МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ІВАНА ФРАНКА

Факультет прикладної математики та інформатики

(повне найменування назва факультету)

Кафедра інформаційних систем

(повна назва кафедри)

ДИПЛОМНА РОБОТА

Розробка системи формування розкладу навчального процесу

Виконав студент групи ПМІ-44

спеціальності 122 – комп'ютерні науки

(шифр і назва спеціальності)

Борейко Д.В.

(підпис) (прізвище та ініціали)

Керівник Горлач В.М.

(підпис) (прізвище та ініціали)

Рецензент

(підпис) (прізвище та ініціали)

2023

Зміст

ВСТУП

Генерування розкладу є важливим завданням організації навчального процесу у будь якому навчальному закладі. В основу покладено необхідність забезпечити кожного максимально зручним і в той же час ефективним набором занять, враховуючи певні правила, яких потрібно при складанні даного розкладу дотримуватись.

Зазвичай є певна людина або набір людей, які відповідають за правильну побудову навчального процесу. Їм потрібно враховувати величезну кількість факторів, наприклад, щоб певна аудиторія не пересікалась у двох різних груп (крім випадку, якщо це лекційне заняття), щоб один викладач не був одночасно на двох різних заняттях і т.д. Також є можливість власних побажань викладачів: у когось немає можливості проводити пару зранку, а хтось навпаки може працювати тільки до обіду, також буває, що в якийсь день викладач в принципі по особистих причинах не може проводити заняття.

Всі такі правила і невеликі побажання повинні бути враховані. Звідси і витікає основна проблема - через величезну кількість занять, які треба правильно розподілити з врахуванням усіх побажань, часто виникають помилки і розклад потрібно переробляти не один раз.

Беручи до уваги все вище сказане можемо зробити висновок що, враховуючи сучасний етап використання інформаційних технологій, є потреба створити зручне і стабільне середовище налагодження і спрощення створення розкладу.

Якщо оцінювати затрати, які потрібні при складанні розкладу, стає очевидним, що використання програми, яка буде сама перевіряти правильність виконання всіх умов, чим значно спростить роботу працівників, є просто необхідним.

Отже, метою даної дипломної роботи є розбір процесу генерування розкладу та пошук методу для його ефективної реалізації.

1. ОСНОВНІ ПОНЯТТЯ

1.1 Що таке розклад

1.1.1 Розклад

Розклад є легко зрозумілим для всіх, але процес його формування не є простим. Розклад - це документ, який регулює робочий графік і впливає на продуктивність вчителів (викладачів), і розглядається як фактор оптимізації навчального процесу. При створенні розкладу важливо враховувати багато обов'язкових факторів, а також можливість ігнорувати деякі з них у певних умовах. У цій роботі ми сфокусуємось на розкладі занять навчального закладу, де важливими факторами будуть:

- Викладачі, які ведуть предмети.
- Студенти.
- Аудиторії, де проводяться заняття.
- Правильне розподілення пар.

Також існують додаткові умови, які можуть бути враховані при складанні розкладу, такі як:

- Вимоги і бажання студентів / викладачів.

Основні умови є законодавчими для правильного складання розкладу, в той час як додаткові умови залишаються не критичними, але бажаними до врахування, адже вони на пряму впливають на комфорт як викладачів так і студентів

1.1.2 Предмети

Предмет – це навчальна дисципліна, яку проводить певний викладач для однієї конкретної групи, або декількох груп студентів разом. Також можливий варіант коли одну пару проводять кілька викладачів.

Є декілька загальних видів занять, такі як практичні заняття, лабораторні, семінари, самостійні заняття, екзамени і т.д. У даному випадку ми обмежимося лекціями та практичними заняттями.

1.1.3 Викладачі

Викладачі – працівники навчального закладу, які проводять заняття і є визначальною складовою навчального процесу. Кожен викладач займає певну посаду, має наукове звання та ступінь, які відповідають його досягненням на кафедрі та можуть відноситися до суб'єктивних параметрів при складанні розкладу.

Викладач зазвичай не може мати пару в двох групах одночасно крім випадку проведення лекцій, а також один викладач не може бути в кількох аудиторіях одночасно, тому це враховується при складанні правильного розкладу.

У кожного викладача можуть бути певні вимоги до розкладу, проте у даній роботі такі вимоги враховуватись не будуть, та це не заперечує їх появи у майбутньому.

1.1.4 Студенти

Студенти поділяються за терміном навчання на курси, та на групи по 20 – 30 осіб. Кожна група має свій унікальний код, який складається з назви, номеру курсу та номеру групи.

1.1.5 Аудиторії

Аудиторія – приміщення, призначене для усних промов, виступів, доповідей, лекцій та ін. перед публікою (студентами, слухачами тощо) у закладі освіти чи іншому громадському закладі.

1.2 Модель розкладу

З використанням попередньої інформації можемо створити певну модель розкладу. [1][2]

Таблиця 1. - Первинна модель розкладу

День	№Пари	Викладач	Предмет	Тип	Група	Аудиторія
…	…	…	…	…	…	…

Створений первинний ключ складається з предмету, типу заняття та групи у якої це заняття повинно проводитись. Ці дані дають достатнє уявлення про створену пару.

1.3 Огляд технологій

Мова програмування C# представляє собою потужний і простий у використанні об'єктно-орієнтований інструмент. Вона надає розробникам можливість створювати програми з великою функціональністю. C# відноситься до мов компілюваного типу, тому має всі переваги таких мов. C# поєднує найкращі можливості таких мов програмування як C++, Java, Visual Basic тощо.

Через велику різноманітність синтаксичних конструкцій та можливості працювати з платформою .Net, C# дозволяє швидше, ніж будь-яка інша мова, розробляти програмні рішення.

Високого рівня надійності в мові програмування C# було досягнуто завдяки роботі CLR (Common Language Runtime). CLR запускає розроблені додатки на віртуальному процесорі, що відрізняється від інших компіляторів. Це означає, що якщо виникають помилки, вони не впливають на роботу інших

програм у системі. Однак, це також призводить до деякого додаткового часу, необхідного для запуску програми. Таким чином, програми, написані на мові програмування C#, мають високу надійність, але можуть бути менш швидкими.

.NET – це платформа розрозблена Microsoft, вона дає можливість створювати програмні рішення. Випуск .NET Framework відбувся у 2002 році. .NET Framework було створено як альтернативу Java-платформі компанії Sun. Однак, основною відмінністю є те, що .NET Framework був офіційно призначений для роботи з операційними системами Microsoft Windows. З тих пір вона пройшла довгий шлях від версії 1.0 до 4.8, і на сьогоднішній день, незважаючи на появу покращеної платформи нового покоління (.NET Core), як і раніше, досить популярна: існує безліч програмних продуктів, бібліотек і фреймворків, які написані та розвиваються під .NET Framework.

У 2016 році була представлена .NET Core - модульна платформа, яка була розроблена як розширення до .NET Framework. Одне з основних відмінностей .NET Core полягає в його кросплатформенності, що означає його сумісність з різними операційними системами. Це відкрило широкі можливості для розробників, що відтоді значно розширило сферу застосування .NET. Кросплатформенність .NET Core відіграла важливу роль у просуванні .NET серед розробників, надаючи їм нові сценарії та більше варіантів використання цієї платформи.

ASP.NET MVC являє собою платформу для створення сайтів і веб-додатків з використанням патерну MVC (model - view - controller).

Шаблон MVC, що лежить в основі платформи, має на увазі взаємодію трьох компонентів: контролера (controller), моделі (model) і вигляду (view).

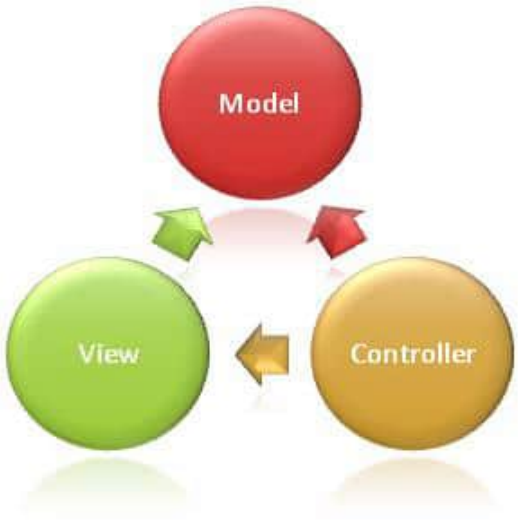
Model представляє набір класів, що описують логіку використовуваних даних та реагує на команди контролера.

Controller представляє собою клас(-и), у якому і починається основна робота написаної програми. Цей клас забезпечує зв'язок між моделлю і

представленням(view). Отримавши певні данні від користувача, використовуючи закладену в нього логіку інтерпретує дії користувача, сповіщаючи модель про необхідність певних змін.

View – це основна частина для користувача, тобто візуал, або інтерфейс створенної програми, веб-застосунку, з яким працює користувач.

Загальну схему взаємодії спрощено можна представити таким чином:



Веб-форми ASP.NET і MVC — це два веб-фреймворки, розроблені Microsoft — обидві вони є хорошим вибором. Жоден з веб-фреймворків не замінюється другим, і не планується їх «злиття» в єдиний фреймворк. Постійна підтримка та розвиток паралельно здійснюються Microsoft, і жоден з них не зникне.

Кожен з цих веб-фреймворків має переваги/недоліки, деякі з яких необхідно враховувати при розробці веб-додатків. Веб-додаток можна розробити за допомогою будь якої технології - це може полегшити розробку для конкретної програми, вибираючи одну технологію порівняно з іншою і навпаки.

Переваги використання ASP.NET MVC замість традиційних веб-форм:

Включає повний контроль над відтвореним HTML.

Забезпечує чітке розділення проблем (SoC).

Більш ніж будь-коли веб-додатки стають такими ж багатими, як і програми, які ви бачите на своїх настільних комп'ютерах. За допомогою MVC це дає вам можливість інтегрувати певні набори інструментів (наприклад, jQuery) з більшою легкістю та безперебійністю, ніж у веб-формах.

Дозволяє розробку на основі тестування (TDD).

За допомогою MVC ви можете легше створювати тести для веб-сторінок. Додатковий рівень тестування забезпечить ще один рівень захисту від несподіваної поведінки.

Оптимізація для пошукових систем (SEO)

URL-адреси є більш дружніми для пошукових систем (наприклад, mywebapplication.com/users/ 1 - отримати користувача з ідентифікатором 1 проти mywebapplication/users/getuser.aspx (ідентифікатор передається під час сеансу)).

Проста інтеграція з фреймворками JavaScript.

Дотримується дизайну веб-сайту Stateless.

Поділ відповідальності. У MVC додаток складається з трьох частин: контролера, представлення та моделі, кожна з яких виконує свої функції. В результаті додаток буде легше підтримуватись модифікуватись в майбутньому. В силу поділу відповідальності додатки mvc мають кращу тестованість, окремі компоненти можна тестувати незалежно один від одного.

Entity Framework — це фреймворк з відкритим вихідним кодом для програм .NET, які підтримуються самими Microsoft. Це підвищує продуктивність розробника, через те, що дозволяє розробникам опрацьовувати дані, використовуючи об'єкти предметних класів, не зосереджуючи увагу на базових таблицях і стовпцях бази даних, де ці дані зберігаються. Це усуває потребу в написанні більшої частини коду , що використовується для взаємодії з базою даних, який зазвичай потрібно написати розробникам. Entity Framework надає

розробникам абстрактний рівень для роботи з таблицею та стовпцями. Це також зменшує розмір коду конкретних програм, а також підвищує читабельність коду завдяки його простоті. Це нова технологія для доступу до даних для програм Microsoft. Остання версія Entity Framework — 7.0.

