ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ІВАНА ФРАНКА

Факультет прикладної математики та інформатики
(повне найменування назва факультету)

Кафедра програмування
(повна назва кафедри)

ДИПЛОМНА РОБОТА

на тему:
РОЗРОБКА ГЕНЕРАТОРА РОЗКЛАДУ ФАКУЛЬТЕТУ З ВИКОРИСТАННЯМ МЕТОДІВ МАШИННОГО НАВЧАННЯ

Виконали: студенти групи ПМІ-41
спеціальності
122 Комп'ютерні науки та інформаційні технології (інформатика)
(шифр і назва спеціальності)

Боліщук С. А.
(підпис) (прізвище та ініціали)

Оскірко М. С.
(підпис) (прізвище та ініціали)

Керівник
доцент Ярошко С. А.
(підпис) (прізвище та ініціали)

Консультант
старший викладач Костів В.Я.
(підпис) (прізвище та ініціали)

Рецензент

(підпис) (прізвище та ініціали)

Львів – 2023

ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ІВАНА ФРАНКА

Факультет прикладної математики та інформатики

Кафедра програмування

Спеціальність 122 Комп'ютерні науки та інформаційні технології (інформатика)
(шифр і назва)

«ЗАТВЕРДЖУЮ»

Завідувач кафедри

_____

" " 20 року

ЗАВДАННЯ

НА ДИПЛОМНУ РОБОТУ СТУДЕНТІВ

Боліщука Святослава Андрійовича
(прізвище, ім'я, по батькові)

Оскірка Максима Сергійовича
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка генератора розкладу факультету з використанням методів машинного навчання.

керівник роботи Ярошко Сергій Адамович, кандидат фізико-матиматичних наук, доцент_____,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

консультант роботи Костів Василь Ярославович, старший викладач

_____,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені Вченою радою факультету від "13" вересня 2022 року №15

2. Строк подання студентом роботи __13 червня 2023 року_____

3. Вихідні дані до роботи

__Дані розкладів занять факультету прикладної математики та інформатики Львівського національного університету імені Івана Франка, офіційні документації бібліотек, технологій та фреймворків

_____

4. Зміст дипломної роботи (перелік питань, які потрібно розробити)

__1) Визначити проблематику(Боліщук, Оскірко)_____

__2) Вибрати необхідні технології(Боліщук, Оскірко)_____

__3) Сформувати архітектуру бази даних(Боліщук, Оскірко)_____

__4) Реалізувати взаємодію з даними(Боліщук)_____

__5) Реалізувати взаємодію з розкладами(Боліщук)_____

__6) Розробити алгоритм перевірки на предмет конфліктів у розкладах(Боліщук)_____

__7) Розробити модель машинного навчання(Боліщук, Оскірко)_____

8) Підготувати історичні дані для навчання(Оскірко)

9) Розробити імпорт/експорт даних(Оскірко)

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

1) зображення діаграми бази даних

2) зображення діаграми потоків даних

3) зображення діаграми ітераційного процесу розробки моделі машинного навчання

4) зображення деяких частин коду системи

5) зображення інтерфейсу та роботи системи

6. Консультанти розділів роботи

Розділ	**Прізвище, ініціали та посада консультанта**	**Підпис, дата**	
		Завдання видав	**Завдання прийняв**

7. Дата видачі завдання ____13 вересня 2022 року_____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної роботи	Строк виконання етапів роботи	Примітка
1	*Вибір технологій*	*жовтень*	*Виконано*
2	*Формування архітектури бази даних*	*листопад*	*Виконано*
3	*Розробка основи інтерфейсу*	*грудень*	*Виконано*
4	*Програмна реалізація взаємодії з даними*	*лютий*	*Виконано*
5	*Програмна реалізація взаємодії з розкладами*	*лютий*	*Виконано*
6	*Розробка алгоритму перевірки на предмет конфліктів у розкладах*	*березень*	*Виконано*
7	*Розробка моделі машинного навчання*	*квітень*	*Виконано*
8	*Підготовка історичних даних для машинного навчання*	*квітень*	*Виконано*
9	*Імпортування/експортування даних в JSON*	*травень*	*Виконано*
10	*Виправлення помилок*	*травень*	*Виконано*
11	*Тестування роботи системи*	*червень*	*Виконано*
12	*Оформлення дипломної роботи*	*червень*	*Виконано*

Студенти _____ **Боліщук С.А.**

(підпис)

_____ **Оскірко М.С.**

(підпис)

Керівник роботи _____ **Ярошко С.А.**

(підпис)

Консультант роботи _____ **Костів В. Я.**

(підпис)

Автореферат

Дипломна робота покликана представити розроблену систему створення, редагування та перегляду розкладів занять факультету, яка б максимально спрощувала даний процес для користувача з допомогою методів машинного навчання. Це веб-додаток, який використовує фреймворк ASP.NET Core і бібліотеку для створення моделей машинного навчання ML.NET.

Основною проблемою створення нового розкладу у новому семестрі є суперечності між різними параметрами пари, яку додає користувач. Наприклад аудиторія може бути зайнята на даний момент у іншому розкладі, або викладач вже має заняття, чи інші можливі колізії. У даній системі всі ці проблеми було враховано і запропоновано рішення.

Всі використовувані при створенні розкладу дані можна редагувати використовуючи систему абсолютно автономно.

Важливим аспектом системи є використання машинного навчання для допомоги користувачеві. Модель пропонує предмети, викладачів та аудиторії, враховуючи заданий у самому ж додатку навчальний план та історичні дані, отримані з раніше використовуваних розкладів.

Зміст

Вступ

Очевидно, щороку виникає складність у побудові великої кількості нових розкладів занять в університеті. В процесі цього завдання необхідно враховувати багато факторів, таких як зміна кількості груп, доступність аудиторій, побажання викладачів та інші. Це, в свою чергу, може спричиняти помилки та суперечності, які потребують оперативного виправлення.

Метою даної роботи є максимальне спрощення процесу створення та редагування розкладів занять шляхом використання технологій .NET, ASP.NET Core MVC та ML.NET. Завдяки цим технологіям ми прагнемо забезпечити автоматизацію цього складного процесу та використати можливості машинного навчання для поліпшення ефективності та точності розкладу.

Дотримуючись принципів ASP.NET Core MVC, ми розробляємо веб-додаток, який дозволить створювати, переглядати та редагувати розклади занять зручним способом. Використання технології .NET дозволяє нам розробити потужні функції для обробки та збереження даних розкладу, а також для взаємодії з іншими системами, наприклад, для доступу до інформації про групи, викладачів та аудиторії.

Крім того, ми використовуємо ML.NET - бібліотеку машинного навчання в .NET, щоб покращити процес створення розкладу та зменшити ймовірність помилок. Машинне навчання дозволяє аналізувати великі обсяги даних про попередні розклади, використовувати цю інформацію для прогнозування оптимального розміщення занять, а також для виявлення конфліктів та суперечностей у розкладі.

Наша робота має на меті спростити та автоматизувати процес створення та редагування розкладів занять в університеті. Ми прагнемо підвищити ефективність та точність розкладу завдяки використанню технологій .NET, ASP.NET Core MVC та ML.NET. Наш додаток надасть робітникам університету потужний інструмент, який допоможе їм легко керувати розкладами та забезпечити оптимальне використання ресурсів.

Суть дипломної роботи

1. Огляд використаних технологій

Мова програмування C#

Мова програмування C# (C-Sharp) - це сучасна об'єктно-орієнтована мова програмування, розроблена компанією Microsoft. C# підтримує основні принципи об'єктно-орієнтованого програмування, такі як наслідування, поліморфізм, абстракція та інкапсуляція. Він також має підтримку для делегатів, подій, інтерфейсів та інших конструкцій ООП.

C# є основною мовою програмування для платформи .NET, що надає широкий спектр функцій і бібліотек для розробки різноманітних додатків. Розробники можуть використовувати .NET Framework для розробки десктопних додатків під Windows або .NET Core для розробки крос-платформних додатків, включаючи веб-додатки і хмарні служби.

C# надає вбудовану підтримку для безпеки типів, що допомагає уникнути помилок виконання через неправильне використання типів даних. Вона також має механізми для обробки винятків та контролю пам'яті.

Поміж інших можливостей, C# підтримує асинхронне програмування з використанням ключових слів async та await, LINQ (Language-Integrated Query) для роботи з даними та велику кількість додаткових бібліотек, які розширюють функціональність мови.

Мова програмування C# є потужним інструментом для розробки різноманітних додатків, від десктопних програм до веб-додатків і хмарних рішень. Вона має широку спільноту розробників і надійну підтримку з боку Microsoft.[6][7]

Мова програмування Javascript

Окрім C# ми також використовували Javascript на Front-end частині.

JavaScript - це мова програмування, яка використовується для розробки веб-додатків. Вона є однією з ключових технологій для фронтенд-розробки, оскільки дозволяє взаємодіяти з користувачем на стороні клієнта. JavaScript забезпечує можливість динамічного змінювати контент сторінки, обробляти події, створювати анімацію та взаємодіяти зі структурою HTML і CSS. Вона є основною мовою для програмування клієнтської частини веб-додатків.

JavaScript підтримує об'єктно-орієнтовану парадигму програмування. Об'єкти в JavaScript можуть бути створені за допомогою літерала об'єкта або за допомогою конструкторів об'єктів. Вона також має прототипне спадкування, що дозволяє створювати ієрархію об'єктів.

JavaScript має велику кількість бібліотек, фреймворків та інструментів, що полегшують розробку веб-додатків. Наприклад, jQuery, React, Angular, Vue.js - це лише кілька з відомих бібліотек та фреймворків JavaScript. JavaScript є однією з найпопулярніших мов програмування, особливо у веб-розробці, і має широку спільноту розробників, що надає багато ресурсів та підтримки.[8]

ASP.NET Core

ASP.NET Core є високопродуктивним, відкритим та крос-платформовим фреймворком для розробки веб-додатків. Він є наступником популярного фреймворка ASP.NET і був розроблений компанією Microsoft.

Одна з головних переваг ASP.NET Core - це можливість розробки додатків, які можна запускати на різних платформах, таких як Windows, macOS і Linux. Це досягнуто за рахунок використання крос-платформового фреймворка .NET Core.

ASP.NET Core пропонує високу швидкодію та продуктивність завдяки оптимізаціям, таким як злиття запитів, кешування, асинхронність і багатопоточність. Він також використовує оптимізований веб-сервер Kestrel.

Також дуже важливим є те, що ASP.NET Core дозволяє використовувати код C# в коді веб сторінки за допомогою так званих tag helper'ів. Це значно збільшує зручність в створенні дизайну та дає більше можливостей.[3][4]

ML.NET

ML.NET — це бібліотека машинного навчання з відкритим кодом, розроблена Microsoft. Дозволяє розробникам створювати власні моделі машинного навчання та інтегрувати їх у свої програми. ML.NET надає набір інструментів, які спрощують процес створення моделей машинного навчання, навчання їх на даних і розгортання їх в різних сценаріях. ML.NET підтримує широкий спектр завдань машинного навчання, включаючи класифікацію, регресію, кластеризацію, рекомендації та виявлення аномалій.

Основні особливості ML.NET:

- Простота використання: ML.NET надає високорівневий API та набір готових компонентів, які спрощують розробку моделей машинного навчання.
- Крос-платформовість: ML.NET підтримує Windows, Linux і macOS.
- Інтеграція з .NET-екосистемою: ML.NET легко інтегрується з іншими компонентами .NET, такими як ASP.NET, Azure, SQL Server тощо.
- Робота в автономному режимі: тренування і використання моделей не вимагає доступу до Інтернету.

Загалом, ML.NET є потужним інструментом для розробки моделей машинного навчання в середовищі .NET.[11][12]

Entity Framework

Entity Framework - це платформа ORM з відкритим вихідним кодом, розроблена Microsoft для додатків .NET. Ця технологія дозволяє розробникам працювати з даними, використовуючи класи, що специфічні для конкретного

домену, замість того, щоб зосереджуватись на базових таблицях і стовпцях бази даних, де ці дані зберігаються. Завдяки Entity Framework розробники можуть працювати на більш абстрактному рівні при роботі з даними, а також створювати та підтримувати програми, орієнтовані на дані, з використанням меншої кількості коду порівняно з традиційними підходами.

Серед підходів до створення бази даних, в Entity Framework найчастіше використовується саме підхід Code First, який ми і застосували в нашому проекті. Він полягає в тому, що розробник фокусується на додатку і створює класи для нього, а не спочатку розробляє свою базу даних, а потім створює класи, що відповідають дизайну бази даних.

Офіційне визначення: «Entity Framework — це об’єктно-реляційний маппер (O/RM), який дозволяє розробникам .NET працювати з базою даних за допомогою об’єктів .NET. Це усуває потребу в більшості коду доступу до даних, який зазвичай потрібно писати розробникам».[1][2]

Microsoft SQL Server

Microsoft SQL Server є системою управління реляційними базами даних, розробленою компанією Microsoft. Цей програмний продукт призначений для зберігання та отримання даних за запитом інших програм, які можуть працювати на тому ж комп'ютері або на інших комп'ютерах в мережі, включаючи Інтернет. Microsoft SQL Server має кілька різних версій, спрямованих на різні аудиторії та різні навантаження, починаючи від невеликих однокористувацьких програм до великих Інтернет-додатків з багатьма одночасними користувачами.

Ми використали саме цю систему керування реляційними базами даних, тому, що вона є вбудована в середовище Visual Studio, яке ми використовували, і це досить зручно при розробці.[5]

Bootstrap

Для дизайну Front-end частини ми використали фреймворк Bootstrap.

Bootstrap є безкоштовним фреймворком з відкритим вихідним кодом, який використовується для розробки інтерфейсів, веб-сайтів і веб-додатків. Він побудований на HTML, CSS і JavaScript і спроектований для спрощення процесу створення адаптивних веб-сайтів і програм. Адаптивний дизайн дозволяє веб-сторінці або додатку адаптуватися до розміру та орієнтації екрана користувача, забезпечуючи оптимальне відображення. Bootstrap містить набір компонентів інтерфейсу користувача, готових макетів та JavaScript-інструментів, що сприяють його реалізації.

Selectize.js

Бібліотека створена за допомогою Bootstrap, яка дозволяє використовувати гарні та функціональні випадаючі меню (dropdown menu), замість вбудованих в html.

Datatable.js

Бібліотека на Javascript, яка дозволяє використовувати на сторінках просунуті таблиці, які можна детально налаштовувати. Ми використали такі можливості таблиці як сортування, пагінація, загальний пошук, пошук по конкретному стовпчику.

Git

Git - це розподілена система керування версіями з відкритим вихідним кодом, яка дозволяє ефективно працювати з проектами різних розмірів - від невеликих до дуже великих. Git є легким у використанні та має компактний розмір, що сприяє швидкій продуктивності. В порівнянні з іншими системами керування версіями, такими як Subversion, CVS, Perforce і ClearCase, Git вирізняється завдяки своїм функціям, таким як ефективне локальне розгалуження, зручність переходу між гілками та підтримка кількох робочих процесів.

Варто наголосити, що вибране нами середовище Visual Studio також дозволяє працювати з Git в інтерфейсі, що є дуже зручно.

2. Перелік основних термінів

Архітектура MVC

MVC - це архітектурний шаблон, що складається з трьох компонентів: Модель (Model), Подання (View) і Контролер (Controller). Його основна ідея полягає в розділенні додатка на логічні частини: модель, яка представляє дані і бізнес-логіку, подання, яке відповідає за відображення інформації користувачеві, і контролер, який керує взаємодією між моделлю та поданням. Хоча початково цей шаблон використовувався для розробки десктопних графічних інтерфейсів користувача, сьогодні він широко застосовується в розробці мобільних додатків і веб-додатків.

Контролер (Controller)

Контролер в архітектурі MVC обробляє будь-який вхідний URL-запит. Контролер — це клас, похідний від базового класу System.Web.Mvc.Controller. Клас контролера містить відкриті методи, які називаються Action методами. Контролер і його Action метод обробляють вхідні запити браузера, отримують необхідні дані моделі та повертають відповідні відповіді.[9]

Подання (View)

У шаблоні Model-View-Controller (MVC) подання обробляє представлення даних програми та взаємодію з користувачем. Подання — це HTML-шаблон із вбудованою розміткою Razor. Розмітка Razor – це код, який взаємодіє з розміткою HTML для створення веб-сторінки, яка надсилається клієнту.

У ASP.NET Core MVC представлення є файлами .cshtml, які використовують мову програмування C# у розмітці Razor. Зазвичай файли перегляду групуються в папки, названі для кожного з контролерів програми. [10]

Модель (Model)

Model - це класи, призначені для керування базою даних

Ці класи моделі використовуються з Entity Framework Core (EF Core) для роботи з базою даних.

Створені класи моделі відомі як класи POCO (Plain Old CLR Objects). Класи POCO не залежать від EF Core. Вони визначають лише властивості даних, які будуть зберігатися в базі даних.

ViewModel

ViewModel являє собою дані, які ви хочете відобразити у своєму поданні, незалежно від того, чи використовуються вони для статичного тексту чи для введених значень (наприклад, текстових полів і спадних списків), які можна додати до бази даних (або відредагувати).

Контекст даних

Контекст даних представляє собою сеанс з базовою базою даних, який дозволяє виконувати операції CRUD (Створення, читання, оновлення, видалення).

Клас контексту в Entity Framework — це клас, похідний від *System.Data.Entity.DbContext* в EF 6 і EF Core. Екземпляр класу контексту представляє шаблони *Unit Of Work* і *Repository*, в яких він може поєднувати кілька змін в рамках однієї транзакції бази даних.

Клас контексту використовується для запиту або збереження даних у базі даних. Він також використовується для налаштування класів домену, пов’язаного з базою даних, змін налаштувань відстеження, кешування, транзакцій тощо.

Модель машинного навчання

Модель машинного навчання - це математична конструкція або алгоритм, який використовується для вирішення завдань штучного інтелекту, тобто виявлення закономірностей, спільних рис чи взаємозв’язків в наборах вхідних

векторів даних і здійсненні прогнозів без явного програмного подання. Для навчання моделей використовуються історичні дані, які потім слугують для виконання завдань регресії, класифікації, виявлення аномалій в даних та інших. Модель фіксує закономірності та зв’язки між даними, виявлені алгоритмом під час процесу навчання.

Модель у ML.NET складається з двох основних компонентів:

- Схема моделі: визначає структуру та типи ознак (вхідні змінні, англ. features) і міток (вихідні змінні, англ. labels), які модель використовує для прогнозування. У процесі навчання модель вчиться асоціювати ознаки з відповідними мітками.
- Параметри моделі: це внутрішні значення або ваги, отримані моделлю під час процесу навчання. Параметри визначають конкретну конфігурацію моделі та використовуються для прогнозування на основі нових вхідних даних.

Після навчання модель може бути збережена у вигляді zip-файлу для подальшого використання в інших програмах.

Ознака (у машинному навчанні)

У машинному навчанні ознака (англ. *feature*) - це певний параметр, характеристика чи властивість спостережуваного явища, яку можливо виміряти і подати здебільшого у числовій формі для використання у алгоритмі машинного навчання. Іншими словами ознаки - це вхідні змінні, характеристики або параметри, які використовуються для прогнозування.

Мітка (у машинному навчанні)

Мітки у машинному навчанні (англ. *label*) - вихідні змінні, значення які мають бути передбачені моделлю на основі вхідних параметрів (ознак). Мітки представляють бажаний результат або ціль завдання машинного навчання.

