються
		3 Ввести валідні дані в поле Пароль	3 Дані в полі Пароль відображаються
		4 Натиснути кнопку Увійти	4 Відкривається Стартова сторінка, але з навігаційним меню авторизованого користувача
	Форма Реєстрації	1 Знаходячись на стартовій сторінці натиснути кнопку "Увійти" на навігаційному меню	1 Відкривається сторінка авторизації
		2 Натиснути на кнопку "Ще не зареєстровані?"	2 Відкриється сторінка реєстрації
		3 Ввести валідні дані в поле ІМ*Я	3 Дані в полі ІМ*Я відображаються
		4 Ввести валідні дані в поле ПРІЗВИЩЕ	4 Дані в полі ПРІЗВИЩЕ відображаються
		5 Ввести валідні дані в поле ПО-БАТЬКОВІ	5 Дані в полі ПО-БАТЬКОВІ відображаються
		6 Ввести валідні дані в поле НОМЕР ТЕЛЕФОНУ	6 Дані в полі НОМЕР ТЕЛЕФОНУ відображаються
		7 Ввести валідні дані в поле EMAIL	7 Дані в полі EMAIL відображаються
		8 Вибрати роль в полі РОЛЬ КОРИСТУВАЧА	8 З*являється випадаючий список з усіма можливими ролями
		9 Ввести валідні дані в поле ПАРОЛЬ	9 Дані в полі ПАРОЛЬ відображаються
		10 Вести валідні дані в поле ПОВТОРІТЬ ПАРОЛЬ	10 Дані в полі ПОВТОРІТЬ ПАРОЛЬ відображаються
		11 Натиснути на кнопку ЗАРЕЄСТРУВАТИСЬ	11 Відкривається стартова сторінка з повідомленням "ДЯКУЄМО ЗА РЕЄСТРАЦІЮ!"

Тестові сценарії для користувача «Адміністратор» наведено у таблиці 4.2.

Таблиця 4.2 - Тестові сценарії для адміністратора

	Модуль	Чек-лист	Очікуваний результат
Адміністратор	**Сторінка Додати захід**	1 Натиснути на кнопку Додати захід на навігаційному меню	1 Відкривається сторінка додавання нового заходу
		2 Заповнити всі поля	2 Дані в полях відображаються
		3 Вибрати дату в полі Дата	3 З'являється календар з вибором дати
		4 Натиснути на кнопку Виберіть файл	4 Відкриється Провідник для вибору необхідного файлу на комп'ютері
		5 Натиснути кнопку Додати	5 Відкривається сторінка з усіма заходами, що містить щойно доданий
	Сторінка Предмети	1 Натиснути на кнопку Предмети на навігаційному меню	1 Відкривається сторінка з переліком предметів та кнопкою для додавання нового предмету
		2 Натиснути на «Додати предмет»	2 Відкривається вікно Додати предмет
		3 Ввести назву предмета	3 Дані в полі Назва відображаються
		4 Натиснути кнопку Додати	4 Відкривається сторінка з переліком предметів, що містить щойно доданий
		5 Натиснути кнопку редагування навпроти будь-якого предмета	5 Відкривається сторінка Редагувати предмет
		6 Ввести нову назву предмета в поле Назва замість поточної	6 Відкривається сторінка з переліком предметів, що містить щойно редагований
		7 Натиснути кнопку видалення навпроти будь-якого предмета	7 Вибраний предмет видаляється, сторінка оновлюється
	Сторінка Вчителі	1 Натиснути на кнопку Вчителі на навігаційному меню	1 Відкривається сторінка з переліком вчителів та кнопкою для додавання нового вчителя
		2 Натиснути «Додати вчителя»	2 Відкривається вікно додавання вчителя
		3 З випадаючого списка Викладач вибрати нещодавно зареєстрованого викладача	3 Дані в полі Викладач відображаються
		4 Зі списку Дисципліни вибрати відповідний до його компетенції предмет	4 Вибраний предмет виділяється синім маркером
		5 Натиснути кнопку Додати	5 Відкривається сторінка з переліком викладачів, що містить щойно доданого
		6 Натиснути кнопку редагування навпроти будь-якого викладача	6 Відкривається сторінка редагування викладача
		7 Вибрати новий предмет зі списку	7 Вибраний предмет виділяється синім маркером
		8 Натиснути кнопку Змінити	8 Відкриється сторінка з переліком викладачів, що містить щойно редагованого викладача
		9 Натиснути кнопку видалення навпроти будь-якого викладача	9 Вибраний викладач видаляється, сторінка оновлюється

Продовження таблиці 4.2

Сторінка Групи	1 Натиснути на кнопку Групи на навігаційному меню	1 Відкривається сторінка з переліком класів та кнопкою для додавання нового класу
	2 Натиснути на кнопку Додати клас	2 Відкривається вікно додавання класу
	3 Ввести назву класу в поле Назва	3 Введені дані відображаються
	4 Вибрати класного керівника з випадаючого списку	4 Вибраний класний керівник відображається
	5 Натиснути кнопку Додати	5 Відкривається сторінка з переліком класів, що містить щойно доданий клас
	6 Натиснути іконку Учні будь-якого класу	6 Відкривається сторінка з переліком учнів поточного класу
	7 Натиснути на кнопку Додати учня	7 Відкривається вікно додавання учня
	8 З випадаючого списка Оберіть учня вибрати нещодавно зареєстрованого учня	8 Дані в полі Оберіть учня відображаються
	9 Натиснути кнопку Додати	9 Відкривається сторінка з переліком учнів класу, що містить щойно доданого учня
	10 Натиснути кнопку видалення навпроти будь-якого учня	10 Вибраний учень видаляється, сторінка оновлюється
	11 Натиснути кнопку Розклад навпроти будь-якого класу	11 Відкривається сторінка з розкладом вибраного класу
	12 Натиснути кнопку Редагувати розклад	12 Відкривається сторінка з розкладом
	13 Натиснути кнопку редагування навпроти будь-якого предмету	13 Відкриється сторінка з додаванням чи редагуванням уроку
	14 З випадаючих списків обрати предмет та вчителя	14 Вибрані дані відображаються
	15 Ввести в поле Аудиторія номер аудиторії	15 Введені дані відображаються
	16 Натиснути кнопку Змінити	16 Відкривається сторінка з розкладом, що містить оновлену інформацію

Тестові сценарії для вчителя наведено у таблиці 4.3.

Таблиця 4.3 - Тестові сценарії для вчителя

	Модуль	Чек-лист	Очікуваний результат
Вчитель	Сторінка групи	1 Натиснути на кнопку Групи на навігаційному меню	1 Відкривається сторінка з переліком класів та кнопкою для додавання нового класу
		2 Натиснути іконку Учні будь-якого класу	2 Відкривається сторінка з переліком учнів поточного класу
		3 Натиснути кнопку Розклад навпроти будь-якого класу	3 Відкривається сторінка з розкладом вибраного класу
		4 Натиснути кнопку Журнал навпроти будь-якого класу	4 Відкривається сторінка з журналом вибраного класу

Продовження таблиці 4.3

Вчитель	Сторінка журнал	1 Натиснути на випадаючий список із дисциплінами	1 Відобразиться список із дисциплінами класу
		2 Обрати дисципліну зі списку	2 Відобразиться журнал класу з обраного предмету за поточний місяць, відобразиться домашнє завдання на поточний день, якщо воно було додано
		3 Натиснути кнопку зі стрілочкою в лівому напрямку	3 Відобразиться сторінка журналу за попередній місяць
		4 Натиснути кнопку зі стрілочкою в правому напрямку	4 Відобразиться сторінка журналу за наступний місяць
		5 Натиснути на поле "Оберіть дату дедлайну"	5 Відобразиться календар, доступними для вибору будуть лише ті дні, у які по розкладу проводиться заняття
		6 Обрати день із доступних	6 Відкриється модальне вікно із заповненим полем дедлайну
		7 Заповнити завдання для модального вікна	7 Дані в полі Завдання відображаються
		8 Натиснути кнопку Додати	8 Домашнє завдання додається в базу даних
		9 Натиснути на будь-яку порожню клітинку журналу	9 Відкриється сторінка із додаванням оцінки
		10 Увести бал від 1 до 12	10 Введена оцінка відобразиться
		11 Натиснути кнопку Додати	11 Оцінка додається, відкривається сторінка із журналом з оновленими оцінками

Тестові сценарії для класного керівника наведено у таблиці 4.4.

Таблиця 4.4 - Тестові сценарії для класного керівника

	Модуль	Чек-лист	Очікуваний результат
Класний керівник	Сторінка Журнал	1 Натиснути кнопку "Сформувати табель для групи"	1 Відкриється модальне вікно, де потрібно ввести додаткову інформацію
		2 Вести термін підрахунку підсумкової оцінки	2 Введена інформація відобразиться
		3 Натиснути кнопку "Сформувати"	3 Для кожного студента групи сформується табель з підсумковими оцінками за певний період
	Сторінка "Моя група"	1 Натиснути на кнопку "Моя група" на навігаційному меню	1 Відобразиться сторінка із категоріями для класу
		2 Натиснути іконку Учні класу	2 Відкривається сторінка з переліком учнів класу
		3 Натиснути кнопку Розклад	3 Відкривається розклад класу
		4 Натиснути кнопку Журнал класу	4 Відкривається сторінка з журналом класу

Продовження таблиці 4.4

Класний керівник	Сторінка Учнів класу	1 Натиснути на іконку батьків навпроти будь-якого учня	1 Відкриється сторінка із переліком батьків учня
		2 Натиснути іконку із повідомленням навпроти учня	2 Відкриється модальне вікно
		3 Заповнити інформацію у модальному вікні	3 Інформація відобразиться
		4 Натиснути кнопку "Додати повідомлення"	4 Введене повідомлення збережеться
		5 Натиснути кнопку "Написати повідомлення учням"	5 Відкриється модальне вікно
		6 Заповнити інформацію у модальному вікні	6 Введена інформація відобразиться
		7 Натиснути кнопку "Додати повідомлення"	7 Введене повідомлення збережеться
	Сторінка із переліком батьків	1 Натиснути іконку із повідомленням навпроти батьків	1 Відкриється модальне вікно
		2 Заповнити інформацію у модальному вікні	2 Інформація відобразиться
		3 Натиснути кнопку "Додати повідомлення"	3 Введене повідомлення збережеться

Тестові сценарії для батьків та учнів наведено у таблиці 4.5.

Таблиця 4.5 - Тестові сценарії для учня та батьків

	Модуль	Чек-лист	Очікуваний результат
Учень/батьки	Сторінка Щоденник	1 Обрати пункт "Щоденник" на навігаційному меню	1 Відобразиться сторінка із щоденником учня
		2 Натиснути кнопку зі стрілочкою в лівому напрямку	2 Відобразиться вміст щоденника за попередній тиждень
		3 Натиснути кнопку зі стрілочкою в правому напрямку	3 Відобразиться вміст щоденника за наступний тиждень
	Сторінка Розклад	1 Обрати пункт "Мій розклад" на навігаційному меню	1 Відобразиться сторінка із розкладом учня
	Сторінка Мої табелі	1 Обрати пункт "Мої табелі" на навігаційному меню	1 Відобразиться сторінка із переліком табелів учня
		2 Натиснути на будь-який табель	2 Відобразиться сторінка із обраним табелем учня із підсумковими оцінками
	Сторінка Домашнього завдання	1 Обрати пункт "Моє домашнє завдання" на навігаційному меню	1 Відобразиться сторінка із переліком домашніх завдань учня
		2 Поставити галочку навпроти будь-якого домашнього завдання	2 Галочка про виконання відобразться, відсоток готовності збільшиться
		3 Забрати галочку навпроти будь-якого домашнього завдання	3 Галочка не буде відображатися, відсоток готовності зменшиться
	Сторінка повідомлень	1 Обрати пункт із іконкою у вигляді дзвоника на навігаційному меню	1 Відобразиться сторінка із переліком всіх повідомлень адресованих учню

1.3 Результати тестування програмного забезпечення

Ручне тестування, як вже було сказано, унеможливлює відправлення некоректних даних на сервер. Завдяки цьому виключається можливість отримання помилки при роботі з базою даних та самою програмою.

Валідація при реєстрації є обов'язковою, оскільки в даному прикладі є введення двох однакових полів, тобто пароль та підтвердження паролю, і вони не можуть відрізнятися. Так само потрібно перевірити адресу на коректність (рисунок 4.1).

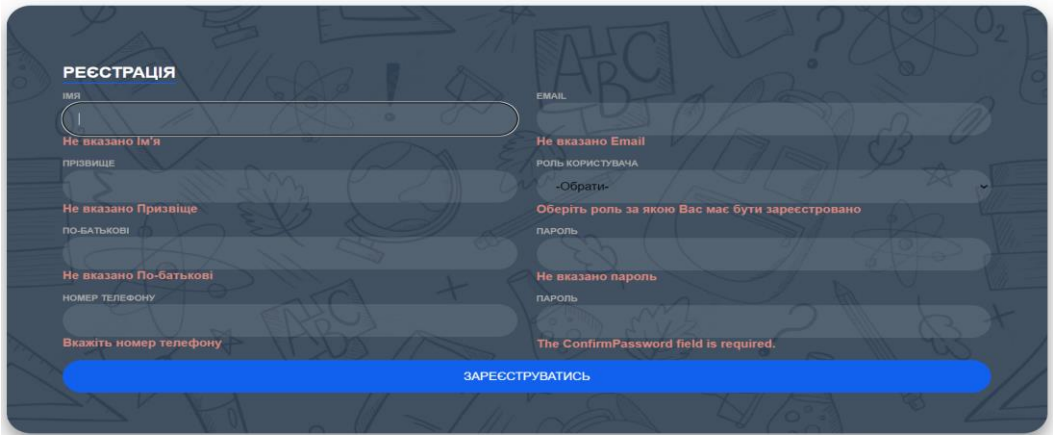


Рисунок 4.1 – Валідація даних при реєстрації

При авторизації користувач проходить аутентифікацію, тобто при веденні логіну, пароль повинен співпадати із тим, що вже записаний у базі даних. Тому валідація, тут також, є дуже важливою (рисунок 4.2).

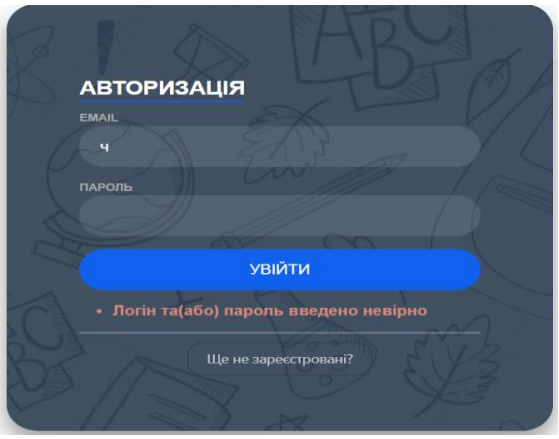


Рисунок 4.2 – Валідація даних при авторизації

Також валідація присутня і в інших формах де вводяться якісь дані.

Для перевірки за допомогою модульного тестування, було написано додаткову програму, яка містить тестові функції, що перевіряють методи контролера основної програми [12].

Приклад класу тестів для контролера Subject:

```
    public class SubjectControllerTests
    {
        [Fact]
        public void IndexReturnsAViewResultWithAListOfSubjects()
        {
            // Arrange
            var mock = new Mock<IAppDBContext>();
            mock.Setup(repo => repo.GetSubjects()).Returns(GetTestSubjects());
            var controller = new SubjectController(mock.Object);

            // Act
            var result = controller.Index();

            // Assert
            var viewResult = Assert.IsType<ViewResult>(result);
            var model = Assert.IsAssignableFrom<List<Subject>>(viewResult.Model);
            Assert.Equal(GetTestSubjects().Count, model.Count());
        }
        [Fact]
        public void EditReturnsAViewResultWithAnSubject()
        {
            // Arrange
            var mock = new Mock<IAppDBContext>();
            mock.Setup(repo =>
repo.GetOneSubject(It.IsAny<int>())).Returns(GetTestSubjects()[1]);
            var controller = new SubjectController(mock.Object);

            // Act
            var result = controller.Edit(1);

            // Assert
            var viewResult = Assert.IsType<ViewResult>(result);
            var model = Assert.IsAssignableFrom<Subject>(viewResult.Model);
        }
        [Fact]
        public void EditNullIdReturnnsNotFoundException()
        {
            // Arrange
            var mock = new Mock<IAppDBContext>();
            mock.Setup(repo =>
repo.GetOneSubject(It.IsAny<int>())).Returns(GetTestSubjects()[1]);
            var controller = new SubjectController(mock.Object);

            // Act
            var result = controller.Edit(0);

            // Assert
            Assert.IsType<NotFoundResult>(result);
        }
        [Fact]
        public void CreateReturnsAView()
        {
            // Arrange
            var mock = new Mock<IAppDBContext>();
```

```
        var controller = new SubjectController(mock.Object);

        // Act
        var result = controller.Create();

        // Assert
        var viewResult = Assert.IsType<ViewResult>(result);
    }

    private List<Subject> GetTestSubjects()
    {

        var users = new List<Subject>
        {
            new Subject { Id=1, Name="Історія України", CreatedDateTime = DateTime.Now},
            new Subject { Id=2, Name="Алгебра", CreatedDateTime = DateTime.Now},
            new Subject { Id=3, Name="Хімія", CreatedDateTime = DateTime.Now}
        };
        return users;
    }    }
```

Результати модульного тестування для даного та інших класів зображено на рисунку 4.3.

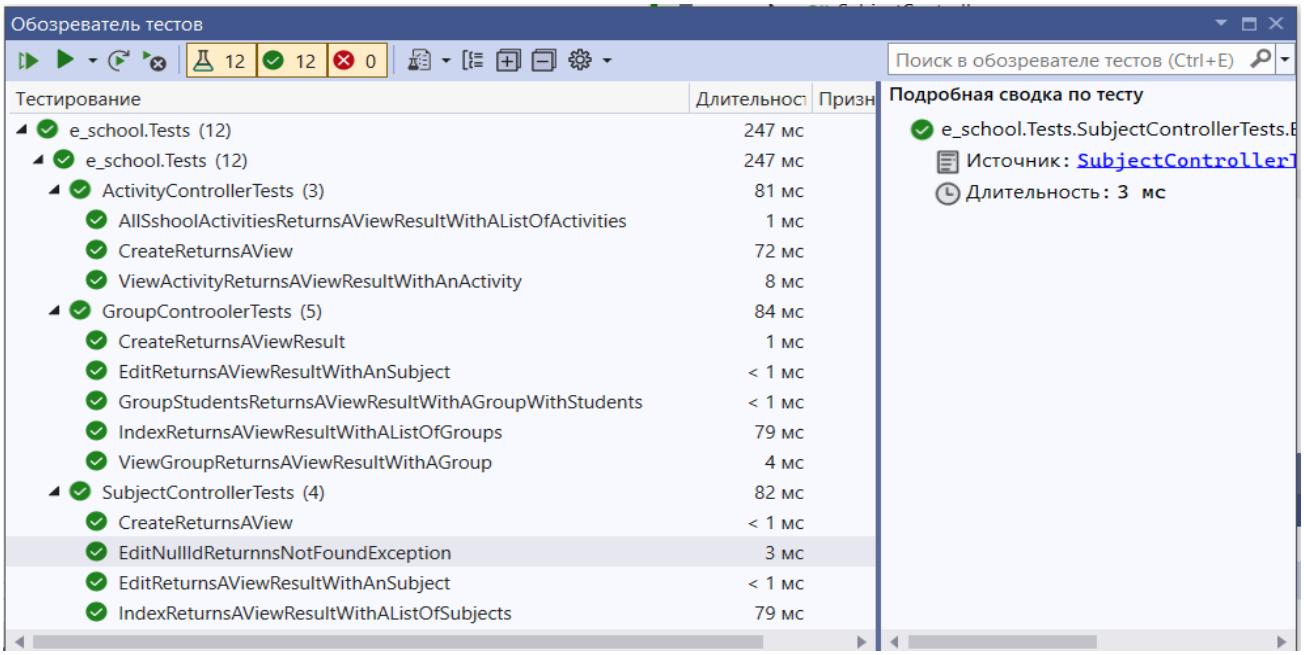


Рисунок 4.3 – Результати модульного тестування

На 4 етапі проводилося тестування програмного забезпечення. Було описано всі тестові сценарії для кожного користувача системи. Після цього проведене ручне тестування та написання модульних тестів. Результатом даного етапу є протестований продукт, готовий до використання.

ВИСНОВКИ

Після виконання всіх завдань дипломного продукту, було отримано повноцінний багатофункціональний сайт для організації учбового процесу.

Під час розробки проводилося ознайомлення із предметною областю продукту та здійснювався змістовний її аналіз. Внаслідок дослідження ринку програмного забезпечення та аналізу схожих розробок, було визначено функціональні вимоги до даного програмного рішення. Для даної системи розроблено технічне завдання, що включає в себе постановку задачі, визначення етапів та стадій робіт. У відповідності до різних категорій проводилося встановлення вимог, виконання яких в подальшому дасть змогу виконати розробку якісного продукту із забезпеченням необхідного функціоналу.

Аналіз схожих розробок дозволив визначити унікальність продукту. Сайт, що розроблятиметься забезпечить коректну організацію учбового процесу школи, із наданням всього необхідного функціоналу, а саме ведення щоденника, заповнення розкладів, генерація домашнього завдання та виставлення оцінок, ведення журналу. Дане програмне забезпечення дозволить швидко та легко перенести навчальний процес у он-лайн, при цьому він буде ідентичний із навчанням оф-лайн.

Під час другого етапу, тобто проектування програмного забезпечення, відбулося формулування основних принципів роботи продукту. Було визначено, що сайт буде побудовано за клієнт-серверною архітектурою та розроблений за принципом MVC. Також проведено проектування бази даних, всіх таблиць та зв’язків між ними. Було створено різнопланові макети сторінок, що забезпечило можливість проведення аналізу, яким чином реалізувати інтерфейс, щоб він був простим, зручним та зрозумілим. Після аналізу доступних технологій та мов програмування, прийнято рішення, що дане програмне забезпечення буде розроблятися за допомогою фреймворку ASP.Net Core та мови програмування C#.

Наступним кроком стало написання самого програмного продукту, за допомогою визначених технологій. В процесі відбулась реалізація адміністративної та користувацької частини. Для кожної з ролей було написано ряд функцій, та розроблено інтерфейс, що є простим та зручним.

Для розуміння, які функції доступні на даному сайті, в роботі надано інструкцію з експлуатації, яка поділяється на декілька частин, відповідно до користувачів системи.

Після написання програмного забезпечення, здійснювалось ручне та модульне тестування, що показало готовність програмного продукту до експлуатації. Також, у веб-додатку присутня валідація, що забезпечує захист системи від введення некоректних даних, та помилок пов'язаних із цим. Модульне тестування показало, що всі методи та функції працюють коректно, тому не повинно виникнути ніяких проблем.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Організація процесу навчання у вищій школі [Електронний ресурс]. / pidru4niki – Режим доступу до ресурсу: https://pidru4niki.com/88905/pedagogika/organizatsiya_protsesu_navchannya_vischiy_shkoli. (дата звернення – 29.04.2021) – Назва з екрану.

2. Найкращі LMS 2020, і як обрати систему дистанційного навчання для вашого бізнесу [Електронний ресурс]. / code-maze – Режим доступу до ресурсу: https://evergreens.com.ua/ua/articles/best-lms-2020.html. (дата звернення – 29.04.2021) – Назва з екрану.

3. Архитектура клиент-сервер [Електронний ресурс]. / org2.knuba – Режим доступу до ресурсу: https://org2.knuba.edu.ua/pluginfile.php/8977/mod_resource/content/2/cli-se.pdf. (дата звернення – 29.04.2021) – Назва з екрану.

4. Views, Partial Views, and Layouts in ASP.NET Core MVC [Електронний ресурс]. / code-maze – Режим доступу до ресурсу: https://code-maze.com/views-partial-views-and-layouts-in-asp-net-core-mvc. (дата звернення – 29.04.2021) – Назва з екрану.

5. Системи керування базами даних MVC [Електронний ресурс]. / miyklas – Режим доступу до ресурсу: https://miyklas.com.ua/p/informatica/9-klas/bazi-danikh-sistemi-keruvannia-bazami-danikh-361840/sistemi-keruvannia-bazami-danikh-352450/re-b0f14256-7ac2-4823-9f53-656061b85eaa. (дата звернення – 29.04.2021) – Назва з екрану.

6. Різниця між логічною та фізичною моделлю даних [Електронний ресурс]. / sawakinome – Режим доступу до ресурсу: https://ua.sawakinome.com/articles/technology/difference-between-logical-and-physical-data-model-2.html /. (дата звернення – 29.04.2021) – Назва з екрану.