Він також надає можливість взаємодії з об'єктами двома способами, як за допомогою LINQ у вигляді LINQ to Entities, так і з використанням Entity SQL.

Підходи в Entity Framework.

Починаючи з початкової версії Entity Framework, підтримувався підхід Database First, який дозволяв генерувати модель EDMX на основі наявної бази даних. Згенерована модель потім використовувалася для підключення до бази даних. У подальшому було додано підхід Model First, який дозволяв розробникам створювати модель EDMX відразу, без наявності готової бази даних. Модель створювалась графічним способом, а потім генерувалась відповідна база даних.

Але з випуском версії 5.0 був доданий підхід Code First, який став основним напрямом розробки в Entity Framework. За допомогою Code First розробники можуть спочатку визначити модель даних за допомогою класів та атрибутів, а потім автоматично створити відповідну базу даних на основі цієї моделі. Таким чином, з плином часу Entity Framework пройшов шлях від підходу Database First до підходів Model First і Code First, дозволяючи розробникам вибирати найзручніший підхід для роботи з базою даних. Це функція Entity Framework. Code First додає конструктор моделей, який перевіряє ваші класи, якими керує контекст, а потім використовує набір правил або умов, щоб визначити, як ці класи та відносини описують модель, і як ця модель має відображатися у вашій базі даних. Все це відбувається під час виконання. Його суть - спочатку пишеться код моделі на C#, а потім вже по даному коду метод генерує базу даних. При цьому модель EDMX вже не використовується. В даному проекті був використаний Code First.

MS SQL Server є однією з найпопулярніших систем управління базами даних (СУБД) у світі. Ця СУБД є універсальною і використовується для різних типів проектів - від невеликих додатків до великих високонавантажених систем.

Однією з важливих особливостей SQL Server є його продуктивність. Він працює дуже швидко і забезпечує ефективну обробку транзакцій і запитів.

Надійність та безпека також є суттєвими характеристиками SQL Server. Він надає можливості шифрування даних, що забезпечує їх захищеність і конфіденційність.

SQL Server відрізняється простотою в використанні і адмініструванні. Він має зручний інтерфейс та інструменти, що спрощують роботу з ним.

Основою SQL Server є база даних, яка використовує реляційну модель. Ця модель баз даних була розроблена ще у 1970 році і досі використовується як стандартна організація даних.

В реляційній моделі дані зберігаються у вигляді таблиць, де кожна таблиця має рядки та стовпці. Кожен рядок представляє окремий об'єкт, а стовпці визначають атрибути цього об'єкта.

MS SQL Server є потужною СУБД, яка надає широкий спектр можливостей для ефективного управління та обробки даних у реляційній моделі.

SQL Server має вбудовану підтримку для .NET Framework, що дозволяє розробникам писати процедури бази даних на будь якій мові платформи .NET. Вони можуть використовувати повний набір бібліотек, який надається в рамках .NET Framework.

Один з важливих аспектів цієї інтеграції полягає в тому, що .NET Framework надає власні механізми керування SQL Server, які не залежать від вбудованих механізмів операційної системи Windows. Це означає, що .NET Framework виділяє додаткову пам'ять та будує власні ресурсні алгоритми спеціально для оптимального використання в структурах SQL Server.

Такий підхід підвищує продуктивність порівняно з загальними алгоритмами операційної системи Windows. Це сприяє ефективному розподілу ресурсів та покращує швидкодію процедур бази даних, які використовують .NET Framework.

Все це робить SQL Server інтегрованою та потужною платформою для розробки баз даних, яка забезпечує високу продуктивність та гнучкість для розробників, які працюють з платформою .NET.

ASP.NET CORE – технологія, яка дає змогу створювати різного роду веб-додатки: від невеличких веб-сайтів до великих веб-програм і веб-сервісів.

ASP.NET Core є повністю відкритим фреймворком (open-source framework), що дозволяє розробникам створювати крос-платформні додатки. Це означає, що ви можете розробляти та запускати додатки на різних операційних системах, таких як Windows, macOS та Linux.

Одним із ключових компонентів ASP.NET Core є серверна частина, яка відповідає за обробку HTTP-запитів та відправку відповідей. Для розгортання веб-додатків на ASP.NET Core ви можете використовувати традиційний веб-сервер IIS (Internet Information Services) на платформі Windows. Але також є альтернативний веб-сервер Kestrel, який є крос-платформним і працює на різних операційних системах.

ASP.NET Core надає розробникам багато гнучких можливостей та високу продуктивність для створення веб-додатків, які можуть працювати на різних платформах та з різними веб-серверами. Це робить його привабливим вибором для розробників, які бажають створити крос-платформні рішення.

В даній версії є одна загальна програмна модель ASP.NET Core MVC. Також було додано ряд додаткових функцій. Прикладом такої функції є тег-хелпери (tag helper), вони дають змогу більш органічно поєднувати синтаксис мови html з кодом на C#.

1.4 Реєстрація та авторизація

Авторизація відіграє важливу роль у різних системах, таких як веб-додатки, мобільні додатки, API тощо. Основні аспекти, які підкреслюють значення та переваги авторизації, включають наступне:

1. Забезпечення безпеки: Авторизація гарантує, що доступ до системи мають лише правомірні користувачі. Це особливо важливо для захисту конфіденційної та критичної інформації. Авторизація дозволяє ідентифікувати користувачів і перевіряти їх права доступу, що допомагає запобігти несанкціонованому доступу та зловживанням.

2. Контроль доступу: Авторизація дозволяє регулювати доступ до функцій і ресурсів системи на основі ролей та прав. Це дозволяє надавати користувачам лише необхідний рівень доступу для виконання їх завдань, забезпечуючи контроль і управління привілегіями.

3. Персоналізація: Авторизація дозволяє надавати різні рівні доступу та функціональність для різних категорій користувачів. Наприклад, адміністратори можуть мати повний доступ до адміністративних інструментів, тоді як звичайним користувачам доступні лише базові функції. Це дозволяє налаштовувати досвід користувачів і забезпечувати оптимальне використання системи.

4. Відстеження дій користувачів: Авторизація дозволяє реєструвати дії користувачів і відстежувати їх активність в системі. Це допомагає виявляти можливі проблеми з безпекою і надає історичні дані про взаємодію користувачів з системою для аналізу та покращення функціональності.

5. Зручне управління: Авторизація дозволяє легко керувати правами доступу користувачів і змінювати їх за потреби. Це дозволяє адміністраторам ефективно керувати правами доступу, додавати нові ролі та відбирати доступ у окремих користувачів.

6. Сумісність зі стандартами: Багато систем авторизації базуються на відкритих стандартах, таких як OAuth і OpenID Connect. Це сприяє сумісності та інтеграції з іншими системами, що підтримують ці стандарти.

В цілому, авторизація є важливим компонентом будь-якої системи, що працює з користувачами та має обмеження доступу до функцій і ресурсів. Використання авторизації допомагає забезпечити безпеку, контроль і гнучкість управління доступом, що є критичними для ефективної та надійної роботи системи.

JWT (JSON Web Token) - це відкритий стандарт для безпечної передачі інформації у вигляді токенів між сторонами. Використання JWT токенів має кілька переваг:

1. Легкість і компактність: JWT токени мають простий формат, який складається з трьох частин: заголовка, набору заявок (claims) і підпису. Це дозволяє легко передавати їх у вигляді рядка в HTTP заголовку або запиті. Такий підхід робить JWT токени ефективними для використання в мережевих комунікаціях.
2. Автентифікація та авторизація: JWT токени дозволяють ідентифікувати та перевіряти автентичність користувачів. Вони містять інформацію про користувача, таку як його ідентифікатор, ім'я, роль і т. д. Ця інформація може бути використана для авторизації доступу до ресурсів і функціональності.
3. Безпека: JWT токени можуть бути підписані за допомогою секретного ключа або алгоритму публічного ключа, такого як RSA. Це дозволяє перевірити цілісність токена і підтвердити його відповідність довіреній стороні. Такий підхід запобігає підробці та зміні токенів, забезпечуючи безпеку обміну даними.

4. Розширюваність: JWT токени можуть містити додаткові заявки, які передають додаткову інформацію про користувача або контекст використання. Це дозволяє розширити функціональність токенів і включити додаткові дані за потреби, що забезпечує гнучкість в управлінні доступом.

5. Незалежність від стану: JWT токени є станозалежними, що означає, що сервер не потребує зберігання стану аутентифікації для кожного користувача. Це полегшує масштабування системи та дозволяє розподілений доступ до ресурсів, оскільки сервер не залежить від зберігання інформації про стан аутентифікації окремих користувачів.

6. Підтримка крос-платформеності: JWT токени підтримуються багатьма мовами програмування і платформами, що робить їх універсальним рішенням для використання в різних системах.

Загалом, використання JWT токенів спрощує реалізацію безпеки та автентифікації у розподілених системах, API та веб-додатках. Вони забезпечують безпеку, зручність та ефективність у передачі інформації та управлінні доступом до ресурсів.

2. РОЗРОБКА ПРОГРАМНОЇ МОДЕЛІ ТА ЇЇ ТЕСТУВАННЯ

2.1 Вибір середовища програмування

Основними вимогами до середовища програмування для виконання даного проекту будуть:

- потужність мови програмування
- легкість написання коду
- швидкість та ефективність роботи
- наявність бібліотек та інструментів, що можуть полегшити розробку
- просте підключення до бази даних
- зручний та простий у використанні функціонал для розробки та адміністрування бази
- зручний інтерфейс та інструментарій для оптимізації та виправлення коду

Беручи до уваги дані вимоги, була вибрана MS Visual Studio 2019 Community з базою даних на основі MS SQL Server. Як основна мова програмування буде використана C#.

2.2 Архітектура бази даних

З метою реалізації схеми, описаної в розділі 1.2, потрібно створити базу даних, яка буде зберігати дані про навчальний заклад які потрібні для побудови розкладу. Виходячи з опису в Таблиці 1, потрібно створити такі таблиці:

1. Таблиця груп
2. Таблиця викладачів
3. Таблиця предметів
4. Таблиця аудиторій
5. Таблиця пар

6. Таблиця дисциплін (загальний вміст предмета, виду пари та викладача який його проводить)
7. Таблиця навчальних програм

Дані таблиці повинні певним чином бути зв’язаними між собою.

Таблиця груп студентів має в собі номер групи, курс, кількість студентів.

Таблиця викладачів несе в собі інформацію про повне ім’я викладача.

Таблиця аудиторій повинна відображати номер аудиторії та кількість місць.

Таблиця предметів зберігає назву предмету та курс на якому він проводиться.

Таблиця навчальних програм зберігає назву програми, курс, та список груп для яких доступна дана програма.

Таблиця дисциплін містить інформацію про предмет, викладача, асистента, тип заняття та навчальну програму, до якої прив’язана дана дисципліна.

Таблиця пар, це таблиця яка об’єднує дані з таблиці дисциплін та додає до них інформацію про день проведення та номер пари.

Беручи до уваги все вище перераховане, створено базу даних побудовану на таких таблицях:

Groups – таблиця груп,

Audiences – таблиця аудиторії,

Teachers – таблиця викладачів,

Subjects – таблиця предметів,

CourseProgram – таблиця навчальних програм,

Program_Subjects – таблиця дисциплін,

Pairs – таблиця пар,

PairToGroup – поєднання групи та пари, яка є в неї.