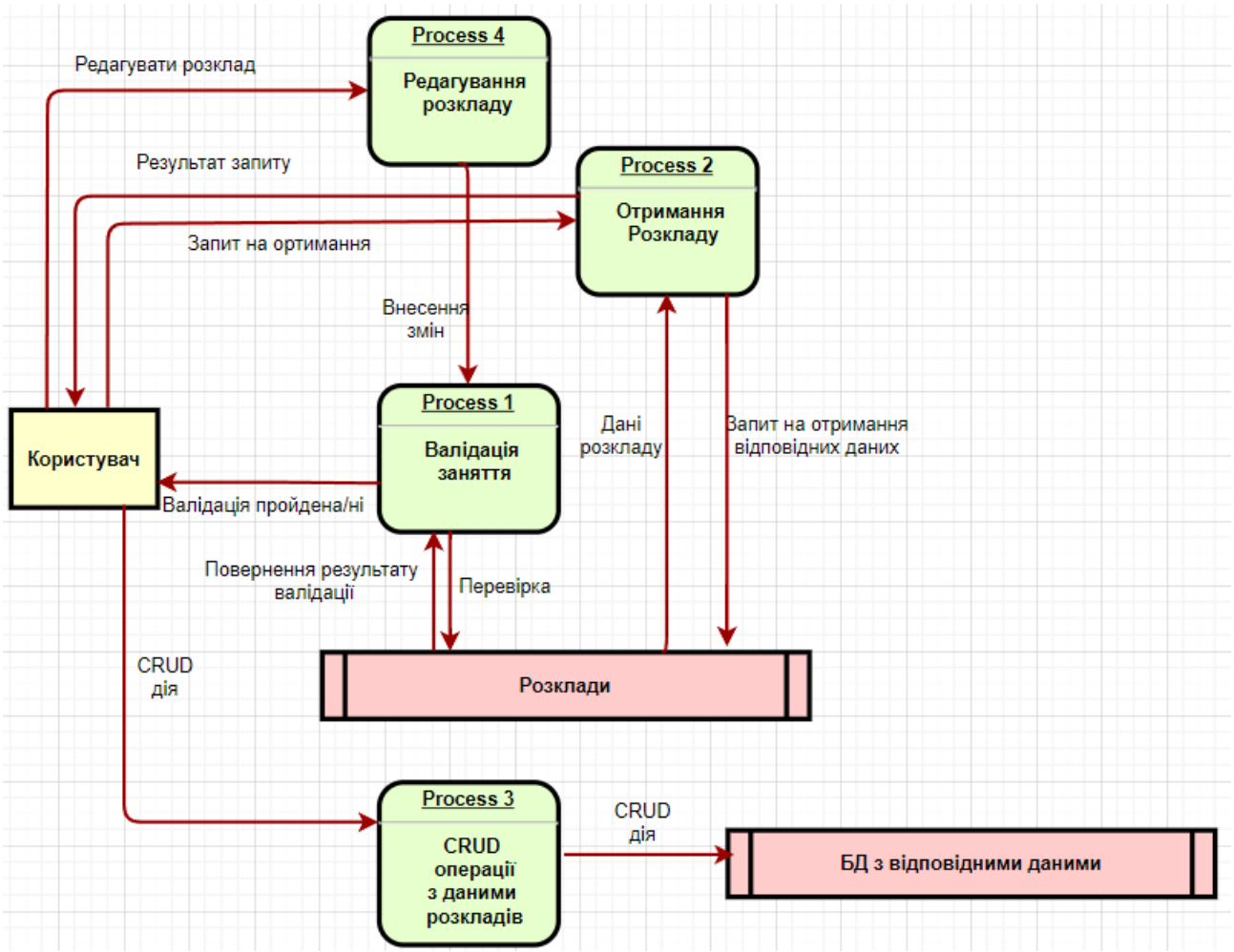
Мультикласова класифікація

(англ. *multi-class classification*) - завдання машинного навчання, яке полягає в тому, щоб визначити приналежність вхідного вектора даних до одного з кількох наперед визначених класів. Іншими словами - це завдання передбачення однієї мітки або класу з набору можливих класів.

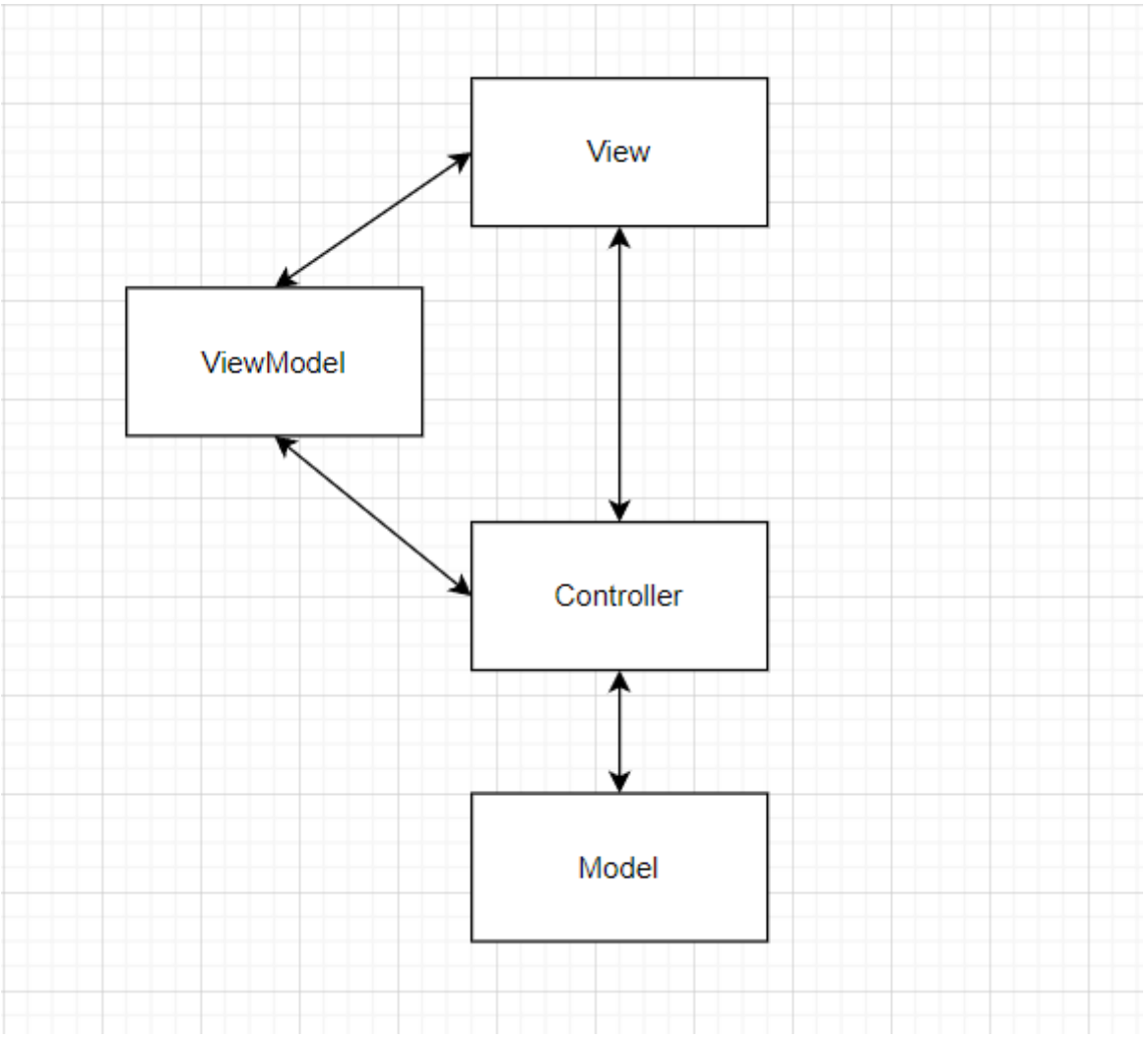
3. Реалізація

3.1 Архітектура

Діаграма потоків даних системи



Система використовує архітектурний паттерн MVC, який можна зобразити так:



Основою системи є шар Model, який представляє собою набір класів моделей даних і клас контексту даних, з допомогою яких Entity Framework Core згенерує базу даних при першому запуску програми. Тобто модель - це клас який містить всі необхідні поля та навігаційні властивості для Entity Framework.

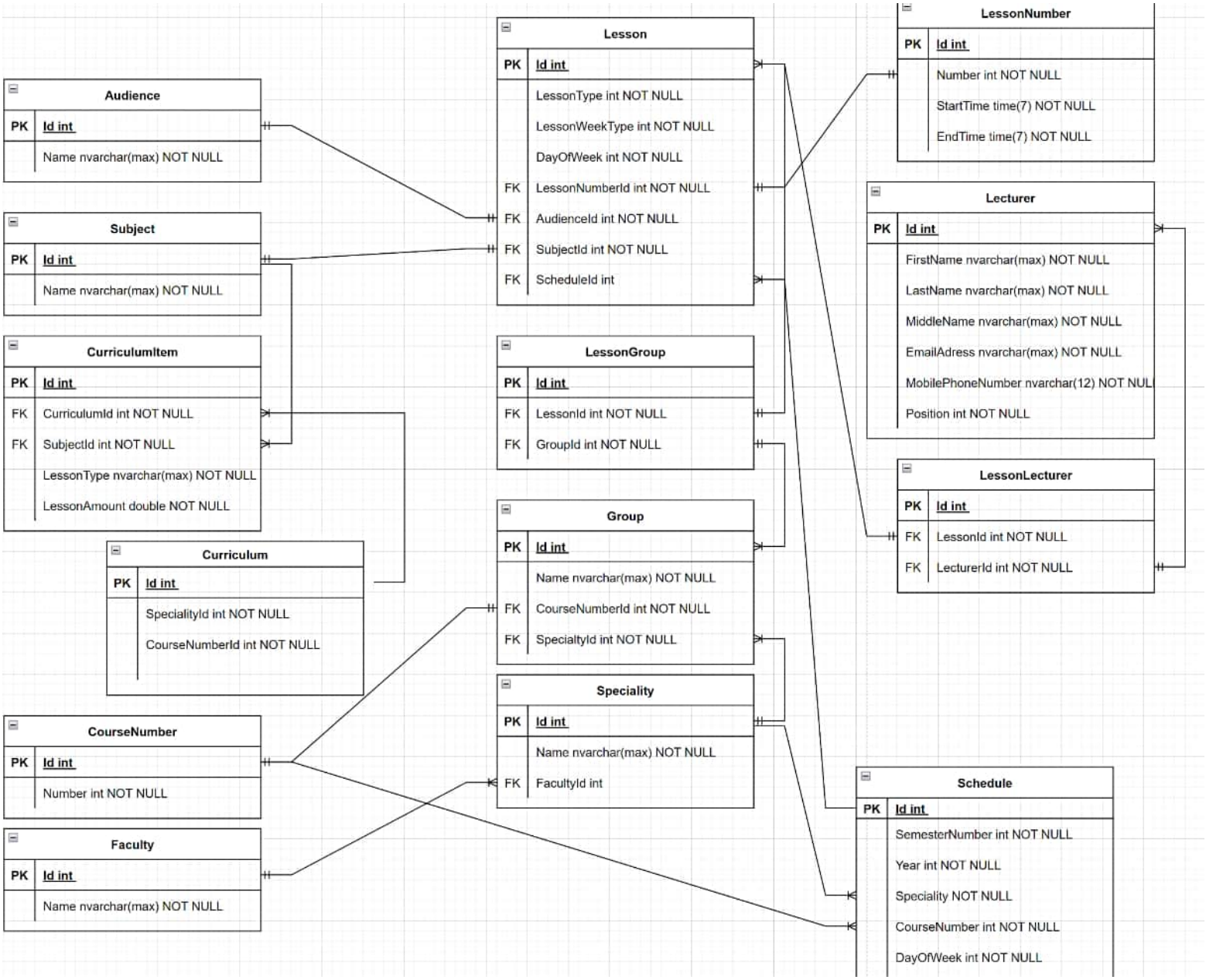
Приклад:

Для прикладу візьмемо модель Спеціальність (див. Додаток А):

Вона має поле Name для назви, а також так звані навігаційні властивості FacultyId для зберігання id факультету, до якого належить спеціальність, Faculty для посилання на сам об’єкт факультету, та Groups - посилання на групи, які належать спеціальності. За допомогою цих властивостей, ми вказуємо Entity

Framework зв'язки які буде містити модель. Тобто якщо подивитись на ER-діаграму бази даних, то ми побачимо, що модель Спеціальність відноситься до одного факультету та містить багато груп.

База даних при цьому матиме вигляд, зображений на діаграмі:



При першому запуску програми, створюється база даних і заповнюється певними даними за замовчуванням(див. Додаток Б). Ці дані визначаються в класі контексту (див. Додаток В).

Шар Controller представляє собою набір класів, які наслідуються від базового класу *Controller*. Контролер - клас який працює з даними, реалізовує

логіку системи, викликає View або приймає від нього дані. Для взаємодії з базою даних кожен з цих класів використовує контекст даних.

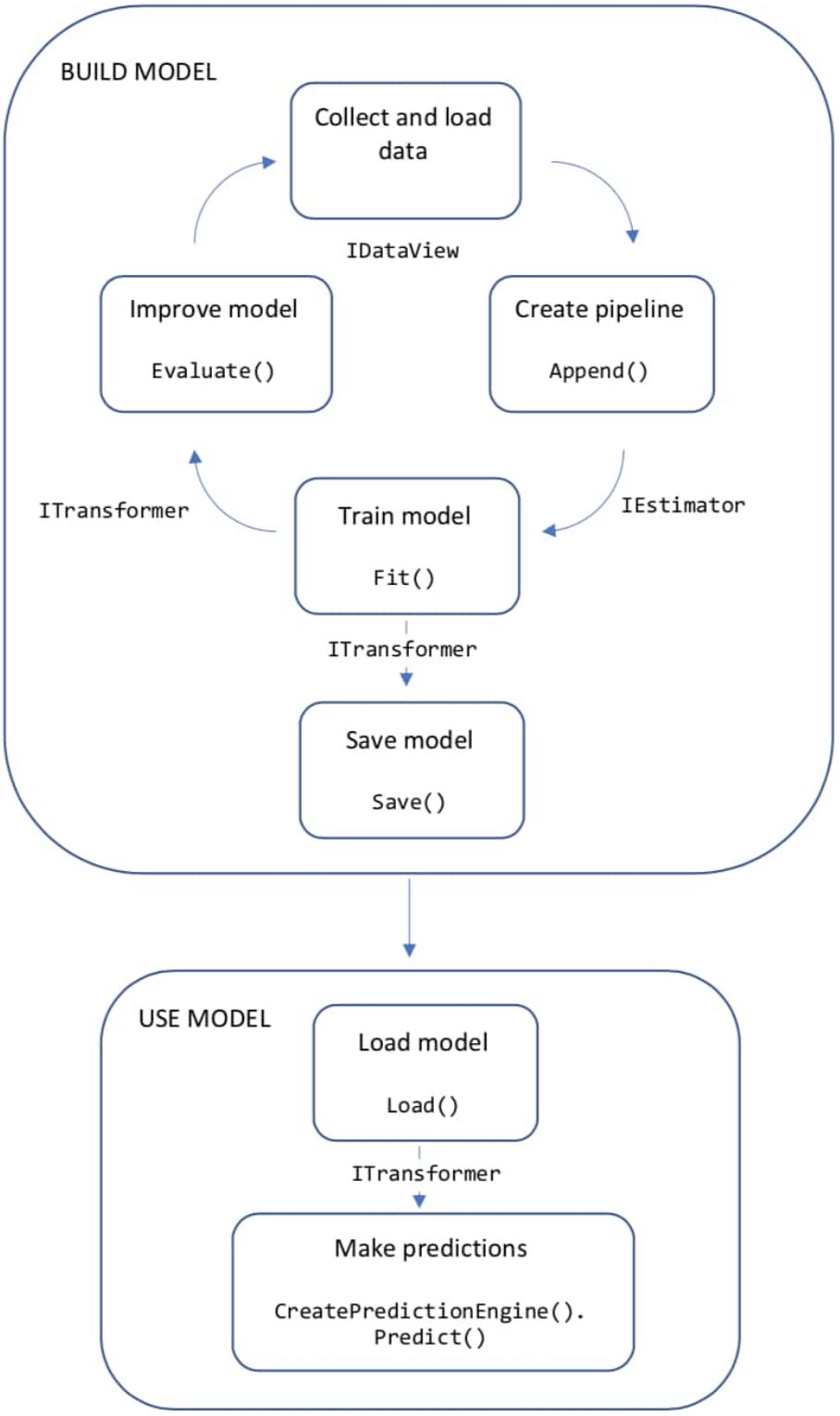
Шар View(представлення) - це самі сторінки, які бачить користувач. Сторінка може приймати дані, та відправляти дані назад в контролер. Для цього можна використати тип моделі яка передається.

Приклад:

Коли ми знаходимось на сторінці і натискаємо відповідну кнопку, то наша сторінка, тобто представлення(View), звертається до контролера. Скажемо, ми хочемо переглянути спеціальності. Зліва на сторінці вибираємо Дані -> Спеціальності. Після цього у відповідний контролер SpecialityController викликається відповідний метод, який визначений в представленні, в нашому випадку метод Configure(див. Додаток Г). В цьому методі з контексту даних вибираються всі спеціальності в базі та викликається представлення, в яке і передаються ці всі дані. І ми бачимо, як в нас відкрилась сторінка з списком спеціальностей. (код представлення див. Додаток Ґ)

Якщо в представлення чи контролер потрібно передати не просто модель, а інші/додаткові дані чи наприклад кілька різних моделей, то можна використовувати *ViewModel.* (див. Додаток Д, Е). Тобто *ViewModel* - це просто клас який містить в собі всі необхідні для передачі дані. При чому не обов'язково при передачі *ViewModel* заповнювати всі поля. Тобто одну *ViewModel* можна використовувати для різних цілей та для кожної з них заповнювати відповідні(не всі) поля.

Наступна діаграма представляє структуру коду програми, а також ітераційний процес розробки моделі машинного навчання з використанням бібліотеки ML.NET (див. Використані джерела [13]):



3.2 Реалізація

Для початку варто згадати, що ми використали готовий layout “NiceAdmin” який є в вільному доступі. Тому за правилами на кожній сторінці внизу в нас є напис © *Copyright NiceAdmin. All Rights Reserved.*
(layout - загальна сторінка сайту яка містить в собі інші сторінки, тобто та частина сайту яка на кожній сторінці однакова)

Функціонально система розділена на дві складові: взаємодію з даними і взаємодію з власне розкладом.

3.2.1 Взаємодія з даними

Для роботи з головною сторінкою були створені відповідне представлення та контролер.

Для взаємодії з моделями, для кожної з них ми створили свій контроллер.

Дані для всіх моделей можливо додавати, видаляти чи редагувати а також проводити пошук необхідних значень (CRUD функціонал) (див. Додаток Є).

Для CRUD операцій для кожної моделі були створені відповідні представлення.

Перелік моделей для редагування:

1. Навчальний план
2. Номер пари
3. Курс
4. Факультет
5. Спеціальність
6. Група
7. Викладач
8. Аудиторія
9. Предмет

В клас контексту були внесені початкові дані, які заповнюються при створені бази даних: близько 200 викладачів, 160 предметів, 60 груп, 50 аудиторій та трішки менше даних для інших моделей. За допомогою нашої програми було створено розклади занять кількох минулих років для 1-4 курсів спеціальностей комп'ютерні науки, кібербезпека та середня освіта та експортовано їх в файли JSON. При створені база даних заповнюється цими розкладами які в подальшому будуть використовуватись для машинного навчання.

3.2.2 Взаємодія з розкладами

Для взаємодії з розкладами в нас є 3 розділи:

- Дивитись
- Редагувати
- Видалити

(див. Додаток Ж)

Після натискання “Редагувати” система пропонує обрати факультет і курс, а також спеціальність і день тижня. Після цього буде згенерована таблиця, яка міститиме часові рамки пар і всі групи відповідного курсу, факультету і спеціальності, обраних раніше. Після натискання “+” на клітинці відповідних групи і номеру пари з’явиться вікно вибору параметрів пари (див. Додатки З). Після натискання “-” на клітинці відповідних групи і номеру пари з’явиться вікно видалення пари.

Також існує можливість завантажити таблицю з JSON файла (внизу сторінки)(див. Додаток И). В такому разі таблиця заповнюється автоматично.

Оскільки для одного заняття має бути можливість вибрати більше одного викладача та групи, ми використали бібліотеку selectize.js. З її допомогою ми помістили на форму створення заняття випадаюче меню з мульти-вибором. Також він має вбудований пошук. (див. Додаток І).

Для заняття можна вибрати тип чисельник/знаменник/кожен тиждень. Залежно від вибору в таблиці заняття відобразиться відповідним чином.

(див. Додаток Ї, Й).

Також заняття можна розтягнути на кілька груп. Відповідно в таблиці ці клітинки об’єднаються (див. Додаток К). Це було досягнуто як tag-хелперами C# так і скриптами на Javascript. Це також працює і з чисельником та знаменником (див. Додаток Л). Як це працює? Якщо коротко, (частина C#) то перед заповненням рядка таблиці, створюється поле lessonPrevious, в який після кожного заповнення клітинки записується поточна пара. Відповідно, перед кожним заповненням клітинки перевіряється, чи поточна пара, яка буде заповнена - така сама, як попередня(lessonPrevious). Якщо така сама, то клітинка залишається пуста, але до неї додається клас (параметр) cellToRemove. (див. Додаток М). (частина Javascript) Після створення таблиці, запускається скрипт на JavaScript, який вибирає всі елементи на сторінці з класом cellToRemove, і для кожного з них робить наступне: збільшує параметр colSpan попереднього (лівого) елемента на 1, та видаляє себе. Параметр colSpan розтягує клітинку по рядку на 1 далі.(див. Додаток Н)

Якщо було додано на пару №1 заняття на 3 групи, наприклад ПМІ-11, 12, 13, то при наступній спробі додати заняття на пару №1 будуть доступні лише групи, які залишилися, тобто ПМІ-14, 15. Це обраховується вже на сервері, тобто в контролері. Просто беруться заняття з бази і перевіряється, які групи ще не мають занять.

Якщо, наприклад, має бути одна пара на групи ПМІ-11 та 13, їх клітинки не об’єднати, бо вони далеко одна від одної, то ми зробили підсвітку для конкретної пари при наведенні на неї. Це було досягнуто за допомогою скриптів на Javascript. Якщо коротко, то при заповненні клітинки в його клас (параметр) записується слово highlight та id самого заняття. Завдяки цьому при наведенні на клітинку з заняттям, відповідний скрипт підсвічує всі клітинки з таким самим id та які містять слово highlight в класі.