7. Чому варто вивчати мову програмування C# [Електронний ресурс]. / quality-assurance-group – Режим доступу до ресурсу: https://www.quality-assurance-group.com/chomu-varto-vyvchaty-movu-programuvannya-c-c-sharp/. (дата звернення

– 29.04.2021) – Назва з екрану.

8. [Електронний ресурс]. / sawakinome – Режим доступу до ресурсу: https://bootstrapdocs.com/v3.0.3/docs /css/. (дата звернення – 29.04.2021) – Назва з екрану.

9. ASP.NET Core Razor Pages: How to implement AJAX requests [Електронний ресурс]. / thereformedprogrammer – Режим доступу до ресурсу: https://www.thereformedprogrammer.net/asp-net-core-razor-pages-how-to-implement-ajax-requests/. (дата звернення – 29.04.2021) – Назва з екрану.

10. ASP .Net Core MVC partial views [Електронний ресурс]. / c-sharpcorner – Режим доступу до ресурсу: https://www.c-sharpcorner.com/article/asp-net-core-2-0-mvc-partial-views/ (дата звернення – 29.04.2021) – Назва з екрану.

11. Unit Testing With xUnit And Moq In ASP.NET Core [Електронний ресурс]. / c-sharpcorner – Режим доступу до ресурсу: https://www.c-sharpcorner.com/article/unit-testing-with-xunit-and-moq-in-asp-net-core/. (дата звернення – 29.04.2021) – Назва з екрану.

12. Unit testing C# in .NET Core using dotnet test and xUnit [Електронний ресурс]. / docs.microsoft – Режим доступу до ресурсу: https://docs.microsoft.com/en-us/dotnet/core/testing/unit-testing-with-dotnet-test. (дата звернення – 29.04.2021) – Назва з екрану.

ДОДАТОК А

(Обов’язковий)

ТЕХНІЧНЕ ЗАВДАННЯ

1. Загальні відомості

1.1. Найменування програмного забезпечення

Найменування розробки: «Програмне забезпечення для організації, управління та впорядкування навчального процесу у загальноосвітніх школах»

1.1. Область застосування

Організація учбового процесу в школах.

1.2. Підстави для розробки

Підставою для розробки є «Завдання на дипломний проект», затверджене завідувачем кафедри інженерії програмного забезпечення.

2. Призначення та цілі створення системи

Критерії ефективності та якості програми:

– система повинна бути простою, не вимагати додаткових знань та навичок з боку користувачів, оскільки використання передбачене для дітей;

– час відповіді системи повинен бути оптимальним у визначених рамках, а саме від 5 до 15 сек. в залежності від того, наскільки складний запит;

– система має бути надійною, не повинна давати збоїв та помилок.

Цілі розробки програми

Проект створюється з метою забезпечення організації учбового процесу в загальноосвітніх школах.

Основна функція системи – це можливість роботи учбового закладу в режимі он-лайн. Тобто, це забезпечення роботи із формуванням списків всіх наявних класів, формування журналу класу, його наповнення та виведення оцінок. Також передбачається організація щоденника для учнів, що забезпечить розсилку домашнього завдання та оцінок учнів. Має бути забезпечена зручна робота із системою. У користувача не повинно виникати питань при роботі із сайтом.

3. Вимоги до системи

3.1. Вимоги до функціональних характеристик:

– авторизація користувачів в системі: має бути можливість реєстрації користувача та присвоєння йому відповідної ролі. В іншому випадку користувач є неавторизованим і не може користуватися багатьма функціями сайту;

– виведення списку всіх класів та основних даних про них;

– виведення списку студентів кожного класу;

– виведення розкладу кожного класу;

– можливість редагування розкладу чи додавання у випадку, якщо його ще немає (така можливість надається адміністратору);

– виведення викладачів та предметів, які вони викладають;

– можливість створення нових викладачів та дисциплін, а також вчителю певних предметів (ця функція доступна для адміністратора);

– додавання домашнього завдання викладачем, після чого робиться автоматична розсилка учням певного класу;

– заповнення оцінок викладачами, після чого здійснюється оновлення журналу та щоденника учнів;

– можливість автоматичного формування щоденника на основі розкладу, а також домашнього завдання та оцінок конкретного учня;

– здійснення виведення щоденника для учнів та батьків;

– написання повідомлень батькам у разі необхідності;

– виведення чек-листа із домашніми заваданнями для учня;

– виведення всіх повідомлень батькам.

3.2. Вимоги до надійності

Розроблюване ПЗ повинно:

– автоматично відновлюватись після збоїв;

– при запуску сайту мати парольний захист;

– давати змогу обмеження доступу до сторінок;

– доступ повинен здійснюватись за ролями.

Час відновлення після відмови, повинен складатися із часу завантаження сторінки сайту, часу перезапуску сервера БД, часу перезапуску браузера.

3.3. Умови експлуатації:

– в системі має бути користувач, що буде відповідати за адміністративну роль (заповнення списків груп, додавання вчителів, предметів);

– адміністратор має бути ознайомленим із принципами роботи сайту, як адміністративної так і користувацької частини, а також повинен вводити правдиві дані для коректної роботи сайту;

– кількість користувачів не має обмежень;

– для роботи із сайтом має бути наявність браузеру, будь якої операційної системи із графічним інтерфейсом та доступом до інтернету;

– особливих вимог до навколишнього середовища немає.

3.4. Вимоги до складу і параметрів технічних засобів

Система повинна задовольняти вимогам на комп’ютері, що знаходиться в наступній мінімальній комплектації:

– 128Мб пам’яті і вище;

– процесор з тактовою частотою 600MHz і вище;

– має бути мережева плата або wifi адаптер.

3.5. Вимоги до інформаційної і програмної сумісності:

– сумісний із будь-якою операційною системою, де є можливість доступу до Інтернету;

– сумісний із будь-яким браузером;

– інформація, що представлена на сайті повинна бути легкосприйнятною та читабельною.

3.6. Вимоги до транспортування та зберігання:

– надається доступ до репозиторію із вихідними даними системи;

– програмна документація надається в електронному та друкованому вигляді;

– умови експлуатації програмного забезпечення схожі з умовами експлуатації ПК.

3.7. Вимоги до мов програмування

Сайт розробляється за допомогою методів мови C# та фреймворку ASP.Net Core. Дане середовище розробки обране в результаті аналізу функціональних можливостей різних середовищ розробки. Воно має такі переваги:

– висока ефективність і швидкість виконання коду;

– наочність та зручність інтерфейсу;

– об’єктно-орієнтованість мови.

4. Вимоги до програмної документації

В ході розробки програми повинні бути підготовлені: текст програми, опис програми, програма і методика випробувань, керівництво користувача.

З огляду на обсяг проекту завдання передбачається вирішувати поетапно. При цьому модулі ПЗ, створені в різний час, повинні передбачати можливість нарощування системи і бути сумісні один з одним, тому документація на прийняте експлуатаційне ПЗ повинна містити повну інформацію.

Програмна документація повинна включати наступні відомості:

– «Інструкція користувачів» складається з опису послідовності роботи сайту, основних режимів роботи, опису основних сторінок сайту, порядку виконання завдань в системі;

– «Інструкція адміністратора», складається з опису кроків для додавання предметів, викладачів та ін.

5 Стадії і етапи розробки програмного продукту

Таблиця А1 – Стадії і етапи розробки програмного продукту

Стадія	Етапи робіт	Опис робіт
Дослідження предметної області	Аналіз предметної області. Огляд існуючих рішень. Розробка технічного завдання.	На цій стадії виконуються роботи які передують проектуванню програмного забезпечення. Розробляється технічне завдання та визначаються основні вимоги до системи. Виконується аналіз схожого існуючого програмного забезпечення.
Проектування системи	Проектування архітектури, інтерфейсу, бази даних. Вибір технологій.	Ця стадія передує кодуванню програмного забезпечення. Виконується детальне проектування всіх компонентів системи, на основі якого створюється проект.
Кодування	Кодування всіх елементів системи.	Після виконання цієї стадії, результатом є готовий, повнофункціональний програмний продукт.
Тестування програмного продукту	Ручне тестування. Модульне тестування.	Тестування виконується для перевірки на помилки чи неточності при роботі з сайтом. Після чого, ці недоліки виправляються та проводиться тестування до тих пір, поки всі елементи системи не будуть працювати правильно.
Оформлення програмної документації		Документація формується у вигляді детального звіту, де описується кожний крок чи етап роботи над програмним продуктом. Для кращого сприйняття окремих етапів будуються діаграми, на яких зображено принципи функціонування системи.

6 Порядок контролю та приймання

Види випробувань

Дане програмне забезпечення буде тестуватися двома способами, а саме: перевірка на коректність введених даних та модульне тестування.

Модульне тестування проводиться для кожного модуля окремо. Модулем може бути будь-який метод, функція, тобто, це найменша частинка програми, яка може бути протестована. Дане тестування виконується, щоб перевірити чи кожен модуль має очікувану поведінку.

Щодо ручного тестування, то воно необхідне для перевірки, як спрацьовує валідація даних, тобто наскільки користувач проінформований про помилки при введені полів. І чи програмне забезпечення не допускає невірного введення даних.

Загальні вимоги до приймання:

– проект повинен успішно пройти всі види тестувань;

– в результаті тестування повинна бути перевірена повна справність системи;

– у випадку неуспішного виконання певного тестування, проект надсилається на доопрацювання.

7 Етапи впровадження

Після проходження всіх тестів програмний продукт переходить в експлуатацію. Реалізація даного етапу зазвичай означає успішне впровадження. В подальшому передбачається супровід програмного забезпечення. Також система може вдосконалюватись та змінюватись.

ДОДАТОК Б

(Обов’язковий)

Діаграма варіантів використання

Рисунок Б.1- - Діаграма варіантів використання для адміністратора

Рисунок Б.2 - Діаграма варіантів використання для вчителя

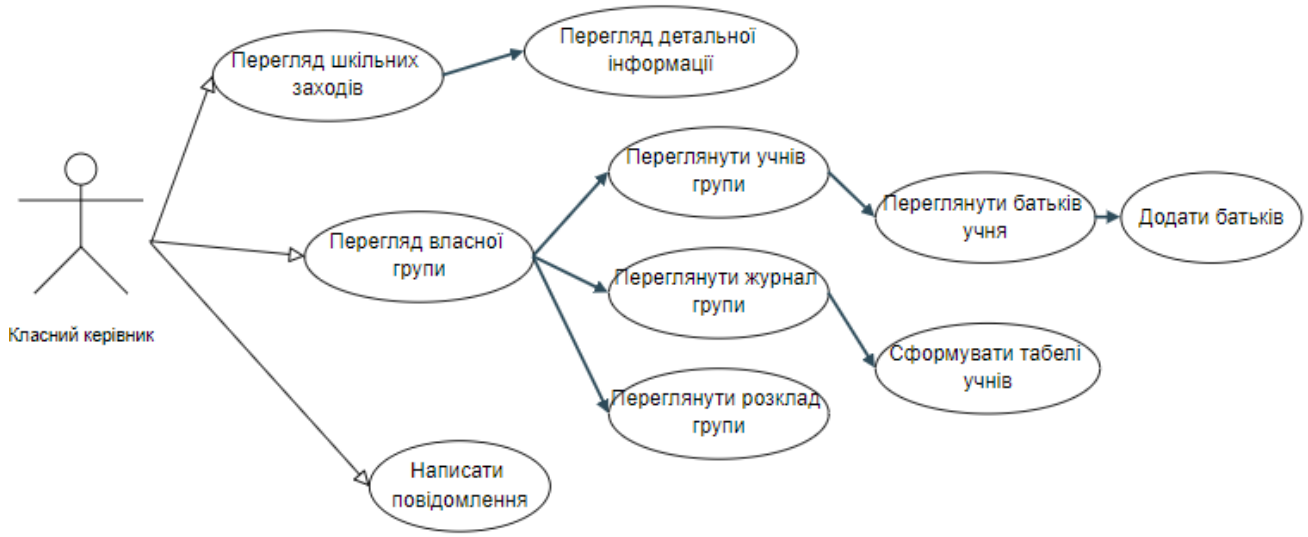


Рисунок Б.3 - Діаграма варіантів використання для класного керівника

Рисунок Б.4 - Діаграма варіантів використання для учня

Рисунок Б.5 - Діаграма варіантів використання для батьків

ДОДАТОК В

(Обов’язковий)

Діаграма розгортання

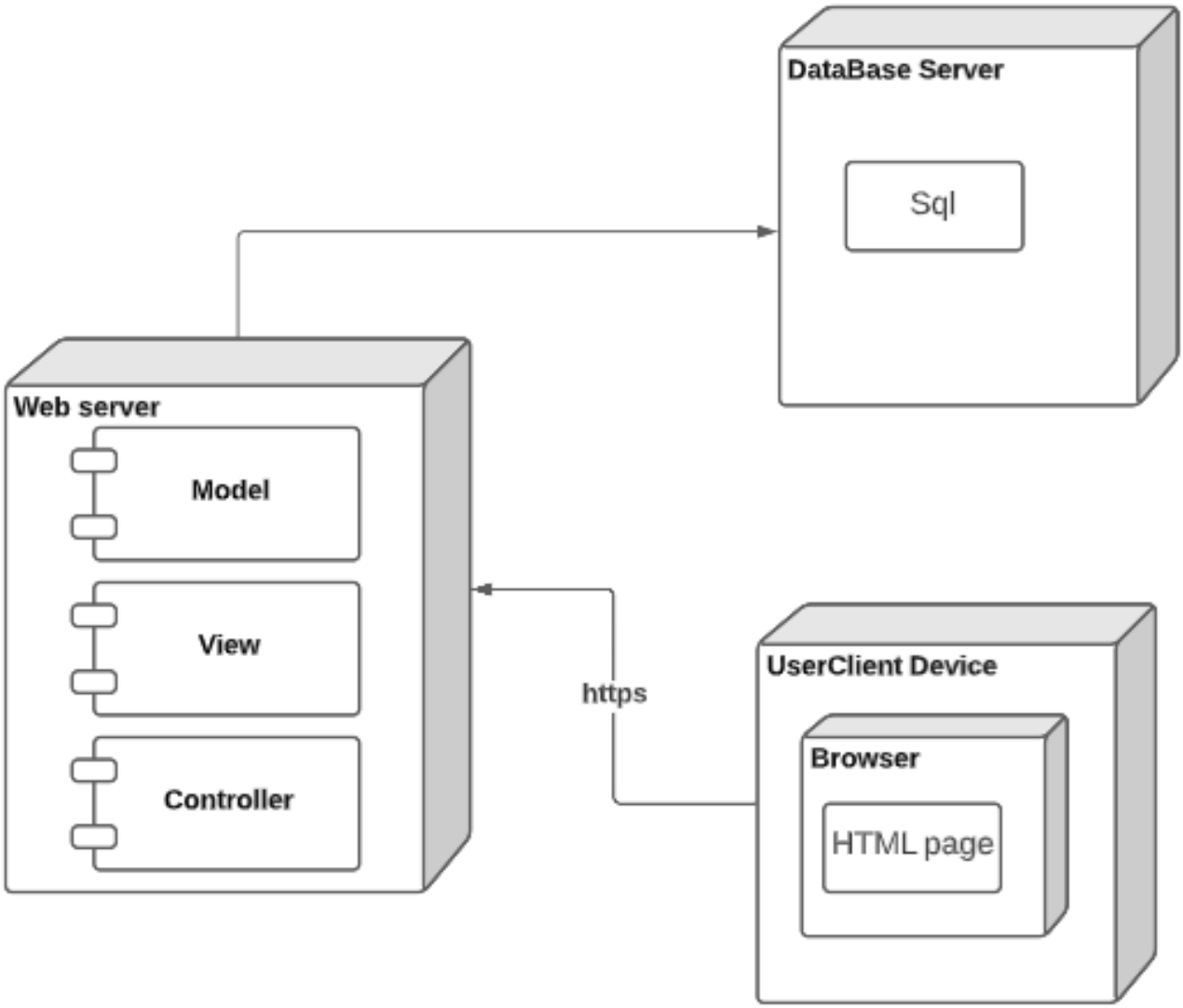


Рисунок В.1 - Діаграма розгортання

ДОДАТОК Г

(Обов’язковий)

Діаграма послідовності

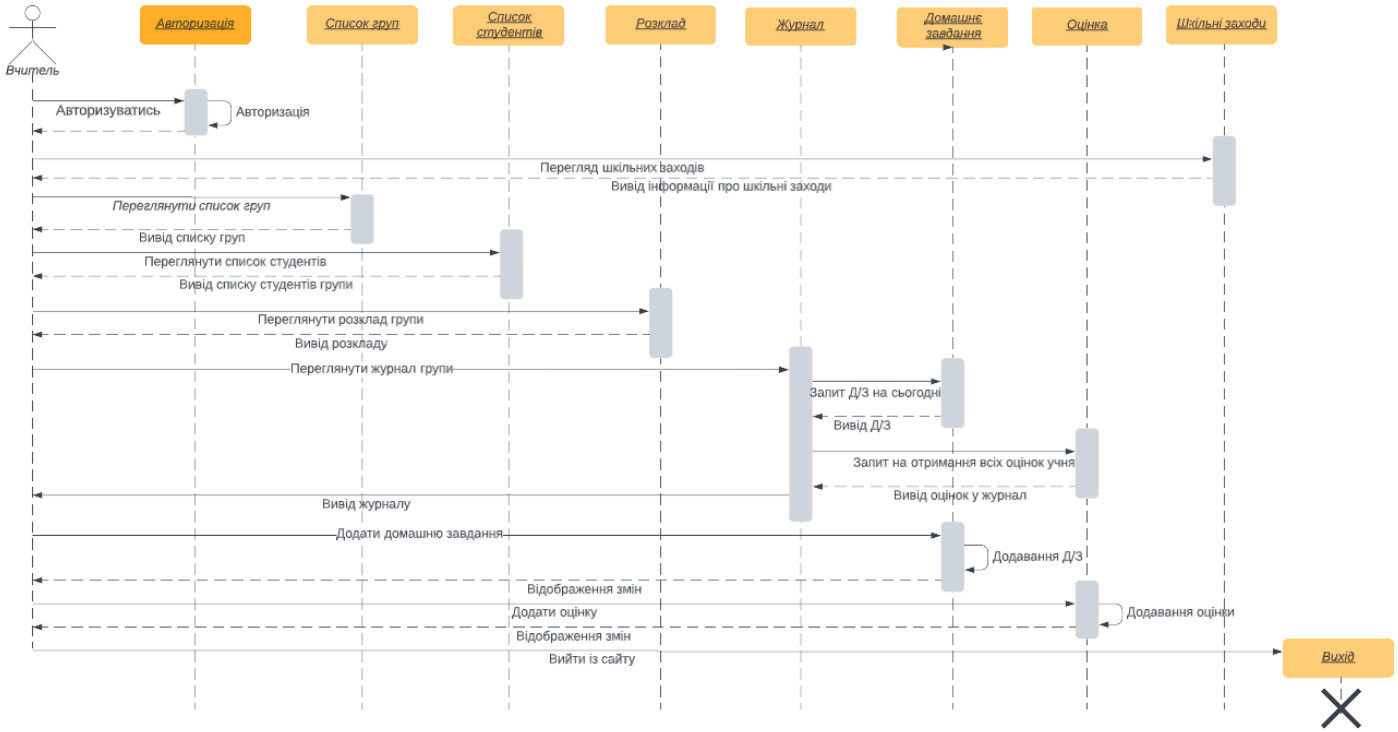


Рисунок Г.1 - Діагарма послідовності для вчителя

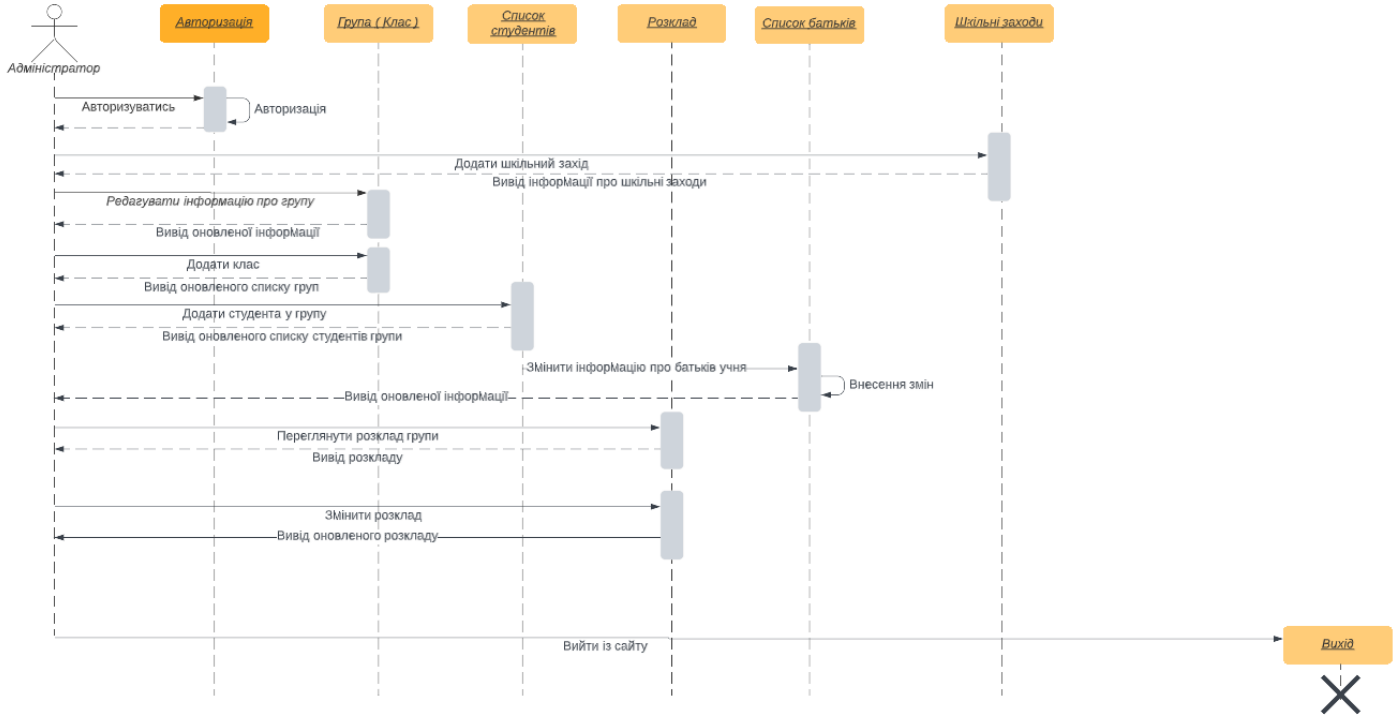


Рисунок Г.2 - Діаграма послідовності для адміністратора

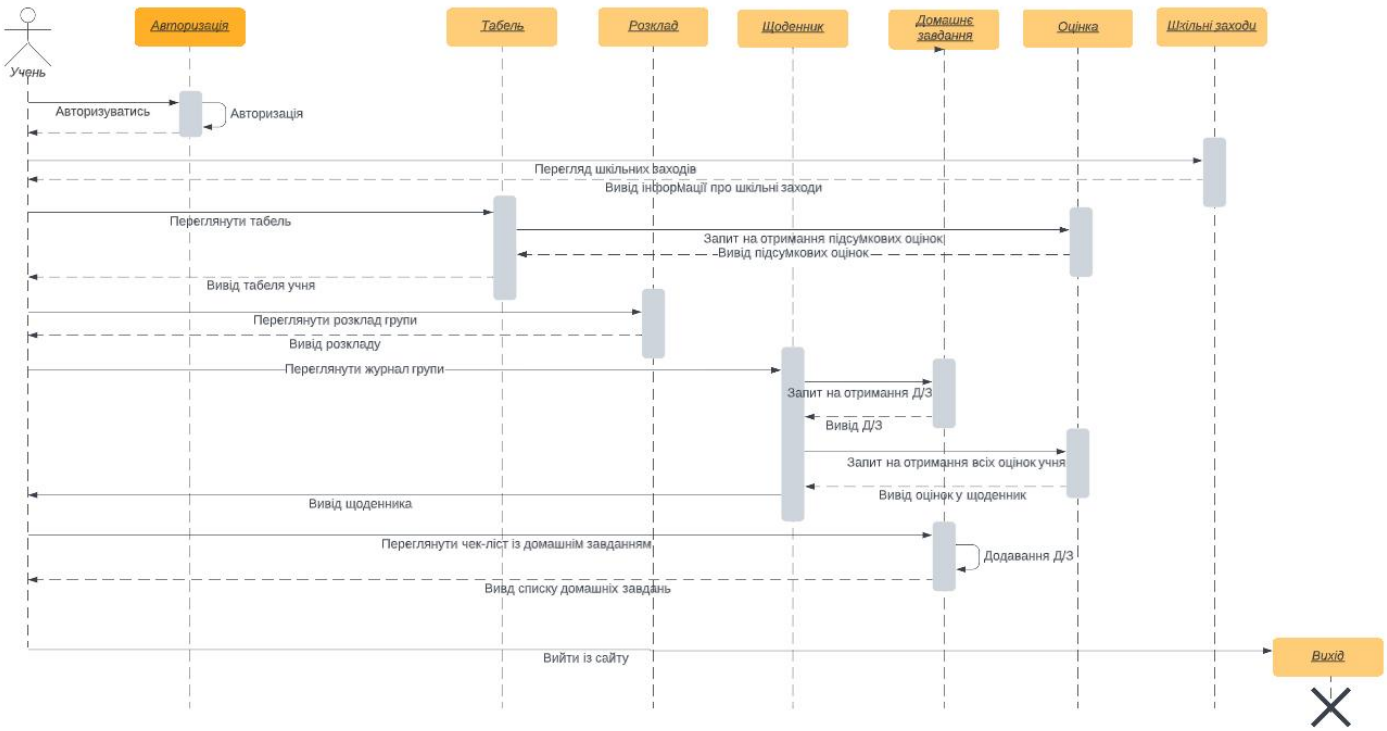


Рисунок Г.3 - Діаграма послідовності для учня

ДОДАТОК Д

(Обов’язковий)