- Subjects
  - Id – id предмета
  - Name – назва предмета
- Groups
  - Id – id групи
  - Name – назва групи
  - Year – рік навчання
  - AmountStudets – кількість студенів
  - CourseProgramId – id навчальної програми
- Audiences
  - Id – id аудиторії
  - Name – назва(номер) аудиторії
  - Capacity – кількість місць
- Teachers
  - Id – id викладача
  - LastName – прізвище викладача
  - Name – ім’я викладача
  - FutherName – по-батькові викладача
- CourseProgram
  - Id – id навчальної програми
  - Name – назва навчальної програми
- Program_Subjects
  - Id – id дисципліни
  - SubjectId – id предмета
  - Type – тип
  - MainTeacher – id основного викладача
  - LabTeacher – id асистента

- ProgramId – id навчальної програми

- Pairs
  - Id – id пари
  - AudienceId – id аудиторії
  - Day – день
  - Number – номер пари
  - Program_SubjectId – id дисципліни
  - IsEveryWeek – чи є пара кожного тижня
- PairToGroup
  - Id – id
  - PairId – id пари
  - GroupId – id групи

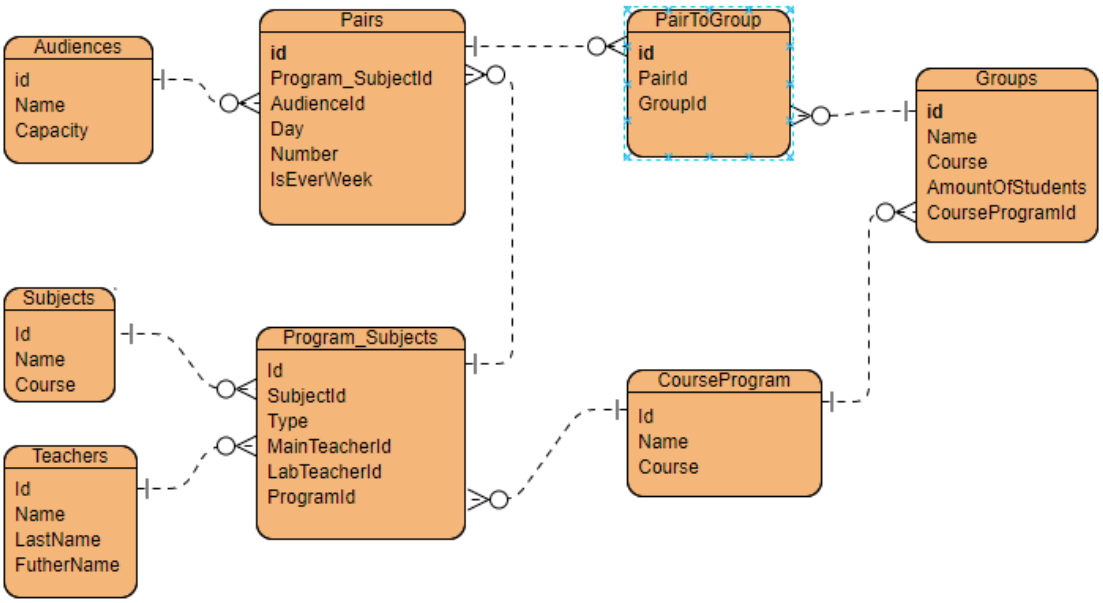


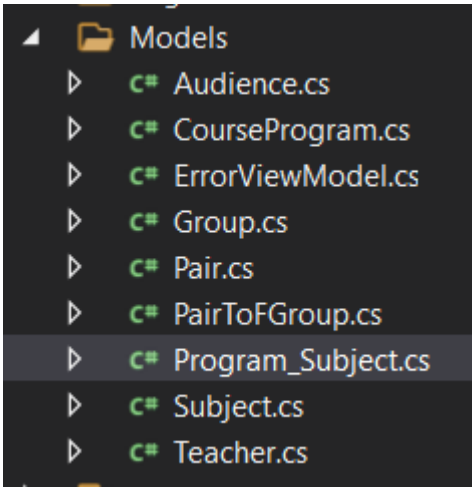
Рис. 2.1 ER-Діаграма бази даних

2.3 Архітектура програмної системи

Програмний продукт спроектований та розроблений згідно шаблону проектування MVC.

2.3.1 Моделі

Модель системи складається з класів Audience, CourseProgram, Group, Pair, PairToGroup, Program_Subject, Subject, Teacher які місять в собі моделі та логіку обробки даних



Клас Audience:

```
20 references
public class Audience
{
    11 references
    public int Id { get; set; }
    [Required]
    21 references
    public string Name { get; set; }
    [Required]
    11 references
    public int Capacity { get; set; }

}
```

Клас CourseProgram:

```
20 references
public class CourseProgram
{
    [Key]
    30 references
    public int Id { get; set; }
    [Required]
    13 references
    public string Name { get; set; }
    [Required]
    9 references
    public int Course { get; set; }
    1 reference
    public virtual List<Program_Subject> Program_Subject { get; set; }
    4 references
    public virtual List<Group> Groups { get; set; }
}
```

Клас Group:

```
32 references
public class Group
{
    30 references
    public int Id { get; set; }
    [Required]
    24 references
    public string Name { get; set; }
    11 references
    public int Course { get; set; }
    11 references
    public int AmountOfStudents { get; set; }
    3 references
    virtual public List<PairToFGroup> PairToFGroups { get; set; }
    12 references
    virtual public CourseProgram CourseProgram { get; set; }
}
```

Клас Pair:

```
43 references
public class Pair
{
    38 references
    public int Id { get; set; }
    44 references
    virtual public Program_Subject Program_Subject { get; set; }
    14 references
    virtual public Audience Audience { get; set; }
    22 references
    virtual public DayOfWeek Day { get; set; }
    18 references
    public int Number { get; set; }
    3 references
    virtual public List<PairToFGroup> PairToFGroups { get; set; }
    15 references
    public int IsEverWeek { get; set; }
    [NotMapped]
    6 references
    public string NameOfPair ...
    1 reference
    public Pair(Program_Subject s, Audience a, DayOfWeek d,int n)...
    1 reference
    public Pair()...
}
```

Клас PairToGroup:

```
10 references
public class PairToFGroup
{
    0 references
    public int Id { get; set; }
    6 references
    virtual public Pair Pair { get; set; }
    6 references
    virtual public Group Group { get; set; }
}
```

Клас Program_Subject:

```
26 references
public class Program_Subject
{
    15 references
    public int Id { get; set; }
    33 references
    virtual public Subject Subject { get; set; }
    22 references
    public string Type { get; set; }
    18 references
    virtual public Teacher MainTeacher { get; set; }
    9 references
    virtual public Teacher LabTeacher { get; set; }
    6 references
    virtual public CourseProgram ProgramId { get; set; }
    [NotMapped]
    4 references
    public string Name { get { return Subject.Name + " " + Type + " " + MainTeacher.ToShortString()
}
```

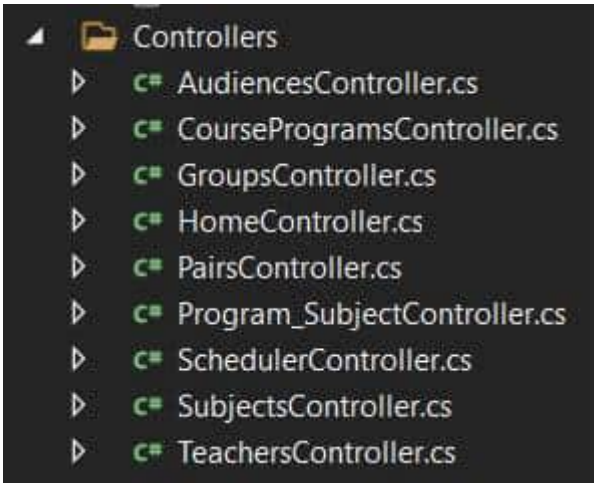
Клас Subject:

```
16 references
public class Subject
{
    15 references
    public int Id { get; set; }
    [Required]
    38 references
    public string Name { get; set; }
    [Required]
    17 references
    public int Course { get; set; }
}
```

Клас Teacher:

```
public class Teacher
{
    16 references
    public int Id { get; set; }
    [Required]
    17 references
    public string LastName { get; set; }
    [Required]
    17 references
    public string Name { get; set; }
    [Required]
    17 references
    public string FutherName { get; set; }
    [NotMapped]
    3 references
    public string FullName { get {return LastName + " " + Name + " " + FutherName; } }
    3 references
    public override string ToString()...
    6 references
    public string ToShortString()...
}
```

2.3.2 Контролери



Контролер HomeController є основою, з якої ми переходимо до всіх інших представлених контролерів, він надає сторінку, на якій користувач може використовувати всі доступні йому можливості програми.

Основою для розкладу служить SchedulerController, в ньому є 2 основні методи Index та MoveToCreator. В Index ми вибираємо програму, та групи зі списку доданих в вибрану програму. MoveToCreator переносить користувача на сторінку заповнення розкладу, з групами, які він обрав у методі Index.

Всі інші контролери мають однакову структуру і три основні методи Create() Edit() та Delete(), (CRUD операції) завдяки яким відбувається створення, редагування та видалення об’єктів.

3. Демонстрація розробленої програми

При запуску програми нас зустрічає привітальна сторінка

Тут ми можемо побачити основні 3 вкладки які доступні звичайному користувачу, а саме :

- Домашня сторінка з загальною інформацією.
- Короткий опис з загальною інформацією
- Сторінка Розкладу, де ми можемо вибрати свою групу та подивитись розклад для неї

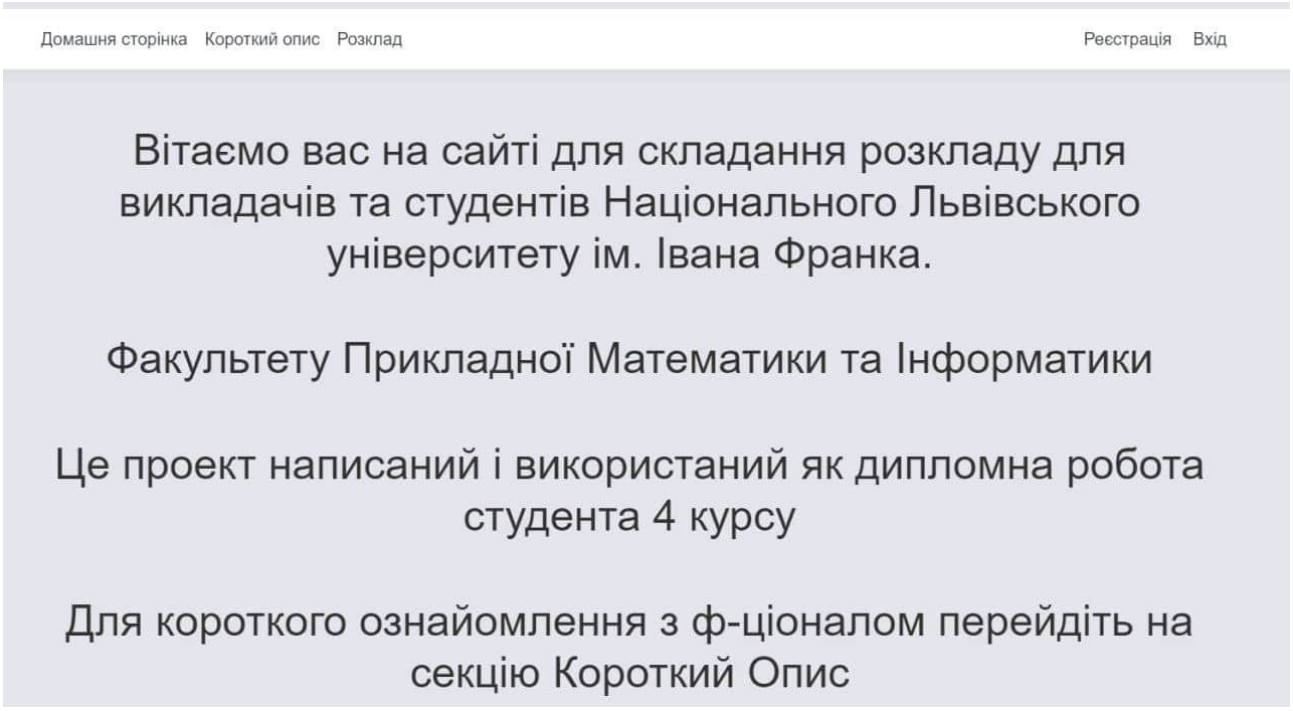


Рис.3.1 Домашня сторінка

Далі можемо перейти до сторінки з описом функціоналу (Короткий Опис)