При натисканні користувачем кнопки “**Завершити розклад**” на сторінці редагування розкладу, система додасть всі заняття у об’єкт розкладу в базі даних “**Schedules**”, для подальшої взаємодії з ними, наприклад перегляду розкладів минулих років. Також готові розклади використовуються для

навчання моделі, що дозволяє пропонувати предмети, аудиторії, викладачів у меню створення пари в розкладі.

Після натискання користувачем кнопки “**Завершити розклад**” система запропонує завантажити готовий розклад у форматі JSON.(див. Додаток О).

Після натискання “**Дивитись**” система пропонує обрати факультет і курс, потім спеціальність, і тоді рік і семестр. І далі можна буде переглянути розклад на всі дні. (див. Додаток П).

Після натискання “**Видалити**” система пропонує обрати факультет. І при натисканні кнопки Видалити появиться повідомлення з підтвердженням. Якщо натиснути “**Підтвердити**”, то буде видалено з бази даних всі заняття, які стосуються вибраного факультету для поточного року та семестру (див. Додаток Р). Це зроблено для того, щоб при втраті розкладу поточного семестру актуальності не потрібно було видаляти вручну кожне заняття. Втрата актуальності не означає, що доведеться видаляти розклад кожного семестру, оскільки система автоматично визначає семестр і очищає сторінку редагування, зберігаючи старий розклад для перегляду. Тобто можливість видалення передбачена для випадків, коли створений розклад визнається таким що не підходить і його необхідно переробити.

3.2.3 Валідація даних в розкладі

При спробі додати пару з викладачем, який вже має заняття в той самий час і в той самий день, система виведе попередження, яке повідомлятиме про те, що це заняття додати не можна і вкаже, яке саме заняття не дозволяє це зробити з вказанням викладача, курсу, предмету, на якому факультеті і спеціальності. З чисельником та знаменником все працює як має бути. Для прикладу, якщо додається заняття по **чисельнику** з викладачем Х, то в цей самий день на цей же ж номер пари можна додати заняття, яке буде містити викладача Х лише для

пари по **знаменнику**, інакше виведеться повідомлення про конфлікт. (див. Додаток С).

При спробі додати пару з аудиторією, яка вже має заняття в той самий час і в той самий день, система виведе попередження, яке повідомлятиме про те, що це заняття додати не можна і вкаже, яке саме заняття не дозволяє це зробити з вказанням аудиторії, викладача, курсу, предмету, на якому факультеті і спеціальності.(див. Додаток Т).

Якщо потрібно додати пару, незважаючи на можливий конфлікт, є можливість вимкнути перевірку на наявність конфліктів з іншими парами.

При виникненні конфлікту користувачеві не доведеться заново заповнювати деякі поля сторінки створення пари, оскільки в такому випадку поля пари відновлюються при наступній спробі створення пари.

3.2.4 Використання машинного навчання

При додаванні нової пари в розклад на сторінці редагування є підказки, які допомагають користувачеві обрати можливий предмет, викладача, аудиторію та тип заняття (лекція/практичне заняття/лабораторне заняття) для обраної пари. (див. Додаток У).

Предмет передбачається моделлю на основі таких вхідних параметрів (ознак):

- день тижня
- номер пари
- спеціальність

Викладач, окрім вище зазначених ознак, використовує як вхід також раніше отриманий предмет.

Аудиторія передбачається моделлю вже з використанням обох результатів минулих прогнозувань.

Така архітектура була обрана задля посилення зв’язків між даними і, як наслідок, отримання більш правдоподібного, реалістичного результату.

Також система прогнозування враховує навчальний план. Він створюється для кожного курсу та спеціальності та містить заняття, які повинні міститись в розкладі. Для кожного заняття вказується предмет, тип, та кількість занять на тиждень. В підказках до предмету відображаються лише предмети, які не досягли норми по кількості занять на тиждень, визначеній у навчальному плані. При цьому, якщо норму досягнуто, то буде запропоновано предмет, ймовірність появи якого найбільша після предмету, який було відхилено, і при цьому який відповідає умові, як, знову ж таки, норма за навчальним планом.

Також буде відображатись підказка для типу заняття на основі наявного навчального плану, яка не використовує машинне навчання.

3.2.5 Реалізація машинного навчання

Специфіка нашої системи полягає у вирішенні задачі мультикласової класифікації[15].

Для кожного курсу та факультету створюються свої моделі. Побудова моделі починається з завантаження даних в спеціальний об'єкт IDataView[17]. В нашому випадку це дані з розкладів занять минулих років. Для цього ми створили клас LessonForLearning з наступними полями: id спеціальності, день тижня, id номеру пари, id викладача, id аудиторії та id предмету. (див. Додаток Ф).

Оскільки ми передбачаємо три параметри, а саме: предмет, викладач і аудиторія, то ми створюємо три моделі. Далі для кожної моделі потрібно створити конвеєр навчання (англ. pipeline). В ньому вказується, які параметри класу LessonForLearning будуть ознаками, а які мітками. Також потрібно вказати алгоритм тренування для кожної моделі.

- Для першої моделі ознаками будуть id спеціальності, день тижня та номер пари, а міткою - id предмету.
- Для другої моделі ознаками будуть id спеціальності, день тижня, номер пари, а також id предмету. Міткою буде id викладача.

- Для третьої моделі ознаками будуть id спеціальності, день тижня, номер пари, id предмету та id викладача. Міткою буде id аудиторії.

В ML.NET є кілька алгоритмів тренування для мультикласової класифікації. Ми випробували всі доступні алгоритми і вибрали OneVersusAll, тому що він показав найкращий результат.

Після цього відбувається навчання моделі і її збереження в zip-файл для подальшого використання. Приклад коду (див. Додаток Х).

Коли відкривається сторінка для створення заняття, то завантажуються моделі з zip-файлів. Далі створюється спеціальний ML.NET об'єкт PredictionEngine і відбувається передбачення предмету. В результаті отримується список всіх предметів, які були розглянуті та ймовірності їх появи за спаданням. Береться предмет з найбільшою ймовірністю і перевіряється, чи можна його можна використати відповідно з навчальним планом, якщо не можна - то перевіряється наступний предмет. Якщо такий предмет було знайдено, то передбачається викладач з використанням раніше отриманого предмету. Далі передбачається аудиторія з використанням предмету та викладача.

Висновок

Внаслідок виконання роботи, наша команда отримала наступні результати:

Боліщук Святослав вніс значний внесок у розробку додатка, зосереджуючись на таких аспектах:

- Розробка архітектури бази даних, що дозволяє ефективно зберігати та організовувати розклади занять.
- Реалізація відображення розкладу, що надає користувачам зручний інтерфейс для перегляду їх розкладу.
- Розробка функціоналу взаємодії з розкладами, зокрема додавання, видалення та валідація занять, що дозволяє користувачам легко редагувати свої розклади.
- Розробка системи для генерації та відображення підказок з використанням машинного навчання

Оскірко Максим також вніс вагомий внесок у розробку додатка, працюючи над такими завданнями:

- Підготовка початкових історичних даних та навчання моделі машинного навчання на їх основі. Це дало змогу використовувати машинне навчання для надання користувачам підказок та рекомендацій.
- Реалізація CRUD-операцій з моделями даних та створення інтерфейсу для їх взаємодії. Це дозволяє ефективно керувати даними розкладу.
- Розробка функціоналу імпорту/експорту розкладів з використанням JSON-файлів, що дозволяє зручно обмінюватися розкладами з іншими системами.
- Концептуалізація алгоритму машинного навчання для зв'язності даних у підказках, що сприяє поліпшенню якості та коректності наданих рекомендацій.

За результатами нашої роботи, можна зробити наступні висновки:

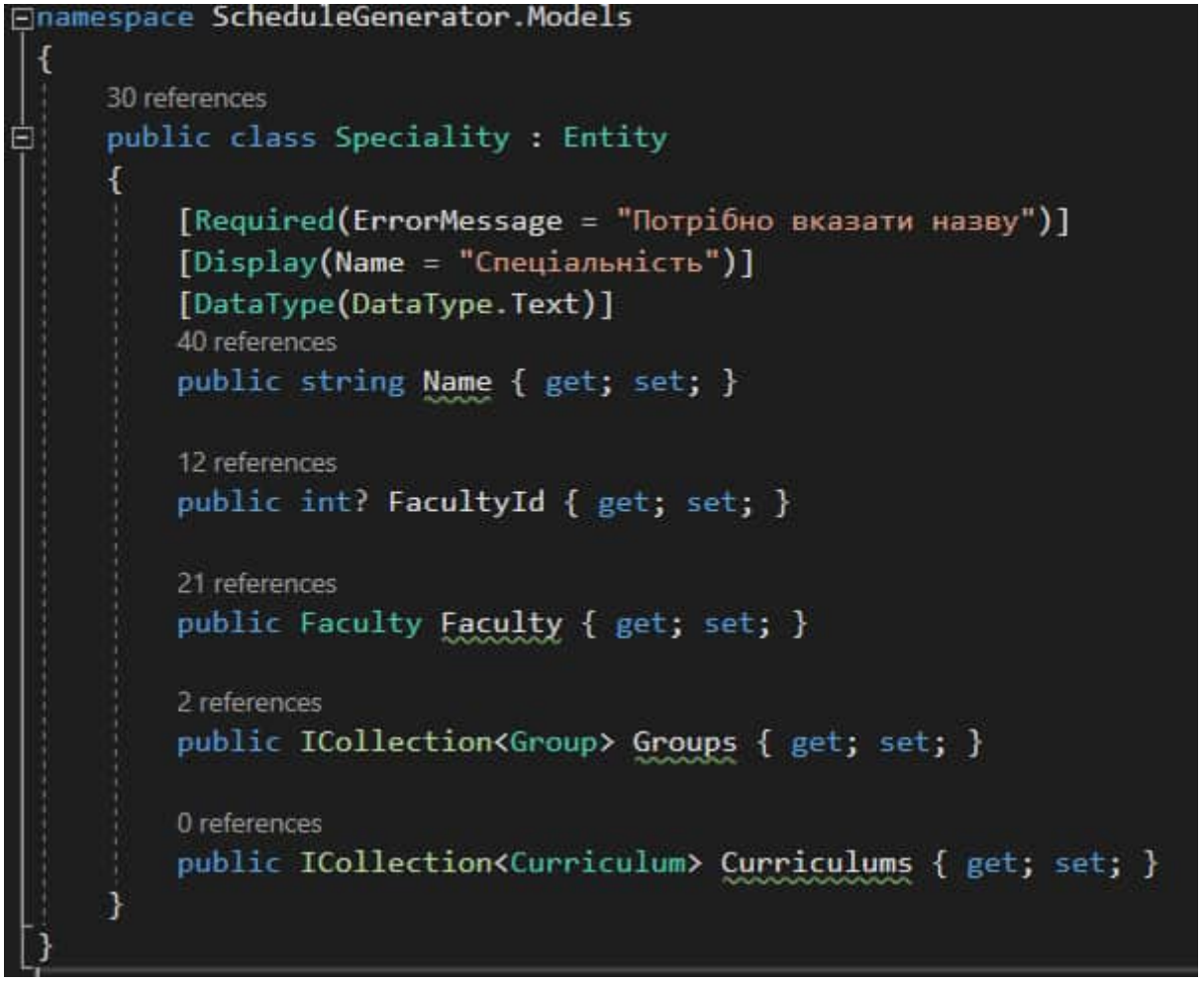
- Розроблений веб-додаток на базі .NET, ASP.NET Core та ML.NET забезпечує ефективне створення, редагування та відображення розкладів занять університету.
- Використання машинного навчання дозволяє надати користувачам цінні підказки та рекомендації, що поліпшують процес планування розкладів та зменшують ймовірність помилок.
- Функціонал імпорту/експорту розкладів дозволяє зручно обмінюватися даними з іншими системами та забезпечує легкість у використанні.

Загалом, робота над проектом дала значні результати і покращила процес створення розкладів занять в університеті. Додаток на базі .NET, ASP.NET Core та ML.NET є потужним інструментом для управління розкладами та надання підтримки користувачам у плануванні їх академічних занять.

Додатки

Додаток А

Вигляд моделі Спеціальність



```
namespace ScheduleGenerator.Models
{
    30 references
    public class Speciality : Entity
    {
        [Required(ErrorMessage = "Потрібно вказати назву")]
        [Display(Name = "Спеціальність")]
        [DataType(DataType.Text)]
        40 references
        public string Name { get; set; }

        12 references
        public int? FacultyId { get; set; }

        21 references
        public Faculty Faculty { get; set; }

        2 references
        public ICollection<Group> Groups { get; set; }

        0 references
        public ICollection<Curriculum> Curriculums { get; set; }
    }
}
```

Додаток Б

Вигляд класу контексту

```
27 references
public class ApplicationDbContext : IdentityDbContext
{
    0 references
    public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options)
        : base(options)
    {
        Database.EnsureCreated();
    }

    4 references
    public DbSet<Group> Groups { get; set; }

    6 references
    public DbSet<Speciality> Specialities { get; set; }

    7 references
    public DbSet<Faculty> Faculties { get; set; }

    4 references
    public DbSet<Lecturer> Lecturers { get; set; }

    3 references
    public DbSet<Lesson> Lessons { get; set; }

    4 references
    public DbSet<Subject> Subjects { get; set; }

    4 references
    public DbSet<Audience> Audiences { get; set; }

    4 references
    public DbSet<LessonNumber> LessonNumbers { get; set; }

    5 references
    public DbSet<CourseNumber> CourseNumbers { get; set; }
```

Додаток В

Початкове заповнення бази даних в контексті

```
modelBuilder.Entity<Speciality>().HasData(
new Speciality[]
    {
        new Speciality
        {
            Id = 1,
            Name = "Комп'ютерні науки",
            FacultyId = 1,
        },
        new Speciality
        {
            Id = 2,
            Name = "Прикладна математика",
            FacultyId = 1,
        },
        new Speciality
        {
            Id = 3,
            Name = "Середня освіта",
            FacultyId = 1,
        },
```

Додаток Г

Вигляд методу Configure в контролері SpecialityController

```
3 references
public async Task<IActionResult> Configure()
{
    var specialities = await _context.Set<Speciality>().Include(sp => sp.Faculty).ToListAsync();

    return View(specialities);
}
```

Додаток Г

Вигляд відповідного представлення *Configure.cshtml*

```
@model IEnumerable<Speciality>

@{
    ViewData["Title"] = "Спеціальності";
}

<div class="card shadow">
    <div class="mt-4">
        <div class="card-body">
            <div class="table-responsive">
                <table class="table" id="Datatable" width="100%" cellspacing="0">
                    <thead>
                        <tr>
                            <th>
                                @Html.DisplayNameFor(model => model.Name)
                            </th>
                            <th>
                                @Html.DisplayNameFor(model => model.Faculty.Name)
                            </th>
                            <th class="dontNeedFilter">
                                Дії
                            </th>
                        </tr>
                    </thead>
                    <tbody>
                        @foreach (var item in Model)
                        {
                            <tr>
                                <td>
                                    @Html.DisplayFor(modelItem => item.Name)
                                </td>
                                <td>
                                    @Html.DisplayFor(modelItem => item.Faculty.Name)
                                </td>
                                <td>
                                    <div class="btn-group-vertical w-100">
                                        <a class="btn btn-warning" asp-controller="Speciality" asp-action="Edit" asp-route-id="@item.Id">Редагувати</a>
                                        <a class="btn btn-danger" asp-controller="Speciality" asp-action="Delete" asp-route-id="@item.Id">Видалити</a>
                                    </div>
                                </td>
                            </tr>
                        }
                    </tbody>
                </table>
            </div>
        </div>
    </div>
</div>
```

Додаток Д

Приклад використання *ViewModel* для перетворення даних з контексту і передачі у представлення

```
0 references
public async Task<IActionResult> Create()
{
    var specialities = await _context.Set<Speciality>().ToListAsync();
    var courseNumbers = await _context.Set<CourseNumber>().ToListAsync();
    var groupVM = new GroupViewModel
    {
        Specialities = specialities.Select(speciality => new SelectListItem
            { Value = speciality.Id.ToString(), Text = speciality.Name.ToString() }).ToList(),
        CourseNumbers = courseNumbers.Select(courseNumber => new SelectListItem
            { Value = courseNumber.Id.ToString(), Text = courseNumber.Id.ToString()}).ToList(),
    };

    return View(groupVM);
}
```