Схема бази даних

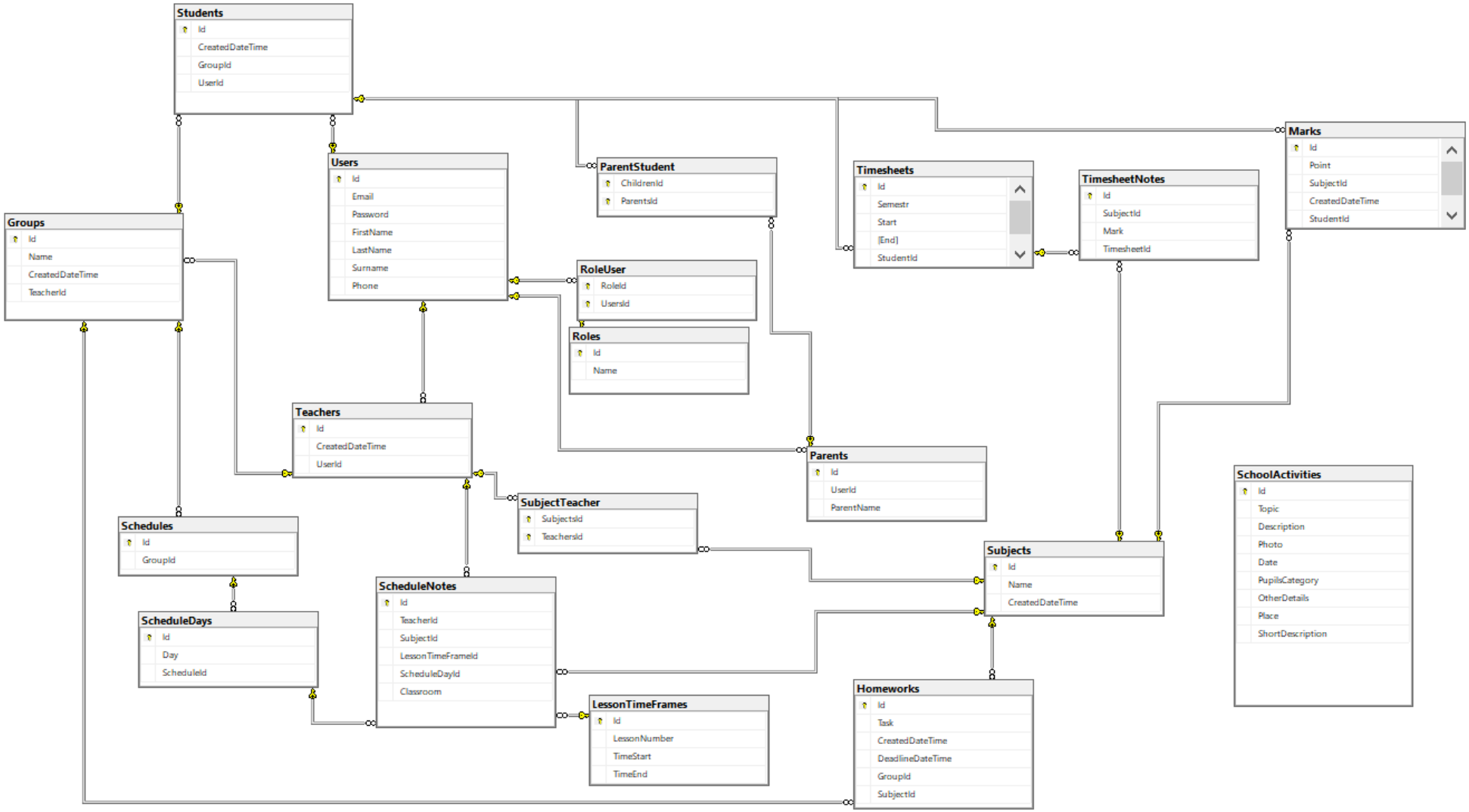


Рисунок Д.13 - Схема бази даних

ДОДАТОК Е

(Обов’язковий)

Діаграма Активностей

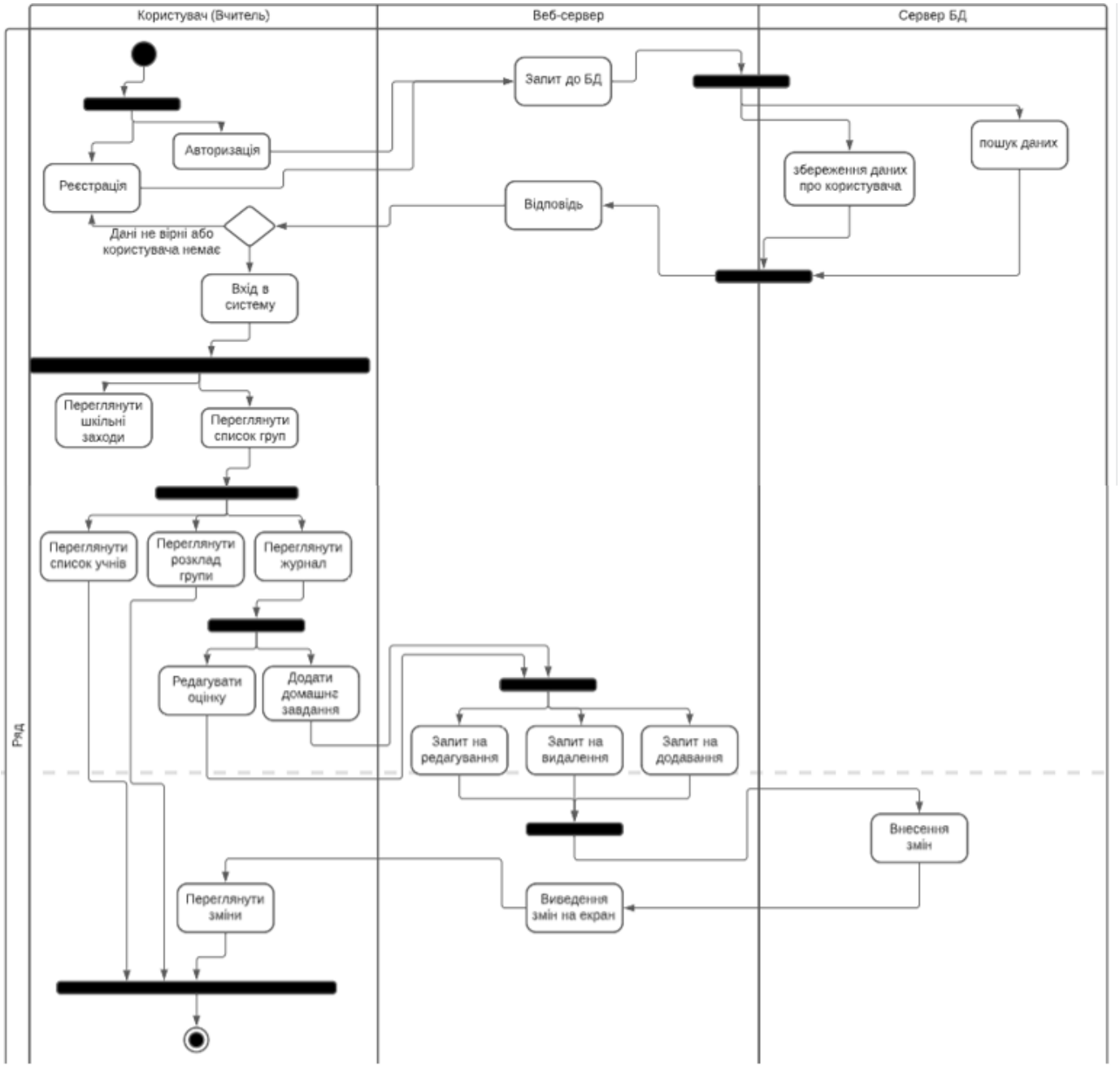


Рисунок Е.1 - Діаграма активностей для вчителя

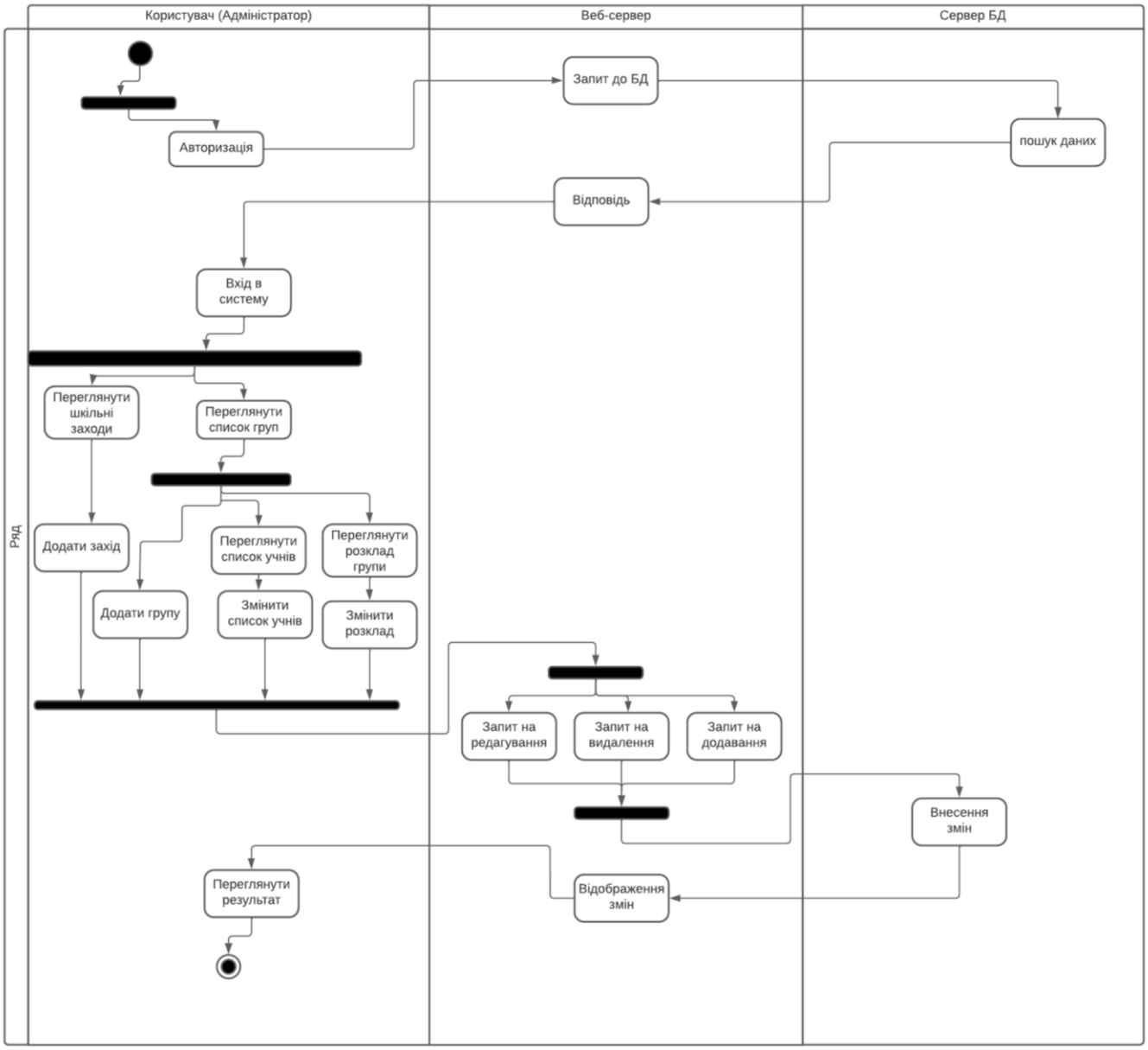


Рисунок Е.2 - Діаграма активностей для адміністратора

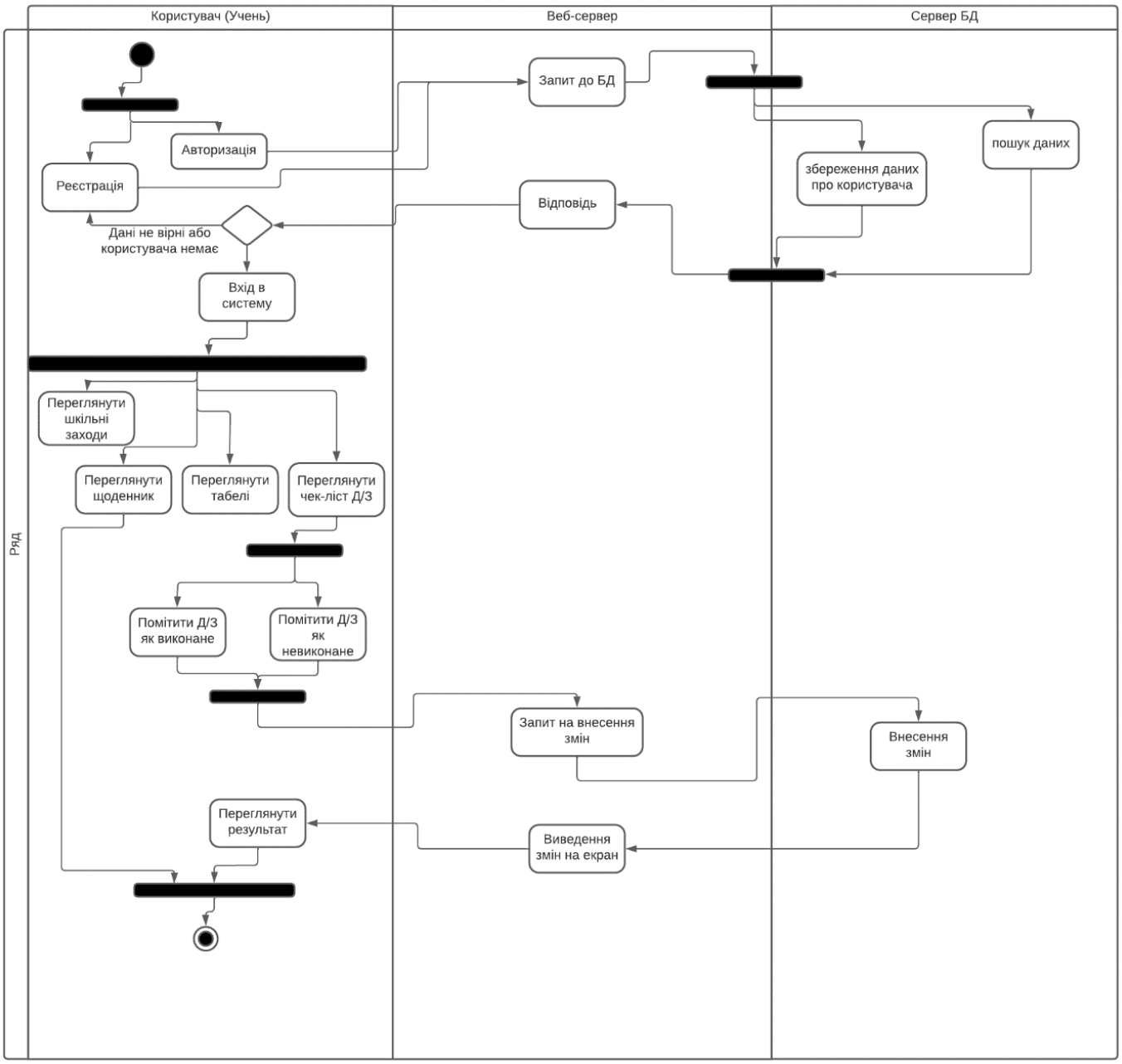


Рисунок Е.3 - Діаграма активностей для учня

ДОДАТОК Ж

(Обов’язковий)