Ця сторінка буде варіюватись в залежності від вашого статусу - адміністратора або звичайного користувача. Якщо ви адміністратор, вам будуть доступні набагато більші можливості для управління системою, а отже і інформації буде більше. Ви зможете переглядати, редагувати та видаляти ресурси, такі як групи, викладачі, аудиторії, предмети, також вам буде доступна вкладка з списком Навчальних програм, які ви також зможете редагувати. Також

вам будуть доступні додаткові вкладки, які містять розширену інформацію та функціональність.

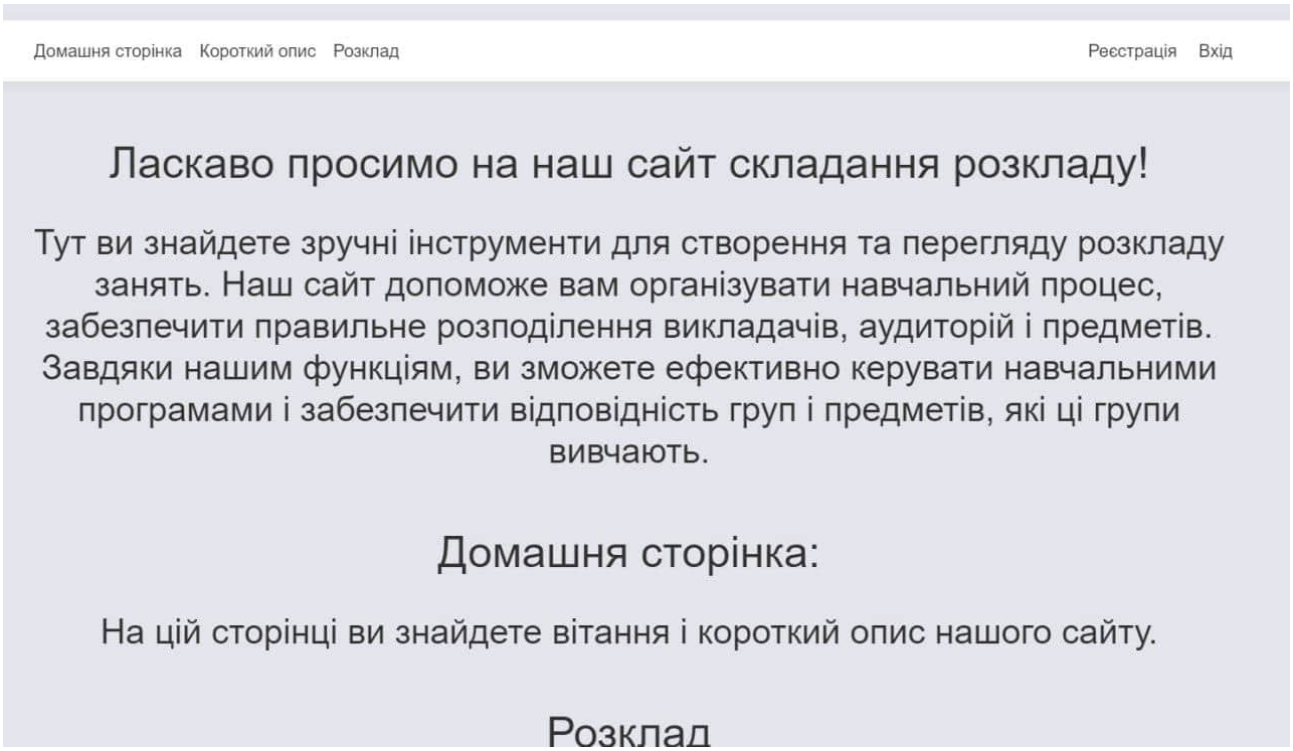


Рис.3.2 Сторінка з коротким описом

З іншого боку, як звичайний користувач, ваші можливості будуть обмежені. Ви зможете переглядати вміст, та виконувати певні дії, які вам дозволено, наприклад знайти та переглянути розклад для своєї групи, але не зможете здійснювати адміністративні функції. Для вас буде доступна основна інформація та набір функцій, необхідних для виконання вашої ролі.

Це диференціювання забезпечує кращий контроль та безпеку системи, дозволяючи адміністраторам мати повний доступ та контроль над системою, а звичайним користувачам - зручний та обмежений доступ до необхідної функціональності.

Далі вам як новому користувачу потрібно буде зареєструватись у системі.

Для цього можна використати кнопку, яка знаходиться у правому верхньому куті хедера і веде на сторінка реєстрації.

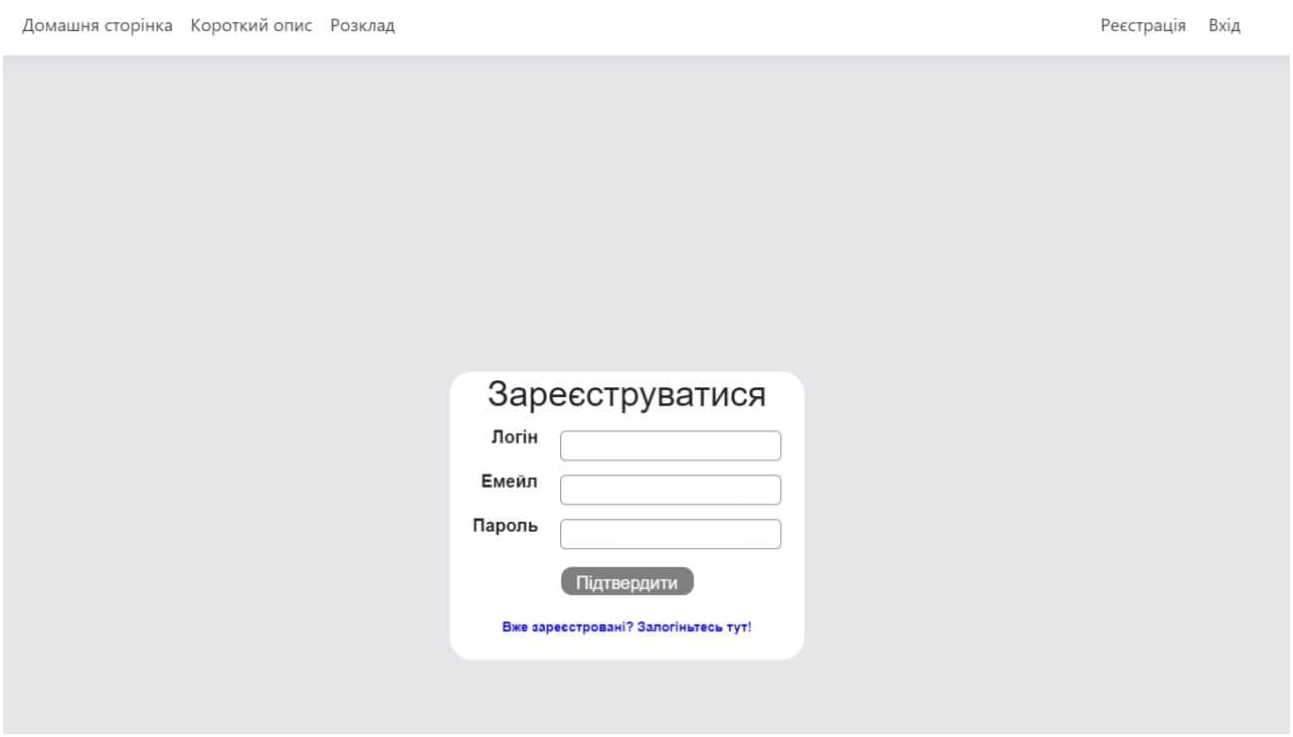


Рис.3.3 Сторінка Реєстрації

Якщо ж у вас уже є аккаунт ви можете використати вкладку Вхід і залогінитись на сайт

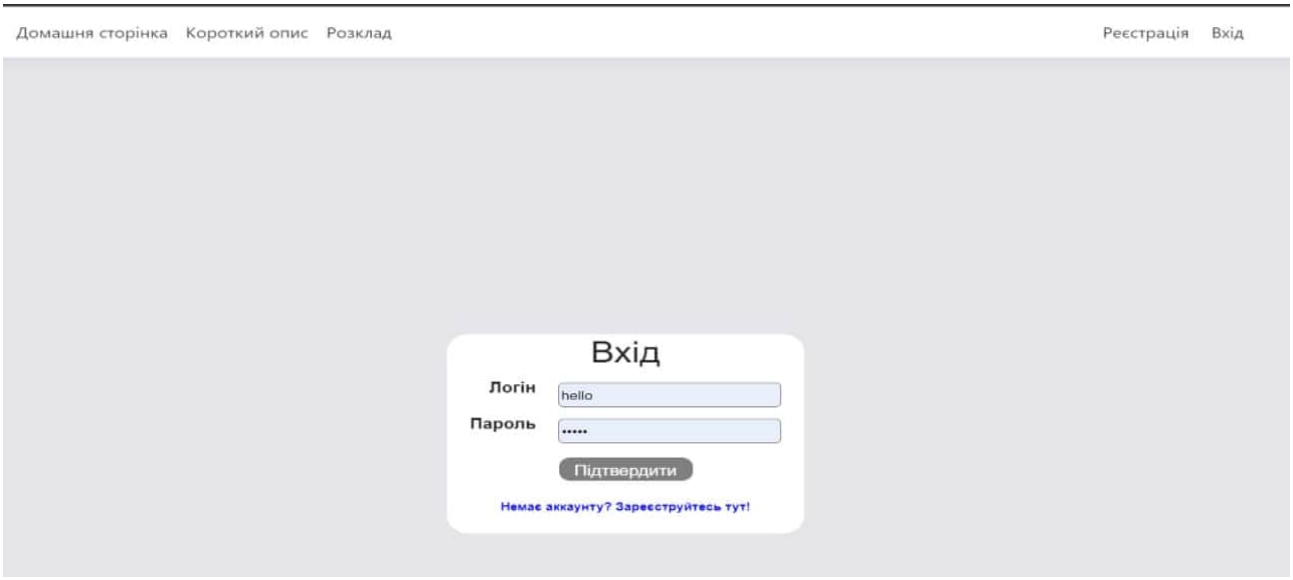


Рис.3.4 Сторінка Входу

Якщо ви увійшли з роллю адміністратора, то кількість доступних на верхній панелі вкладок для вас збільшиться