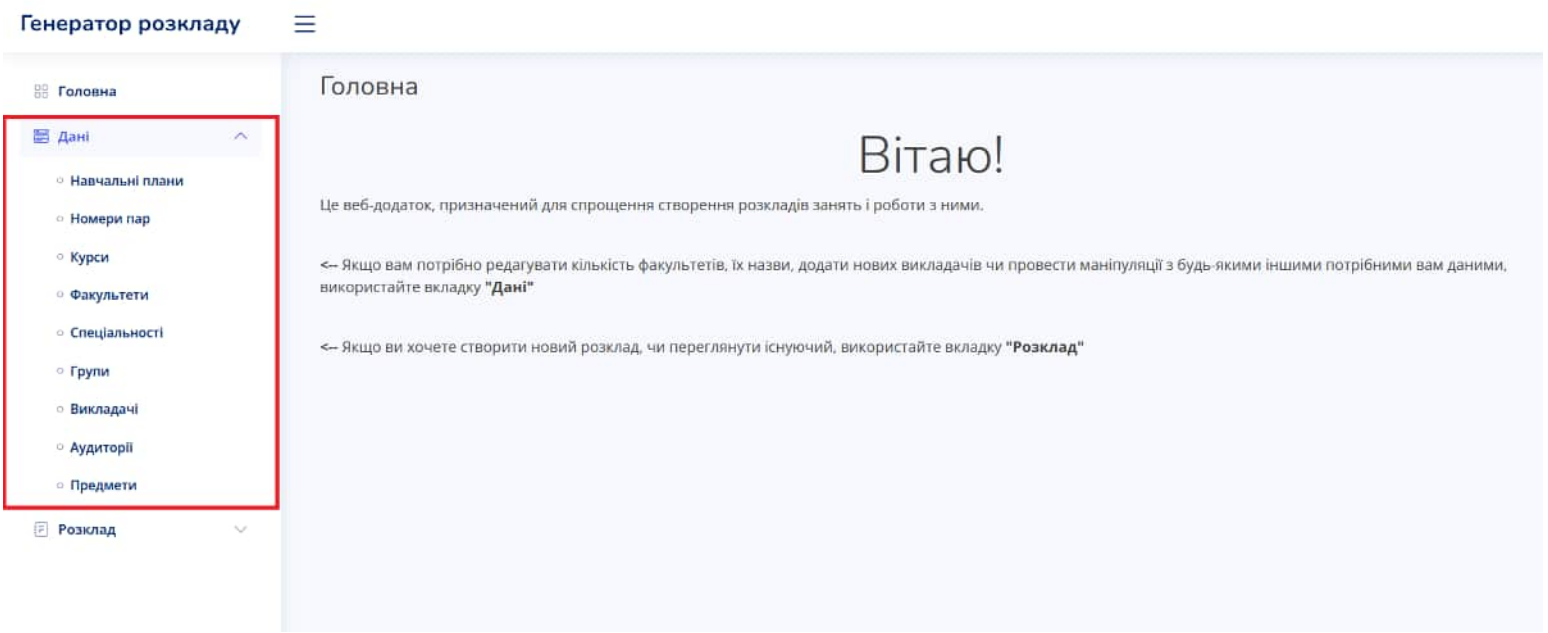
Додаток Е

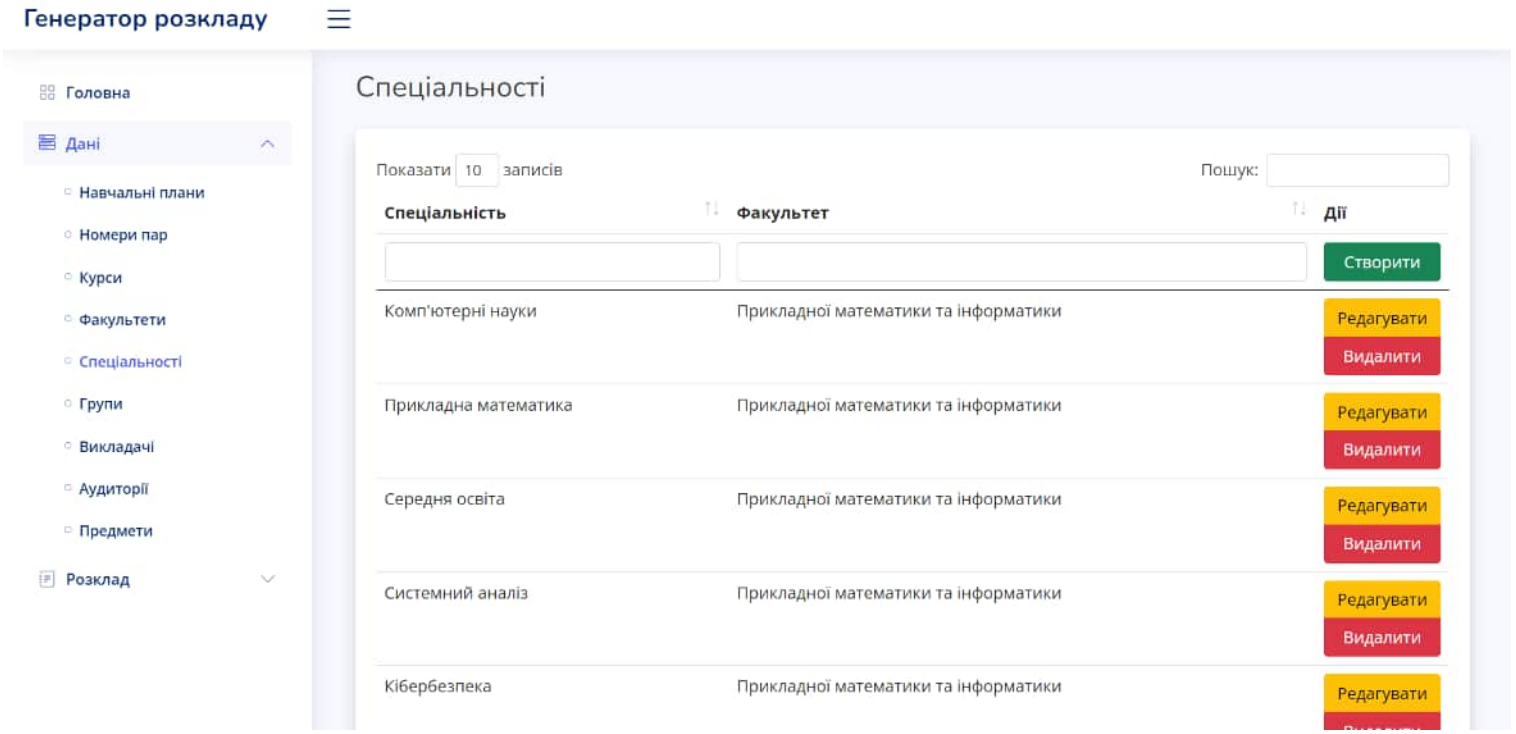
Відповідний *ViewModel*, що використовується

```
using Microsoft.AspNetCore.Mvc.Rendering;
using ScheduleGenerator.Models;
using System.ComponentModel.DataAnnotations;

namespace ScheduleGenerator.ViewModels
{
    14 references
    public class GroupViewModel
    {
        17 references
        public Group Group { get; set; }

        [Required(ErrorMessage = "Потрібно вказати назву")]
        [Display(Name = "Назва групи")]
        6 references
        public string GroupName { get; set; }

        [Display(Name = "Спеціальність")]
        9 references
        public List<SelectListItem> Specialities { get; set; }

        [Display(Name = "Курс")]
        6 references
        public int? CourseNumberId { get; set; }

        4 references
        public List<SelectListItem> CourseNumbers { get; set; }

        3 references
        public int? SpecialityId { get; set; }
    }
}
```

Додаток Є

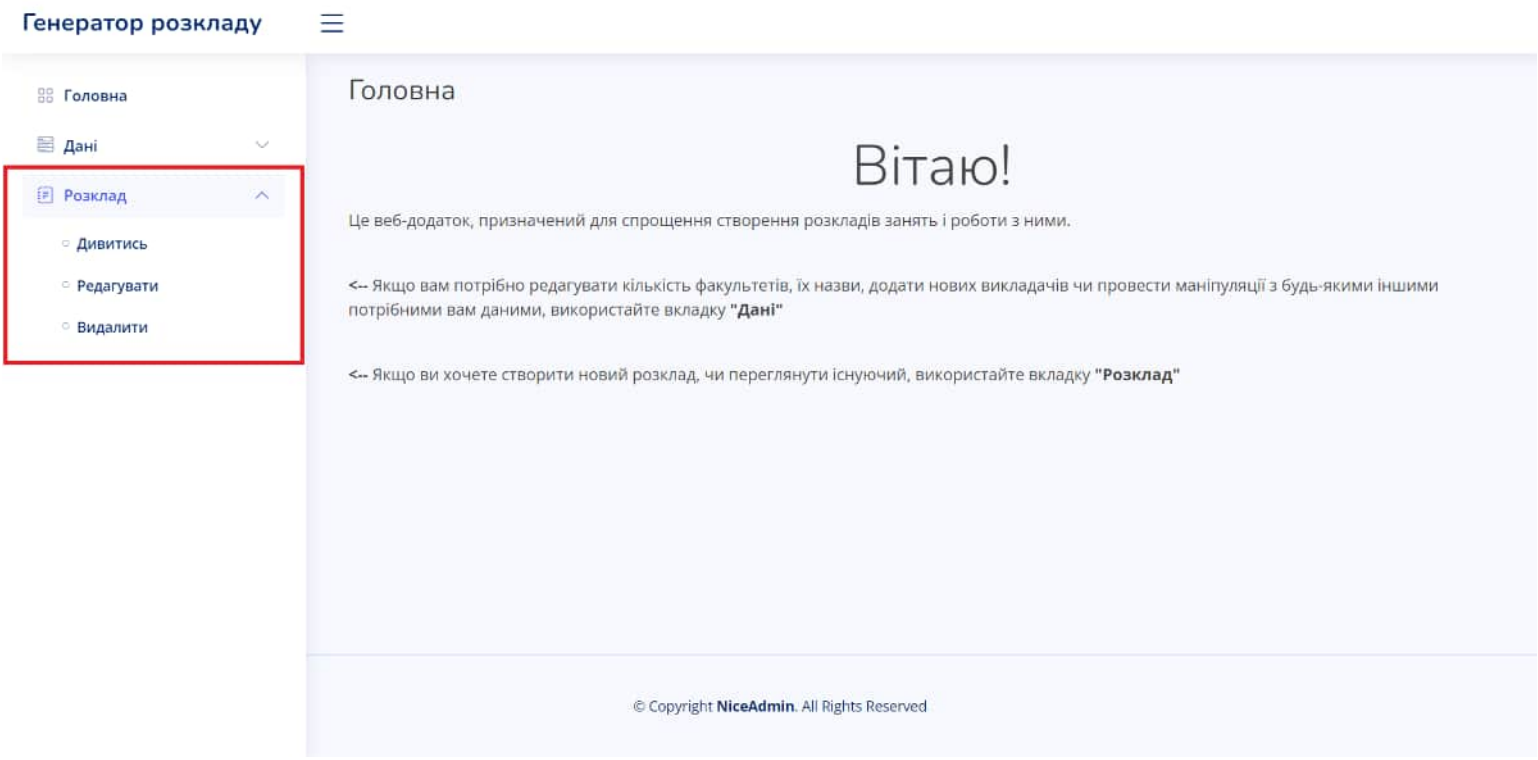
Опції взаємодії з даними (CRUD) (2 зображення)





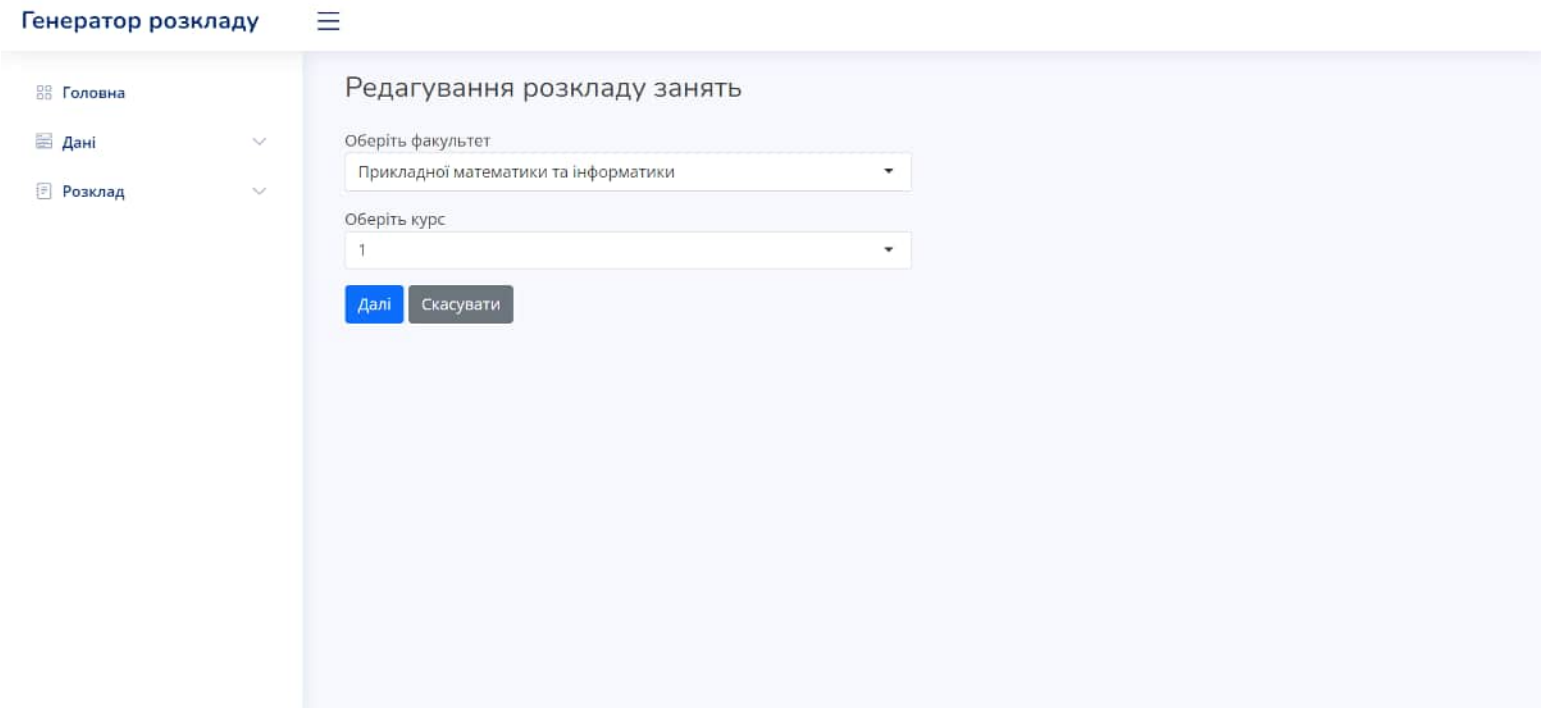
Додаток Ж

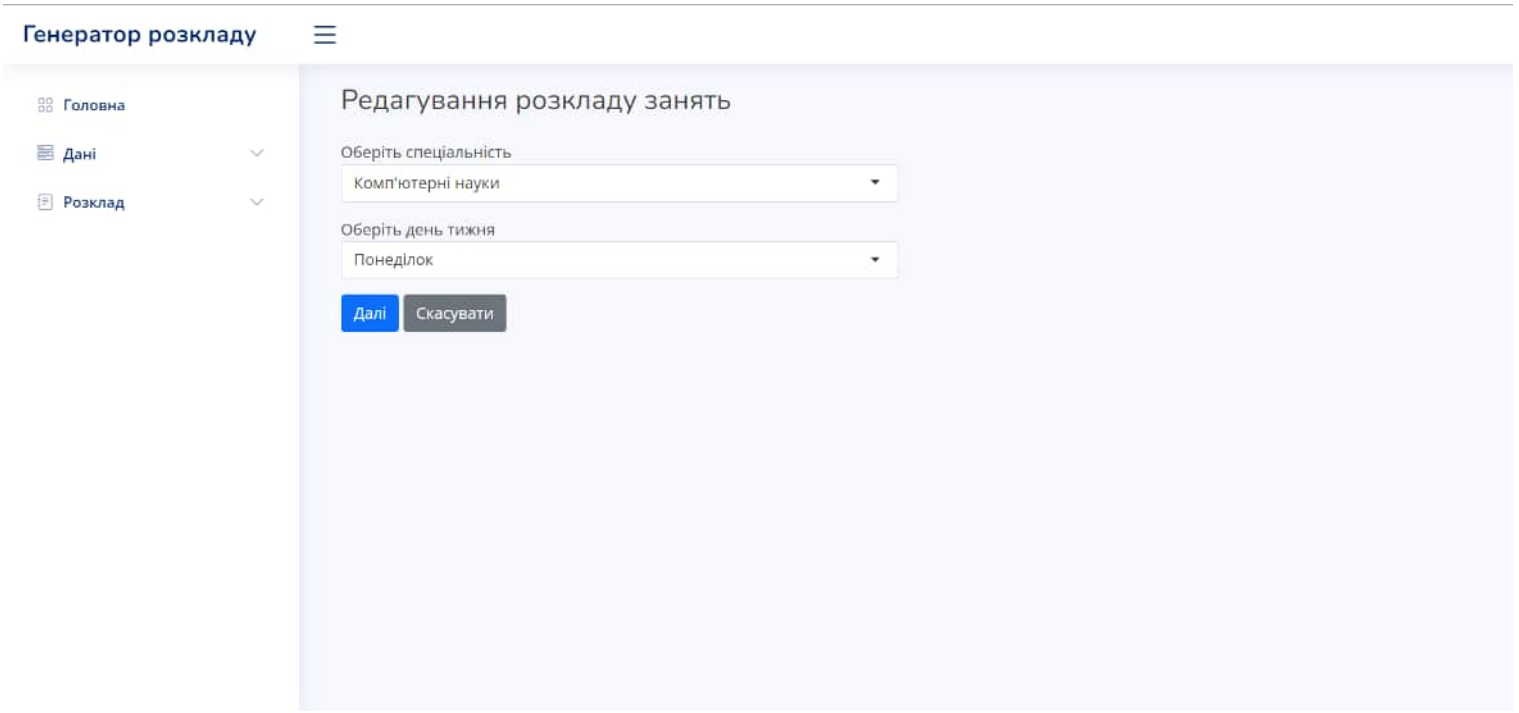
Розділи для взаємодії з розкладами

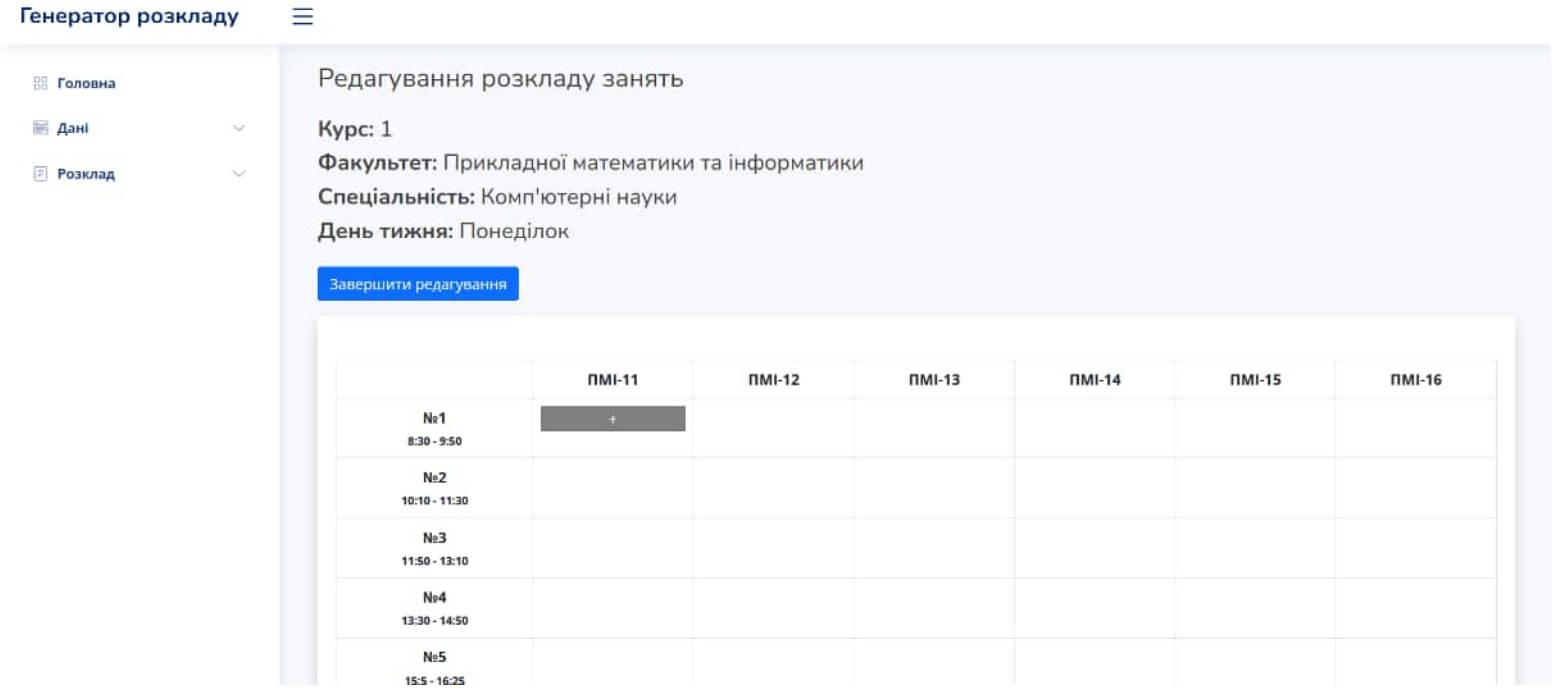


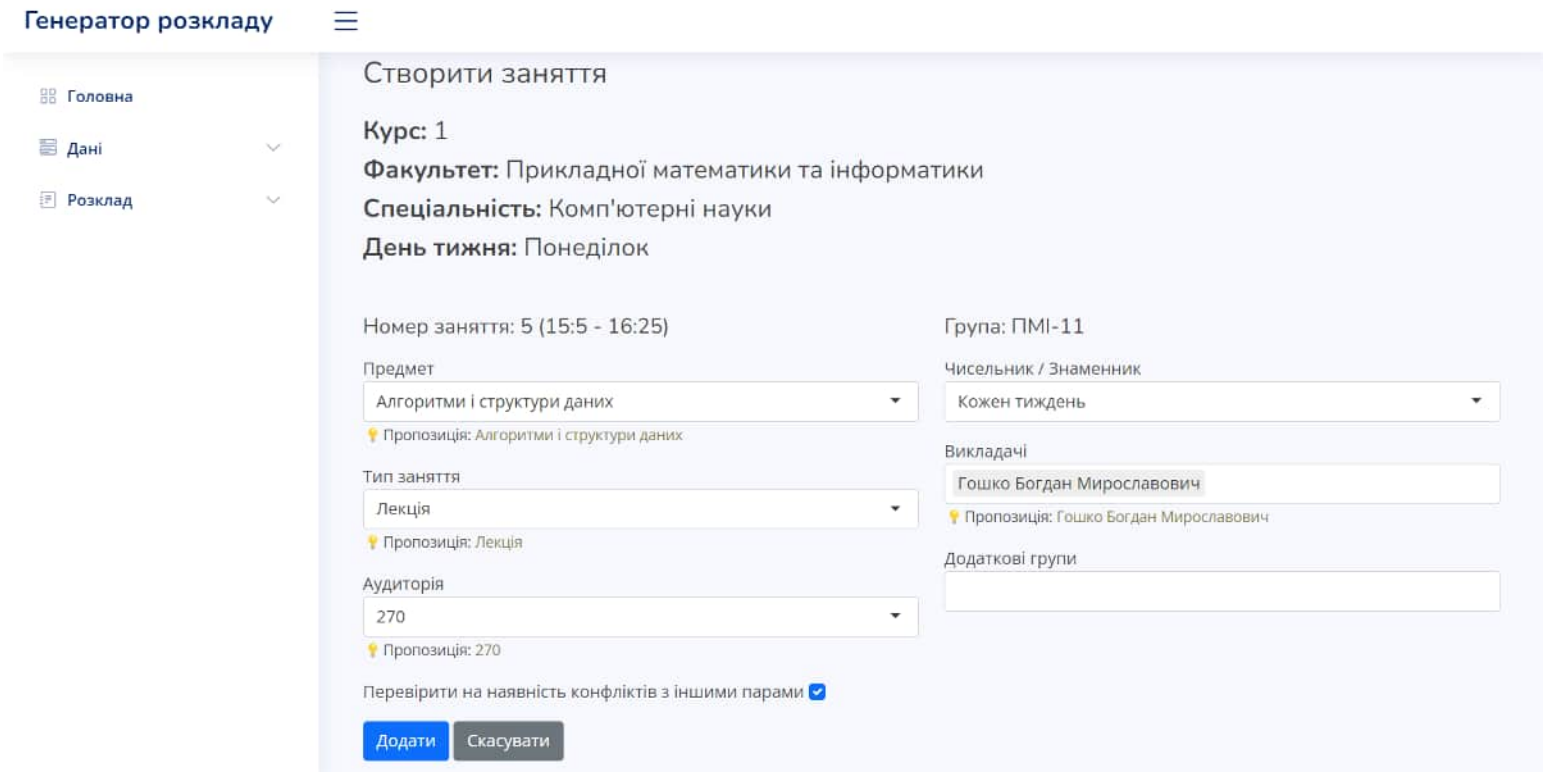
Додаток З

Редагування розкладу (4 зображення)