Код (лістинг) програми

Controllers

```
using e_school.Data;
using e_school.Models;
using e_school.Models.VM;
using Microsoft.AspNetCore.Mvc;

namespace e_school.Controllers
{
    public class ActivityController : Controller
    {
        private readonly IAppDBContext _db;
        public ActivityController(IAppDBContext db)
        {
            _db = db;
        }

        public IActionResult Create()
        {
            return View();
        }
        [HttpPost]
        [ValidateAntiForgeryToken]
        public IActionResult Create(SchoolActivityVM schoolActivity)
        {
            SchoolActivity schoolActivity1 = new SchoolActivity();
            schoolActivity1.Topic = schoolActivity.Topic;
            schoolActivity1.Description = schoolActivity.Description;
            schoolActivity1.Date = schoolActivity.Date;
            schoolActivity1.PupilsCategory = schoolActivity.PupilsCategory;
            schoolActivity1.ShortDescription = schoolActivity.ShortDescription;
            schoolActivity1.OtherDetails = schoolActivity.OtherDetails;
            schoolActivity1.Place = schoolActivity.Place;
            if (schoolActivity.Photo != null)
            {
                byte[] imageData = null;
                // считываем переданный файл в массив байтов
                using (var binaryReader = new
BinaryReader(schoolActivity.Photo.OpenReadStream()))
                {
                    imageData = binaryReader.ReadBytes((int)schoolActivity.Photo.Length);
                }
                // установка массива байтов
                schoolActivity1.Photo = imageData;

            }
            _db.SchoolActivities.Add(schoolActivity1);
            _db.SaveChanges();

            return RedirectToAction("AllSchoolActivities");
        }

        public IActionResult AllSchoolActivities()
        {
            List<SchoolActivity> schoolActivities = _db.GetSchoolActivities();
            return View(schoolActivities);
        }
        public IActionResult ViewActivity(int id)
        {
            SchoolActivity schoolActivitiy = _db.GetSchoolActivity(id);
            return PartialView(schoolActivitiy);
        }
    }
}
```

```
using e_school.Data;
using e_school.Models;
using e_school.Models.VM;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using System.Globalization;

namespace e_school.Controllers
{
    public class DiaryController : Controller
    {
        private readonly AppDBContext _db;
        public DiaryController(AppDBContext db)
        {
            _db = db;
        }
        [Authorize(Roles = "Student")]
        public IActionResult StudentDiary(int id, DateTime date)
        {
            System.Globalization.CultureInfo cultureinfo = new
System.Globalization.CultureInfo("en-US");
            DateTime dt = DateTime.Parse(date.ToString(), cultureinfo);
            Student student = _db.Students.Include(c=>c.Group).FirstOrDefault(c => c.Id
== id);
            var day = dt.DayOfWeek;
            switch (day)
            {
                case DayOfWeek.Sunday:
                    {
                        dt = dt.AddDays(-6);
                    }
                    break;
                case DayOfWeek.Friday:
                    {
                        dt = dt.AddDays(-4);
                    }
                    break;
                case DayOfWeek.Thursday:
                    {
                        dt = dt.AddDays(-3);
                    }
                    break;
                case DayOfWeek.Wednesday:
                    {
                        dt = dt.AddDays(-2);
                    }
                    break;
                case DayOfWeek.Tuesday:
                    {
                        dt = dt.AddDays(-1);
                    }
                    break;
                case DayOfWeek.Saturday:
                    {
                        dt = dt.AddDays(-5);
                    }
                    break;
                default:
                    { }
                    break;
            }
            Schedule schedule = _db.Schedules.Include(g => g.Group).Include(s =>
s.ScheduleDays).ThenInclude(n => n.ScheduleNotes).ThenInclude(t => t.Subject).Include(s
```

```
=> s.ScheduleDays).ThenInclude(n => n.ScheduleNotes).ThenInclude(t =>
t.LessonTimeFrame).Include(s => s.ScheduleDays).ThenInclude(n =>
n.ScheduleNotes).ThenInclude(t => t.Teacher).ThenInclude(u => u.User).FirstOrDefault(c =>
c.GroupId == student.GroupId);
            CultureInfo myCI = new CultureInfo("uk-UA");
            List<DiaryNoteViewModel> diaryNotes = new List<DiaryNoteViewModel>();
            List<DiaryDayViewModel> diaryDays = new List<DiaryDayViewModel>();
            foreach (var scday in schedule.ScheduleDays)
            {
                foreach(var note in scday.ScheduleNotes)
                {
                    Mark mark = new Mark();
                    //Homework homework = new Homework();
                    List<Mark> marks = _db.Marks.Where(b => b.StudentId == student.Id &&
b.SubjectId == note.SubjectId && b.CreatedDateTime !=null).ToList();
                    if (marks != null)
                    {
                        mark = marks.Find(b => b.CreatedDateTime.Date == dt.Date);
                    }

                    Homework homework = _db.Homeworks.Where(b => b.GroupId ==
student.GroupId && b.SubjectId == note.SubjectId && b.DeadlineDateTime ==
dt.Date).FirstOrDefault();

                    diaryNotes.Add(new DiaryNoteViewModel(note, mark, homework));
                }

                diaryDays.Add(new DiaryDayViewModel(dt,
myCI.DateTimeFormat.GetDayName(dt.DayOfWeek).ToString(), diaryNotes));
                diaryNotes = new List<DiaryNoteViewModel>();
                dt = dt.AddDays(1);
            }
            DiaryViewModel diaryPart1 = new DiaryViewModel();
            diaryPart1.DiaryDays = new List<DiaryDayViewModel>();
            DiaryViewModel diaryPart2 = new DiaryViewModel();
            diaryPart2.DiaryDays = new List<DiaryDayViewModel>();
            List<DiaryViewModel> diaryParts=new List<DiaryViewModel>();
            int i = 0;
            int maxLines = 0;
            foreach(var dDay in diaryDays)
            {
                if(maxLines < dDay.DiaryNotes.Count)
                {
                    maxLines = dDay.DiaryNotes.Count;
                }
                if (i < (diaryDays.Count+1) / 2)
                {
                    diaryPart1.DiaryDays.Add(dDay);
                }
                else
                {
                    diaryPart2.DiaryDays.Add(dDay);
                }
                i++;
            }
            ViewBag.Date = date;
            ViewBag.MaxLines = maxLines;
            diaryParts.Add(diaryPart1);
            diaryParts.Add(diaryPart2);
            return View(diaryParts);
        }


        }}
```

```
using e_school.Data;
using e_school.Models;
using e_school.Models.VM;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.Rendering;
using Microsoft.EntityFrameworkCore;
using System.Security.Claims;

namespace e_school.Controllers
{
    public class GroupController : Controller
    {

        private readonly IAppDBContext _db;
        public GroupController(IAppDBContext db)
        {
            _db = db;
        }
        public IActionResult Index()
        {
            //var roles = ((ClaimsIdentity)User.Identity).Claims.Where(c => c.Type ==
ClaimTypes.Role).Select(c => c.Value).ToList().Contains("Student");
            IEnumerable<Group> groupList = _db.GetGroups();
            return View(groupList);
        }

        public IActionResult ViewGroup(int groupId)
        {
            return View(_db.GetGroupById(groupId));
        }
        public IActionResult Create()
        {
            IEnumerable<Teacher> teacherList = _db.Teachers.Include(m=>m.User).ToList();
            TeacherViewModel model = new TeacherViewModel();
            model.TeachersList = _db.Teachers.Where(t=>_db.Groups.FirstOrDefault(y =>
y.TeacherId == t.Id) == null).Include(m => m.User).Select(x => new ItemList { Value =
x.Id, Text = x.User.LastName+" "+ x.User.FirstName +" "+ x.User.Surname }).ToList();
            ViewBag.teachers = model;

            return View();
        }

        [HttpPost]

        [ValidateAntiForgeryToken]
        public IActionResult Create(Group group)
        {
            /*if (ModelState.IsValid)
            {*/
            var teacher = _db.Teachers.FirstOrDefault(t => t.Id==group.TeacherId);
            var user = _db.Users.Include(r=>r.Role).FirstOrDefault(u =>
u.Id==teacher.UserId);
            user.Role.Add(_db.Roles.FirstOrDefault(c=>c.Name=="Curator"));
            _db.Users.Update(user);
                _db.Groups.Add(group);
                _db.SaveChanges();
                return RedirectToAction("Index");
           // }
            //return View(group);
        }
        public IActionResult Edit(int? id)
        {
            if (id == null || id == 0)
            {
```

```
                return NotFound();
            }
            IEnumerable<Teacher> teacherList = _db.Teachers.Include(m =>
m.User).ToList();
            TeacherViewModel model = new TeacherViewModel();
            model.TeachersList = _db.Teachers.Include(m => m.User).Select(x => new
ItemList { Value = x.Id, Text = x.User.LastName + " " + x.User.FirstName + " " +
x.User.Surname }).ToList();
            ViewBag.teachers = model;

            var category =
_db.Groups.Include(t=>t.Teacher).ThenInclude(u=>u.User).FirstOrDefault(c => c.Id == id);
            ViewBag.selectedTeacher = category.TeacherId;
            if (category == null)
            {
                category = new Group();
            }
            return View(category);

        }
        [HttpPost]
        [ValidateAntiForgeryToken]
        public IActionResult Edit(Group group)
        {
            var teacher = _db.Teachers.FirstOrDefault(t => t.Id == group.TeacherId);
            var user = _db.Users.Include(r => r.Role).FirstOrDefault(u => u.Id ==
teacher.UserId);
            if (!user.Role.Contains(_db.Roles.FirstOrDefault(c => c.Name == "Curator")))
            user.Role.Add(_db.Roles.FirstOrDefault(c => c.Name == "Curator"));
            _db.Users.Update(user);
            _db.Groups.Update(group);
                _db.SaveChanges();
                return RedirectToAction("Index");

        }

        public IActionResult Delete(int? id)
        {
            if (id == null || id == 0)
            {
                return NotFound();
            }
            var group = _db.Groups.FirstOrDefault(c => c.Id == id);
            if (group == null)
            {
                group = new Group();
            }
            return View(group);

        }
        [HttpPost]
        [ValidateAntiForgeryToken]
        public IActionResult DeletePost(int? id)
        {
            var group = _db.Groups.Find(id);
            if (group == null)
            {
                return NotFound();
            }
            _db.Groups.Remove(group);
            _db.SaveChanges();
            return RedirectToAction("Index");
        }
        public IActionResult GroupStudents(int? id)
```

```
        {
            if (id == null || id == 0)
            {
                return NotFound();
            }
            var group = _db.GetGroupWithStudentsById(id);
            if (group == null)
            {
                group = new Group();
            }
            return View(group);

        }
        public IActionResult AddStudentToGroup(int? id)
        {
            if (id == null || id == 0)
            {
                return NotFound();
            }
            var group = _db.Groups.FirstOrDefault(c => c.Id == id);
            if (group == null)
            {
                group = new Group();
            }
            UserViewModel model = new UserViewModel();
            model.UsersList = _db.Users.Where(t => t.Role.Contains(_db.Roles.Where(c =>
c.Name == "Student").First()) && _db.Students.FirstOrDefault(y=>y.UserId==t.Id) ==
null).Select(x => new ItemList { Value = x.Id, Text = x.LastName + " " + x.FirstName + "
" + x.Surname }).ToList();
            ViewBag.Users = model;
            ViewBag.Group = group;
            return View();


        }

        [HttpPost]

        [ValidateAntiForgeryToken]
        public IActionResult AddStudentToGroup(Student student)
        {
            /*if (ModelState.IsValid)
            {*/
            // student.Id = null;
            Student st1 = new Student();
            st1.UserId=student.UserId;
            st1.GroupId=student.GroupId;
            _db.Students.Add(st1);
            _db.SaveChanges();
            return RedirectToAction("GroupStudents", new { id = student.GroupId});
            // }
            //return View(group);
        }



    }
}
using e_school.Data;
using e_school.Models;
using e_school.Models.VM;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
```

```
namespace e_school.Controllers
{
    public class HomeworkController : Controller
    {
        private readonly AppDBContext _db;
        public HomeworkController(AppDBContext db)
        {
            _db = db;
        }
        public IActionResult GetStudentHomework(int id)
        {
            Student student = _db.Students.FirstOrDefault(c => c.Id == id);
            Group group = _db.Groups.FirstOrDefault(c => c.Id == student.GroupId);
            var homeworks = _db.Homeworks.Include(h => h.Subject).Where(t => t.GroupId ==
student.GroupId).OrderByDescending(t=>t.DeadlineDateTime).ToList();
            var groups = from homework in homeworks
                         group homework by homework.DeadlineDateTime;
            List<StudentHomeworDay> studentHomeworDays = new List<StudentHomeworDay>();
            foreach (var homeworkDay in groups)
            {
                StudentHomeworDay studentHomeworDay = new StudentHomeworDay();
                studentHomeworDay.DateTime = homeworkDay.Key;
                studentHomeworDay.homeworkStudentVMs = new List<HomeworkStudentVM>();
                foreach (var homeworkNote in homeworkDay)
                {
                    var model = new HomeworkStudentVM();

                    model.Student = student;
                    model.StudentId = student.Id;
                    model.Homework = homeworkNote;
                    model.HomeworkId = homeworkNote.Id;
                    var homDone = _db.HomeworkDones.Where(t => t.HomeworkId ==
homeworkNote.Id && t.StudentId == id).FirstOrDefault();
                    if (homDone != null)
                    {
                        model.isDone = homDone.IsDone;
                    }
                    else model.isDone = false;
                    studentHomeworDay.homeworkStudentVMs.Add(model);
                }
                studentHomeworDay.Percent = 100 /
studentHomeworDay.homeworkStudentVMs.Count() *
studentHomeworDay.homeworkStudentVMs.Where(r=>r.isDone==true).Count();
                studentHomeworDays.Add(studentHomeworDay);

            }
            return View(studentHomeworDays);
        }

        public void AddHomeworkDone(int studentId, int homeworkId, bool isDone)
        {
            HomeworkDone homeworkDone =
_db.HomeworkDones.FirstOrDefault(r=>r.HomeworkId==homeworkId && r.StudentId==studentId);
            if (homeworkDone == null)
            {
                homeworkDone = new HomeworkDone();
                homeworkDone.HomeworkId = homeworkId;
                homeworkDone.StudentId = studentId;
                homeworkDone.IsDone = isDone;
                _db.HomeworkDones.Add(homeworkDone);
            }
            else { homeworkDone.IsDone = isDone;
                _db.HomeworkDones.Update(homeworkDone);
            }
```

```
            _db.SaveChanges();

        }
    }
}
using e_school.Data;
using e_school.Models;
using e_school.Models.VM;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using System.Globalization;

namespace e_school.Controllers
{
    public class JournalController : Controller
    {
        private readonly AppDBContext _db;
        public JournalController(AppDBContext db)
        {
            _db = db;
        }
        public IActionResult GroupJournal(int groupId, int subjectId)
        {
            ViewBag.subjectId = subjectId;
            Schedule schedule = _db.Schedules.Include(s => s.ScheduleDays).ThenInclude(c
=> c.ScheduleNotes).Where(g => g.GroupId == groupId).FirstOrDefault();
            List<ScheduleNote> scheduleNotes = _db.ScheduleNotes.Where(c =>
c.ScheduleDay.Schedule.GroupId == groupId).Include(s => s.Subject).ToList();
            List<Subject> subjects = new List<Subject>();
            foreach (ScheduleNote scheduleNote in scheduleNotes)
            {
                if (subjects.Find(r => r.Id == scheduleNote.Subject.Id) == null)
                {
                    subjects.Add(scheduleNote.Subject);
                }
            }
            SubjectViewModel subjectVM = new SubjectViewModel();
            subjectVM.SubjectsList = subjects.Select(x => new ItemList { Value = x.Id,
Text = x.Name }).ToList();
            ViewBag.subjects = subjectVM;
            Group group = _db.Groups.FirstOrDefault(g => g.Id == groupId);
            ViewBag.Group = group;

            return View();
        }
        public List<int> GetDayNumbers(int subjectId, int groupId)
        {
            Schedule schedule = _db.Schedules.Include(s => s.ScheduleDays).ThenInclude(c
=> c.ScheduleNotes).FirstOrDefault(b => b.GroupId == groupId);
            List<ScheduleDay> days = new List<ScheduleDay>();

            foreach (var day in schedule.ScheduleDays)
            {
                foreach (var note in day.ScheduleNotes)
                {
                    if (note.SubjectId == subjectId)
                    {
                        days.Add(day);
                        break;
                    }
                }
            }
            Subject subject = _db.Subjects.FirstOrDefault(s => s.Id == subjectId);
            List<int> dayNumbers = new List<int>();
```

```
            foreach (var day in days)
            {
                switch (day.Day)
                {
                    case "Понеділок":
                        dayNumbers.Add(5);
                        break;
                    case "Вівторок":
                        dayNumbers.Add(4);
                        break;
                    case "Середа":
                        dayNumbers.Add(3);
                        break;
                    case "Четвер":
                        dayNumbers.Add(2);
                        break;
                    case "П'ятниця":
                        dayNumbers.Add(1);
                        break;
                    case "Субота":
                        dayNumbers.Add(0);
                        break;

                }

            }
            return dayNumbers;
        }
        public IActionResult GetSubjectTable(int groupId, int subjectId, int addMonth)
        {
            Subject subject = _db.Subjects.FirstOrDefault(s => s.Id == subjectId);
            List<int> dayNumbers = GetDayNumbers(subjectId, groupId);

            List<DateTime> lessonDates = GetDayDates(dayNumbers, addMonth);

            List<Student> students = _db.Students.Where(c => c.GroupId ==
groupId).Include(u => u.User).OrderBy(p => p.User.LastName).ToList();
            List<JournalDate> journalDates = new List<JournalDate>();

            foreach (var student in students)
            {
                JournalDate journalDate = new JournalDate();
                journalDate.Student = student;
                journalDate.Subject = subject;
                List<DateStudentMark> dateStudentMarks = new List<DateStudentMark>();
                foreach (var lessonDate in lessonDates)
                {
                    DateStudentMark dateStudentMark = new DateStudentMark();
                    dateStudentMark.DayDate = lessonDate;
                    Mark mark = new Mark();
                    List<Mark> marks = _db.Marks.Where(b => b.StudentId == student.Id &&
b.SubjectId == subjectId && b.CreatedDateTime != null).ToList();
                    if (marks != null)
                    {
                        mark = marks.Find(b => b.CreatedDateTime == lessonDate);
                        if (mark != null)
                        {
                            dateStudentMark.Mark = mark;
                        }
                    }
                    dateStudentMarks.Add(dateStudentMark);
                }
                journalDate.DateStudentMarks = dateStudentMarks;
                journalDates.Add(journalDate);
```

```
            }
            ViewBag.lessonDates = lessonDates;
            ViewBag.SubjectId = subjectId;
            return PartialView(journalDates);
        }

        //public IActionResult SubjectGroupJournal(int groupId, int subjectId)
        //{
        //    Subject subject = _db.Subjects.FirstOrDefault(s => s.Id == subjectId);
        //    List<int> dayNumbers = GetDayNumbers(subjectId, groupId);

        //    List<DateTime> lessonDates = GetDayDates(dayNumbers, -1);

        //    List<Student> students = _db.Students.Where(c => c.GroupId ==
groupId).Include(u => u.User).OrderBy(p => p.User.LastName).ToList();
        //    List<JournalDate> journalDates = new List<JournalDate>();

        //    foreach (var student in students)
        //    {
        //        JournalDate journalDate = new JournalDate();
        //        journalDate.Student = student;
        //        journalDate.Subject = subject;
        //        List<DateStudentMark> dateStudentMarks = new List<DateStudentMark>();
        //        foreach (var lessonDate in lessonDates)
        //        {
        //            DateStudentMark dateStudentMark = new DateStudentMark();
        //            dateStudentMark.DayDate = lessonDate;
        //            Mark mark = new Mark();
        //            List<Mark> marks = _db.Marks.Where(b => b.StudentId == student.Id
&& b.SubjectId == subjectId && b.CreatedDateTime != null).ToList();
        //            if (marks != null)
        //            {
        //                mark = marks.Find(b => b.CreatedDateTime == lessonDate);
        //                if (mark != null)
        //                {
        //                    dateStudentMark.Mark = mark;
        //                }
        //            }
        //            dateStudentMarks.Add(dateStudentMark);
        //        }
        //        journalDate.DateStudentMarks = dateStudentMarks;
        //        journalDates.Add(journalDate);

        //    }
        //    ViewBag.lessonDates = lessonDates;


        //    return View(journalDates);
        //}
        public List<DateTime> GetDayDates(List<int> dayNumbers, int addMonth)
        {
            List<DateTime> days = new List<DateTime>();
            var currentDay = DateTime.Today.AddMonths(addMonth);
            var nextSaturday = currentDay.AddDays(Math.Abs((int)DayOfWeek.Saturday -
(int)currentDay.DayOfWeek));
            while (currentDay <= DateTime.Today.AddMonths(addMonth+1))
            {
                foreach (var day in dayNumbers)
                {
                    DateTime dt = nextSaturday.AddDays(-day);
                    days.Add(dt);
                }
                currentDay = currentDay.AddDays(7);
```

```
                nextSaturday = currentDay.AddDays(Math.Abs((int)DayOfWeek.Saturday -
(int)currentDay.DayOfWeek));
            }
            return days;
        }
        public IActionResult AddMark(int subjectId, DateTime date, int studentId)
        {
            var mark = new Mark();
            List<Mark> marks = _db.Marks.Include(s => s.Student).ThenInclude(p =>
p.User).Include(y => y.Subject).Where(u => u.SubjectId == subjectId && u.StudentId ==
studentId && u.CreatedDateTime != null).ToList();
            if (marks != null)
            {
                mark = marks.Find(b => b.CreatedDateTime == date);
            }
            Student student = _db.Students.Include(u => u.User).FirstOrDefault(b => b.Id
== studentId);
            Subject subject = _db.Subjects.FirstOrDefault(t => t.Id == subjectId);
            if (mark == null)
            {
                mark = new Mark();
                mark.SubjectId = subjectId;
                mark.Student = student;
                mark.CreatedDateTime = date;
                mark.StudentId = studentId;
                mark.Subject = subject;
            }

            Group group = _db.Groups.Where(g =>
g.Students.Contains(student)).FirstOrDefault();
            ViewBag.groupId = group.Id;
            return View(mark);
        }
        [HttpPost]
        [ValidateAntiForgeryToken]
        public IActionResult AddMark(Mark mark)
        {
            Student student = _db.Students.FirstOrDefault(b => b.Id == mark.StudentId);
            Group group = _db.Groups.Where(g =>
g.Students.Contains(student)).FirstOrDefault();
            List<Mark> marks = _db.Marks.Where(u => u.SubjectId == mark.SubjectId &&
u.StudentId == mark.StudentId && u.CreatedDateTime != null).ToList();
            Mark mark1 = new Mark();
            if (marks != null)
            {
                mark1 = marks.Find(b => b.CreatedDateTime == mark.CreatedDateTime);
            }
            if (mark1 != null)
            {
                mark1.Point = mark.Point;
                _db.Marks.Update(mark);
            }
            else
            {
                _db.Marks.Add(mark);
            }

            _db.SaveChanges();
            return RedirectToAction("GroupJournal", new { groupId = group.Id, subjectId =
mark.SubjectId });
        }

        public IActionResult AddHomework(int subjectId, int groupId, DateTime deadLine)
        {
```

```
            var homework = new Homework();

            homework = _db.Homeworks.Include(c => c.Subject).Include(p =>
p.Group).Where(u => u.SubjectId == subjectId && u.GroupId == groupId &&
u.DeadlineDateTime== deadLine).FirstOrDefault();
            if (homework == null)
            {
                homework = new Homework();
                homework.SubjectId = subjectId;
                homework.GroupId = groupId;
                homework.Group = _db.Groups.FirstOrDefault(b => b.Id == groupId);
                homework.CreatedDateTime = DateTime.Today;
                homework.Subject = _db.Subjects.FirstOrDefault(t => t.Id == subjectId);
                homework.DeadlineDateTime = deadLine;
            }
            return PartialView(homework);
        }
        [HttpPost]
        [ValidateAntiForgeryToken]
        public IActionResult AddHomework(Homework homework)
        {
            //Student student = _db.Students.FirstOrDefault(b => b.Id == mark.StudentId);
            //Group group = _db.Groups.Where(g =>
g.Students.Contains(student)).FirstOrDefault();
            //List<Mark> marks = _db.Marks.Where(u => u.SubjectId == mark.SubjectId &&
u.StudentId == mark.StudentId && u.CreatedDateTime != null).ToList();
            //Mark mark1 = new Mark();
            //if (marks != null)
            //{
            //    mark1 = marks.Find(b => b.CreatedDateTime == mark.CreatedDateTime);
            //}
            //if (mark1 != null)
            //{
            //    mark1.Point = mark.Point;
            //    _db.Marks.Update(mark);
            //}
            //else
            //{
            //    _db.Marks.Add(mark);
            //}
            _db.Homeworks.Add(homework);
            _db.SaveChanges();
            return RedirectToAction("GroupJournal", new { groupId = homework.GroupId,
subjectId = homework.SubjectId });
        }

        [HttpGet]
        public JsonResult GetDays(int groupId, int subjectId)
        {
            Schedule schedule = _db.Schedules.Include(s => s.ScheduleDays).ThenInclude(c
=> c.ScheduleNotes).FirstOrDefault(b => b.GroupId == groupId);
            List<ScheduleDay> days = new List<ScheduleDay>();

            foreach (var day in schedule.ScheduleDays)
            {
                foreach (var note in day.ScheduleNotes)
                {
                    if (note.SubjectId == subjectId)
                    {
                        days.Add(day);
                        break;
                    }
                }
            }
```

```
            Subject subject = _db.Subjects.FirstOrDefault(s => s.Id == subjectId);
            List<int> dayNumbers = new List<int>();
            foreach (var day in days)
            {
                switch (day.Day)
                {
                    case "Понеділок":
                        dayNumbers.Add(1);
                        break;
                    case "Вівторок":
                        dayNumbers.Add(2);
                        break;
                    case "Середа":
                        dayNumbers.Add(3);
                        break;
                    case "Четвер":
                        dayNumbers.Add(4);
                        break;
                    case "П'ятниця":
                        dayNumbers.Add(5);
                        break;
                    case "Субота":
                        dayNumbers.Add(6);
                        break;

                }

            }
            return Json(dayNumbers);
        }

        public JsonResult GetHomework(int groupId, int subjectId)
        {
            var homework = _db.Homeworks.FirstOrDefault(s => s.GroupId == groupId &&
s.SubjectId == subjectId && s.DeadlineDateTime == DateTime.Today);
            if (homework != null)
            {
                return Json(homework.Task);
            }
            return Json("");
        }
    }
}
using e_school.Data;
using e_school.Models;
using e_school.Models.VM;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using System.Security.Claims;

namespace e_school.Controllers
{
    public class NotificationController : Controller
    {   private AppDBContext _db;
        public NotificationController(AppDBContext context)
        {
            _db = context;
        }
        public IActionResult GetAllNotifications(int id)
        {
            ((ClaimsIdentity)User.Identity).RemoveClaim(User.FindFirst(x => x.Type ==
"countNotification"));
            List<Notification> list =
_db.Notifications.Include(y=>y.FromUser).Where(c=>c.UserId==id).ToList();
```

```
            foreach(var item in list)
            {
                item.IsRead = true;
                _db.Notifications.Update(item);
            }
            _db.SaveChanges();

            return View(list);
        }

        public IActionResult AddGroupNotification(int groupId)
        {GroupNotificationVM model= new GroupNotificationVM();
            model.GroupId = groupId;
            model.Group = _db.Groups.FirstOrDefault(r=>r.Id == groupId);
            return PartialView(model);
        }
        [HttpPost]
        public IActionResult AddGroupNotification(GroupNotificationVM model)
        {
            var group =
_db.Groups.Include(r=>r.Students).ThenInclude(u=>u.User).FirstOrDefault(r => r.Id ==
model.GroupId);
            foreach(var student in group.Students)
            {
                Notification notification = new Notification();
                notification.UserId = student.User.Id;
                notification.FromUserId = Int32.Parse(User.FindFirst(x => x.Type ==
"id").Value);
                notification.Text= model.Text;
                notification.Title = model.Title;
                notification.CreatedDate = DateTime.Now;
                notification.IsRead = false;
                _db.Notifications.Add(notification);
                _db.SaveChanges();

            }
            return RedirectToAction("GroupStudents", "Group", new { id = model.GroupId
});
        }


        public IActionResult AddUserNotification(int userId)
        {
            Notification model = new Notification();
            model.UserId = userId;
            model.User = _db.Users.FirstOrDefault(r=>r.Id==userId);
            return PartialView(model);

        }
        [HttpPost]

        public IActionResult AddUserNotification(Notification notification)
        {

            notification.FromUserId = Int32.Parse(User.FindFirst(x => x.Type ==
"id").Value);
            notification.CreatedDate = DateTime.Now;
            notification.IsRead = false;
            _db.Notifications.Add(notification);
            _db.SaveChanges();
            Student student = _db.Students.FirstOrDefault(r =>
r.UserId==notification.UserId);
            return RedirectToAction("GroupStudents", "Group", new { id = student.GroupId
});
```

```
        }

    }
}
using e_school.Data;
using e_school.Models;
using e_school.Models.VM;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;

namespace e_school.Controllers
{
    public class ParentController : Controller
    {
        private readonly AppDBContext _db;
        public ParentController(AppDBContext db)
        {
            _db = db;
        }
        public IActionResult Index(int studentId)
        { ViewBag.Student =
_db.Students.Include(u=>u.User).FirstOrDefault(t=>t.Id==studentId);
            List<Parent> parents =
_db.Parents.Include(c=>c.Children).ThenInclude(s=>s.User).Include(u=>u.User).Where(f =>
f.Children.Contains(_db.Students.FirstOrDefault(s => s.Id == studentId))).ToList();
            return View(parents);
        }
        public IActionResult AddParent(int studentId)
        {
            AddParentVM parent = new AddParentVM();
            parent.StudentId = studentId;
            parent.Student=
_db.Students.Include(u=>u.User).FirstOrDefault(u=>u.Id==studentId);
            UserViewModel model = new UserViewModel();
            model.UsersList = _db.Users.Where(t => t.Role.Contains(_db.Roles.Where(c =>
c.Name == "Parent").First())).Select(x => new ItemList { Value = x.Id, Text = x.LastName
+ " " + x.FirstName + " " + x.Surname }).ToList();
            ViewBag.Users = model;

            return PartialView(parent);
        }
        [HttpPost]
        public IActionResult AddParent(AddParentVM addParentVM)
        {
            Student Student = _db.Students.Include(u => u.User).FirstOrDefault(u => u.Id
== addParentVM.StudentId);
            Parent parent = _db.Parents.FirstOrDefault(c=>c.UserId==addParentVM.UserId);
            if (parent == null)
            {
                parent = new Parent();
                parent.ParentName = addParentVM.ParentName;
                parent.UserId = addParentVM.UserId;
                parent.Children = new List<Student>();

                parent.Children.Add(Student);
                _db.Parents.Add(parent);
            }
            else
            {
                parent.Children.Add(Student);
                _db.Parents.Update(parent);
            }
            _db.SaveChanges();
            return RedirectToAction("Index", new { studentId = addParentVM.StudentId});
```

```
        }
        public IActionResult Delete(int? id, int studentId)
        {
            var parent = _db.Parents.Find(id);
            if (parent == null)
            {
                return NotFound();
            }
            var student = _db.Students.Include(s => s.Parents).FirstOrDefault(r => r.Id
== studentId);
            student.Parents.Remove(parent);
            _db.Students.Update(student);
            _db.Parents.Remove(parent);
            _db.SaveChanges();
            return RedirectToAction("Index", new { studentId = studentId});
        }
    }
}
using e_school.Data;
using e_school.Models;
using e_school.Models.VM;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;

namespace e_school.Controllers
{
    public class ScheduleController : Controller
    {
        private readonly AppDBContext _db;
        public ScheduleController(AppDBContext db)
        {
            _db = db;
        }
        public IActionResult GroupSchedule(int? id)
        {
                if (id == null || id == 0)
                {
                    return NotFound();
                }
            var group = _db.Groups.FirstOrDefault(c => c.Id == id);
            ViewBag.group = group;
            List<String> days = new List<String>()
            {
                "Понеділок", "Вівторок", "Середа", "Четвер", "П'ятниця", "Субота"
            };

            ViewBag.days = days;
            List<LessonTimeFrame> lessonTimeFrames = new List<LessonTimeFrame>();
            lessonTimeFrames = _db.LessonTimeFrames.ToList();

            ViewBag.frames = lessonTimeFrames;
            Schedule model = _db.Schedules.Include(g => g.Group).Include(s =>
s.ScheduleDays).ThenInclude(n => n.ScheduleNotes).ThenInclude(t => t.Subject).Include(s
=> s.ScheduleDays).ThenInclude(n => n.ScheduleNotes).ThenInclude(t =>
t.Teacher).ThenInclude(u => u.User).FirstOrDefault(c => c.GroupId == id);


            return View(model);

        }

        public IActionResult Create(int? id)
        {
            var group = _db.Groups.FirstOrDefault(c => c.Id == id);
```

```
            ViewBag.group = group;
            List<String> days = new List<String>()
            {
                "Понеділок", "Вівторок", "Середа", "Четвер", "П'ятниця", "Субота"
            };

            ViewBag.days = days;
            List<LessonTimeFrame> lessonTimeFrames = new List<LessonTimeFrame>();
            lessonTimeFrames = _db.LessonTimeFrames.ToList();

            ViewBag.frames = lessonTimeFrames;
            Schedule model = _db.Schedules.Include(g => g.Group).Include(s =>
s.ScheduleDays).ThenInclude(n => n.ScheduleNotes).ThenInclude(t => t.Subject).Include(s
=> s.ScheduleDays).ThenInclude(n => n.ScheduleNotes).ThenInclude(t =>
t.Teacher).ThenInclude(u => u.User).FirstOrDefault(c => c.GroupId == id);


            return View(model);
        }
        public IActionResult CreateNote(int id, string day, int frameId)
        {
            ScheduleAddVM model = new ScheduleAddVM();
            model.LessonTimeFrameId = frameId;
            model.LessonTimeFrame = _db.LessonTimeFrames.FirstOrDefault(c => c.Id ==
frameId);
            model.ScheduleDayName = day;
            model.GroupId = id;
            Schedule schedule = _db.Schedules.FirstOrDefault(c => c.GroupId == id);
            if (schedule != null)
            {
                ScheduleDay scday = _db.ScheduleDays.Include(s => s.Schedule).Where(c =>
c.ScheduleId == schedule.Id).FirstOrDefault(c => c.Day == model.ScheduleDayName);
                if (scday != null)
                {
                    ScheduleNote note = _db.ScheduleNotes.Include(s =>
s.ScheduleDay).Where(c => c.ScheduleDayId == scday.Id).FirstOrDefault(c =>
c.LessonTimeFrameId == model.LessonTimeFrameId);
                    if (note != null)
                    {
                        model.TeacherId = note.TeacherId;
                        model.SubjectId = note.SubjectId;
                    }
                }
            }

            IEnumerable<Teacher> teacherList = _db.Teachers.Include(m =>
m.User).ToList();
            TeacherViewModel teacherVM = new TeacherViewModel();
            teacherVM.TeachersList = _db.Teachers.Include(m => m.User).Select(x => new
ItemList { Value = x.Id, Text = x.User.LastName + " " + x.User.FirstName + " " +
x.User.Surname }).ToList();
            ViewBag.teachers = teacherVM;
            SubjectViewModel subjectVM = new SubjectViewModel();
            subjectVM.SubjectsList = _db.Subjects.Select(x => new ItemList { Value =
x.Id, Text = x.Name }).ToList();
            ViewBag.subjects = subjectVM;


            return View(model);
        }
        public ActionResult GetSubjects(int id)
        {
            TeacherViewModel teacherVM = new TeacherViewModel();
```

```
            teacherVM.TeachersList = _db.Teachers.Where(t =>
t.Subjects.Contains(_db.Subjects.Where(c => c.Id == id).First())).Select(x => new
ItemList { Value = x.Id, Text = x.User.LastName + " " + x.User.FirstName + " " +
x.User.Surname }).ToList();
            ViewBag.teachers2 = teacherVM;
            return PartialView();
        }
        [HttpPost]
        public IActionResult CreateNote(ScheduleAddVM model)
        {
            Schedule schedule = _db.Schedules.FirstOrDefault(c => c.GroupId ==
model.GroupId);
            if (schedule == null)
            {
                schedule = new Schedule();
                schedule.GroupId = model.GroupId;
                _db.Schedules.Add(schedule);
                _db.SaveChanges();
                schedule = _db.Schedules.FirstOrDefault(c => c.GroupId == model.GroupId);
            }

            ScheduleDay day=
_db.ScheduleDays.Include(s=>s.Schedule).Where(c=>c.ScheduleId==schedule.Id).FirstOrDefaul
t(c => c.Day == model.ScheduleDayName);
            if (day == null)
            {
                day = new ScheduleDay();
                day.Day=model.ScheduleDayName;
                day.ScheduleId=schedule.Id;
                _db.ScheduleDays.Add(day);
                _db.SaveChanges();
                day = _db.ScheduleDays.Include(s => s.Schedule).Where(c => c.ScheduleId
== schedule.Id).FirstOrDefault(c => c.Day == model.ScheduleDayName);
            }

            ScheduleNote note = _db.ScheduleNotes.Include(s => s.ScheduleDay).Where(c =>
c.ScheduleDayId == day.Id).FirstOrDefault(c => c.LessonTimeFrameId ==
model.LessonTimeFrameId);
            if (note == null)
            {
                note = new ScheduleNote();
                note.SubjectId = model.SubjectId;
                note.ScheduleDayId = day.Id;
                note.TeacherId = model.TeacherId;
                note.LessonTimeFrameId = model.LessonTimeFrameId;
                note.Classroom = model.Classroom;
                _db.ScheduleNotes.Add(note);
                _db.SaveChanges();
            }
            else
            {
                note.SubjectId = model.SubjectId;
                note.TeacherId = model.TeacherId;
                note.Classroom = model.Classroom;
                _db.ScheduleNotes.Update(note);
                _db.SaveChanges();

            }

            return RedirectToAction("Create", new { id = model.GroupId });
        }
    }
}
using e_school.Data;
```

```
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;

namespace e_school.Controllers
{
    public class StudentController : Controller
    {
        private readonly AppDBContext _db;
        public StudentController(AppDBContext db)
        {
            _db = db;
        }

        public IActionResult Delete(int? id)
        {
            var student = _db.Students.Find(id);
            if (student == null)
            {
                return NotFound();
            }
            var group = _db.Groups.Include(s=>s.Students).FirstOrDefault(r => r.Id ==
student.GroupId);
            group.Students.Remove(student);
            _db.Groups.Update(group);
            _db.Students.Remove(student);
            _db.SaveChanges();
            return RedirectToAction("GroupStudents", "Group", new {id=group.Id});
        }

    }
}
using e_school.Data;
using e_school.Models;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;

namespace e_school.Controllers
{
    public class SubjectController : Controller
    {

        private readonly IAppDBContext _db;
        public SubjectController(IAppDBContext db)
        {
            _db = db;
        }
        public IActionResult Index()
        {
            IEnumerable<Subject> subjectList = _db.GetSubjects();
            return View(subjectList);
        }
        public IActionResult Create()
        {
            return View();
        }
        [HttpPost]
        [ValidateAntiForgeryToken]
        public IActionResult Create(Subject subject)
        {
            if (ModelState.IsValid)
            {
                _db.Subjects.Add(subject);
                _db.SaveChanges();
                return RedirectToAction("Index");
```

```
            }
            return View(subject);
        }
        public IActionResult Edit(int? id)
        {
            if (id == null || id == 0)
            {
                return NotFound();
            }
            var subject = _db.GetOneSubject(id);
            if (subject == null)
            {
                subject = new Subject();
            }
            return View(subject);

        }
        [HttpPost]
        [ValidateAntiForgeryToken]
        public IActionResult Edit(Subject subject)
        {
            if (ModelState.IsValid)
            {
                _db.Subjects.Update(subject);
                _db.SaveChanges();
                return RedirectToAction("Index");
            }
            return View(subject);
        }



        public IActionResult DeletePost(int? id)
        {
            var subject= _db.Subjects.Find(id);
            if (subject == null)
            {
                return NotFound();
            }
            _db.Subjects.Remove(subject);
            _db.SaveChanges();
            return RedirectToAction("Index");
        }
    }
    }

using e_school.Data;
using e_school.Models;
using e_school.Models.VM;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.Rendering;
using Microsoft.EntityFrameworkCore;

namespace e_school.Controllers
{
    public class TeacherController : Controller
    {
        private readonly AppDBContext _db;
        public TeacherController(AppDBContext db)
        {
            _db = db;
        }
        public IActionResult Index()
        {
```

```
            IEnumerable<Teacher> teacherList =
_db.Teachers.Include(u=>u.User).Include(s=>s.Subjects);
            return View(teacherList);
        }
        public IActionResult Create()
        {

            UserViewModel model = new UserViewModel();
            model.UsersList = _db.Users.Where(t => t.Role.Contains(_db.Roles.Where(c =>
c.Name == "Teacher").First()) && _db.Teachers.FirstOrDefault(y => y.UserId == t.Id) ==
null).Select(x => new ItemList { Value = x.Id, Text = x.LastName + " " + x.FirstName + "
" + x.Surname }).ToList();
            ViewBag.Users = model;
            TeacherSubjectsViewModel model1 = new TeacherSubjectsViewModel();
            List<long> subjectsIds = new List<long>();

                model1.drpSubjects = _db.Subjects.Select(x => new SelectListItem { Text =
x.Name, Value = x.Id.ToString() }).ToList();


            return View(model1);

        }
        [HttpPost]
        [ValidateAntiForgeryToken]
        public IActionResult Create(TeacherSubjectsViewModel model)
        {

            Teacher teacher = new Teacher();
            List<Subject> subj = new List<Subject>();
                if (model.SubjectsIds.Length > 0)
                {
                    foreach (var subjectid in model.SubjectsIds)
                    {
                    Subject sbj = new Subject();
                    subj.Add(_db.Subjects.FirstOrDefault(c => c.Id == subjectid));
                    }
                }
                teacher.Subjects=subj;
                teacher.UserId = model.UserId;
                _db.Teachers.Add(teacher);
                _db.SaveChanges();

            return RedirectToAction("index");

        }
        public IActionResult Edit(int? id)
        {
            if (id == null || id == 0)
            {
                return NotFound();
            }
            var user = _db.Teachers.Include(u => u.User).FirstOrDefault(t => t.Id ==
id).User;
            UserViewModel model = new UserViewModel();
            model.UsersList = _db.Users.Where(t => t.Role.Contains(_db.Roles.Where(c =>
c.Name == "Teacher").First()) && _db.Teachers.FirstOrDefault(y => y.UserId == t.Id) ==
null).Select(x => new ItemList { Value = x.Id, Text = x.LastName + " " + x.FirstName + "
" + x.Surname }).ToList();
            ViewBag.Users = model;

            TeacherSubjectsViewModel model1 = new TeacherSubjectsViewModel();
            var teacher = _db.Teachers.Include(s=>s.Subjects).FirstOrDefault(t => t.Id ==
id);
```

```
            long[] subjectsIds = new long[teacher.Subjects.Count];
            int i = 0;
            foreach (var s in teacher.Subjects)
            {
                subjectsIds[i] = s.Id; i++;
            }


            model1.drpSubjects = _db.Subjects.Select(x => new SelectListItem { Text =
x.Name, Value = x.Id.ToString() }).ToList();
            model1.User = user;
            model1.UserId = user.Id;
            model1.SubjectsIds = subjectsIds;
            return View(model1);

        }
        [HttpPost]
        [ValidateAntiForgeryToken]
        public IActionResult Edit(TeacherSubjectsViewModel model)
        {

            Teacher teacher = _db.Teachers.Include(s=>s.Subjects).FirstOrDefault(d =>
d.UserId == model.UserId);
            List<Subject> subj = new List<Subject>();
            if (model.SubjectsIds.Length > 0)
            {
                foreach (var subjectid in model.SubjectsIds)
                {
                    Subject sbj = new Subject();
                    subj.Add(_db.Subjects.FirstOrDefault(c => c.Id == subjectid));
                }
            }
            foreach(Subject sub in subj)
            {
                if (teacher.Subjects.Find(t => t.Id == sub.Id) == null)
                {
                    teacher.Subjects.Add(sub);
                }
            }

                    teacher.Subjects.RemoveAll(t=> !model.SubjectsIds.Contains(t.Id));

            teacher.UserId = model.UserId;

            _db.Teachers.Update(teacher);
            _db.SaveChanges();

            return RedirectToAction("index");

        }

        public IActionResult DeletePost(int? id)
        {
            var teacher = _db.Teachers.Find(id);
            if (teacher == null)
            {
                return NotFound();
            }
            List<ScheduleNote> notes=_db.ScheduleNotes.Where(x => x.TeacherId ==
id).ToList();
            foreach(var it in notes)
            {
                _db.ScheduleNotes.Remove(it);
            }
```

```
            _db.SaveChanges();
            _db.Teachers.Remove(teacher);
            _db.SaveChanges();
            return RedirectToAction("Index");
        }
    }
}
using e_school.Data;
using e_school.Models;
using e_school.Models.VM;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;

namespace e_school.Controllers
{
    public class TimesheetController : Controller
    {
        private readonly AppDBContext _db;

        public TimesheetController(AppDBContext db)
        {
            _db = db;
        }
        public IActionResult CreateTimeSheets(int groupId)
        {
            CreateGroupTimesheetVM timesheet = new CreateGroupTimesheetVM();
            ViewBag.Date = DateTime.Now;
            timesheet.Group = _db.Groups.FirstOrDefault(x => x.Id == groupId);
            return PartialView(timesheet);

        }
        [HttpPost]
        [ValidateAntiForgeryToken]
        public IActionResult CreateTimeSheets(CreateGroupTimesheetVM timesheetVM)
        {
            Group group = _db.Groups.Include(s => s.Students).FirstOrDefault(g => g.Id ==
timesheetVM.GroupId);
            List<ScheduleNote> scheduleNotes = _db.ScheduleNotes.Where(c =>
c.ScheduleDay.Schedule.GroupId == timesheetVM.GroupId).Include(s => s.Subject).ToList();
            List<Subject> subjects = new List<Subject>();
            foreach (ScheduleNote scheduleNote in scheduleNotes)
            {
                if (subjects.Find(r => r.Id == scheduleNote.Subject.Id) == null)
                {
                    subjects.Add(scheduleNote.Subject);
                }
            }
            SubjectViewModel subjectVM = new SubjectViewModel();
            foreach (Student student in group.Students)
            {
                if (_db.Timesheets.Where(x => x.StudentId == student.Id && x.Semestr ==
timesheetVM.Semestr && x.Start.Year == timesheetVM.Start.Year && x.End.Year ==
timesheetVM.End.Year).FirstOrDefault() == null)
                {
                    Timesheet timesheet = new Timesheet();
                    timesheet.Student = student;
                    timesheet.StudentId = student.Id;
                    timesheet.Start = timesheetVM.Start;
                    timesheet.End = timesheetVM.End;
                    timesheet.Semestr = timesheetVM.Semestr;
                    timesheet.Notes = new List<TimesheetNote>();
                    foreach (Subject subject in subjects)
                    {
                        TimesheetNote note = new TimesheetNote();
```

```
                        note.Subject = subject;
                        if (_db.Marks.Where(x => x.SubjectId == subject.Id && x.StudentId
== student.Id && x.CreatedDateTime >= timesheetVM.Start && x.CreatedDateTime <=
timesheetVM.End).ToList().Count() != 0)
                        {
                            var t = _db.Marks.Where(x => x.SubjectId == subject.Id &&
x.StudentId == student.Id && x.CreatedDateTime >= timesheetVM.Start && x.CreatedDateTime
<= timesheetVM.End).Average(p => p.Point);
                            note.Mark = (int)Math.Round(t, 0);
                        }
                        else note.Mark = 0;
                        timesheet.Notes.Add(note);


                    }
                    _db.Timesheets.Add(timesheet);
                }
                else
                {
                    var timesheet = _db.Timesheets.Where(x => x.StudentId == student.Id
&& x.Semestr == timesheetVM.Semestr && x.Start.Year == timesheetVM.Start.Year &&
x.End.Year == timesheetVM.End.Year).FirstOrDefault();
                    timesheet.Start = timesheetVM.Start;
                    timesheet.End = timesheetVM.End;
                    foreach (var note in timesheet.Notes)
                    {
                        if (_db.Marks.Where(x => x.SubjectId == note.SubjectId &&
x.StudentId == timesheet.StudentId && x.CreatedDateTime >= timesheetVM.Start &&
x.CreatedDateTime <= timesheetVM.End).ToList() != null)
                        {
                            var t = _db.Marks.Where(x => x.SubjectId == note.SubjectId &&
x.StudentId == timesheet.StudentId && x.CreatedDateTime >= timesheetVM.Start &&
x.CreatedDateTime <= timesheetVM.End).Average(p => p.Point);
                            note.Mark = (int)Math.Round(t, 0);
                        }
                        else note.Mark = 0;
                        _db.TimesheetNotes.Update(note);
                    }
                }
            }
            _db.SaveChanges();

            return RedirectToAction("GroupJournal", new { groupId = group.Id, subjectId =
1 });

        }

        public IActionResult StudentTimeSheet(int studentId)
        {
            List<Timesheet> timesheet =
_db.Timesheets.Include(s=>s.Notes).ThenInclude(p=>p.Subject).Where(i=>i.StudentId==studen
tId).ToList();


            return View(timesheet);

        }
        public IActionResult ViewTimeSheet(int id)
        {
            Timesheet timesheet = _db.Timesheets.Include(s => s.Notes).ThenInclude(p =>
p.Subject).FirstOrDefault(x=>x.Id==id);


            return View(timesheet);
```

```
        }

    }
}
```