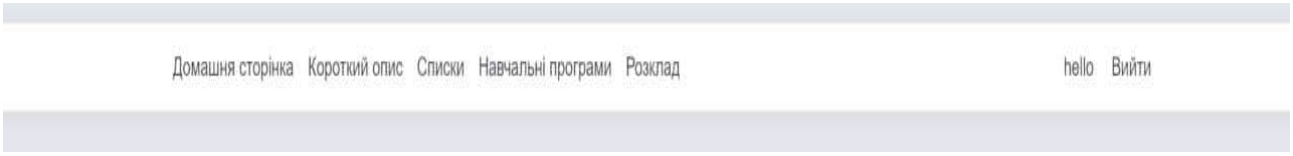


Рис.3.5 Панель користувача

Списки

Це сторінка з 4 основними таблицями:

Таблиця груп, викладачів, предметів, аудиторій. Тут користувачу необхідно заповнити таблиці, використовуючи кнопку Створити, для подальшого створення програм та складання розкладу

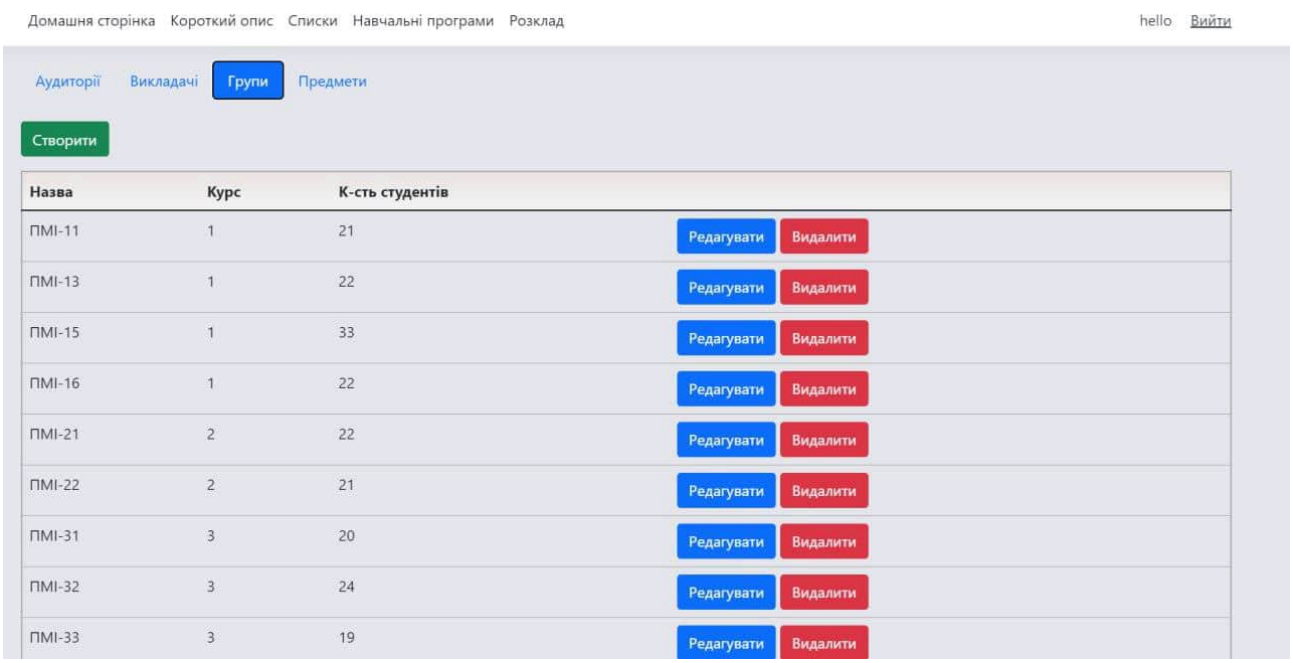


Рис 3.6 Списки

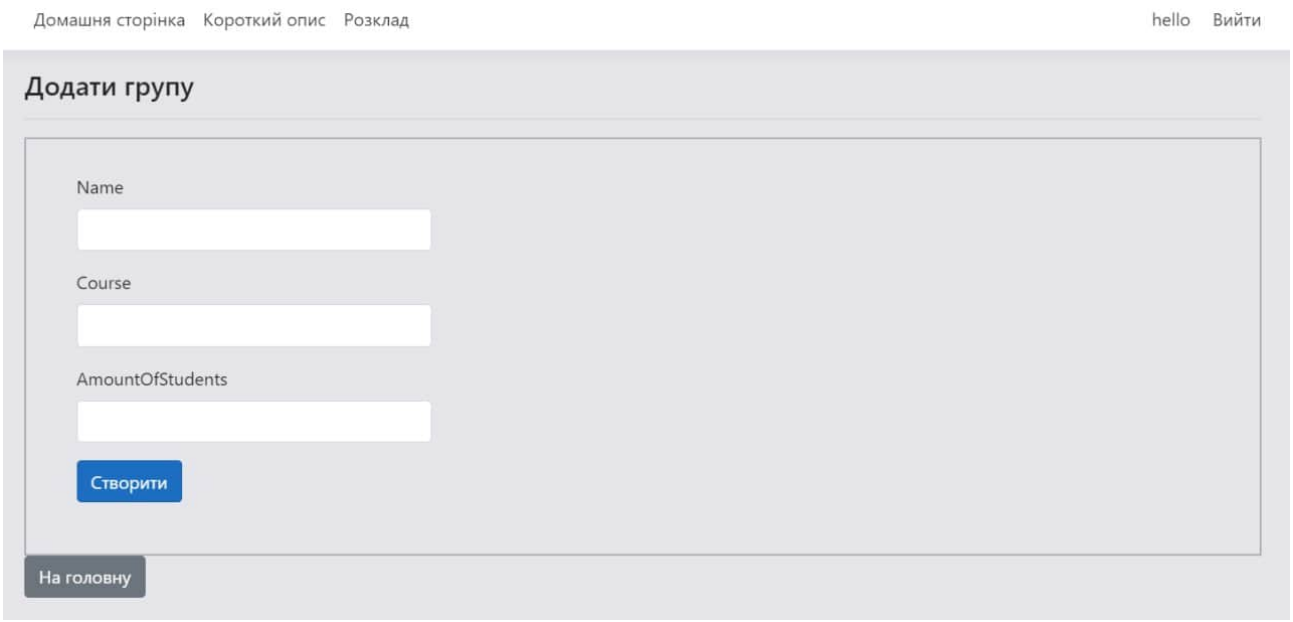


Рис 3.7 Приклад форми створення груп

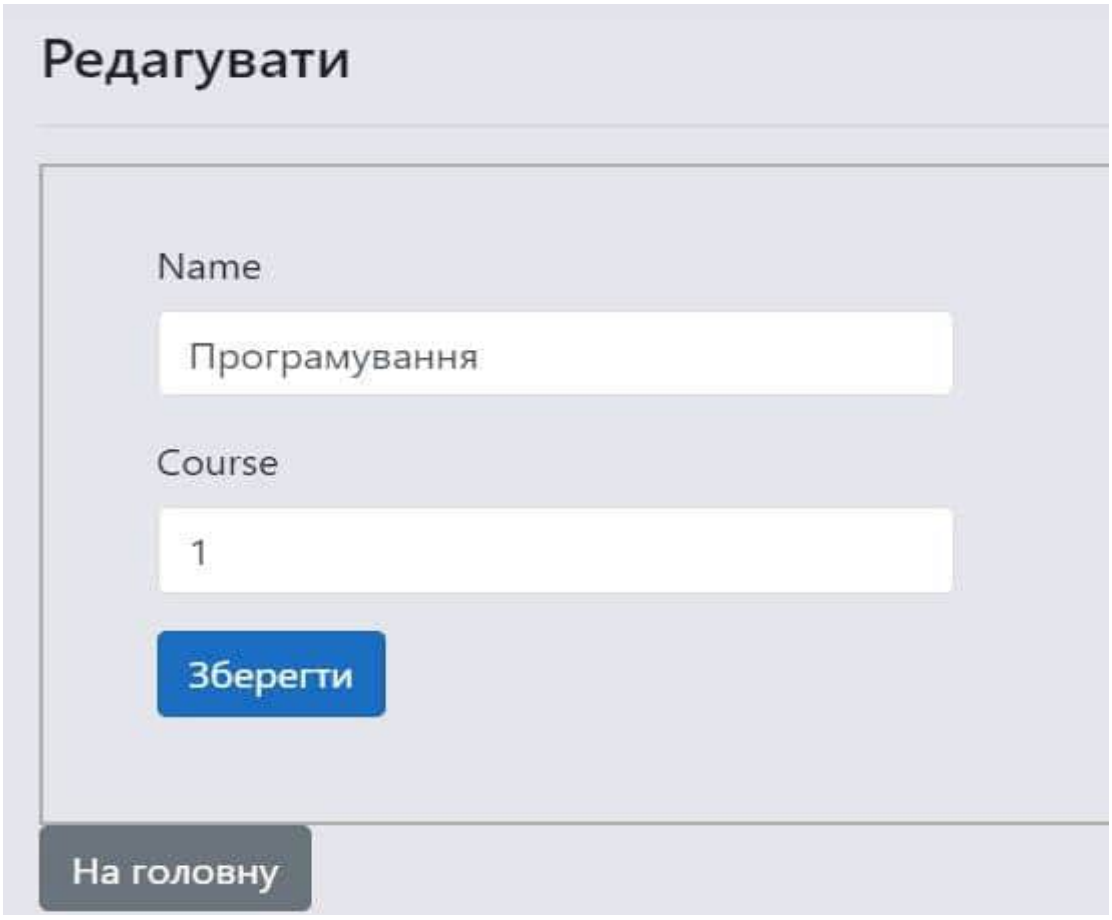


Рис 3.8 Приклад форми редагування предметів

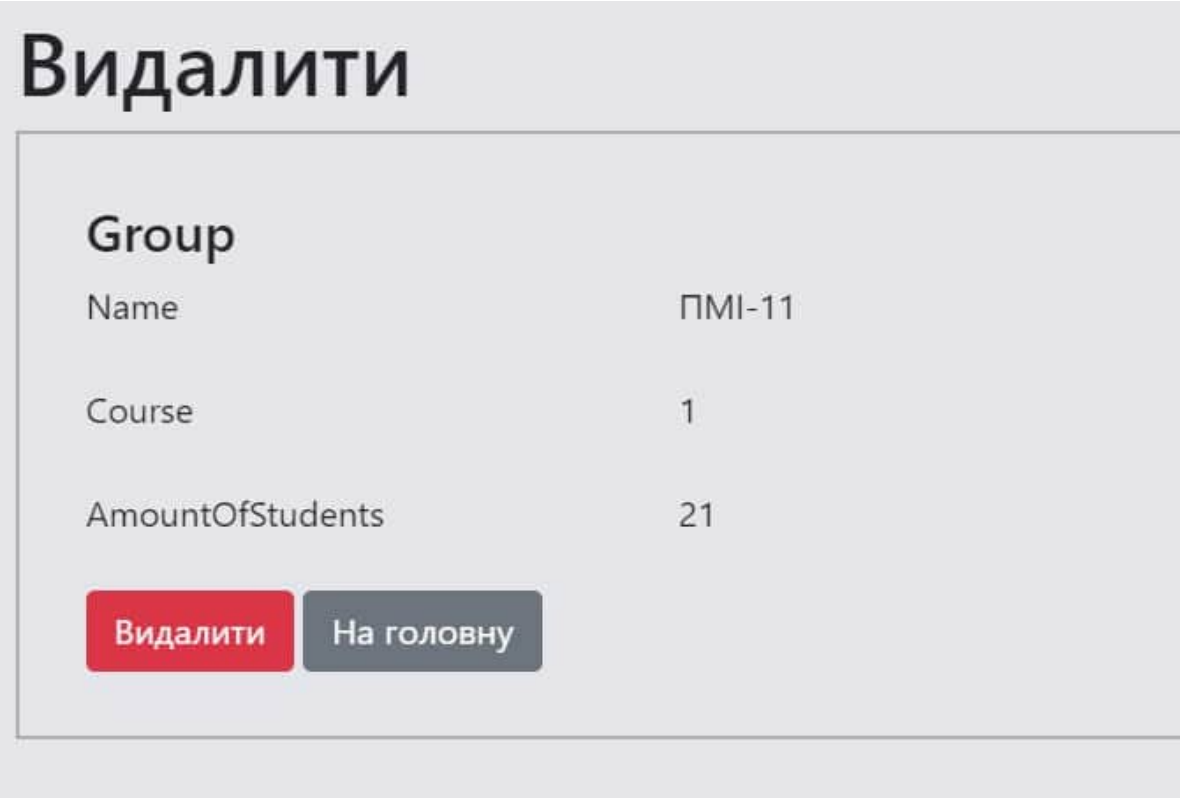


Рис 3.9 Приклад форми видалення групи

Далі сторінка з Навчальними програмами, де ми можемо створювати, редагувати, наповнювати та видаляти програми