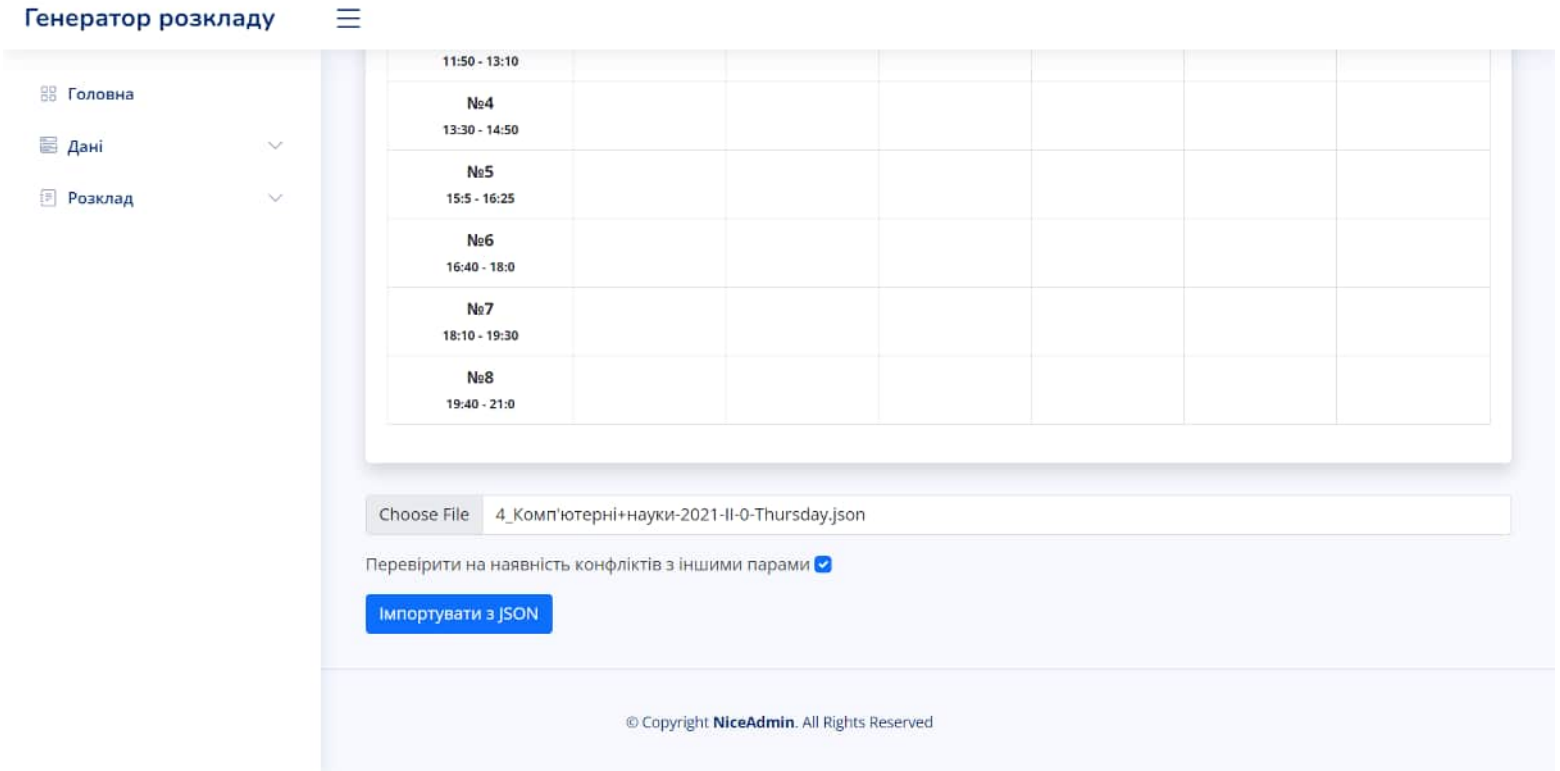

Генератор розкладу
Головна
Дані
Розклад
Редагування розкладу занять
Оберіть спеціальність
Комп'ютерні науки
Оберіть день тижня
Понеділок
Далі
Скасувати


Генератор розкладу
Головна
Дані
Розклад
Редагування розкладу занять
Курс: 1
Факультет: Прикладної математики та інформатики
Спеціальність: Комп'ютерні науки
День тижня: Понеділок
Завершити редагування
ПМІ-11
ПМІ-12
ПМІ-13
ПМІ-14
ПМІ-15
ПМІ-16
№1
8:30 - 9:50
№2
10:10 - 11:30
№3
11:50 - 13:10
№4
13:30 - 14:50
№5
15:5 - 16:25



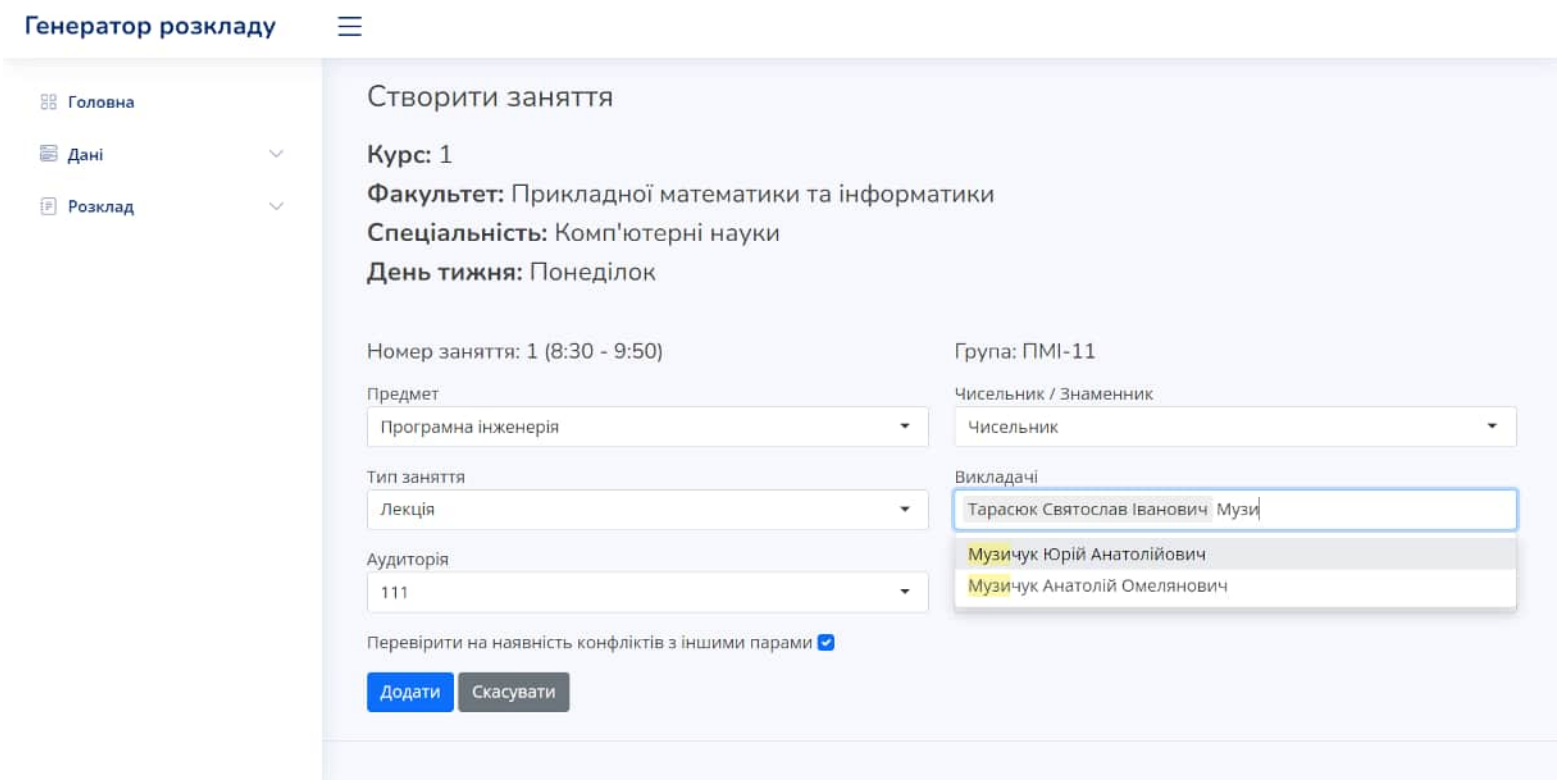
Додаток И

Імпортування розкладу



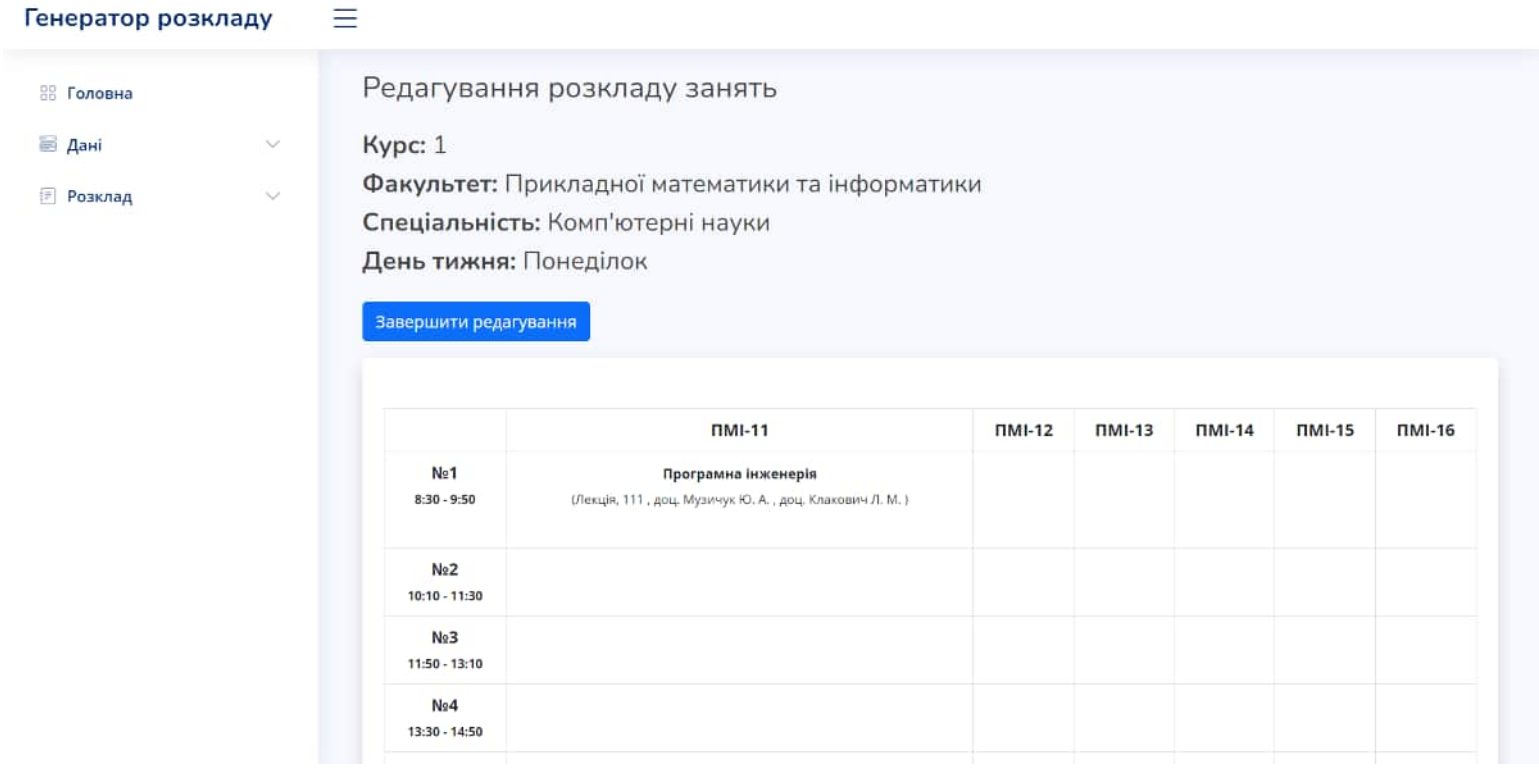
Додаток І

Випадаюче меню з мульти-вибором (selectize.js)