Models

```
using System.ComponentModel.DataAnnotations;

namespace e_school.Models
{
    public class Curator
    {
        [Key]
        public int Id { get; set; }
        public int TeacherId { get; set; }
        public Teacher Teacher { get; set; }
        public List<Group> Groups { get; set; }

    }
}


using System.ComponentModel.DataAnnotations;

namespace e_school.Models
{
    public class Diary
    {
        [Key]
        public int Id { get; set; }
        public List<DiaryNote> DiaryNotes { get; set; }
        public int StudentId { get; set; }
        public Student Student { get; set; }
    }
}
using System.ComponentModel.DataAnnotations;

namespace e_school.Models
{
    public class Group
    {
        [Key]
        public int Id { get; set; }
        public string Name { get; set; }
        public DateTime CreatedDateTime { get; set; } = DateTime.Now;
        public List<Student> Students { get; set; }
        public int TeacherId { get; set; }
        public Teacher Teacher { get; set; }
        public Group()
        {
            Students = new List<Student>();
        }
    }
}
using System.ComponentModel;
using System.ComponentModel.DataAnnotations;

namespace e_school.Models
{
    public class Homework
    {
        [Key]
```

```
        public int Id { get; set; }
        [Required]
        [DisplayName("Завдання")]
        [MinLength(2, ErrorMessage = "Довжина назви повина бути щонайменше 2 символи")]
        public string Task { get; set; }
        public DateTime CreatedDateTime { get; set; } = DateTime.Now;
        public DateTime DeadlineDateTime { get; set; }

        public int GroupId { get; set; }
        public Group Group { get; set; }
        public int SubjectId { get; set; }
        public Subject Subject { get; set; }
    }
}
namespace e_school.Models
{
    public class HomeworkDone
    {
        public int Id { get; set; }
        public bool IsDone { get; set; }
        public int HomeworkId { get; set; }
        public Homework Homework { get; set; }
        public int StudentId { get; set; }
        public Student Student { get; set; }
    }
}
using System.ComponentModel.DataAnnotations;

namespace e_school.Models
{
    public class LessonTimeFrame
    {
        [Key]
        public int Id { get; set; }
        public int LessonNumber { get; set; }
        public DateTime TimeStart { get; set; }
        public DateTime TimeEnd { get; set; }
    }
}
using System.ComponentModel.DataAnnotations;

namespace e_school.Models
{
    public class Mark
    {
        [Key]
        public int Id { get; set; }

        public int Point { get; set; }
        public DateTime CreatedDateTime { get; set; } = DateTime.Now;

        public int StudentId { get; set; }
        public Student Student { get; set; }
        public int SubjectId { get; set; }
        public Subject Subject { get; set; }

    }
}
using System.ComponentModel.DataAnnotations;

namespace e_school.Models
{
    public class Notification
    {
```

```
        [Key]
        public int Id { get; set; }
        public String Text { get; set; }
        public String Title { get; set; }
        public DateTime CreatedDate { get; set; }
        public bool IsRead { get; set; }
        public int UserId { get; set; }
        public User User { get; set; }
        public int FromUserId { get; set; }
        public User FromUser { get; set; }


    }
}
using System.ComponentModel.DataAnnotations;

namespace e_school.Models
{
    public class Parent
    {
        [Key]
        public int Id { get; set; }
        public int UserId { get; set; }
        public User User { get; set; }
        public String ParentName { get; set; }
        public List<Student> Children { get; set; }
    }
}
using System.ComponentModel.DataAnnotations;

namespace e_school.Models
{
    public class Role
    {
        [Key]
        public int Id { get; set; }
        public String Name { get; set; }
        public List<User> Users { get; set; }
        public Role()
        {
            Users = new List<User>();
        }

    }
}
using System.ComponentModel.DataAnnotations;

namespace e_school.Models
{
    public class Schedule
    {
        [Key]
        public int Id { get; set; }
        public int GroupId { get; set; }
        public Group Group { get; set; }
        public List<ScheduleDay> ScheduleDays { get; set; }
    }
}
using System.ComponentModel.DataAnnotations;

namespace e_school.Models
{
    public class ScheduleDay
    {
```

```
        [Key]
        public int Id { get; set; }
        public string Day { get; set; }
        public int ScheduleId { get; set; }
        public Schedule Schedule { get; set; }
        public List<ScheduleNote> ScheduleNotes { get; set; }
    }
}
using System.ComponentModel.DataAnnotations;

namespace e_school.Models
{
    public class ScheduleNote
    {
        [Key]
        public int Id { get; set; }
        public int TeacherId { get; set; }
        public Teacher Teacher { get; set; }
        public int SubjectId { get; set; }
        public Subject Subject { get; set; }
        public int ScheduleDayId { get; set; }
        public ScheduleDay ScheduleDay { get; set; }
        public int LessonTimeFrameId { get; set; }
        public LessonTimeFrame LessonTimeFrame { get; set; }
        public String Classroom { get; set; }


    }
}
namespace e_school.Models
{
    public class SchoolActivity
    {
        public int Id { get; set; }
        public string Topic { get; set; }
        public string Description { get; set; }
        public byte[] Photo { get; set; }
        public DateTime Date { get; set; }
        public string PupilsCategory { get; set; }
        public string? ShortDescription { get; set; }
        public string? Place { get; set; }
        public string? OtherDetails { get; set; }
    }
}
using Microsoft.EntityFrameworkCore;
using System.ComponentModel;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;

namespace e_school.Models
{
    public class Student
    {
        [Key]
        public int Id { get; set; }
        public DateTime CreatedDateTime { get; set; } = DateTime.Now;
        public int GroupId { get; set; }
        public Group Group { get; set; }

        public int UserId { get; set; }
        public User User { get; set; }
        public List<Parent> Parents { get; set; }
```

```
    }
}
using System.ComponentModel.DataAnnotations;

namespace e_school.Models
{
    public class Subject
    {
        [Key]
        public int Id { get; set; }
        [Required]
        public string Name { get; set; }
        public DateTime CreatedDateTime { get; set; } = DateTime.Now;
        public List<Teacher>? Teachers { get; set; }
    }
}
using System.ComponentModel.DataAnnotations;

namespace e_school.Models
{
    public class Teacher
    {
        [Key]
        public int Id { get; set; }
        public DateTime CreatedDateTime { get; set; } = DateTime.Now;
        public List<Subject> Subjects { get; set; }

        public int UserId { get; set; }
        public User User { get; set; }

    }
}
namespace e_school.Models
{
    public class Timesheet
    {
        public int Id { get; set; }
        public int Semestr { get; set; }
        public DateTime Start{ get; set; }
        public DateTime End { get; set; }
        public int StudentId { get; set; }
        public Student Student { get; set; }
        public List<TimesheetNote> Notes { get; set; }


    }
}
namespace e_school.Models
{
    public class TimesheetNote
    {
        public int Id { get; set; }
        public Subject Subject { get; set; }
        public int SubjectId { get; set; }
        public int Mark { get; set; }
        public Timesheet Timesheet { get; set; }
        public int TimesheetId { get; set; }
    }
}
using System.ComponentModel.DataAnnotations;

namespace e_school.Models
{
    public class User
```

```
    {
        [Key]
        public int Id { get; set; }
        public string Email { get; set; }
        public string Password { get; set; }
        public string FirstName { get; set; }
        public string LastName { get; set; }
        public string Surname { get; set; }
        public List<Role> Role { get; set; }
        public string Phone { get; set; }
    }
}
```