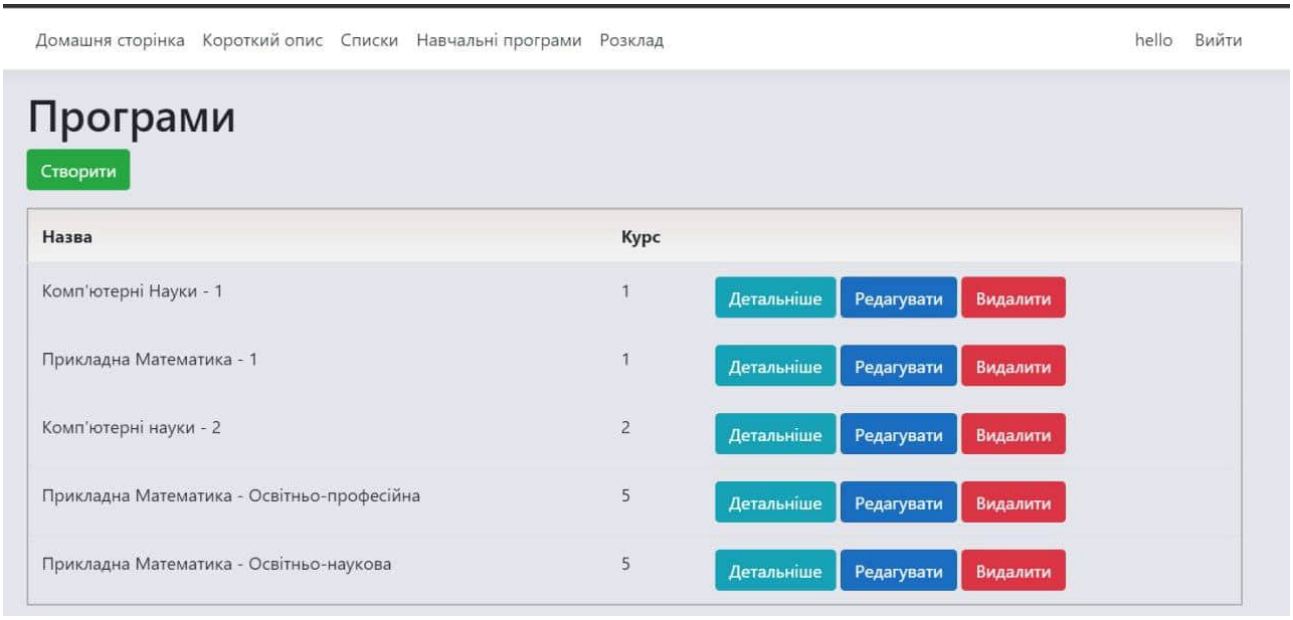


Рис 3.10 Сторінка навчальних програм

Основним методом тут є вкладка “Детальніше”. В ній ми додаємо предмети до навчального курсу та групи, у яких дані предмети будуть проводитись.

Додати ми можемо тільки ті предмети і групи студентів, в яких курс співпадає з відповідним курсом Навчальної програми

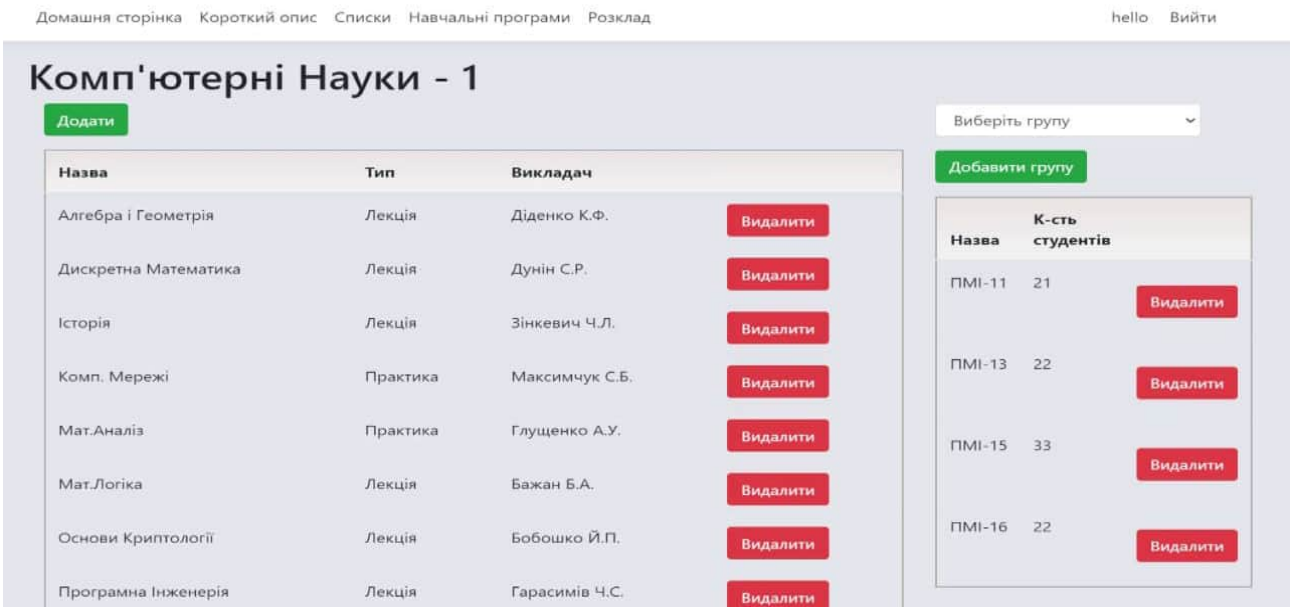


Рис 3.11 Сторінка редагування програми

Після заповненої бази та створених навчальних програм, можемо перейти до заповнення розкладу.

Для початку ми повинні вибрати бажаний курс, для якого буде створюватись розклад, після чого в списку залишаться тільки ті групи, які ми попередньо додали то даного курсу.

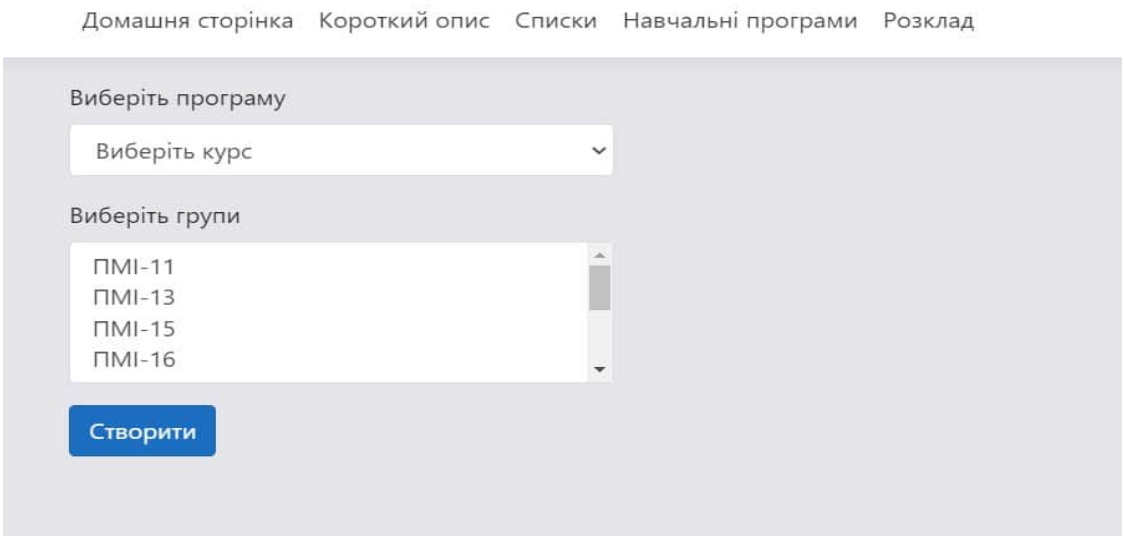


Рис 3.12 Сторінка вибору навчальної програми та груп для заповнення розкладу

Після того як ми вибрали програму та групи, нас переносить на сторінку для заповнення розкладу.

Рис 3.13 Сторінка з розкладом

При натисканні на знак <+> переходимо на сторінку створення пари для вибраної групи