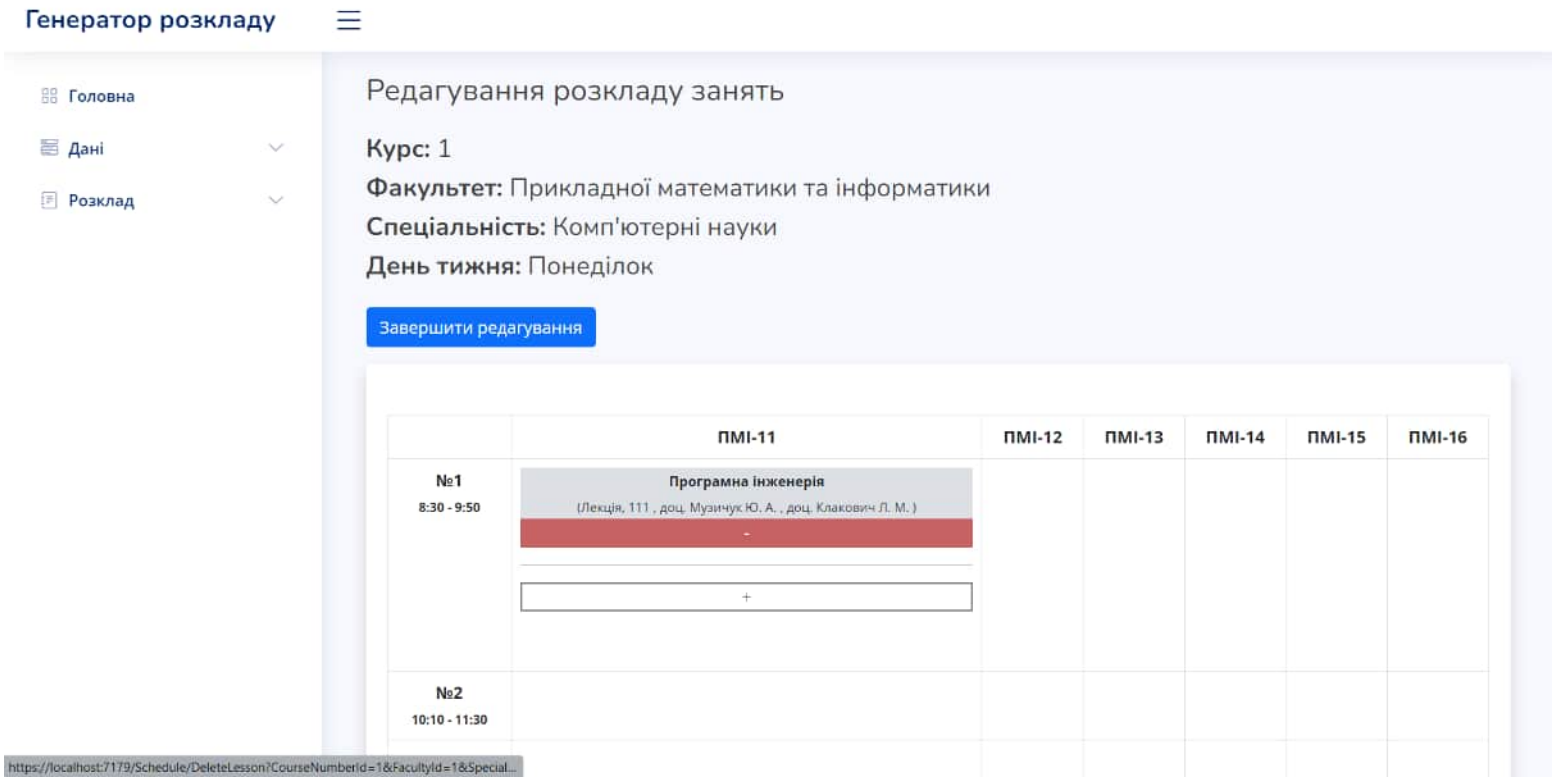
Додаток Ї

Заняття кожен тиждень



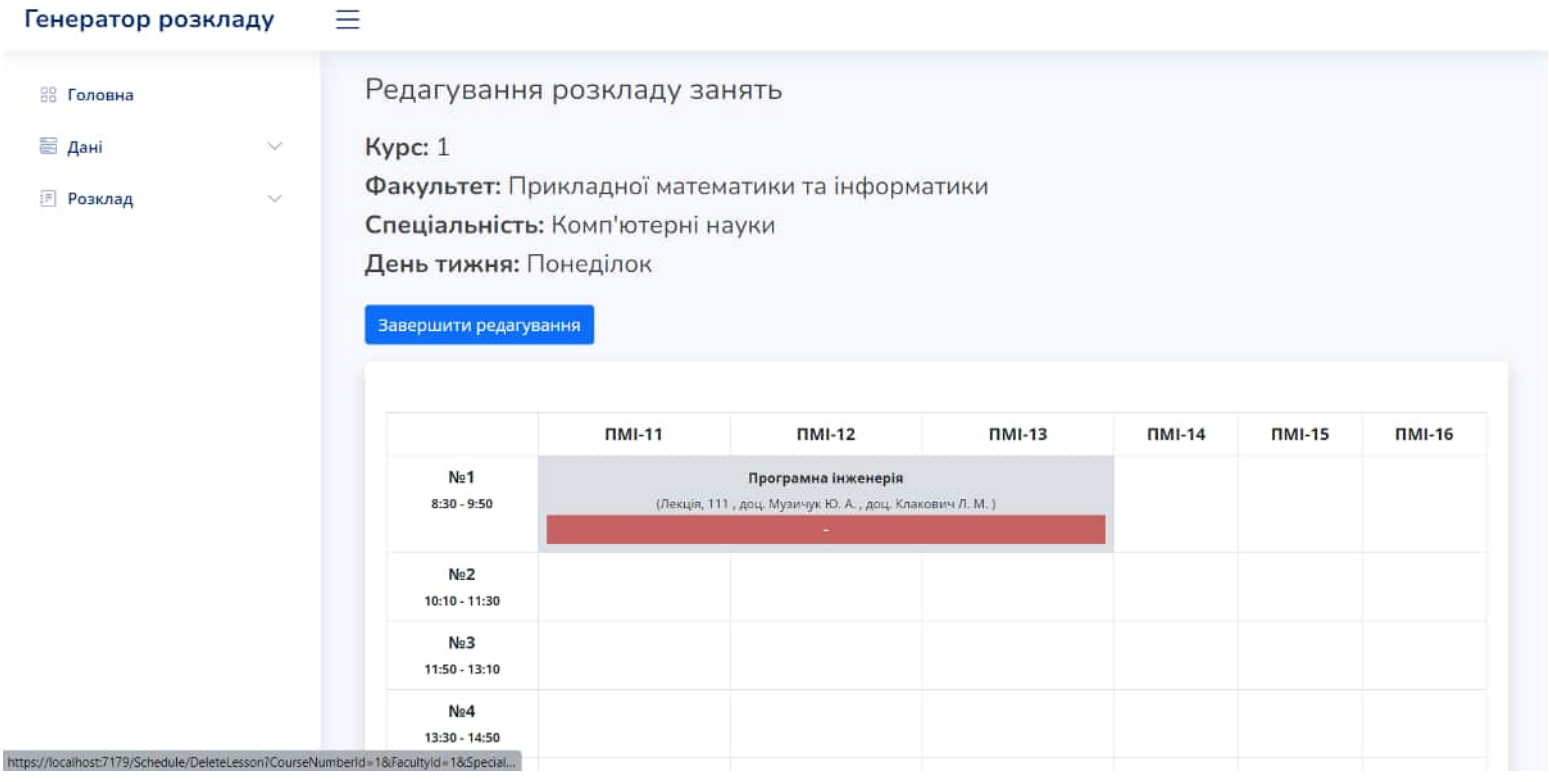
Додаток Й

Заняття по чисельнику



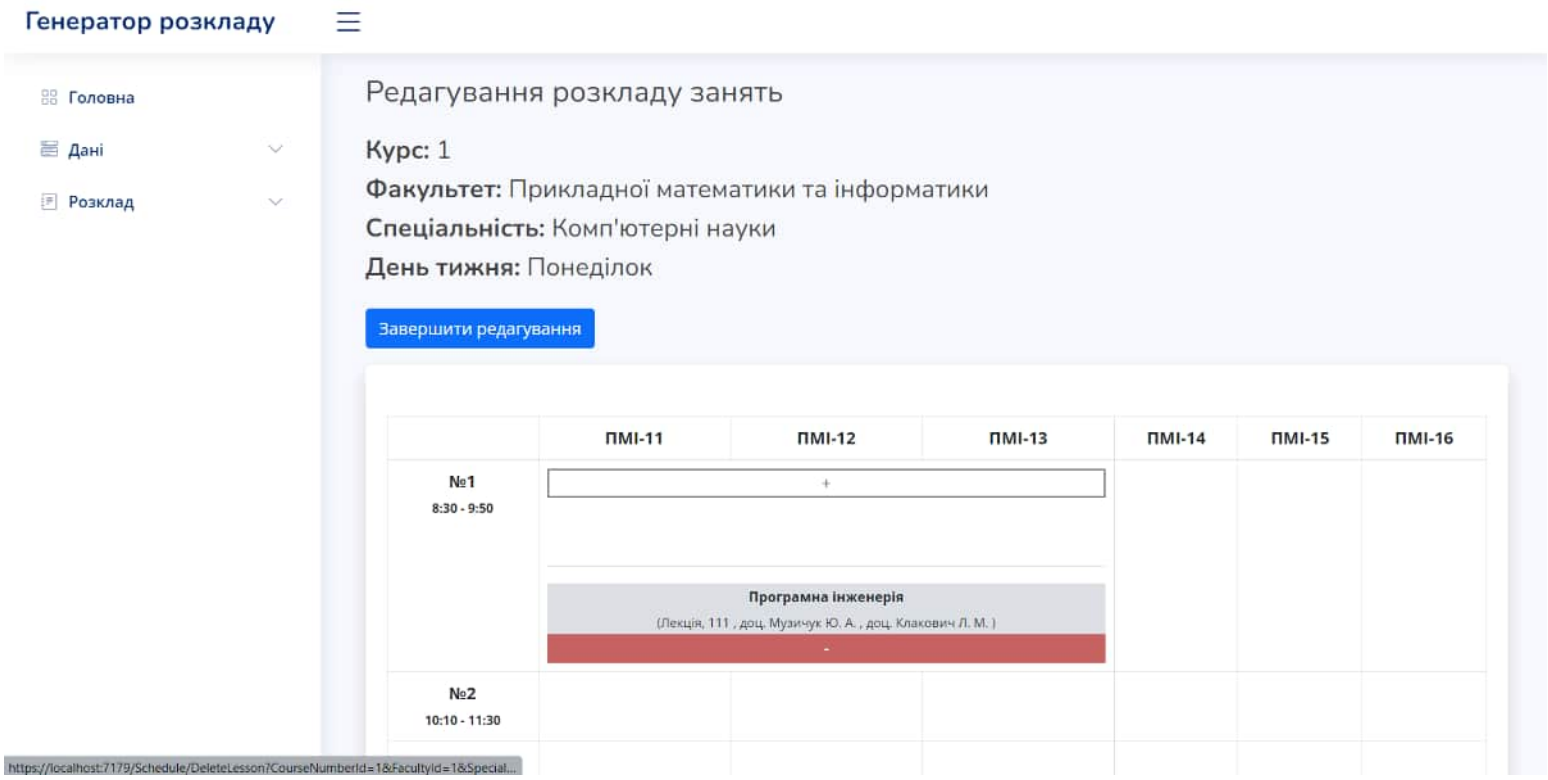
Додаток К

Заняття на кілька груп



Додаток Л

Заняття на кілька груп, знаменник



Додаток М

Об’єднання клітинок C#

```
else
{
    @if (lessonPrevious != lesson)
    {
        var classId = $"l{lesson.Id}";
        <td class="highlight @classId" onmouseover="HighlightCommon(this, true)" onmouseleave="HighlightCommon(this, false)">
            <span class="textMedium"><b>@lesson.Subject.Name</b></span><br>
            <span class="textSmall">
                (@lesson.LessonType.GetDisplayName(), @lesson.Audience.Name
                @foreach (var lecturer in lesson.Lecturers)
                {
                    if (lecturer.Position == Position.Assistant)
                    {
                        <span>, ас. @lecturer.LastName @lecturer.FirstName[0]. @lecturer.MiddleName[0].</span>
                    }
                    else if (lecturer.Position == Position.Docent)
                    {
                        <span>, доц. @lecturer.LastName @lecturer.FirstName[0]. @lecturer.MiddleName[0].</span>
                    }
                    else if (lecturer.Position == Position.Professor)
                    {
                        <span>, проф. @lecturer.LastName @lecturer.FirstName[0]. @lecturer.MiddleName[0].</span>
                    }
                }
                )
            </span>
            <a asp-controller="Schedule" asp-action="DeleteLesson"
asp-route-CourseNumberId="@Model.CourseNumberId"
asp-route-FacultyId="@Model.FacultyId"
asp-route-SpecialityId="@Model.SpecialityId"
asp-route-DayOfWeek="@Model.DayOfWeek"
asp-route-LessonId="@lesson.Id">
                <div class="removeLessonButton">
                    <span class="noSelectText">-</span>
                </div>
            </a>
        </td>
    }
    else if (lessonPrevious == lesson)
    {
        <td class="cellToRemove">
            left lesson
        </td>
    }
}
lessonPrevious = lesson;
```

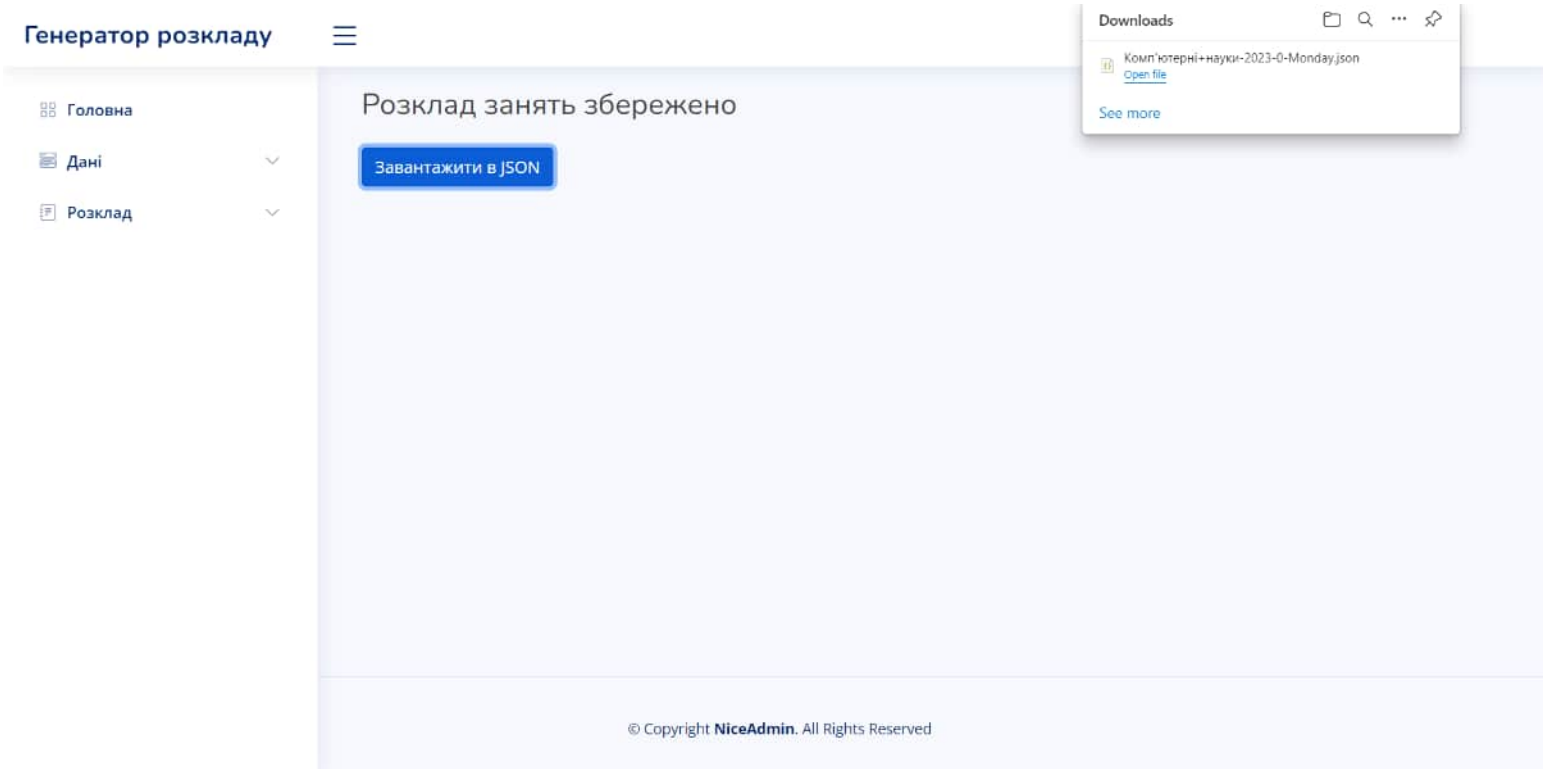
Додаток Н

Об’єднання клітинок Js

```
345
346     <script>
347         // combining cells
348             let cellsToRemove = document.querySelectorAll(".cellToRemove");
349             cellsToRemove.forEach(cell => {
350                 cell.previousElementSibling.colSpan += 1;
351                 cell.remove();
352             });
353     </script>
354
```