Views

```
@using System.Security.Claims
@model IEnumerable<Group>

<div class="container p-3">
    <div class="row pt-4">
        <div class = "col-6">
            <h2>Список класів</h2>
        </div>
         @if (((ClaimsIdentity)User.Identity).Claims.Where(c => c.Type ==
ClaimTypes.Role).Select(c => c.Value).ToList().Contains("Admin"))
            {
            <div class="cal-6 text-end">
                <a asp-controller="Group" asp-action="Create" class="btn btn-primary">
                Додати клас
            </a>
        </div>}
    </div>

</div>

<table class="table table-bordered table-striped" style="width:100%">
    <thead>
        <tr>
            <th>
                Назва групи
            </th>
            <th>
                Куратор
            </th>
            <th>
                Учні
            </th>
            <th>
                Розклад
            </th>
                    @if (!((ClaimsIdentity)User.Identity).Claims.Where(c => c.Type ==
ClaimTypes.Role).Select(c => c.Value).ToList().Contains("Admin"))
            {

                <th>
                    Журнал
            </th>}
            else{
            <th></th><th></th>}
        </tr>
    </thead>
    <tbody>
        @foreach (var item in Model)
```

```
        {
            <tr>
                <td>
                    @item.Name
                </td>
                <td>
                  @(item.Teacher.User.LastName+" "+item.Teacher.User.FirstName+"
"+item.Teacher.User.Surname)
                </td>
                <td style="text-align:center">
                    <div class="w-75 btn-group" role="group" >
                        <a asp-controller="Group" asp-action="GroupStudents" asp-route-
id="@item.Id" style="margin:auto"><i class="bi bi-people" style="font-size:
30px"></i></a>
                    </div>
                </td>
                <td  style="text-align:center">
                    <div class="w-75 btn-group" role="group">
                        <a asp-controller="Schedule" asp-action="GroupSchedule" asp-
route-id="@item.Id" style="margin:auto" ><i class="bi bi-list-ol" style="font-size:
30px"></i></a>
                    </div>
                </td>
                @if (!((ClaimsIdentity)User.Identity).Claims.Where(c => c.Type ==
ClaimTypes.Role).Select(c => c.Value).ToList().Contains("Admin"))
                {
                    <td style="text-align:center">
                        <div class="w-75 btn-group" role="group" >
                       <a asp-controller="Journal" asp-action="GroupJournal"  asp-route-
groupId="@item.Id" style="margin:auto"><i class="bi bi-journals" style="font-size:
30px"></i></a>
                    </div>
                </td>}
                else{
                <td style="text-align:center">
                    <div class="w-75 btn-group" role="group">
                        <a asp-controller="Group" asp-action="Edit" asp-route-
id="@item.Id" class="btn btn-primary"><i class="bi bi-pencil-square"  style="font-size:
18px; margin:auto"></i></a>
                    </div>
                </td>
                <td style="text-align:center">
                    <div class="w-75 btn-group" role="group">
                        <a asp-controller="Group" asp-action="Delete" asp-route-
id="@item.Id" class="btn btn-warning"><i class="bi bi-trash"  style="font-size: 18px;
margin:auto"></i></a>
                    </div>
                </td>}
            </tr>
        }
    </tbody>
</table>



@model Group
@{
    ViewData["Title"] = "ViewGroup";
}

<h1>Клас @Model.Name</h1>
<div class="container">
    <div class="row">
         <div class=" col-4" role="group"  style="text-align:center">
```

```
                        <a asp-controller="Group" asp-action="GroupStudents" asp-route-
id="@Model.Id" style="margin:auto"><i class="bi bi-people" style="font-size:
200px"></i><br />Учні</a>
                    </div>

                    <div class=" col-4" role="group" style="text-align:center">
                        <a asp-controller="Schedule" asp-action="GroupSchedule" asp-
route-id="@Model.Id" style="margin:auto" ><i class="bi bi-list-ol" style="font-size:
200px"></i><br />Розклад</a>
                    </div>

                    <div class=" col-4" role="group" style="text-align:center">
                       <a asp-controller="Journal" asp-action="GroupJournal" asp-route-
groupId="@Model.Id" style="margin:auto"><i class="bi bi-journals" style="font-size:
200px"></i><br />Журнал</a>
                    </div>

    </div>

</div>



@using System.Security.Claims
@model Group

<div class="container p-3">
    <div class="row pt-4">
        <div class = "col-6">
            <h2>Список учнів класу @Model.Name</h2>
        </div>
        <div class="cal-6 text-end">
            @if (!((ClaimsIdentity)User.Identity).Claims.Where(c => c.Type ==
ClaimTypes.Role).Select(c => c.Value).ToList().Contains("Admin"))
            {
                <a asp-controller="Notification" asp-action="AddGroupNotification" asp-
route-groupId="@Model.Id" class="btn btn-primary compItem">
                    Написати повідомлення учням
            </a>}
            else{
            <a asp-controller="Group" asp-action="AddStudentToGroup" asp-route-
id="@Model.Id" class="btn btn-primary">
                Додати учня
            </a>}
        </div>
    </div>

</div>

<table class="table table-bordered table-striped" style="width:100%">
    <thead>
        <tr>
            <th>
                Ім'я
            </th>
            <th>
                Email адреса
            </th>
            <th>
                Номер телефону
            </th>
            <th>
                Батьки
            </th>
```

```
            <th></th>
                            @if (!((ClaimsIdentity)User.Identity).Claims.Where(c =>
c.Type == ClaimTypes.Role).Select(c => c.Value).ToList().Contains("Admin"))
            {
                <th></th>
            }
        </tr>
    </thead>
    <tbody>
        @foreach (var item in Model.Students)
        {
            <tr>
                <td>
                    @(item.User.LastName+" "+item.User.FirstName+" "+item.User.Surname)
                </td>
                <td>@item.User.Email</td>
                <td>@item.User.Phone</td>
                <td>
                    <div class="w-75 btn-group" role="group">
                         <a asp-controller="Parent" asp-action="Index" asp-route-
studentId="@item.Id" style="margin:auto"><i class="bi bi-people" style="font-size:
30px"></i></a>
                    </div>
                </td>
                                @if (!((ClaimsIdentity)User.Identity).Claims.Where(c =>
c.Type == ClaimTypes.Role).Select(c => c.Value).ToList().Contains("Admin"))
                {
                    <td style="text-align:center">
                        <div class="w-75 btn-group" role="group">
                         <a asp-controller="Notification" asp-
action="AddUserNotification" asp-route-userId="@item.User.Id" style="margin:auto"
class="userNotif"><i class="bi bi-envelope-plus" style="font-size: 30px"></i></a>
                    </div>
                </td>}
                <td style="text-align:center">
                    <div class="w-75 btn-group" role="group">
                        <a asp-controller="Student" asp-action="Delete" asp-route-
id="@item.Id" class="btn btn-warning"><i class="bi bi-trash"  style="font-size: 18px;
margin:auto"></i></a>
                    </div>
                </td>
            </tr>
        }
    </tbody>
</table>

<div id="modDialog" class="modal fade">
    <div id="dialogContent" class="modal-dialog"></div>
</div>
@section scripts{
<script type="text/javascript">

    $(function () {
        $.ajaxSetup({ cache: false });
       $(".compItem").click(function (e) {
            e.preventDefault();
            $.get(this.href, function (data) {
                $('#dialogContent').html(data);
                $('#modDialog').modal('show');
            });
        });
               $(".userNotif").click(function (e) {
           debugger;
            e.preventDefault();
```

```
            $.get(this.href, function (data) {
                $('#dialogContent').html(data);
                $('#modDialog').modal('show');
            });
        });

    })
      </script>}



@model Group

<form method="post">
    <div class="border p-3 mt-4">
        <div class="row pb-2">
            <h2 class="text-primary">Додати клас</h2>
            <hr/>
        </div>
        <div class="mb-3">
            <label asp-for="Name"></label>
            <input asp-for="Name" class="form-control"/>
            <span asp-validation-for="Name" class="text-danger"></span>
        </div>

        <div class="mb-3">
        <label>Класний керівник</label>
        <select id="drpEmpList" class="form-control" asp-for="TeacherId"  asp-
items="@(new SelectList(ViewBag.teachers.TeachersList, "Value", "Text"))">
                    <option selected
value="@Model.TeacherId">@(Model.Teacher.User.LastName+ " "
+Model.Teacher.User.FirstName+" " +Model.Teacher.User.Surname)</option>
                </select>
                <input type="hidden" asp-for="TeacherId" />
        </div>
        <button type="submit" class="btn-primary btn">Додати</button>
    </div>
</form>

@section Scripts{
    @{
        <partial name="_ValidationScriptsPartial"/>
    }
                                                  }



@model Group

<form method="post">
    <div class="border p-3 mt-4">
        <div class="row pb-2">
            <h2 class="text-primary">Додати клас</h2>
            <hr/>
        </div>
        <div class="mb-3">
            <label asp-for="Name"></label>
            <input asp-for="Name" class="form-control"/>
            <span asp-validation-for="Name" class="text-danger"></span>
        </div>

        <div class="mb-3">
        <label>Класний керівник</label>
```

```
        <select id="drpEmpList" class="form-control" asp-for="TeacherId" asp-items="@(new
SelectList(ViewBag.teachers.TeachersList, "Value", "Text"))">
                    <option value="">--Обрати--</option>
                </select>
                <input type="hidden" asp-for="TeacherId" />
        </div>
        <button type="submit" class="btn-primary btn">Додати</button>
    </div>
</form>

@section Scripts{
    @{
        <partial name="_ValidationScriptsPartial"/>
    }
                                                }


@using Syncfusion.EJ2
@using System.Security.Claims

<div class="container row">
    <div class="row col-7">
       <fieldset class="scheduler-border col-11" >
        <legend class="scheduler-border">Домашнє завдання на сьогодні</legend>
            <input id="getHomework" class="form-control"/>
       </fieldset>
       <div class="col-1"></div></div>
    <fieldset class="scheduler-border col-5 align-items-end" id="addTaskFieldset">
        <legend class="scheduler-border">Додати домашнє завдання</legend>
        <div class="form-group">
            <label class="control-label">Оберіть дату дедлайну</label>
            <input id="datepicker" type="text" class="form-control"
value="@DateTime.Now.ToString("dd-MM-yyyy")" />
        </div>
    </fieldset>
</div>
@if (((ClaimsIdentity)User.Identity).Claims.Where(c => c.Type ==
ClaimTypes.Role).Select(c => c.Value).ToList().Contains("Curator")  && User.FindFirst(x
=> x.Type == "groupId").Value==ViewBag.Group.Id.ToString())
            {
<a href="#" id="createTimesheet" class="btn btn-primary" style="margin-
top:20px">Сформувати табель для групи</a>}


<div class="container">

    <input type="text" id="subjectId" value="@ViewBag.SubjectId" hidden />
    <input value="@ViewBag.Group.Id" class="form-control" id="groupId" hidden />

    <div class="border p-3 mt-4">
        <div class="mb-1">
            <h5>Журнал класу @ViewBag.Group.Name</h5>
        </div>

        <div class="mb-3">
            <div class="row">
                <div class="col-2">
                    <h6>Дисципліна:</h6>
                </div>
                <div class="col-10">
                    <select id="subject" class="form-control dropdown" asp-items="@(new
SelectList(ViewBag.subjects.SubjectsList, "Value", "Text"))">
                        <option value="" class="dropdown-item">--Обрати--</option>
                    </select>
```

```
                </div>

            </div>
        </div>
        <input value=-1 id="addMonth" hidden/>

        <a href="#" id ="previousPage"   ><i class="bi bi-arrow-left-circle" style="font-
size: 30px"></i></a>
        <a href="#" id="nextPage"  ><i class="bi bi-arrow-right-circle" style="font-size:
30px"></i></a>
        <div class="mb-3">
            <div id="subjectTable"></div>
        </div>

    </div>
</div>
<div id="modDialog" class="modal fade">
    <div id="dialogContent" class="modal-dialog"></div>
</div>


@section scripts{
<link rel="stylesheet" href="https://code.jquery.com/ui/1.12.1/themes/base/jquery-
ui.css">
<script src="https://code.jquery.com/jquery-1.12.4.js"></script>
<script src="https://code.jquery.com/ui/1.12.1/jquery-ui.js"></script>
<script type="text/javascript">

                $(function () {

                    var arr=[];
                    function unavailable(date) {
    var day = date.getDay();
    var dayVisibility = false;
    for(var i=0; i<arr.length;i++){
        if(arr[i]==day){
    dayVisibility = true;
    break;
        }
    }
            return [dayVisibility,  ''];
        }
      //  function loadDatePicker(setting) {
      //  $("#datepicker").datepicker("destroy");
      //  if(setting == 'noWeekends') {
      //    $( "#datepicker" ).datepicker({ beforeShowDay: $.datepicker.noWeekends,
minDate: +2, maxDate: "+1M" });
      //  }
      //  else if(setting == 'noSunday') {
      //    $( "#datepicker" ).datepicker({ beforeShowDay: noSunday, minDate: +2,
maxDate: "+1M" });
      //  }
      //  $( "#datepicker" ).datepicker("refresh");
      //}

             $("#datepicker").change(function()
                    {
                    var deadline = $(this).val();
                        var subjectId = $("#subjectId").val();
                        var groupId = $("#groupId").val();
                        $('#subjectId').val(subjectId);
                        $.ajax({
                            type: 'GET',
                            url: '@Url.Action("AddHomework")',
```

```
                            data: { "groupId": groupId, "subjectId":subjectId, "deadline"
: deadline},
                            success: function (data) {

                                 $('#dialogContent').html(data);
                                $('#modDialog').modal('show');

                            }
                        });

                    });

                     $("#createTimesheet").click(function()
                    {debugger;
                        var groupId = $("#groupId").val();

                        $.ajax({
                            type: 'GET',
                            url: '/Timesheet/CreateTimeSheets',
                            data: { "groupId": groupId},
                            success: function (data) {

                                 $('#dialogContent').html(data);
                                $('#modDialog').modal('show');

                            }
                        });

                    });
                $("#datepicker").datepicker({
                    minDate: 0,
                    beforeShowDay: unavailable
                });


                       $(document).ready(function(data) {
                var subjectId = $("#subjectId").val();
                if(subjectId!=0){
                $('#subject').val(subjectId);
                        var groupId = $("#groupId").val();
                        $.ajax({
                            type: 'GET',
                            url: '@Url.Action("GetSubjectTable")',
                            data: { "groupId": groupId, "subjectId":subjectId,
"addMonth":-1 },
                            success: function (data) {

                                $('#subjectTable').replaceWith(data);
                            }
                        });
                        $.ajax({
                            type: 'GET',
                            url: '@Url.Action("GetDays")',
                            data: { "groupId": groupId, "subjectId":subjectId },
                            success: function (data) {
                                arr=data;
                                $("#addTaskFieldset").prop('disabled', false);
                            }
                        });
                         $.ajax({
                            type: 'GET',
                            url: '@Url.Action("GetHomework")',
                            data: { "groupId": groupId, "subjectId":subjectId },
                            success: function (data) {
```

```
                            $('#getHomework').val(data);
                        }
                    });
                    }
                    else{
                        $("#addTaskFieldset").prop('disabled', true);
                    }

            });
                    $('#subject').change(function()
                    {
                        var subjectId = $(this).val();
                        var groupId = $("#groupId").val();
                        $('#subjectId').val(subjectId);

                        $.ajax({
                            type: 'GET',
                            url: '@Url.Action("GetSubjectTable")',
                            data: { "groupId": groupId, "subjectId":subjectId,
"addMonth":-1 },
                            success: function (data) {

                                $('#subjectTable').replaceWith(data);

                            }
                        });
                        $.ajax({
                            type: 'GET',
                            url: '@Url.Action("GetDays")',
                            data: { "groupId": groupId, "subjectId":subjectId },
                            success: function (data) {
                                arr=data;
                                $("#addTaskFieldset").prop('disabled', false);
                            }
                        });
                         $.ajax({
                            type: 'GET',
                            url: '@Url.Action("GetHomework")',
                            data: { "groupId": groupId, "subjectId":subjectId },
                            success: function (data) {
                                $('#getHomework').val(data);
                            }
                        });
                    });

                     $('#previousPage').click(function()
                    {
                        var subjectId = $("#subjectId").val();
                        var groupId = $("#groupId").val();
                        var addMonth= $("#addMonth").val()-1;
                        $("#addMonth").val(addMonth);

                        $.ajax({
                            type: 'GET',
                            url: '@Url.Action("GetSubjectTable")',
                            data: { "groupId": groupId, "subjectId":subjectId,
"addMonth":addMonth },
                            success: function (data) {

                                $('#subjectTable').replaceWith(data);

                            }
                        });
                    });
```

```
                    $('#nextPage').click(function()
                    {
                        var subjectId = $("#subjectId").val();
                        var groupId = $("#groupId").val();
                        var addMonth= parseInt($("#addMonth").val())+1;
                        $("#addMonth").val(addMonth);

                        $.ajax({
                            type: 'GET',
                            url: '@Url.Action("GetSubjectTable")',
                            data: { "groupId": groupId, "subjectId":subjectId,
"addMonth":addMonth },
                            success: function (data) {

                                $('#subjectTable').replaceWith(data);

                            }
                        });
                    });



                })
</script>
}



@using e_school.Models.VM
@model IEnumerable<JournalDate>
@{
    ViewData["Title"] = "SubjectGroupJournal";
}

<h1>SubjectGroupJournal</h1>
<div class="container p-3">
    <div class="row pt-4">
        <div class = "col-6">
            <h2>Category List</h2>
        </div>
        <div class="cal-6 text-end">
            <a asp-controller="Category" asp-action="Create" class="btn btn-primary">
                Додати оцінку
            </a>
        </div>
    </div>

</div>
<h2>@ViewBag.SubjectId</h2>

<table class="table table-bordered table-striped" style="width:100%">
    <thead>
        <tr>
            <th>
                Учень
            </th>
             @foreach (var date in ViewBag.lessonDates){
            <th>
                @date.ToString("dd/MM")
            </th>
             }
        </tr>
    </thead>
    <tbody>
```

```
        @foreach (var item in Model)
        {
            <tr>
                <td>@item.Student.User.LastName</td>

                    @foreach (var dsm in item.DateStudentMarks){
                    if (dsm.Mark != null) {
                        <td><a asp-controller="Journal" asp-action="AddMarkPartial"
class="compItem" asp-route-subjectId=@item.Subject.Id asp-route-date=@dsm.DayDate asp-
route-studentId=@item.Student.Id>
                @dsm.Mark.Point
            </a></td>
                    } else {
                        <td><a asp-controller="Journal" asp-action="AddMarkPartial"
class="compItem" asp-route-subjectId=@item.Subject.Id asp-route-date=@dsm.DayDate asp-
route-studentId=@item.Student.Id>_</a></td>

                    }  }

            </tr>

        }
    </tbody>
</table>
<div id="modDialog" class="modal fade" >
    <div id="dialogContent" class="modal-dialog"></div>
</div>
@section scripts
{
    <script type="text/javascript">

        $(function () {
            $.ajaxSetup({ cache: false });
           $(".compItem").click(function (e) {

                e.preventDefault();
                $.get(this.href, function (data) {
                    $('#dialogContent').html(data);
                    $('#modDialog').modal('show');
                });
            });
        })
    </script>
}



@using e_school.Models.VM
@model IEnumerable<JournalDate>
<div id="subjectTable">

<table class="table table-bordered table-striped table-jou" style="width:100%">
    <thead>
        <tr>
            <th colspan="2">
                Учень
            </th>
             @foreach (var date in ViewBag.lessonDates){
            <th>
                @date.ToString("dd/MM")
            </th>
             }
        </tr>
```

```
    </thead>
    <tbody>

        @foreach (var item in Model)
        {
            <tr>
                <td colspan="2">@(item.Student.User.LastName +" "+
item.Student.User.FirstName.Substring(0,1)+".
"+item.Student.User.Surname.Substring(0,1)+".")</td>

                    @foreach (var dsm in item.DateStudentMarks){
                    if (dsm.Mark != null) {
                        <td><a asp-controller="Journal" asp-action="AddMark"
class="compItem" asp-route-subjectId=@item.Subject.Id asp-route-date=@dsm.DayDate asp-
route-studentId=@item.Student.Id>
                @dsm.Mark.Point
            </a></td>
                    } else {
                        <td><a asp-controller="Journal" asp-action="AddMark"
class="compItem" asp-route-subjectId=@item.Subject.Id asp-route-date=@dsm.DayDate asp-
route-studentId=@item.Student.Id style="color:transparent" >_</a></td>

                    }  }

            </tr>

        }
    </tbody>
</table>

</div>


@model Mark

<form method="post" asp-action="AddMark">
    <div class="border p-3 mt-4">
        <div class="row pb-2">
            <h2 class="text-primary">Додати оцінку</h2>
            <hr/></div>
        <div class="mb-3">
            <label asp-for="StudentId">@(Model.Student.User.LastName+"
"+Model.Student.User.FirstName+" "+Model.Student.User.Surname)</label>
            <input asp-for="StudentId" class="form-control" hidden/>
        </div>
        <div class="mb-3">
            <label asp-for="SubjectId">@Model.Subject.Name</label>
            <input asp-for="SubjectId" class="form-control" hidden/>
        </div>
        <div class="mb-3">
            <label asp-for="CreatedDateTime">Дата</label>
            <input asp-for="CreatedDateTime" class="form-control" readonly/>
        </div>
        <div class="mb-3">
            <label asp-for="Point">Оцінка</label>
            <input type="number" min=1 max=12  asp-for="Point" class="form-control" />
            <span asp-validation-for="Point" class="text-danger"></span>
        </div>
        <button type="submit" class="btn-primary btn">Додати</button>
        <button asp-action="GroupJournal" asp-route-groupId="@ViewBag.groupId" asp-route-
subjectId="@Model.SubjectId" class="btn-primary btn">Повернутися назад</button>
    </div>
</form>
```

```
@model Homework
<div class="modal-content">
    <div class="modal-header">
        @*<button class="close" data-dismiss="modal-content" area-
hidden="true">X</button>*@
        <h4>Домашнє завдання</h4>
    </div>
    <div class="modal-body">
<form method="post" asp-action="AddHomework">
    <div class="border p-3 mt-4">

        <div class="mb-3">
            <label asp-for="GroupId">Клас: @Model.Group.Name</label>
            <input asp-for="GroupId" class="form-control" hidden/>
        </div>
        <div class="mb-3">
            <label asp-for="SubjectId">Назва дисципліни: @Model.Subject.Name</label>
            <input asp-for="SubjectId" class="form-control" hidden/>
        </div>
        <div class="mb-3">
            <label asp-for="CreatedDateTime">Дата</label>
            <input asp-for="CreatedDateTime" class="form-control" readonly/>
        </div>
        <div class="mb-3">
            <label asp-for="DeadlineDateTime">Дедлайн</label>
            <input asp-for="DeadlineDateTime" class="form-control" readonly/>
            <span asp-validation-for="DeadlineDateTime" class="text-danger"></span>
        </div>
        <div class="mb-3">
            <label asp-for="Task">Завдання</label>
            <input asp-for="Task" class="form-control"/>
            <span asp-validation-for="Task" class="text-danger" ></span>
        </div>

        <button type="submit" class="btn-primary btn">Додати</button>
        @*<button asp-action="GroupJournal" asp-route-groupId="@Model.GroupId" asp-route-
subjectId="@Model.SubjectId" class="btn-primary btn">Повернутися назад</button>*@
    </div>
</form></div></div>
@using System.Security.Claims
@model Schedule

<div class="container p-3">
    <div class="row pt-4">
        <div class="col-12">
            <h2>Розклад класу @ViewBag.group.Name</h2>
        </div>

    </div>

</div>
@if (@Model != null)
{  @if(((ClaimsIdentity)User.Identity).Claims.Where(c => c.Type ==
ClaimTypes.Role).Select(c => c.Value).ToList().Contains("Admin")){
    <a asp-controller="Schedule" asp-action="Create" asp-route-id="@ViewBag.group.Id"
class="btn btn-primary" style="margin-bottom:5px">
        Редагувати розклад
    </a>}
    <table class="table table-bordered table-striped" style="width:100%">

        @foreach (var item in @ViewBag.days)
        {
            if (Model.ScheduleDays.Find(p => p.Day == item) != null)
            {
```

```
                <tr>
                    <td colspan="5" style="background-color:#78c2ad;
color:white">@item</td>
                </tr>
                <tr>
                    <td>№</td>
                    <td>Час</td>
                    <td>Предмет</td>
                    <td>Вчитель</td>
                    <td>Аудиторія</td>

                </tr>
                @foreach (var frame in @ViewBag.frames)
                {
                    if (Model.ScheduleDays.Find(p => p.Day == item).ScheduleNotes.Find(v
=> v.LessonTimeFrameId == frame.Id) != null)
                    {
                        <tr>
                            <td>
                                @frame.LessonNumber
                            </td>
                            <td>
                                @(frame.TimeStart.ToString("HH:mm")+"-
"+frame.TimeEnd.ToString("HH:mm"))
                            </td>
                            <td>

@Model.ScheduleDays.Find(p=>p.Day==item).ScheduleNotes.Find(v=>v.LessonTimeFrameId==frame
.Id).Subject.Name
                            </td>
                            <td>

@(Model.ScheduleDays.Find(p=>p.Day==item).ScheduleNotes.Find(v=>v.LessonTimeFrameId==fram
e.Id).Teacher.User.LastName +" "+
Model.ScheduleDays.Find(p=>p.Day==item).ScheduleNotes.Find(v=>v.LessonTimeFrameId==frame.
Id).Teacher.User.FirstName.Substring(0,1)+".
"+Model.ScheduleDays.Find(p=>p.Day==item).ScheduleNotes.Find(v=>v.LessonTimeFrameId==fram
e.Id).Teacher.User.Surname.Substring(0,1)+".")
                            </td>
                            <td>

@(Model.ScheduleDays.Find(p=>p.Day==item).ScheduleNotes.Find(v=>v.LessonTimeFrameId==fram
e.Id).Classroom)
                            </td>


                        </tr>
                    }
                }

            }
        }
    </table>
}
else
{
    <div class="col-12">
        <a asp-controller="Schedule" asp-action="Create" asp-route-id="@ViewBag.group.Id"
class="btn btn-primary">
            Додати розклад
        </a>
    </div>
                                                }

@using e_school.Models.VM
```

```
@model ScheduleAddVM

<div class="container p-3">
    <div class="row pt-4">
        <div class = "col-12">
            <h2>@Model.ScheduleDayName</h2>
            <h3>@Model.LessonTimeFrame.LessonNumber урок</h3>

        </div>

    </div>

</div>
<form method="post">

    <div class="border p-3 mt-4">
        <div class="row pb-2">
            <h6>@(Model.LessonTimeFrame.TimeStart.ToString("HH:mm")+" -
"+Model.LessonTimeFrame.TimeEnd.ToString("HH:mm"))</h6>
        </div>
        <div class="row pb-2">
            <input hidden asp-for="ScheduleDayName" value="@Model.ScheduleDayName"/>
            <input hidden asp-for="LessonTimeFrameId" value="@Model.LessonTimeFrameId"/>
            <hr/>
        </div>

        <div class="mb-3">
            <label asp-for="SubjectId">Предмет</label>
                <select id="subject" class="form-control dropdown" asp-for="SubjectId"
asp-items="@(new SelectList(ViewBag.subjects.SubjectsList, "Value", "Text"))">
                    <option value="" class="dropdown-item">--Обрати--</option>
                </select>
        </div>
        <div class="mb-3">
            <label asp-for="Teacher">Вчитель</label>
            <select id="teacher" class="form-control" asp-for="TeacherId" asp-
items="@(new SelectList(ViewBag.teachers.TeachersList, "Value", "Text"))">
                    <option value="">--Обрати--</option>
            </select>
        </div>
        <input type="hidden" asp-for="GroupId" value="@Model.GroupId"/>
        <div class="mb-3">
            <label asp-for="Classroom">Аудиторія</label>
            <input asp-for="Classroom" class="form-control"/>
        </div>
        <button type="submit" class="btn-primary btn">Змінити</button>
    </div>

</form>

@section scripts{
<script type="text/javascript">
    $(function () {

        $('#subject').change(function()
        {
            // получаем выбранный id
            var id = $(this).val();
            $.ajax({
                type: 'GET',
                url: '@Url.Action("GetSubjects")/' + id,
                success: function (data) {

                    // заменяем содержимое присланным частичным представлением
```

```
                    $('#teacher').replaceWith(data);
                }
            });
        });

    })
</script>
}


@model Schedule

<div class="container p-3">
    <div class="row pt-4">
        <div class = "col-12">
            <h2>Розклад класу @ViewBag.group.Name</h2>
        </div>

    </div>

</div>
<table class="table table-bordered table-striped" style="width:100%">

        @foreach (var item in @ViewBag.days)
        {
            <tr>
                    <td colspan="6" style="background-color:#78c2ad;
color:white">@item</td>
                </tr>
                <tr>
                    <td>№</td>
                <td>Час</td>
                <td>Предмет</td>
                <td>Вчитель</td>
                <td>Аудиторія</td>

            </tr>
             @foreach (var frame in @ViewBag.frames)
        {
            <tr>
                <td>
                    @frame.LessonNumber
                </td>
                <td>
                    @(frame.TimeStart.ToString("HH:mm")+"-
"+frame.TimeEnd.ToString("HH:mm"))
                </td>
                @if (Model != null)
                {
                    if (Model.ScheduleDays.Find(p => p.Day == item) != null)
                    {
                        if (Model.ScheduleDays.Find(p => p.Day ==
item).ScheduleNotes.Find(v => v.LessonTimeFrameId == frame.Id) != null)
                        {
                        <td>

@Model.ScheduleDays.Find(p=>p.Day==item).ScheduleNotes.Find(v=>v.LessonTimeFrameId==frame
.Id).Subject.Name
                        </td>
                        <td>

@(Model.ScheduleDays.Find(p=>p.Day==item).ScheduleNotes.Find(v=>v.LessonTimeFrameId==fram
e.Id).Teacher.User.LastName +" "+
Model.ScheduleDays.Find(p=>p.Day==item).ScheduleNotes.Find(v=>v.LessonTimeFrameId==frame.
```

```
Id).Teacher.User.FirstName.Substring(0,1)+".
"+Model.ScheduleDays.Find(p=>p.Day==item).ScheduleNotes.Find(v=>v.LessonTimeFrameId==fram
e.Id).Teacher.User.Surname.Substring(0,1)+".")
                        </td>
                        <td>

@(Model.ScheduleDays.Find(p=>p.Day==item).ScheduleNotes.Find(v=>v.LessonTimeFrameId==fram
e.Id).Classroom)
                        </td>
                        }
                        else {<td>-</td><td>-</td><td>-</td>}
                    }
                    else {<td>-</td><td>-</td><td>-</td>}
                }
                else {<td>-</td><td>-</td><td>-</td>}
                <td>
                     <div class="w-75 btn-group" role="group">
                        <a asp-controller="Schedule" asp-action="CreateNote" asp-route-
id="@ViewBag.group.Id" asp-route-day="@item" asp-route-frameId="@frame.Id" class="btn
btn-primary"><i class="bi bi-pencil-square"></i></a>
                    </div>
                </td>

            </tr>

        }

        }

</table>
```