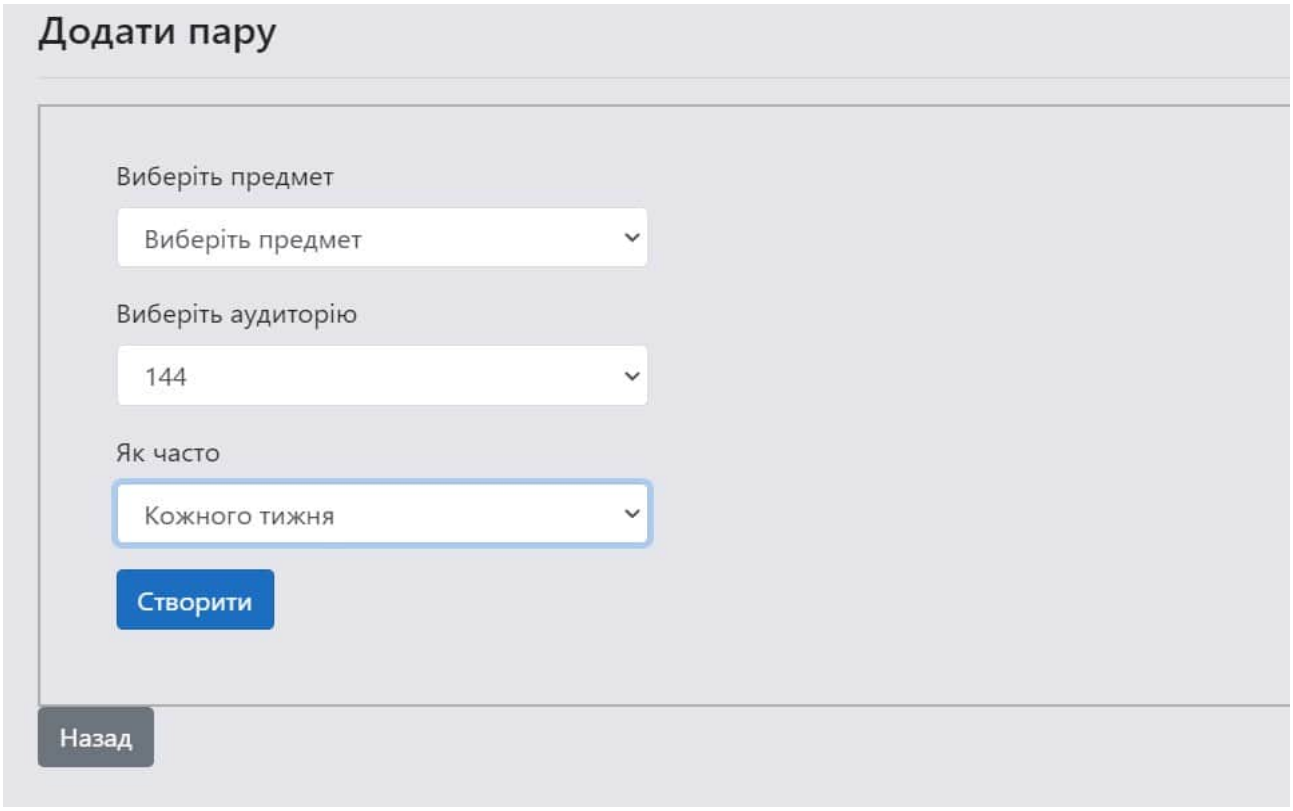


Рис 3.14 Сторінка створення пари

Рис 3.15 Фрагмент заповненого розкладу

Тут подано приклад конфліктів. У першому випадку через те, що у першої групи тип заняття практика, а у другої лекція, і через те, що пари проводяться в різних приміщеннях.

Рис 3.16 Приклад Конфлікту

Ще один вид конфлікту може бути у випадку, коли кількість студентів у групі перевищує вмістимість аудиторії. [3]

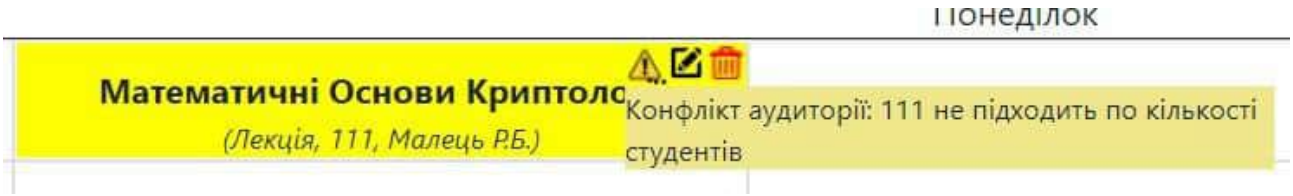


Рис 3.17 Приклад Конфлікту розміру групи та вмістимості аудиторії

Коли розклад буде заповнено, у вас буде можливість викачати його як csv файл, який після цього можна відкрити у програмі Excel для особистого перегляду/редагування.

На цьому моменті є декілька важливих етапів, адже просто так форматований CSV файл відкрити в Excel не вийде. Оскільки запис даних відбувається з особливим форматуванням, тому для правильного показу потрібно виконати декілька кроків, які будуть описані нижче