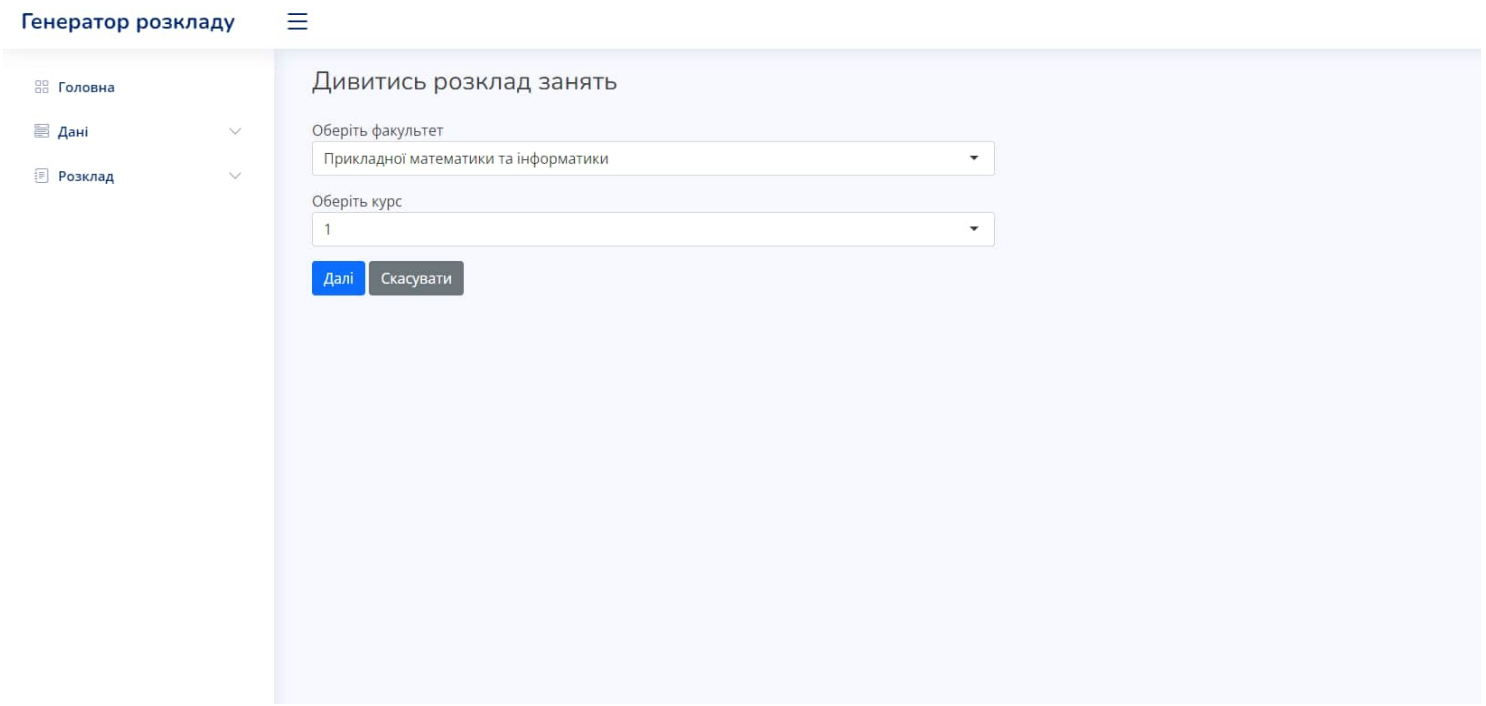
Додаток О

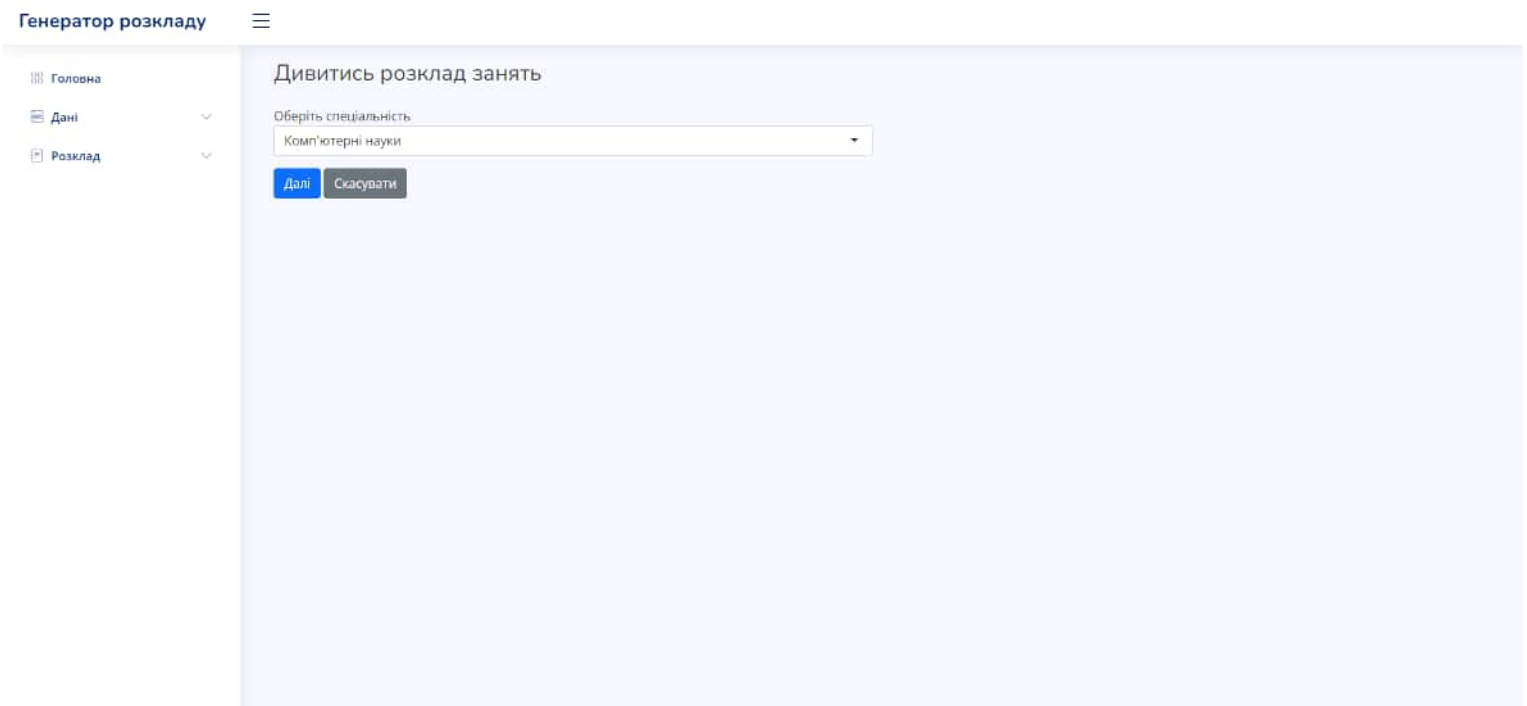
Завантаження розкладу

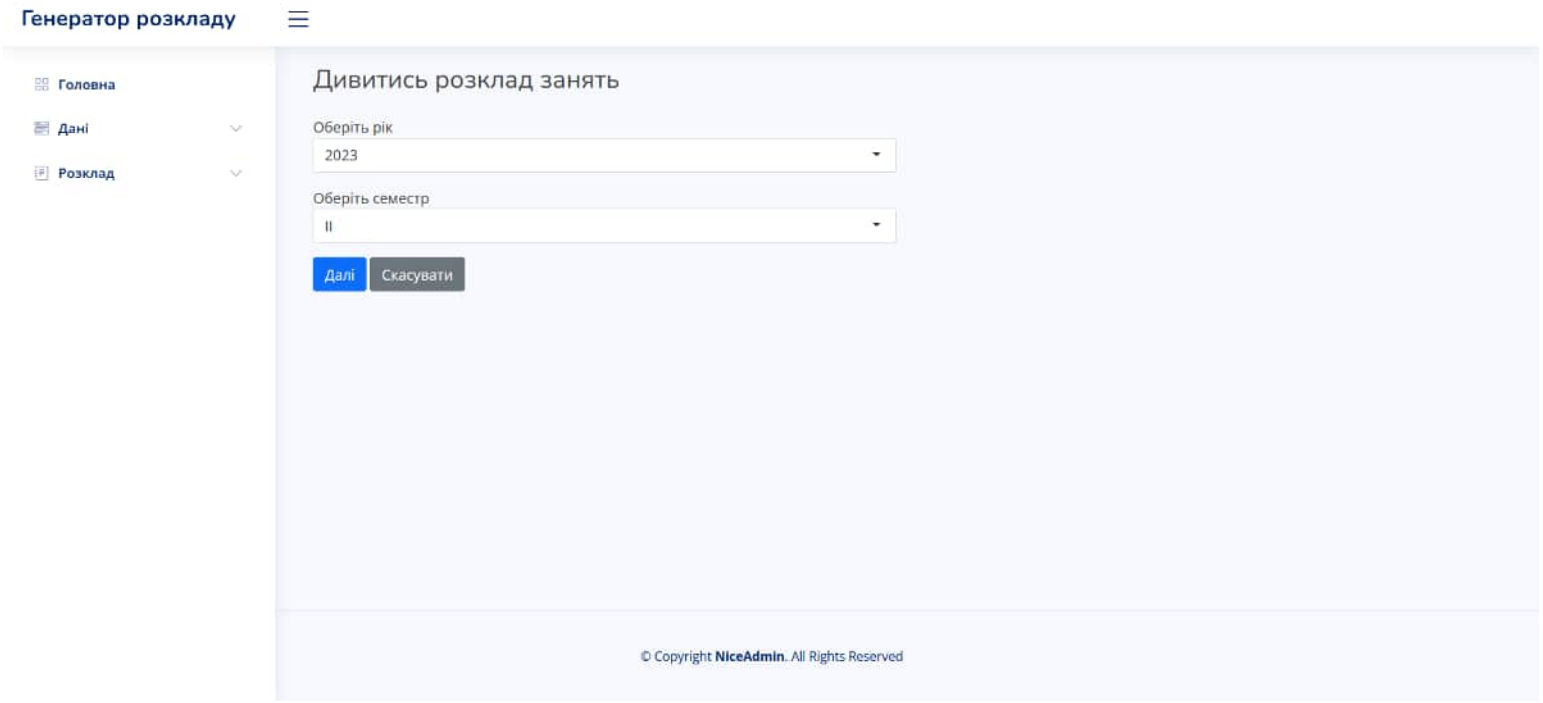


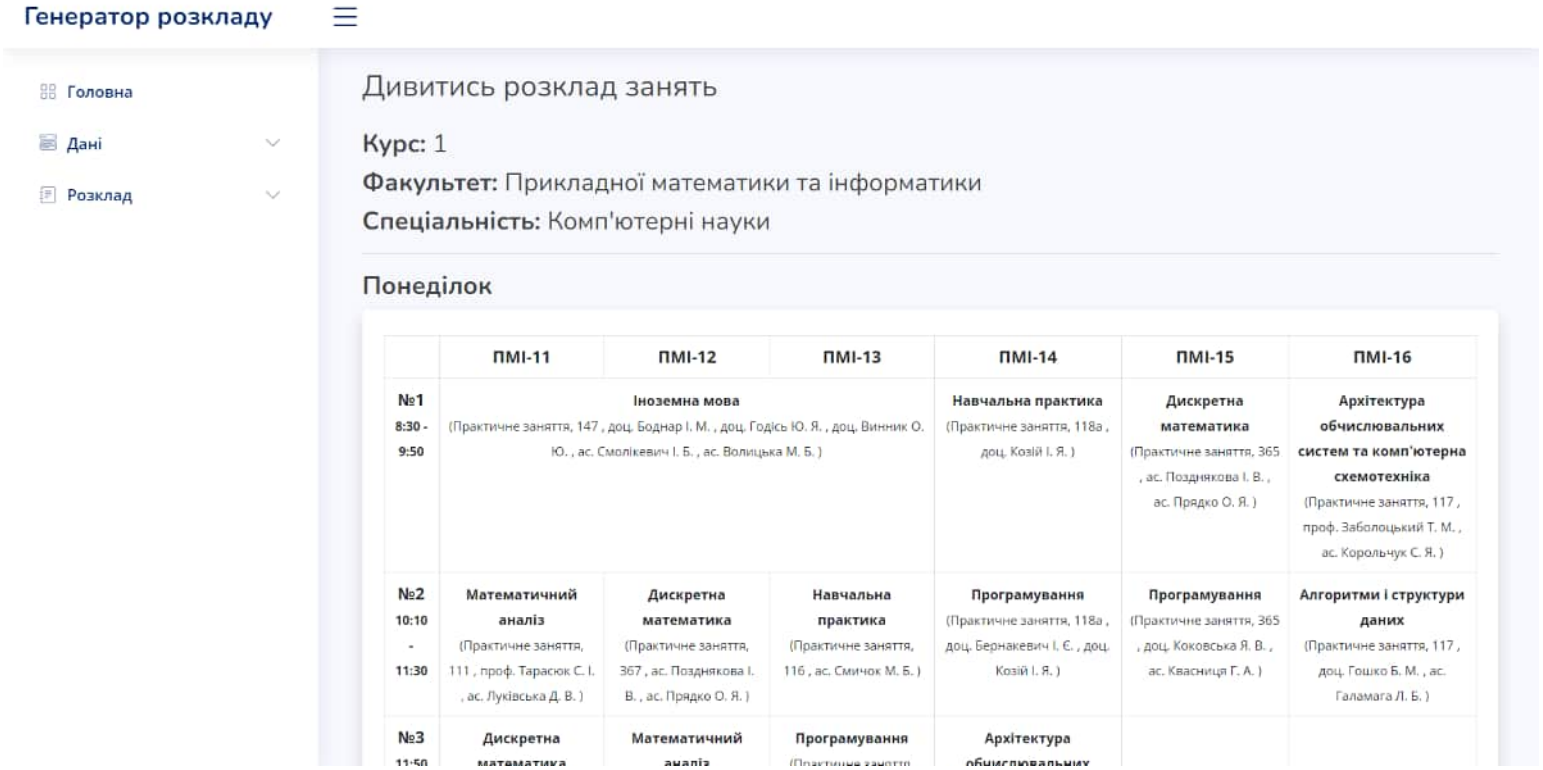
Додаток П

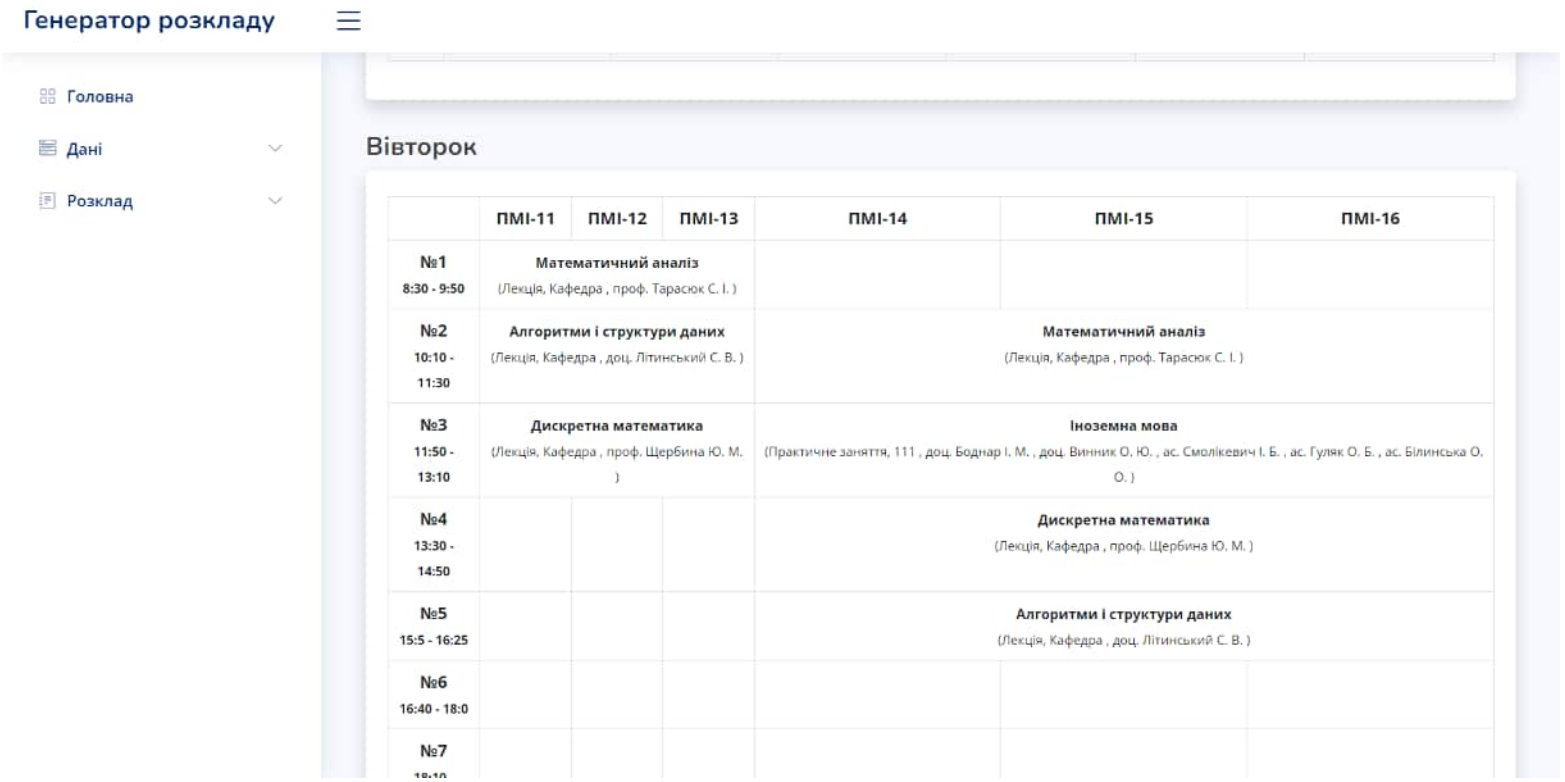
Перегляд розкладу(5 зображень)





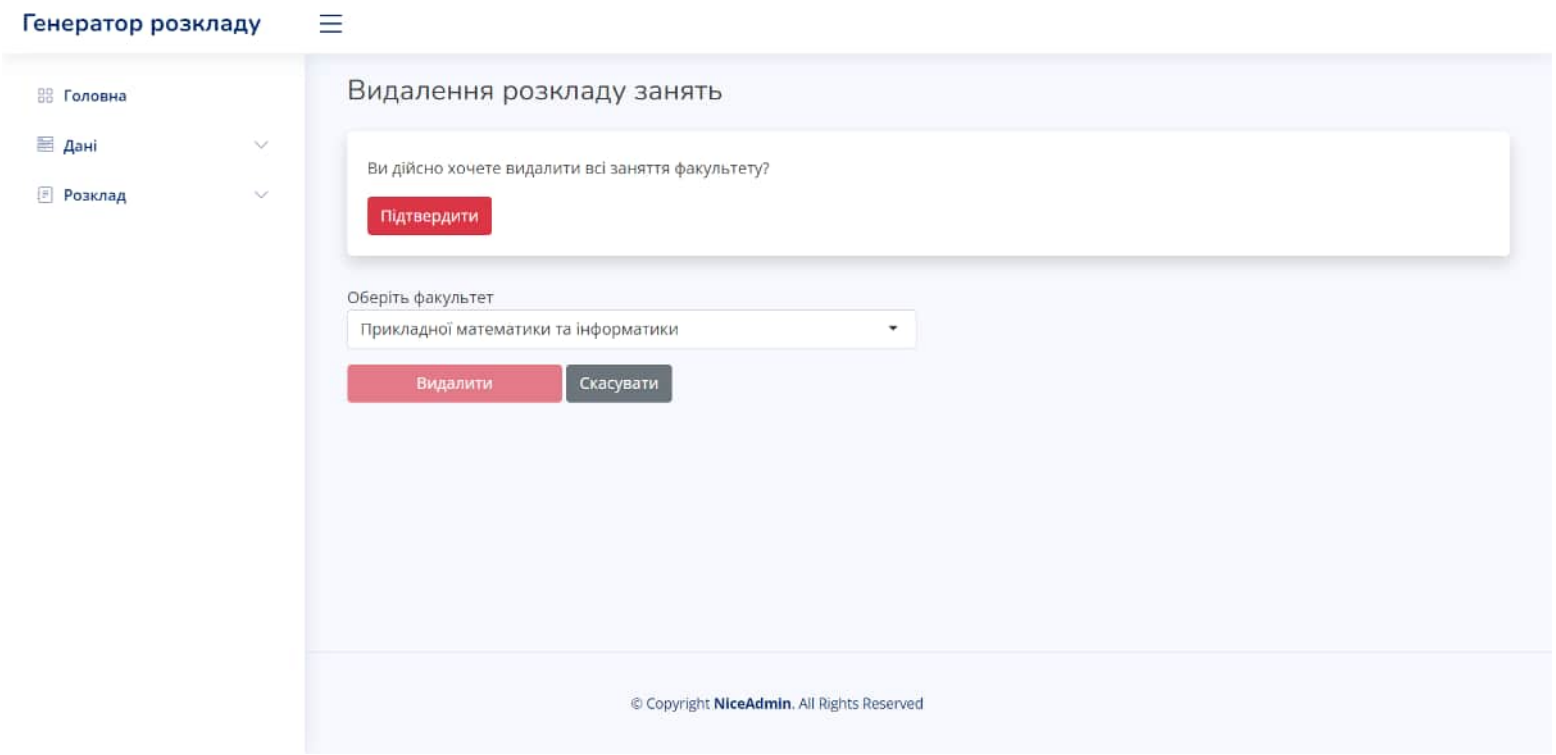

Генератор розкладу
Головна
Дані
Розклад
Дивитись розклад занять
Оберіть рік
2023
Оберіть семестр
II
Далі
Скасувати
© Copyright NiceAdmin. All Rights Reserved


Генератор розкладу
Головна
Дані
Розклад
Дивитись розклад занять
Курс: 1
Факультет: Прикладної математики та інформатики
Спеціальність: Комп'ютерні науки
Понеділок
ПМІ-11
ПМІ-12
ПМІ-13
ПМІ-14
ПМІ-15
ПМІ-16
№1 8:30 - 9:50
Іноземна мова (Практичне заняття, 147 , доц. Боднар І. М. , доц. Годісь Ю. Я. , доц. Винник О. Ю. , ас. Смолікевич І. Б. , ас. Волицька М. Б.)
Навчальна практика (Практичне заняття, 118а , доц. Козій І. Я.)
Дискретна математика (Практичне заняття, 365 , ас. Позднякова І. В. , ас. Прядко О. Я.)
Архітектура обчислювальних систем та комп'ютерна схемотехніка (Практичне заняття, 117 , проф. Заболоцький Т. М. , ас. Корольчук С. Я.)
№2 10:10 - 11:30
Математичний аналіз (Практичне заняття, 111 , проф. Тарасюк С. І. , ас. Луківська Д. В.)
Дискретна математика (Практичне заняття, 367 , ас. Позднякова І. В. , ас. Прядко О. Я.)
Навчальна практика (Практичне заняття, 116 , ас. Смичок М. Б.)
Програмування (Практичне заняття, 118а , доц. Бернакевич І. Є. , доц. Козій І. Я.)
Програмування (Практичне заняття, 365 , доц. Коковська Я. В. , ас. Квасниця Г. А.)
Алгоритми і структури даних (Практичне заняття, 117 , доц. Гошко Б. М. , ас. Галамага Л. Б.)
№3 11:50
Дискретна математика
Математичний аналіз
Програмування
Архітектура обчислювальних