ДОДАТОК З

(Обов’язковий)

Презентаційні матеріали

Дипломний проект

На тему

«Програмне забезпечення для організації, управління та впорядкування навчального процесу у загальноосвітніх школах»

Виконала: студентка групи ІПЗс-19-1 Дрищ К.В.
Керівник: канд. пед. наук, доцент Праворська Н.І.

Мета проекту

Мета проекту – розробити систему у вигляді сайту, що дозволить автоматично здійснювати навчання учнів шкіл он-лайн. Таке програмне забезпечення повинно надати можливість простої та зрозумілої роботи в даному середовищі.

Завдання

Були поставлені такі задачі:

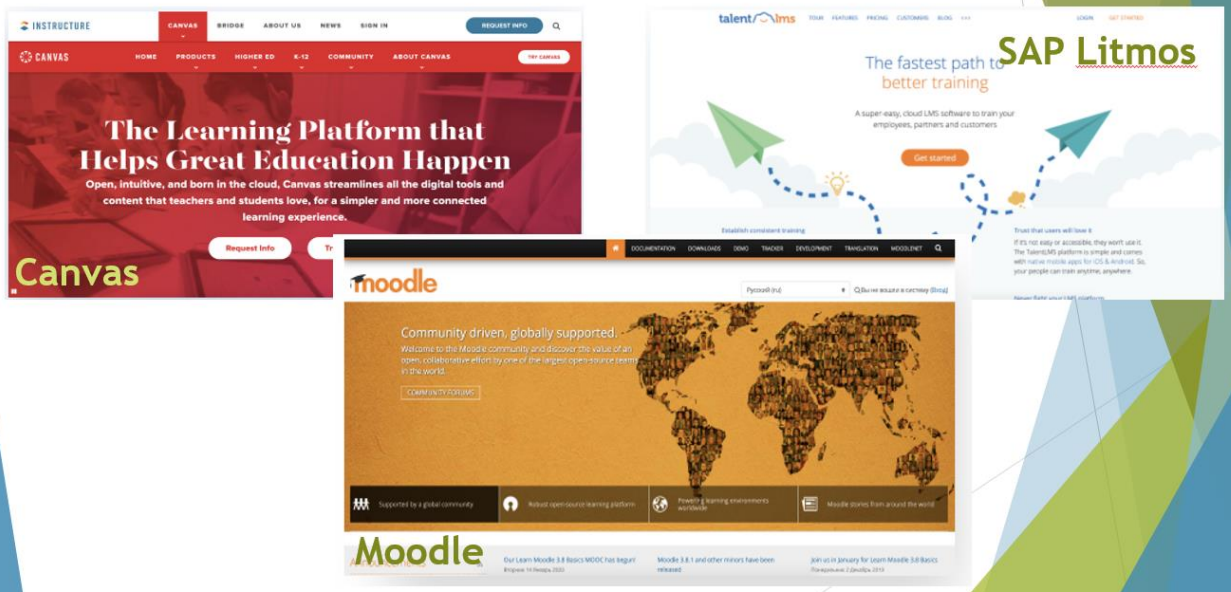
- аналіз та дослідження організації учбового процесу;
- розробка технічного завдання;
- визначення функціональних характеристик;
- визначення вимог та специфікацій;
- визначення типів архітектур та зразків проектування;
- детальне проектування модулів та даних;
- розробка бази даних для програми;
- розробка програмних модулів та інтерфейсу;
- тестування даного ПЗ.

Актуальність

Організація учбового процесу це завжди актуальне питання, а враховуючи сьогоднішню ситуацію виникла надважлива потреба у переході на он-лайн платформи.

Школи намагаються певним чином організовувати весь учбовий процес он-лайн для забезпечення безпеки учнів. Проте інколи такий раптовий перехід є дуже складним або взагалі практично неможливий, оскільки організація учбового процесу в школі є вузькоспеціалізованим напрямком, що потребує відповідного сервісу, який насправді важко знайти. Такий сервіс має забезпечувати повноцінну роботу між учнями і вчителями.

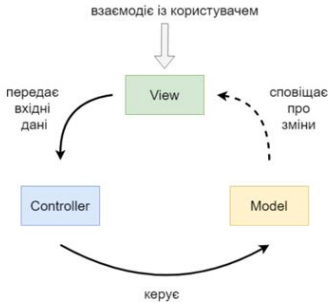
Аналоги

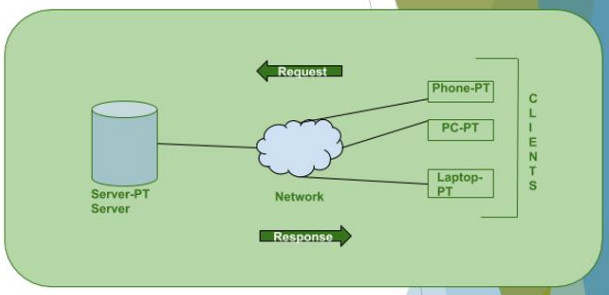


Функціональні вимоги

- реєстрація/авторизація в системі;
- розмежування доступу користувачів за ролями;
- можливість роботи із групами, а саме додавання/редагування;
- можливість реєстрації користувачів із додаванням їх у базу даних;
- можливість додавання/редагування розкладу;
- робота із щоденником, що передбачає виведення предметів із врахуванням домашнього завдання та оцінок учня;
- чек-ліст із домашніми завданнями учня, враховуючи ступінь завершеності завдань;
- формування табелів;
- виставлення оцінок вчителями та розсилка домашнього завдання;
- додавання та редагування вчителів із вибором предметів, при цьому передбачається, що кожен вчитель може викладати декілька дисциплін;
- в системі повинна бути можливість додавання і виведення шкільних заходів для всіх категорій користувачів;
- передбачається можливість роботи у даній системі і для батьків, а саме перегляд щоденника дітей, перегляд шкільних заходів, отримання повідомлень від куратора чи вчителів.

Архітектура та застосовані технології





ASP.NET Core

C# SQL JS CSS B

Діаграма варіантів використання

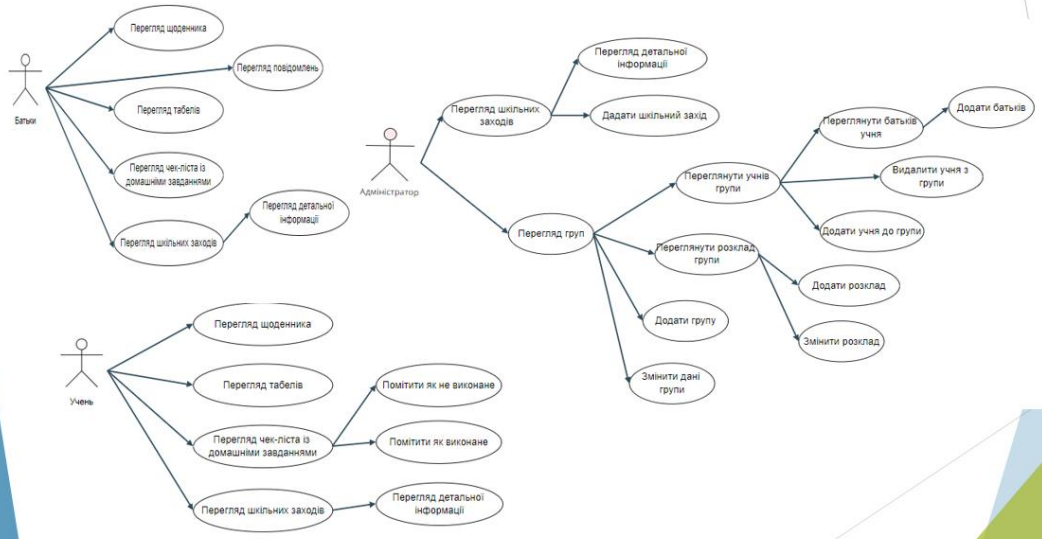
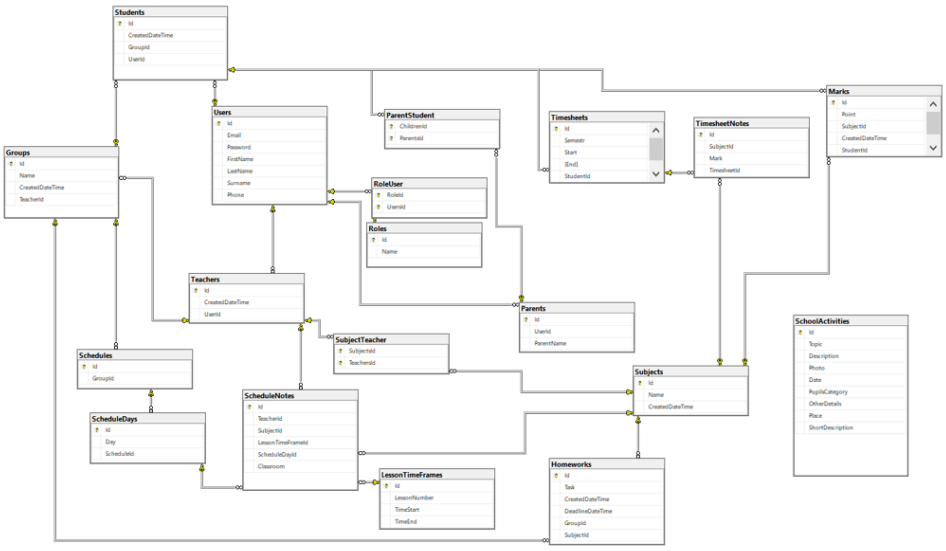
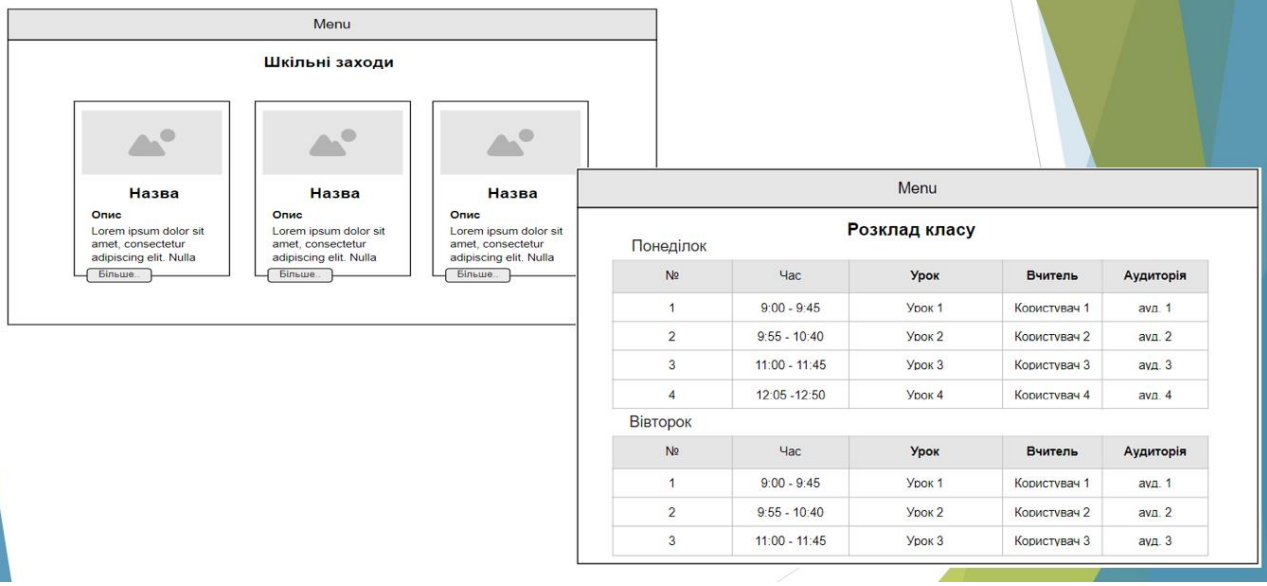
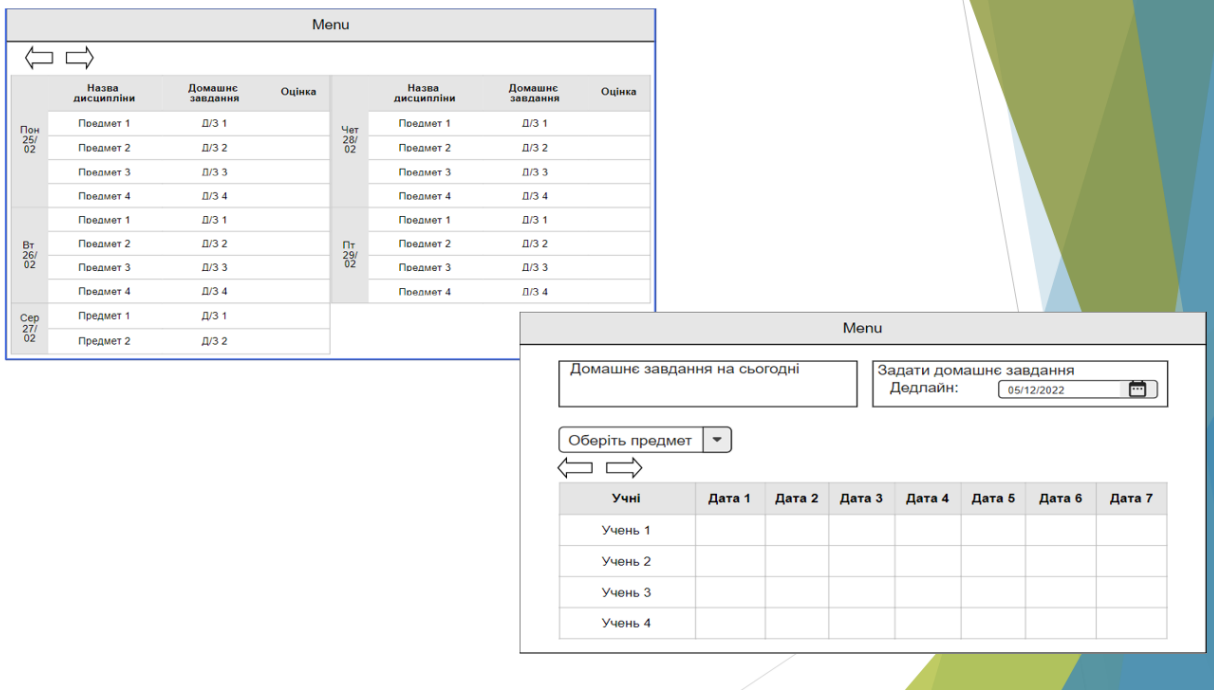


Схема бази даних

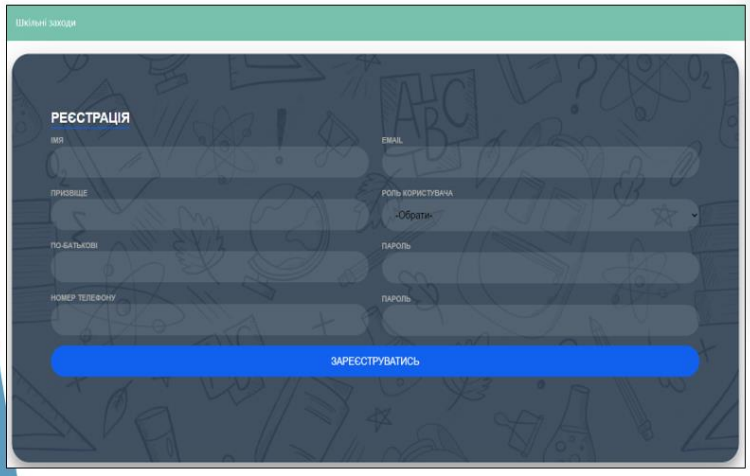


Макетування

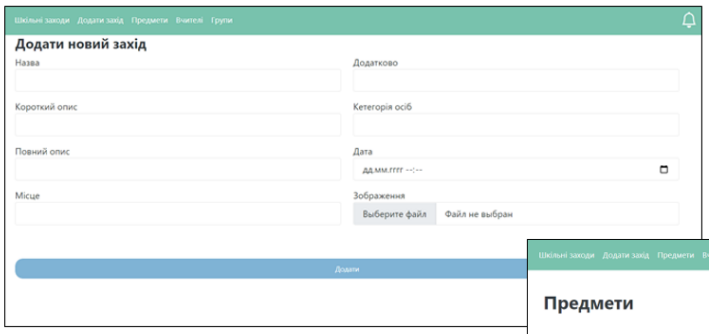


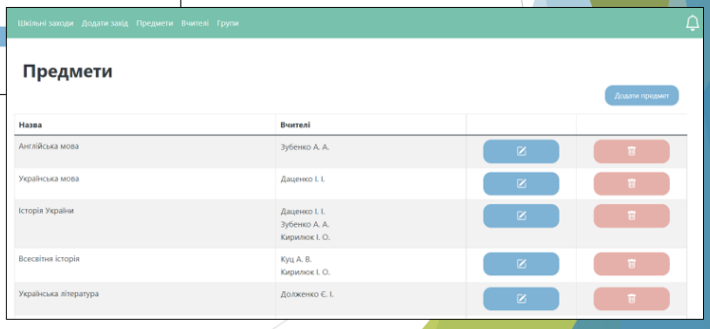


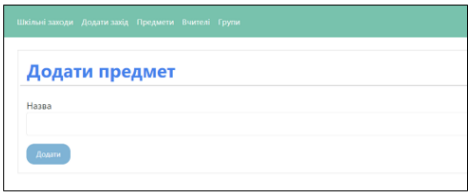
Авторизація/Реєстрація



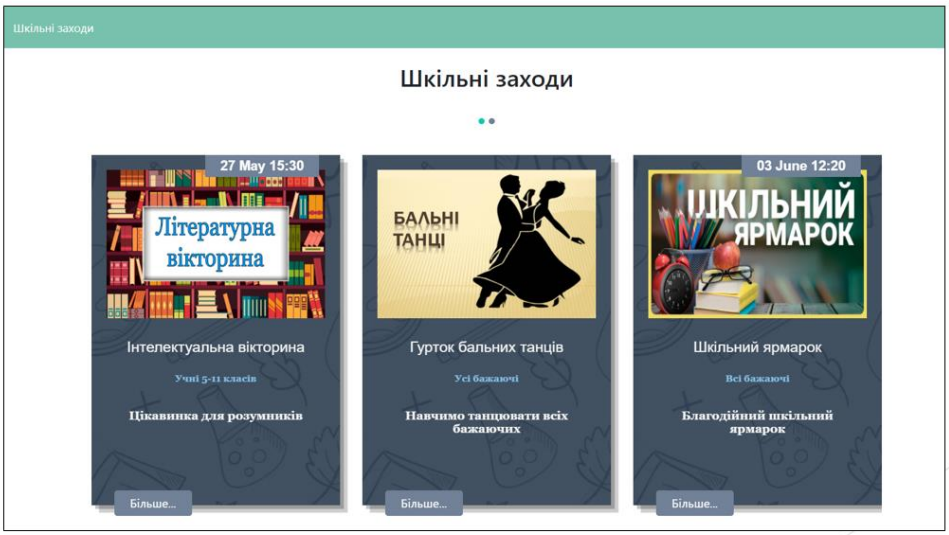
Додавання шкільних заходів та предметів



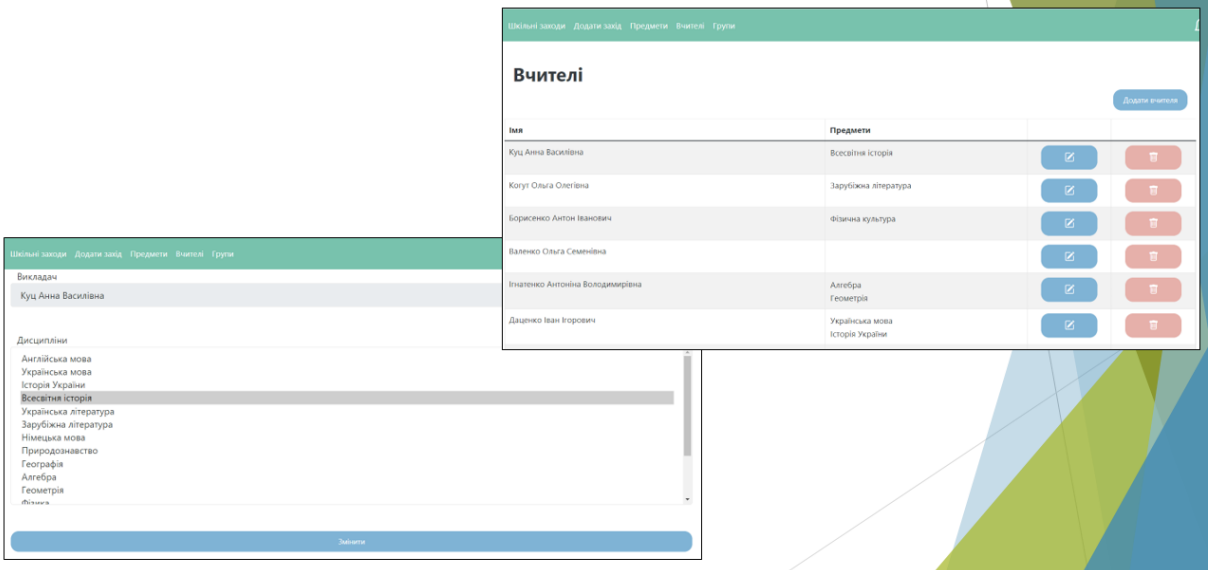




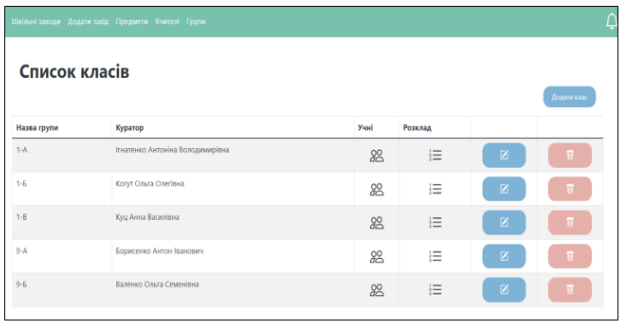
Сторінка із перерахунком шкільних заходів