1. Натискаємо на кнопку <Експортувати в CSV>

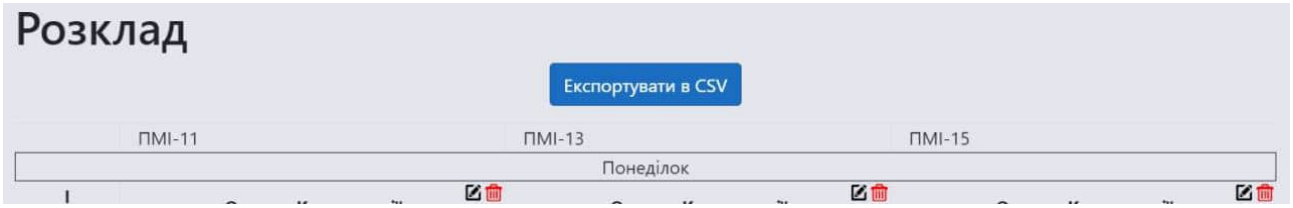


Рис 3.18 Інструкція з виводу файлу 1

2. Далі у нас відбудеться скачування файлу

Рис 3.19 Інструкція з виводу файлу 2

3. Створюємо новий документ

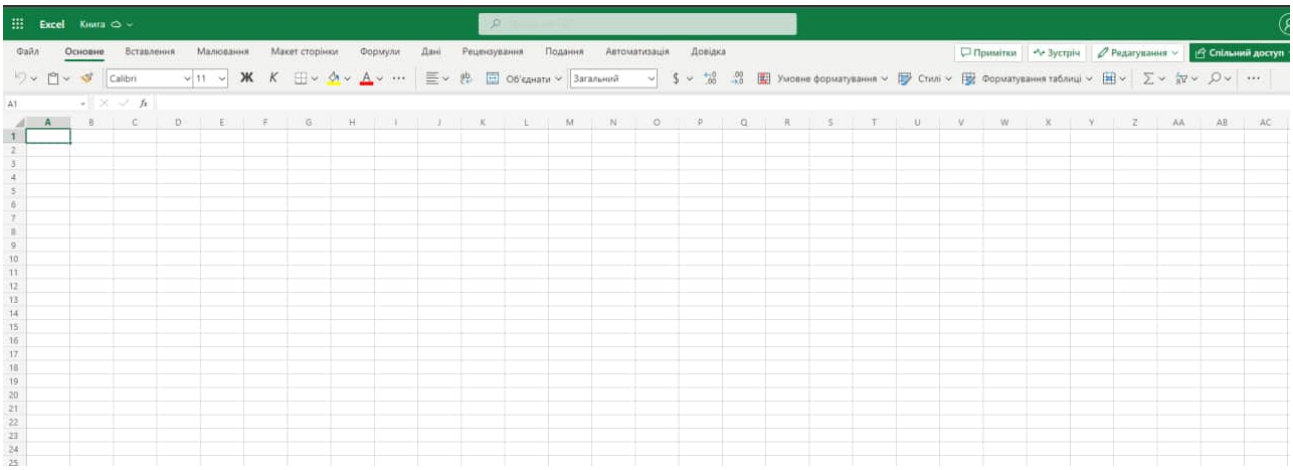


Рис 3.20 Інструкція з виводу файлу 3

4. Переходимо на вкладку <Дані>, та вибираємо <З тексту>

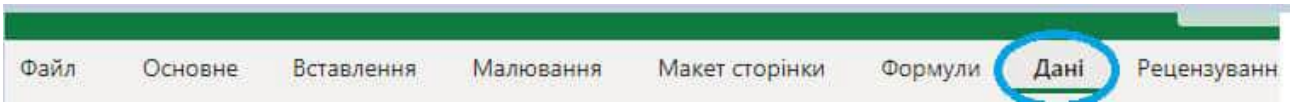


Рис 3.21 Інструкція з виводу файлу 4

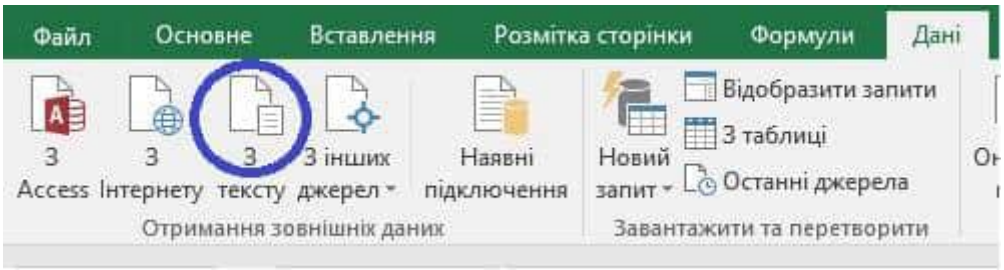


Рис 3.22 Інструкція з виводу файлу 5

5. Далі вибираємо завантажений файл

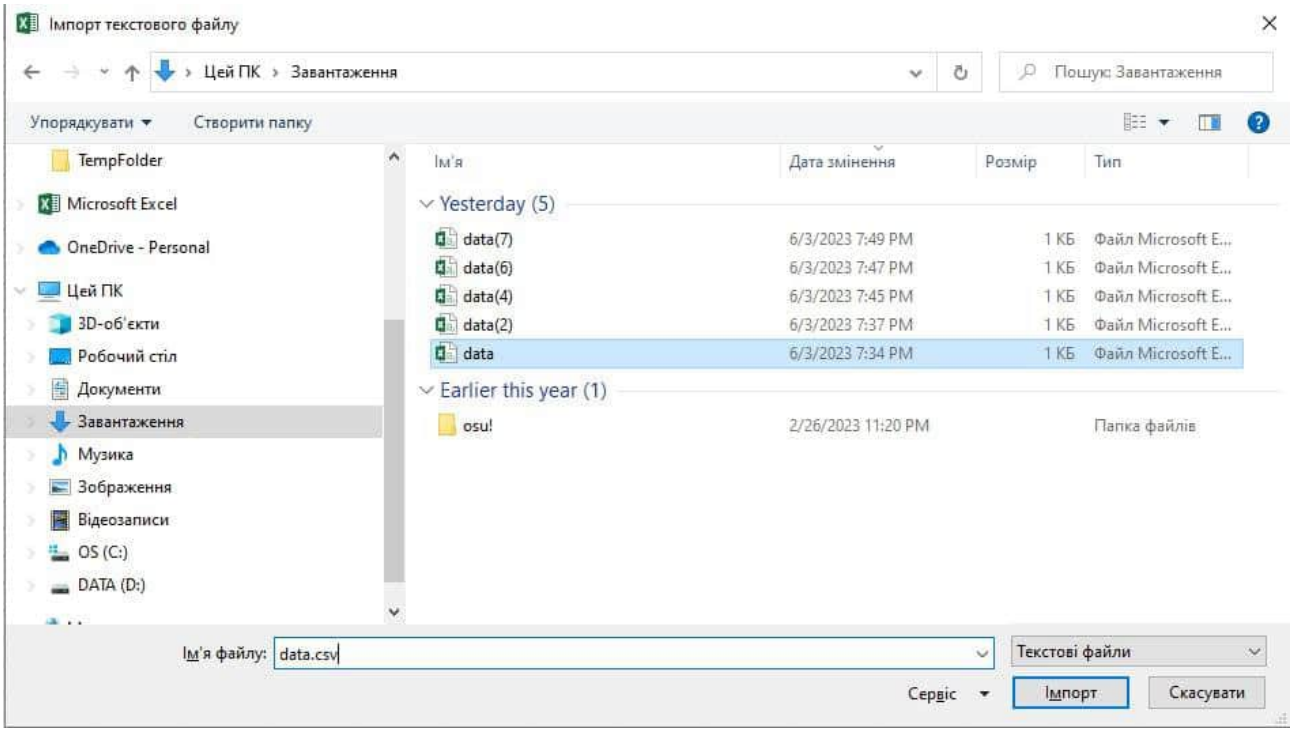


Рис 3.23 Інструкція з виводу файлу 6

6. В Майстрі імпорту тексту вибираємо пункт <З роздільниками> та переходимо далі

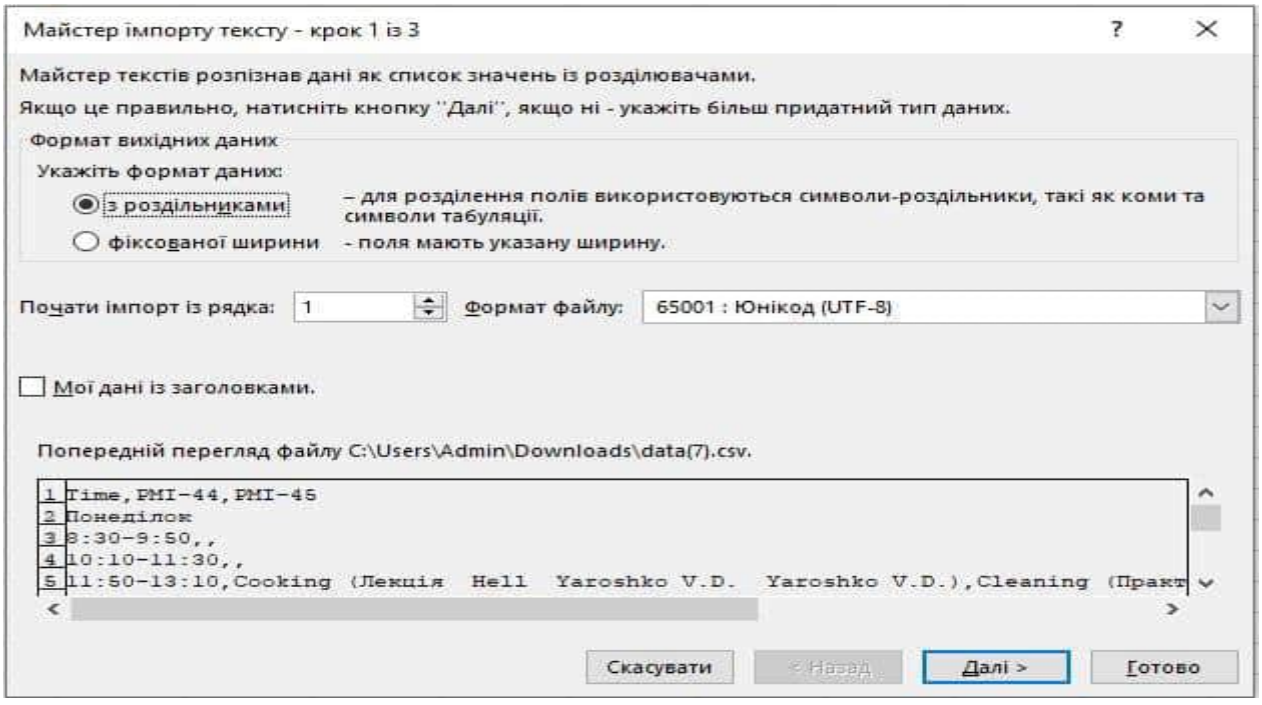


Рис 3.24 Інструкція з виводу файлу 7

7. На кроці 2 вибираємо роздільники <Кома>, та натискаємо <Готово>

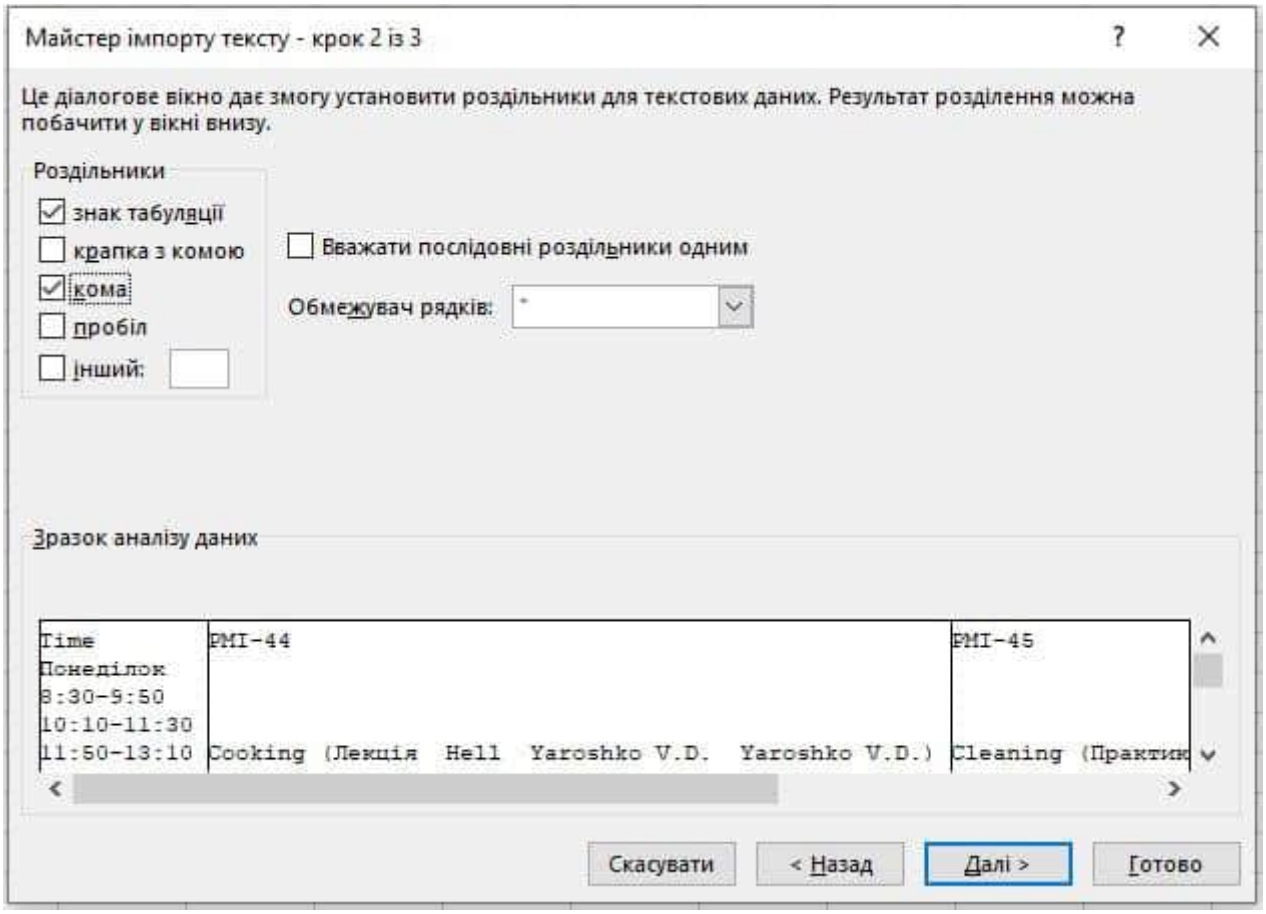


Рис 3.25 Інструкція з виводу файлу 8

8. Після цього у спиваючому вікні вибираємо місце, куди вставити дані, або просто натискаємо <OK>

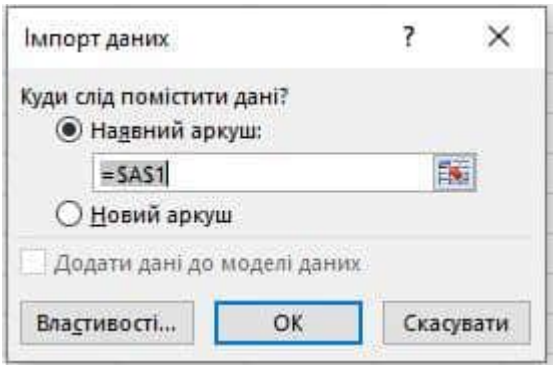


Рис 3.26 Інструкція з виводу файлу 9

9. Після чого отримуємо готовий результат

	A	B	C	D
1		ПМІ-11	ПМІ-13	ПМІ-15
2	Понеділок			
3	8:30-9:50	Основи Криптології (Лекція 111 Бобошко Й.П.)	Основи Криптології (Лекція 111 Бобошко Й.П.)	Основи Криптології (Лекція 111 Бобошко Й.П.)
4	10:10-11:30	Комп. Мережі (Практика 143 Максимчук С.Б. Фарина Ч.І.)	Програмування (Практика 144 Ярошко А.О.)	Мат.Аналіз (Практика 444 Глущенко А.У.)
5	11:50-13:10			Програмування (Практика 144 Ярошко А.О.)
6	13:30-14:50			
7	15:05-16:25			
8	16:40-18:00			
9	18:10-19:30			
10	19:40-21:00			
11				
12	Вівторок			
13	8:30-9:50	Програмування (Практика 144 Ярошко А.О.)	Мат.Аналіз (Практика 143 Глущенко А.У.)	Комп. Мережі (Практика 444 Максимчук С.Б. Фарина Ч.І.)
14	10:10-11:30			
15	11:50-13:10			
16	13:30-14:50			
17	15:05-16:25			
18	16:40-18:00			
19	18:10-19:30			
20	19:40-21:00			
21				
22	Середа			
23	8:30-9:50			
24	10:10-11:30			
25	11:50-13:10			
26	13:30-14:50			
27	15:05-16:25			
28	16:40-18:00			
29	18:10-19:30			
30	19:40-21:00			

Рис 3.27 Інструкція з виводу файлу 10

ВИСНОВКИ

В процесі виконання даної роботи було досліджено область складання розкладів та розроблено веб-застосунок формування розкладу для навчального закладу. Було розібрано проблематику створення розкладу і запропоновані варіанти для її вирішення та практичне представлення цих рішень у готовому веб-застосунку.

Дана дипломна робота була виконана у вигляді веб-застосунку, для реалізації було обрано середовище програмування MS VS 2019 та об’єктно орієнтована мова C#, база даних була реалізована завдяки MS SQL Server, і в сукупності правильна архітектура програми дає користувачу можливість зручно складати навчальний процес.

У випадку, коли в нас виявляється той чи інший вид конфлікту, програма дає всі можливі та необхідні методи для вирішення даної проблеми, а отже і економить величезну кількість часу для адміністраторів застосунку, який вони могли б потратити, при складанні розкладу вручну.

Завдяки розподілу користувачів по ролям за допомогою реєстрації, з’явився зручний механізм контролю та розподілу керування можливостями системи.

Також є можливість виводу розкладу у форматі CSV для його локального збереження та редагування.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1) Shostak, K. Yanovskaya and S. Rossokha, "Avtomatizatsiya protsessa sostavleniya raspisaniya zanyatiy na osnove tenzornogo ischisleniya v uchebnom komplekse" no. 9, S. 263–266, 2012.

http://nbuv.gov.ua/UJRN/aktit_2012_9_50

2) https://journals.onaft.edu.ua/index.php/atbp/article/view/1370/1623

3) Ruban, S. Dudenko, Yu. Busyhin, M. Kolmykov, O. Trublin, "Analiz suchasnoho prohramnoho zabezpechennya dlya avtomatyzatsiyi protsesu skladannya rozkladu navchal’nykh zanyat’", Systemy obrobky informatsiyi, vol. 115, no. 8, S. 305-310, 2013.

4) https://docs.microsoft.com/en-us/dotnet/

5) https://docs.microsoft.com/en-us/ef/

6) https://www.youtube.com/watch?v=C5cnZ-gZy2I&t=9972s

7) https://cyberleninka.ru/article/n/programna-sistema-formuvannya-rozkladu-zanyat-u-zakladi-vischoyi-osviti/viewer