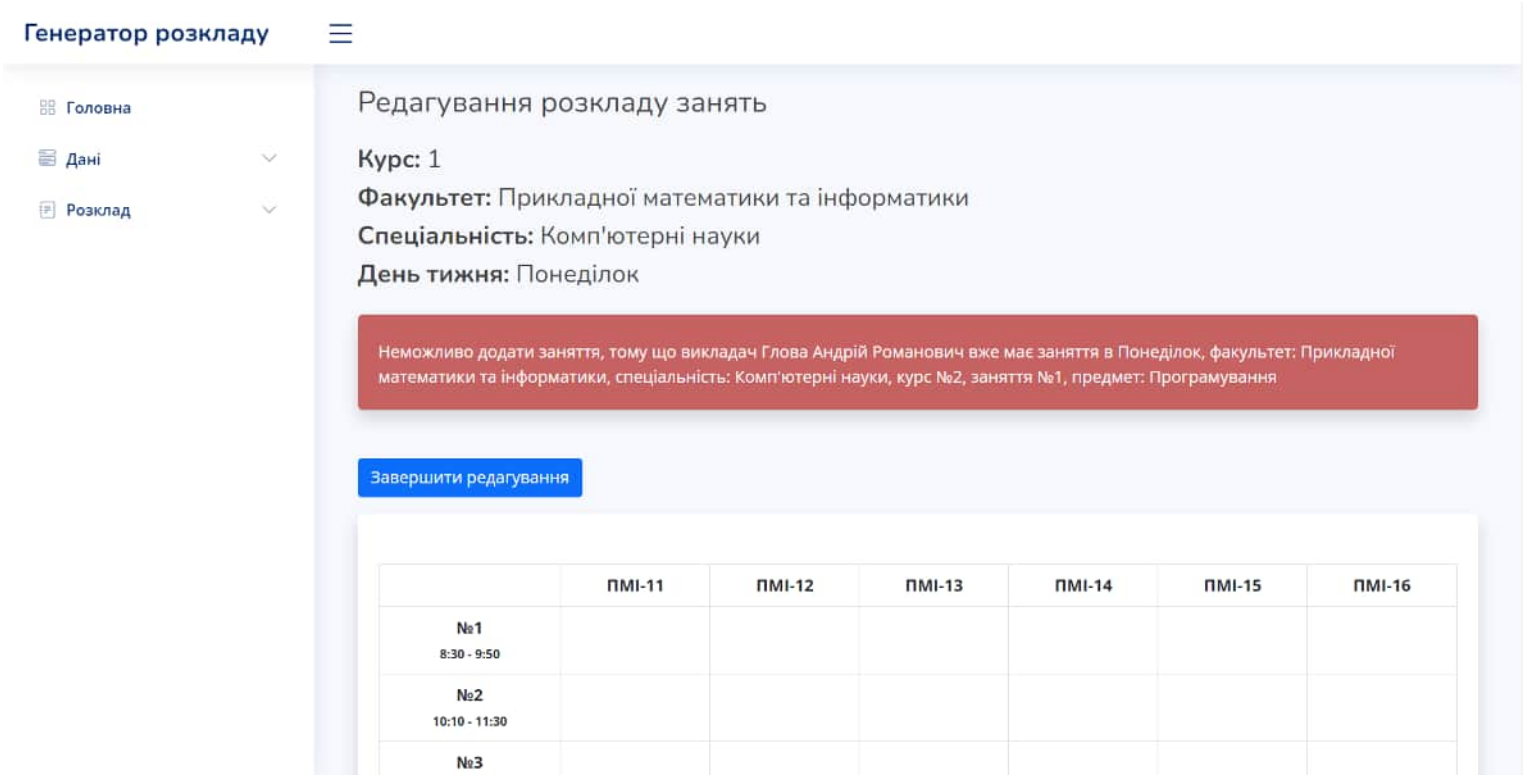
Додаток Р

Видалення розкладу



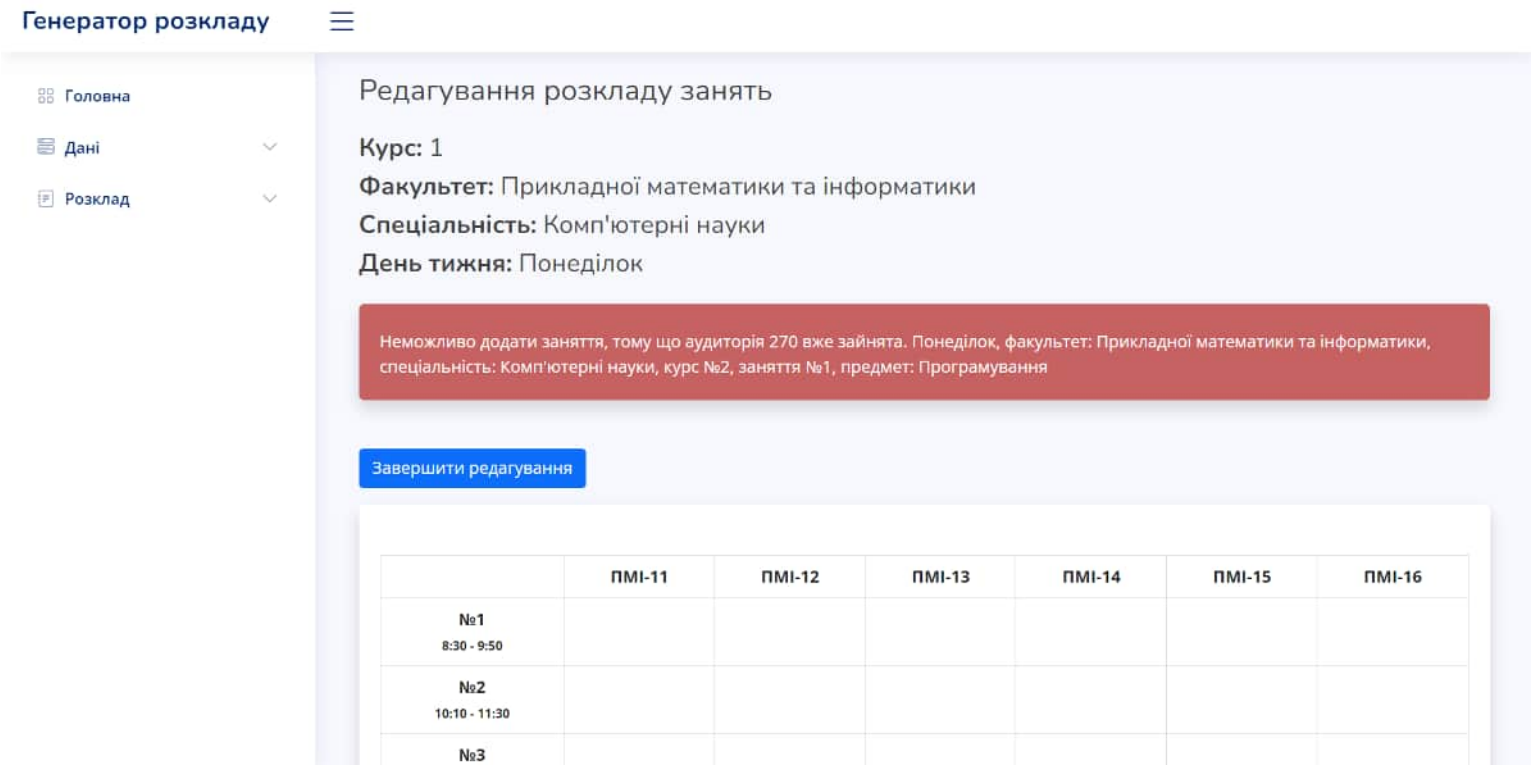
Додаток С

Повідомлення про конфлікт викладача в розкладі



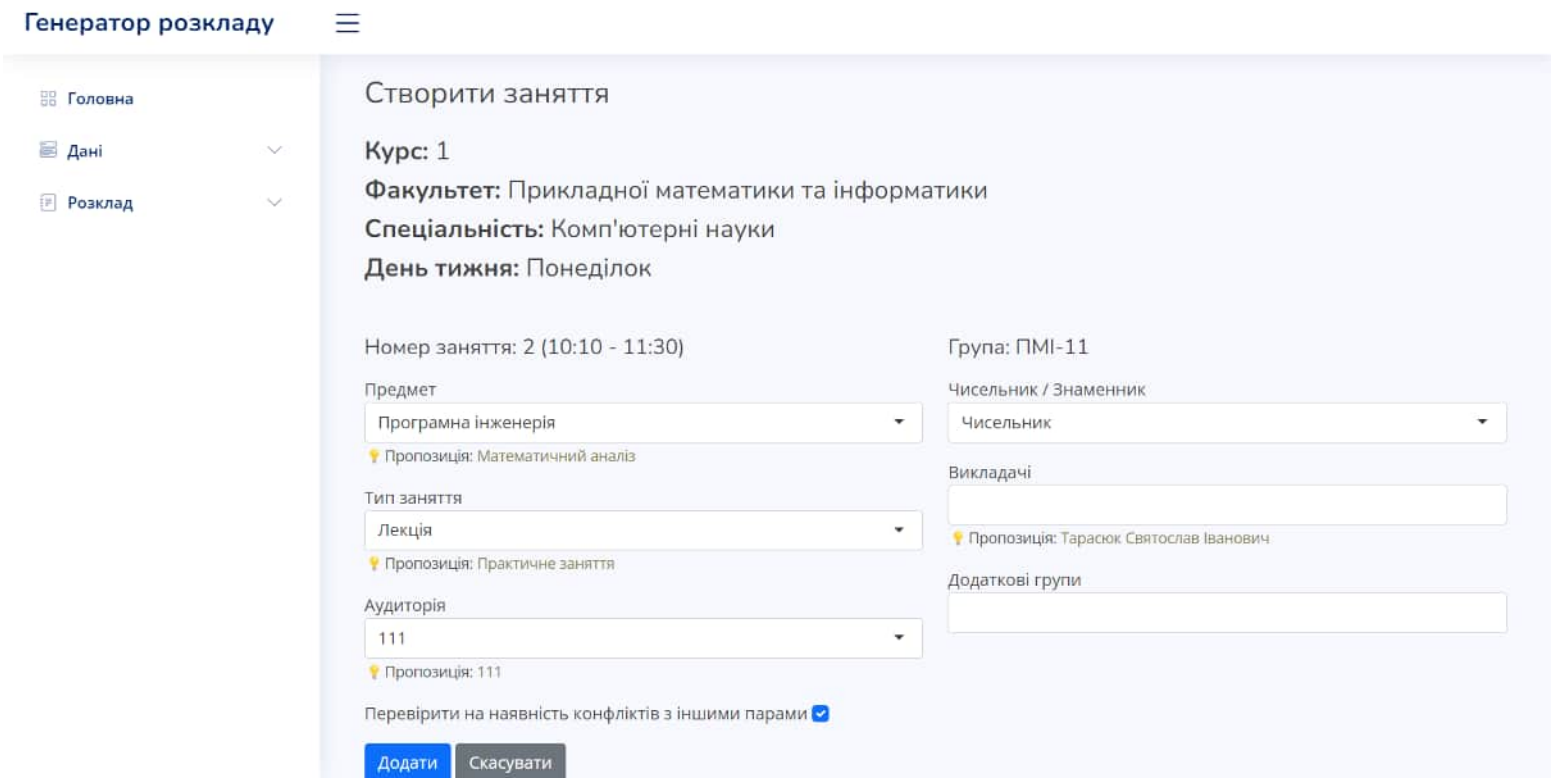
Додаток Т

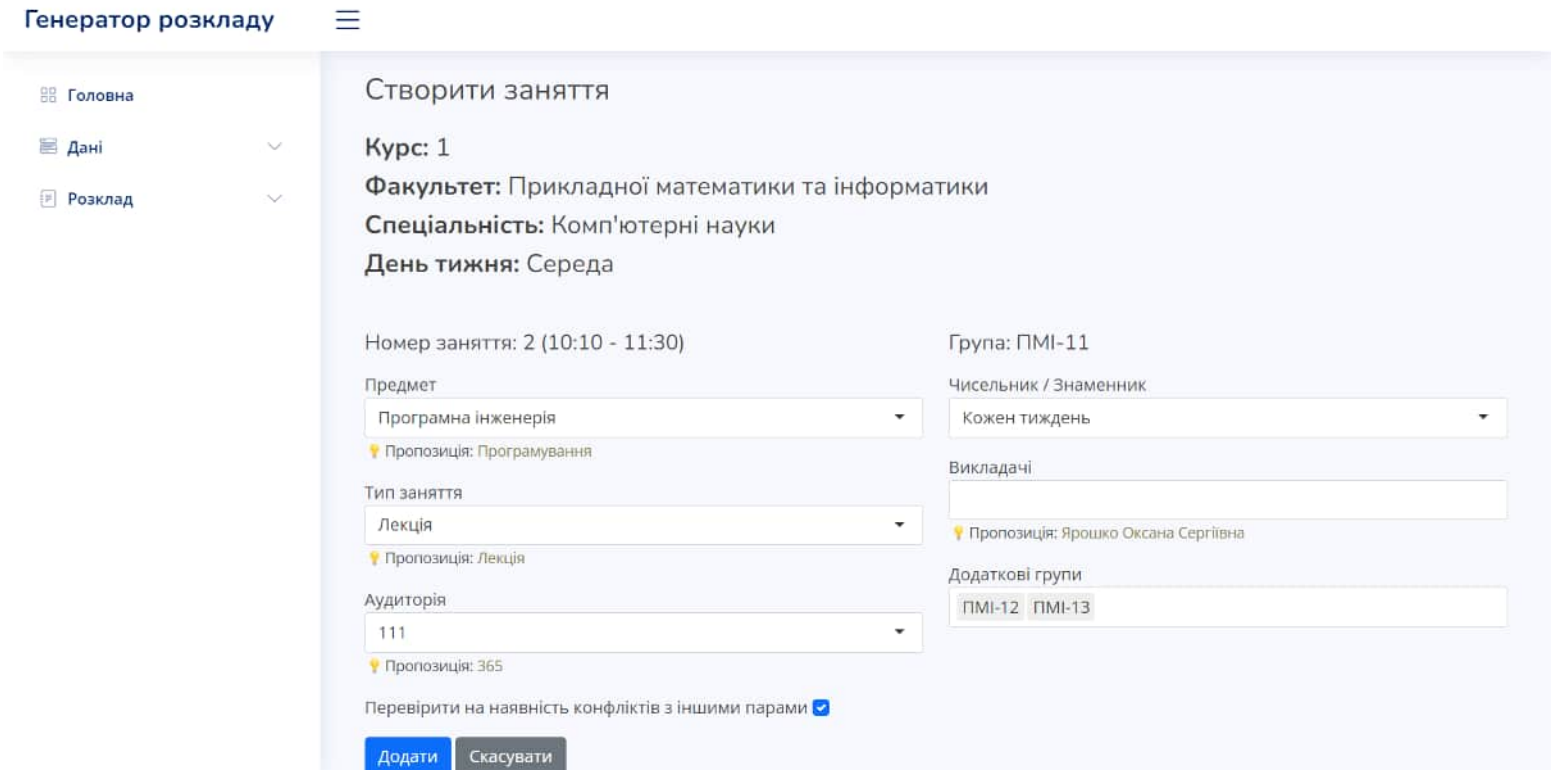
Повідомлення про конфлікт аудиторії в розкладі

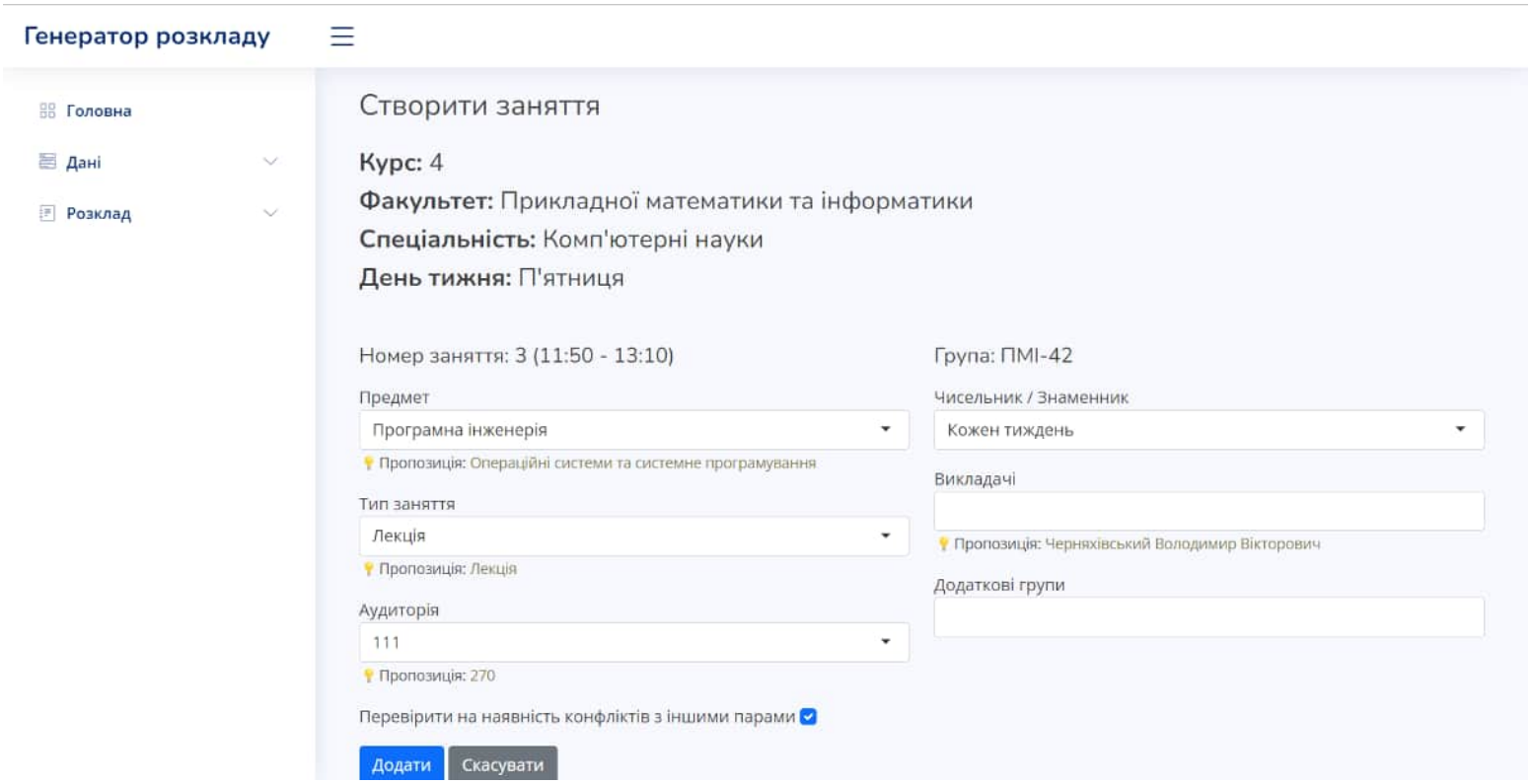


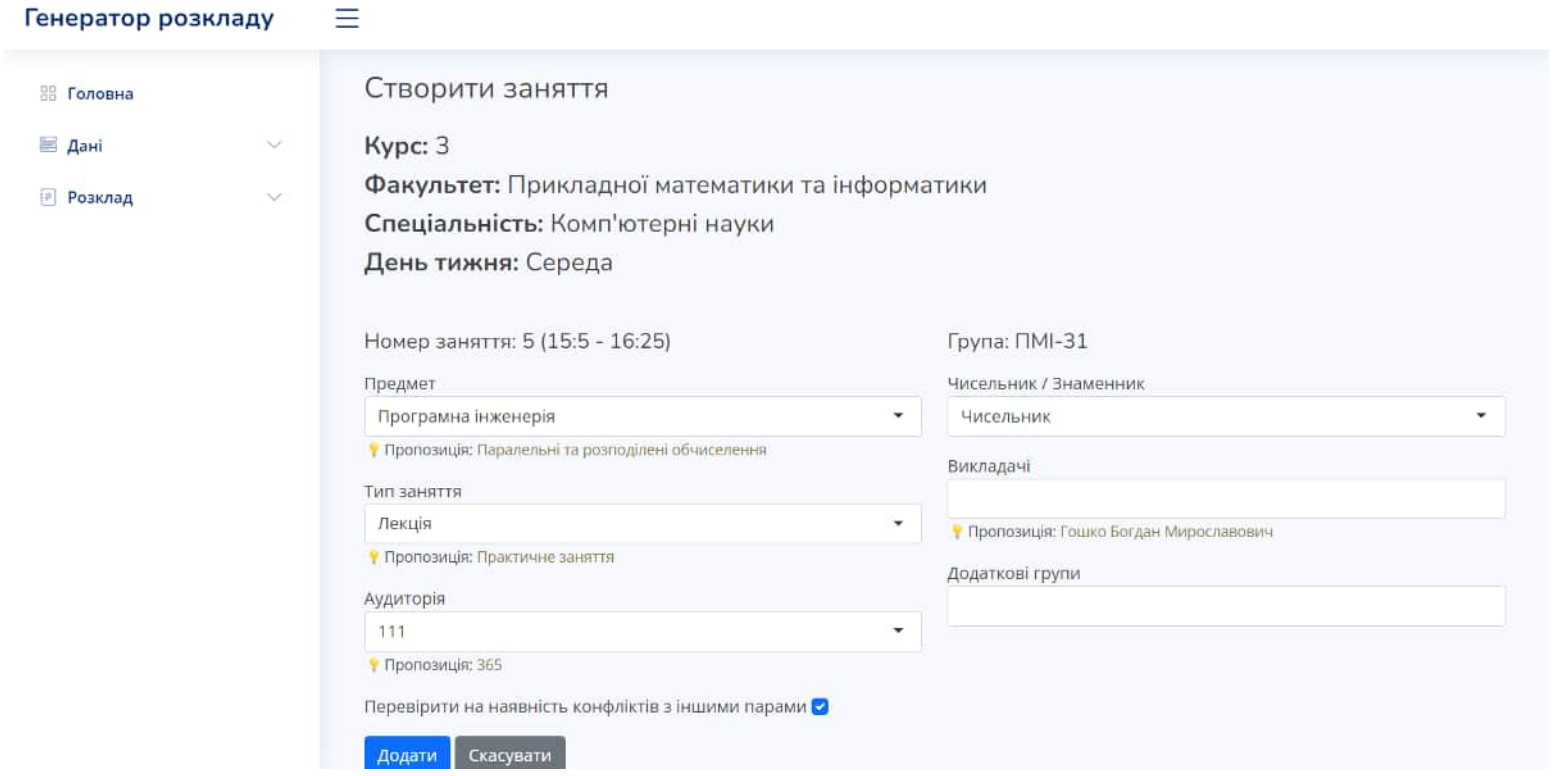
Додаток У

Підказки при створенні заняття (4 зображення)






Генератор розкладу
Головна
Дані
Розклад
Створити заняття
Курс: 4
Факультет: Прикладної математики та інформатики
Спеціальність: Комп'ютерні науки
День тижня: П'ятниця
Номер заняття: 3 (11:50 - 13:10)
Група: ПМІ-42
Предмет
Програмна інженерія
Пропозиція: Операційні системи та системне програмування
Чисельник / Знаменник
Кожен тиждень
Викладачі
Пропозиція: Черняхівський Володимир Вікторович
Тип заняття
Лекція
Пропозиція: Лекція
Додаткові групи
Аудиторія
111
Пропозиція: 270
Перевірити на наявність конфліктів з іншими парами
Додати
Скасувати


Генератор розкладу
Головна
Дані
Розклад
Створити заняття
Курс: 3
Факультет: Прикладної математики та інформатики
Спеціальність: Комп'ютерні науки
День тижня: Середа
Номер заняття: 5 (15:5 - 16:25)
Група: ПМІ-31
Предмет
Програмна інженерія
Пропозиція: Паралельні та розподілені обчислення
Чисельник / Знаменник
Чисельник
Викладачі
Пропозиція: Гошко Богдан Мирославович
Тип заняття
Лекція
Пропозиція: Практичне заняття
Додаткові групи
Аудиторія
111
Пропозиція: 365
Перевірити на наявність конфліктів з іншими парами
Додати
Скасувати

Додаток Ф

Клас LessonForLearning

```
31 references
public class LessonForLearning
{
    9 references
    public string SpecialityId { get; set; }

    9 references
    public string DayOfWeek { get; set; }

    9 references
    public string LessonNumberId { get; set; }

    4 references
    public string LecturerId { get; set; }

    2 references
    public string AudienceId { get; set; }

    4 references
    public string SubjectId { get; set; }
}
```

Додаток Х

Створення, тренування та збереження моделей для передбачення предмету та викладача

```
public void Train()
{
    pipelineForSubjectId = _mlContext.Transforms.Conversion.MapValueToKey(inputColumnName: "SubjectId", outputColumnName: "Label")
        .Append(_mlContext.Transforms.Categorical.OneHotEncoding(inputColumnName: "SpecialityId", outputColumnName: "SpecialityIdCategorized"))
        .Append(_mlContext.Transforms.Categorical.OneHotEncoding(inputColumnName: "DayOfWeek", outputColumnName: "DayofWeekCategorized"))
        .Append(_mlContext.Transforms.Categorical.OneHotEncoding(inputColumnName: "LessonNumberId", outputColumnName: "LessonNumberIdCategorized"))
        .Append(_mlContext.Transforms.Concatenate(outputColumnName: "Features",
            inputColumnNames: "SpecialityIdCategorized", "DayofWeekCategorized", "LessonNumberIdCategorized"));

    var EstimatorChain<KeyToValueMappingTransformer>? trainingPipelineForSubjectId = pipelineForSubjectId
        .Append(_mlContext.MulticlassClassification.Trainers
            .OneVersusAll(_mlContext.BinaryClassification.Trainers
                .AveragedPerceptron(labelColumnName: "Label", numberOfIterations: 10, featureColumnName: "Features"), labelColumnName: "Label"))
        .Append(_mlContext.Transforms.Conversion.MapKeyToValue(outputColumnName: "PredictedSubjectId", inputColumnName: "PredictedLabel"));
    trainedModelSubject = trainingPipelineForSubjectId.Fit(data);
    predEngineSubject = _mlContext.Model.CreatePredictionEngine<LessonForLearning, LessonPrediction>(trainedModelSubject);
    _mlContext.Model.Save(trainedModelSubject, data.Schema, Path.Combine(modelPath, "modelSubject.zip"));

    pipelineForLecturerId = _mlContext.Transforms.Conversion.MapValueToKey(inputColumnName: "LecturerId", outputColumnName: "Label")
            .Append(_mlContext.Transforms.Categorical.OneHotEncoding(inputColumnName: "SpecialityId", outputColumnName: "SpecialityIdCategorized"))
            .Append(_mlContext.Transforms.Categorical.OneHotEncoding(inputColumnName: "DayOfWeek", outputColumnName: "DayofWeekCategorized"))
            .Append(_mlContext.Transforms.Categorical.OneHotEncoding(inputColumnName: "LessonNumberId", outputColumnName: "LessonNumberIdCategorized"))
            .Append(_mlContext.Transforms.Categorical.OneHotEncoding(inputColumnName: "SubjectId", outputColumnName: "SubjectIdCategorized"))
            .Append(_mlContext.Transforms.Concatenate(outputColumnName: "Features",
            inputColumnNames: "SpecialityIdCategorized", "DayofWeekCategorized", "LessonNumberIdCategorized", "SubjectIdCategorized"));

    var EstimatorChain<KeyToValueMappingTransformer>? trainingPipelineForLecturerId = pipelineForLecturerId
        .Append(_mlContext.MulticlassClassification.Trainers
            .OneVersusAll(_mlContext.BinaryClassification.Trainers
                .AveragedPerceptron(labelColumnName: "Label", numberOfIterations: 10, featureColumnName: "Features"), labelColumnName: "Label"))
        .Append(_mlContext.Transforms.Conversion.MapKeyToValue(outputColumnName: "PredictedLecturerId", inputColumnName: "PredictedLabel"));
    trainedModelLecturer = trainingPipelineForLecturerId.Fit(data);
    predEngineLecturer = _mlContext.Model.CreatePredictionEngine<LessonForLearning, LessonPrediction>(trainedModelLecturer);
    _mlContext.Model.Save(trainedModelLecturer, data.Schema, Path.Combine(modelPath, "modelLecturer.zip"));
```

Використані джерела

1. https://www.entityframeworktutorial.net/what-is-entityframework.aspx
2. https://docs.microsoft.com/en-us/ef/
3. https://docs.microsoft.com/en-us/aspnet/core/?view=aspnetcore-6.0
4. https://github.com/dotnet/aspnetcore
5. https://www.microsoft.com/en-us/sql-server/sql-server-2019?rtc=1
6. https://docs.microsoft.com/en-us/dotnet/csharp/
7. https://uk.wikipedia.org/wiki/C_Sharp
8. https://uk.wikipedia.org/wiki/JavaScript
9. https://www.tutorialsteacher.com/mvc/mvc-controller
10. https://docs.microsoft.com/en-us/aspnet/core/mvc/views/overview?view=aspnetcore-6.0
11. https://learn.microsoft.com/en-us/dotnet/machine-learning/
12. https://learn.microsoft.com/en-us/dotnet/api/?view=ml-dotnet
13. https://learn.microsoft.com/en-us/dotnet/machine-learning/how-does-mldotnet-work
14. https://learn.microsoft.com/en-us/dotnet/machine-learning/?WT.mc_id=dotnet-35129-website
15. https://learn.microsoft.com/en-us/dotnet/machine-learning/tutorials/github-issue-classification
16. https://learn.microsoft.com/en-us/dotnet/machine-learning/how-to-guides/save-load-machine-learning-models-ml-net
17. https://learn.microsoft.com/en-us/dotnet/machine-learning/how-to-guides/load-data-ml-net#load-data-from-other-sources
18. https://learn.microsoft.com/en-us/dotnet/machine-learning/how-to-guides/prepare-data-ml-net