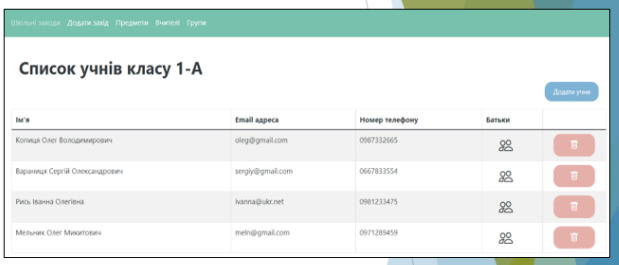


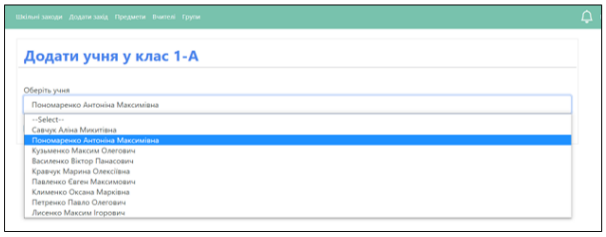
Додавання/редагування вчителів



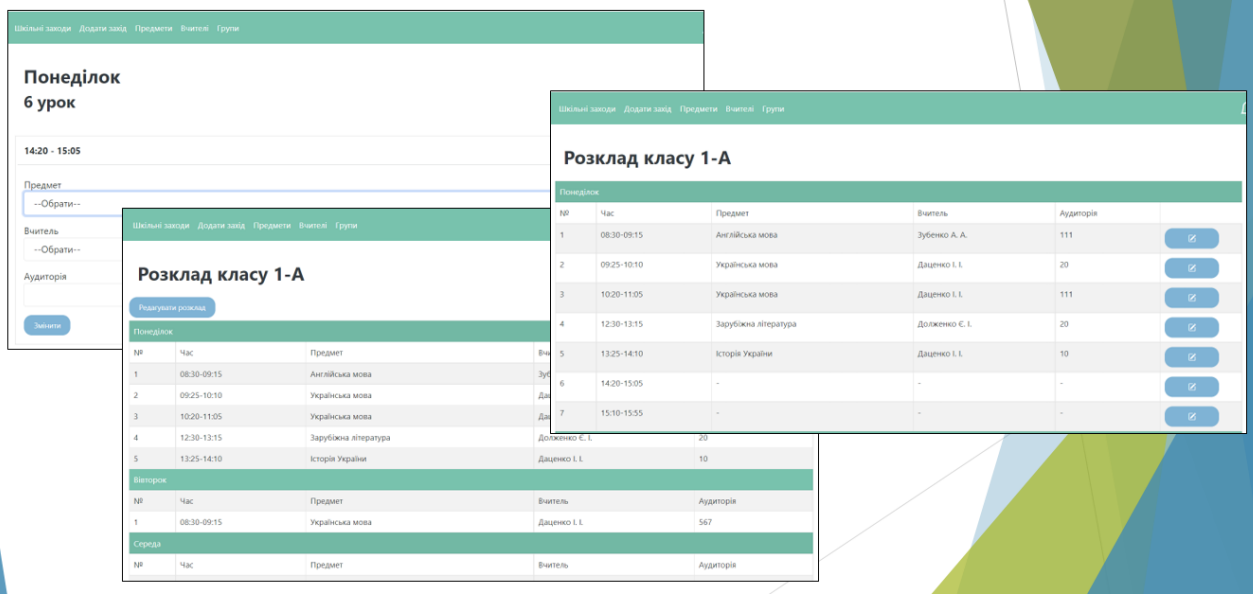
Робота з учнівськими групами (класами)



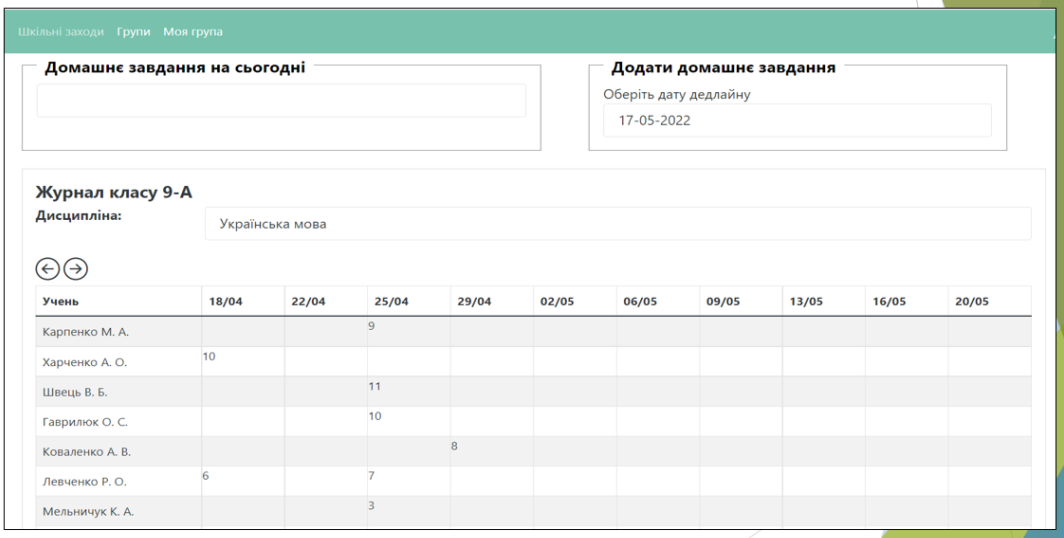




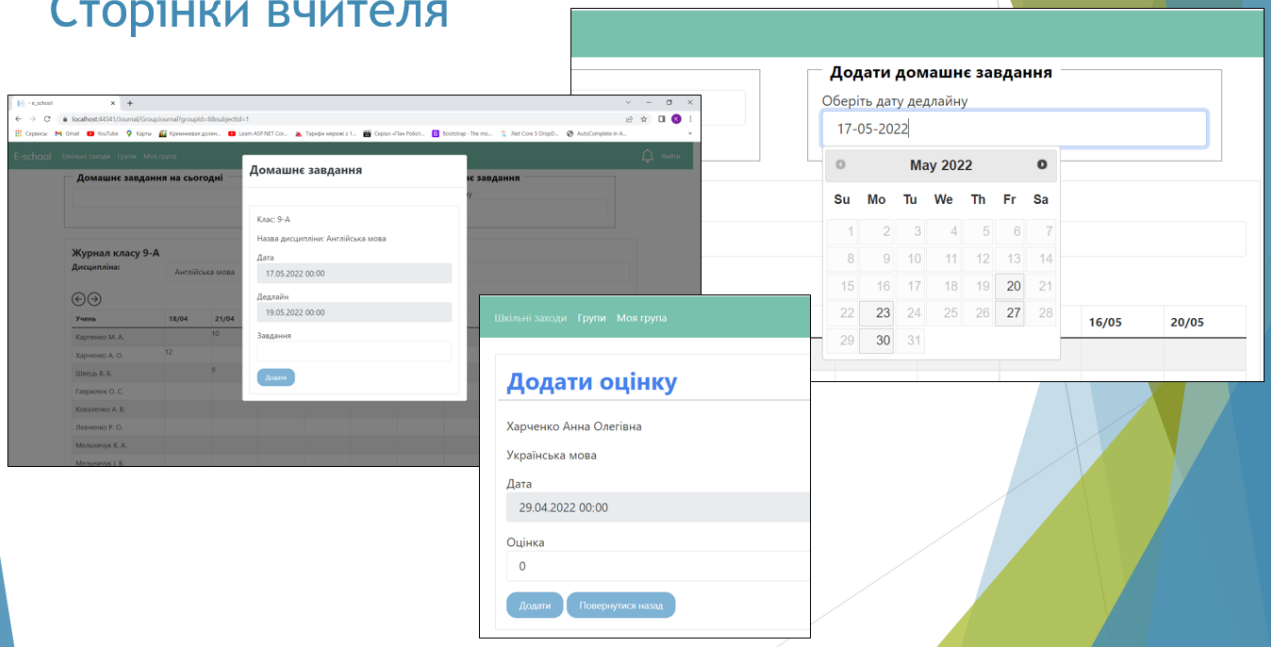
Вигляд та редагування розкладу



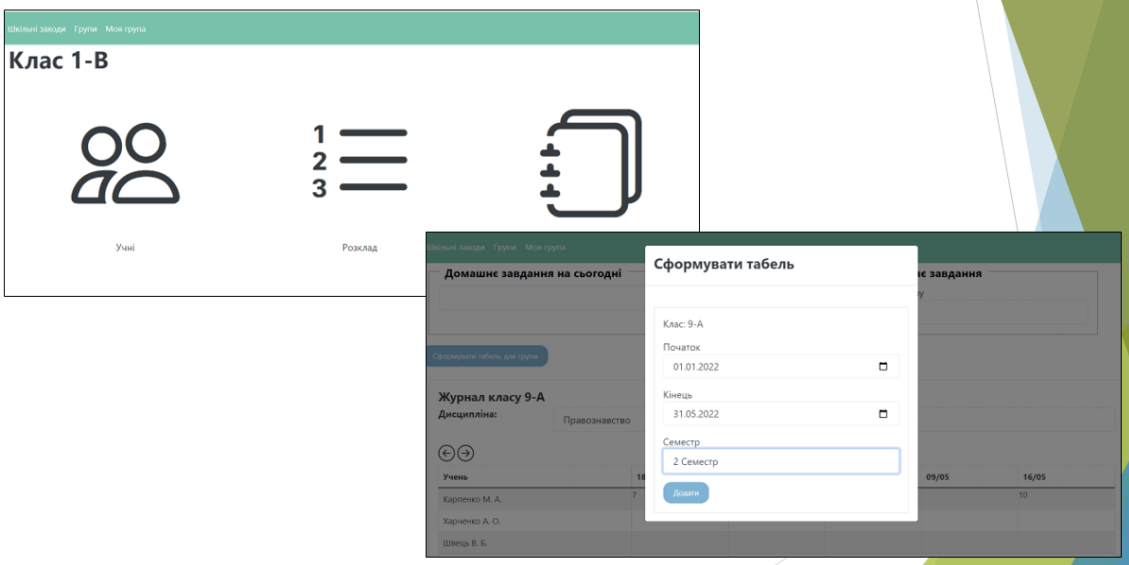
Журнал



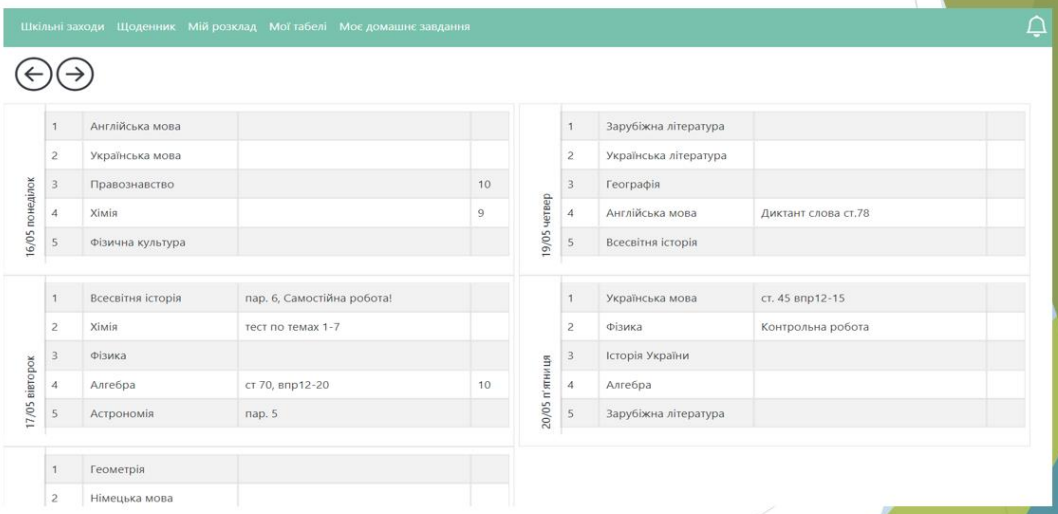
Сторінки вчителя



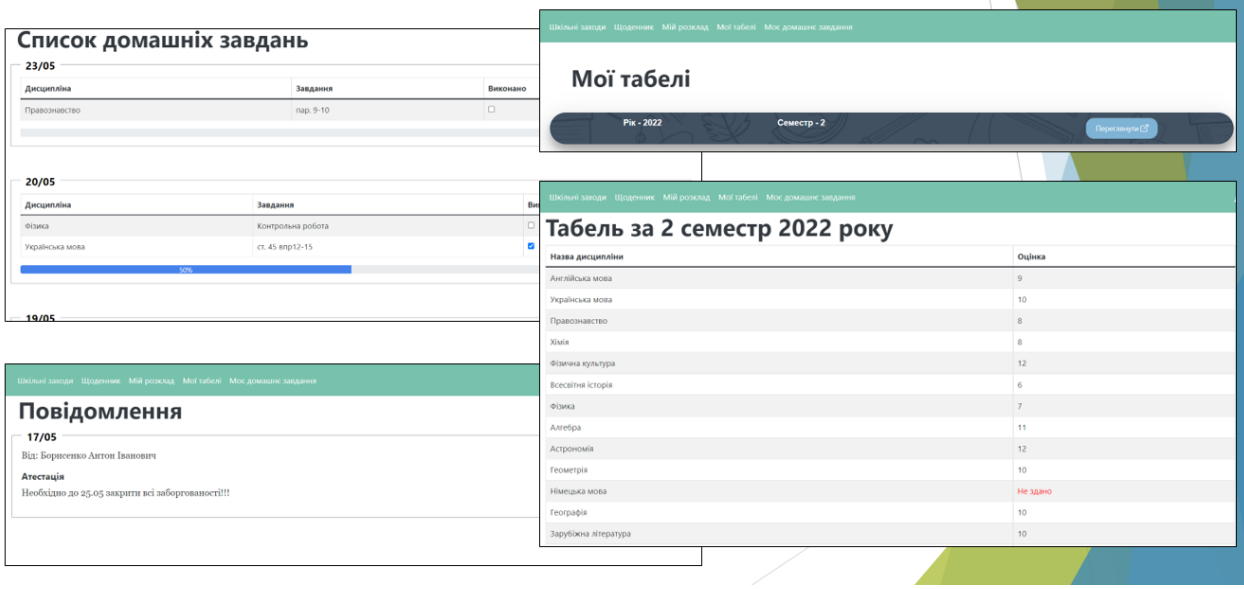
Робота класного керівника



Щоденник учня



Сторінки учня



Висновок

- Після виконання всіх етапів дипломного проекту було створено багатофункціональний сайт, що забезпечить організацію учбового процесу.
- Було здійснено проектування всіх деталей, враховуючи інтерфейс та архітектуру.
- Сайт має простий, зрозумілий інтерфейс.
- Забезпечується розмежування доступу для: батьків, учнів, вчителів, класних керівників.
- Система забезпечує необхідний функціонал для роботи учнів та вчителів.
- Існує роль адміністратора що відповідає за заповнення контенту.
- Сайт відповідає всім сформованим вимогам.

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

здобувача вищої освіти
Дрищ К.В.
Прізвище, ініціали

факультет ІТ, 3 курс, група ІПЗс-19-1

ЗАЯВА

З правилами чинного Положення «Про дотримання академічної доброчесності в Хмельницькому національному університеті» від 26.09.2020 (зі змінами від 26.11.2020), згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування заходів дисциплінарної та академічної відповідальності, ознайомлена. Про використання програмно-технічних засобів для перевірки кваліфікаційних робіт здобувачів вищої освіти на плагіат оповіщена та надаю свою згоду на обробку та збереження університетом моєї роботи в інституційному репозитарії університету.

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

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

31.05
дата

підпис

UNICHECK

Ім'я користувача:
Кафедра ІПЗ

ID перевірки:
1011345630

Дата перевірки:
26.05.2022 19:41:40 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
26.05.2022 19:42:33 EEST

ID користувача:
100005589

Назва документа: **Дипломний_проект_Дрищ_К_В_ІПЗс_19_1**

Кількість сторінок: **157** Кількість слів: **22848** Кількість символів: **181385** Розмір файлу: **10.43 MB** ID файлу: **1011231407**

11.3%
Схожість

Найбільша схожість: **1.61%** з джерелом з Бібліотеки (ID файлу: **1008265198**)

9.23% Джерела з Інтернету	552	Сторінка 159
3.51% Джерела з Бібліотеки	131	Сторінка 166

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

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

Замінені символи	1

Thu May 26 18:44:37 EEST 2022, Хіврич Володимир Русланович, Хмельницький національний університет, ХНУ

Anti-Plagiarism v-15.257

Максимальне співпадіння з одним документом 7.0%

Словники перевірки: en_US, ru_RU, ua_UA. **Помилок в документах: 8%**

ID: 104061 Назва: Програмне забезпечення для організації, управління та впорядкування навчального процесу у загально-освітніх школах Додано в БД: 2022-05-26 Автора: К. В. Дрищ Керівники: Н. І. Праворська Консультанти: Опоненти:	Документ		Сумарний збіг по Базі Даних	
	Символи	Лексеми	Символи	Лексеми
	75558	761	9540 (13%)	106 (14%)

Джерело плагіату

ID	Опис	Наявність плагіату в документі	
		Символи	Лексеми

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА ДИПЛОМНИЙ ПРОЕКТ
освітнього ступеня «Бакалавр»

Дипломник Дрищ Катерина Володимирівна

Тема Програмне забезпечення для організації, управління та впорядкування навчального процесу у загально-освітніх школах

Спеціальність 121 – Інженерія програмного забезпечення

Обсяг дипломного проекту:

Кількість листів креслень _____; кількість сторінок записки 73

1.Короткий зміст пояснювальної записки та прийнятих рішень У дипломному проекті було проведено: дослідження предметної області (опис предметної області, аналіз існуючих методів та рішень, визначення технічного завдання та вимог до проекту), проектування сайту (проектування архітектури, проектування бази даних, проектування інтерфейсу). Після чого здійснена реалізація проекту (структура програмних модулів, реалізація бази даних, реалізація сайту, інструкція з експлуатації), тестування та зроблено висновки

2. Висновок про відповідність проекту поставленому завданню Дипломний проект виконаний відповідно до поставленого завдання та з дотриманням всіх вимог.

3. Характеристика виконання кожного розділу проекту, ступінь використання останніх досягнень науки і техніки та передових методів роботи В ході виконання першого розділу проводилося ознайомлення із предметною областю продукту та здійснювався змістовний її аналіз. Внаслідок дослідження ринку програмного забезпечення та аналізу схожих розробок, було визначено функціональні вимоги до даного програмного рішення. Для даної системи під час цього етапу розроблено технічне завдання. Під час опрацювання другого розділу відбулося формулювання основних принципів роботи продукту. Спроектовано архітектуру програмного забезпечення. Було визначено, що сайт буде побудовано за клієнт-серверною архітектурою та розроблений за принципом MVC. Також проведено проектування інтерфейсу та бази даних, всіх таблиць та зв'язків між ними. В ході третього розділу відбулась реалізація адміністративної та користувацької частини. Для кожної з ролей було написано ряд функцій, та розроблено інтерфейс, що є простим та зручним. Також було розроблено інструкцію з експлуатації. Під час виконання четвертого розділу програмне забезпечення протестовано та визначено, що воно працює коректно та відповідає всім поставленим вимогам.

4. Позитивні сторони проекту Тематика дипломного проекту є актуальною, оскільки перехід шкіл на систему навчання онлайн став необхідністю. Програмне забезпечення містить всі необхідні інструменти для реалізації навчального процесу. Система є вузькоспеціалізованою і орієнтованою на навчальний процес саме у школах, що забезпечує її попит на ринку.

5. Негативні сторони проекту У проекті не реалізовано зворотній зв'язок батьків з викладачами, тому немає можливості спілкуватися напряму.

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

7. Відгук про дипломний проект в цілому Дипломний проект виконаний в повному обсязі із детальним, поетапним описом кожного кроку розробки програмного забезпечення. Надано різні типи діаграм для наочного представлення певних розділів програмного продукту. Розробка повністю відповідає темі дипломного проекту.

8. Інші зауваження

9. Оцінка дипломного проекту Дипломний проект виконаний у повному обсязі, відповідає поставленій задачі та заслуговує на оцінку «відмінно».

РЕЦЕНЗЕНТ Говорущенко Тетяна Олександрівна, доктор технічних наук, професор, завідувач кафедри комп'ютерної інженерії та інформаційних систем

"01" червня 2022 р. _____
(підпис)

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

Підтверджуємо ознайомлення з результатом звіту подібності щодо роботи, генерованого системою виявлення текстових збігів/ідентичності/схожості:
Назва: «Програмне забезпечення для організації управління та впорядкування навчального процесу у загально-освітніх школах»

Автор: Дрищ Катерина Володимирівна
Спеціальність: 121 – Інженерія програмного забезпечення
Освітня програма: Освітньо-професійна програма «Інженерія програмного забезпечення»
Науковий керівник: Праворська Наталія Іванівна, канд. пед. наук, доцент
Після аналізу звіту подібності зроблено такий висновок:

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

Підтвердження:
Більшість запозичень в роботі виявлені у стандартних бланках (титулка, завдання на дипломну роботу, структура змісту). Також в якості запозичень було визначено деякі фрагменти вихідного коду, що є стандартними при написанні будь-якої програми і це не є плагіатом.
У роботі виявлено 11,3 % запозичень, що є допустимою нормою. Робота приймається до захисту.

Керівник _____ Н. І. Праворська

Гарант ОП _____ Л. П. Бедратюк

Завідувач кафедри _____ Л. П. Бедратюк