ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО–НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра інженерії програмного забезпечення

Пояснювальна записка

до бакалаврської роботи
на ступінь вищої освіти бакалавр

на тему: «**РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ПЛАНУВАННЯ ТА ОБЛІКУ НАУКОВОЇ РОБОТИ КАФЕДРИ ЗАСОБАМИ MySql, C#**»

Виконав: студент 4 курсу, групи ПД–43
спеціальності
121 Інженерія програмного забезпечення
(шифр і назва спеціальності/спеціалізації)

Данилін С. О.
(прізвище та ініціали)

Керівник: Садовенко В.С.
(прізвище та ініціали)

Рецензент _____
(прізвище та ініціали)

Київ –2023

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра Інженерії програмного забезпечення
Ступінь вищої освіти -«Бакалавр»
Спеціальність підготовки – 121 «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного забезпечення
_____Негоденко О.В.
"____" _____2023 року

З А В Д А Н Н Я
НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТА

ДАНИЛІНА СЕРГІЯ ОЛЕКСАНДРОВИЧА

(прізвище, ім'я, по батькові)

1. Тема роботи: «Розробка програмного забезпечення для планування та обліку наукової роботи кафедри засобами C#, MySql»

Керівник роботи: Садовенко В.С. к.ф.-м.н., доцент кафедри ІПЗ.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом вищого навчального закладу від «24» лютого 2023 року №26
2. Строк подання студентом роботи «1» червня 2023 року
3. Вхідні дані до роботи
   3.1 Інструменти для розробки: Visual Studio 2022, SDK .NET 5, ASP.NET MVC
   3.2 Методи створення програмного продукту;
   3.3 Методи обробки та зберігання даних

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити).
   4.1Вимоги та оцінка якості системи.

4.2Опис проектування системи.

4.3Опис використаних технологій.

4.4Висновки

5. Перелік демонстраційного матеріалу

5.1 Титульний слайд

5.2 Мета, об’єкт, та предмет дослідження

5.3 Задачі дипломної роботи

5.4 Аналіз аналогів

5.5 Вимоги до програмного забезпечення

5.6 Програмні засоби реалізації

5.7 Діграма класів

5.8 Схема бази даних

5.9 Екранні форми

6. Висновки

6.1 Дата видачі завдання «25» лютого 2023

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської роботи	Строк виконання етапів роботи	Примітка
1	Вибір теми проекту	25.02.2023	
2	Постановка задачі	28.02.2023	
3	Розробка візуальної частини	10.03.2023	
4	Моделювання та розробка бази даних	15.03.2023	
5	Розробка функціональної частини	25.03.2023	
6	Висновки, оформлення роботи	05.04.2023	
7	Розробка демонстраційних матеріалів	20.04.2023	
8	Попередній захист роботи	24.05.2023	
9	Здача роботи	01.06.2023	

Студент _____ Данилін С.О.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Садовенко В.С.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Текстова частина бакалаврської роботи: 47 сторінок, 11 малюнків, 10 джерел.

НАУКОВА РОБОТА КАФЕДРИ, ASP.NET, MVC, КЛІЄНТ-СЕРВЕРНА АРХІТЕКТУРА, ВЕБ ДОДАТОК

Об'єктом дослідження є процес планування та обліку наукової роботи кафедри.

Предметом дослідження є розробка програмного забезпечення для планування та обліку наукової роботи кафедри на мові C#, з використанням бази даних MySql.

Мета дослідження полягає в розробці програмного забезпечення для планування та обліку наукової роботи кафедри, яке дозволить ефективно вести облік наукових робіт, планувати їх виконання та відстежувати результати.

Описано архітектуру та основні принципи розробки. Додаток розроблений на платформі .NET з використанням мови C# для серверу і HTML & CSS та JavaScript для клієнтської частини.

В якості бази даних було взято MySql та бібліотеку Entity Framework Core для взаємодії з нею.

Отже, розроблено та описано веб-додаток, завданням якого є полегшення процесу керування науковими роботами а саме конкурсами наукових робіт студентів, підвищення ефективності роботи кафедри.

Даний додаток може бути використано будь-якими закладами, які виконують наукові роботи.

Галузь використання – академічна сфера.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

MySql – система управління базами даних.

JSON (JavaScript Object Notation) - це текстовий формат обміну даними між комп'ютерами.

HTTP (HyperText Transfer Protocol) - це протокол передачі гіпертексту.

URL (Uniform Resource Locator) - єдиний вказівник на ресурс, стандартизована адреса певного ресурсу.

API (Application Programming Interface) - це набір визначень підпрограм, протоколів взаємодії та засобів для створення програмного забезпечення.

GET запит - метод передавання даних від клієнта до сервера з метою отримання інформації, зазначеної за допомогою конкретних параметрів.

POST запит – метод для надсилання сутностей до певного ресурсу.

SMTP (Simple Mail Transfer Protocol) - це комунікаційний протокол для пересилання електронної пошти.

TLS (Transport Layer Security) - криптографічний протокол, що надає можливості безпечної передачі даних в інтернеті для навігації, отримання пошти, спілкування, обміну файлами, тощо.

POP3 (Post Office Protocol) - це протокол, що використовується клієнтом для доступу до повідомлень електронної пошти на сервері.

IMAP (Internet Message Access Protocol) - мережевий протокол прикладного рівня для доступу до електронної пошти.

Вступ

Розробка програмного забезпечення для планування та обліку наукової роботи кафедри є важливою темою в сучасному вищому навчальному закладі. Все більшою мірою наукова робота кафедри стає однією з найважливіших складових діяльності вищого навчального закладу, що відображається на рейтингу закладу та професійній репутації викладачів. Тому, зручне та ефективне програмне забезпечення для планування та обліку наукової роботи кафедри, є надзвичайно важливим інструментом для управління науково-дослідною роботою.

Обґрунтування вибору теми та її актуальність: Згідно зі статистикою, більшість вищих навчальних закладів мають кафедри, які займаються науковою роботою. Однак, не завжди наукова робота кафедр належним чином організована та управляється. Це може призвести до нездійснення дослідницьких планів, недооцінки результатів наукових робіт та, в кінцевому рахунку, до зниження рівня професійної репутації викладачів та рейтингу вищого навчального закладу в цілому. Тому, розробка програмного забезпечення для планування та обліку наукової роботи кафедри є актуальною темою в наш час.

Об'єктом дослідження є процес планування та обліку наукової роботи кафедри.

Предметом дослідження є розробка програмного забезпечення для планування та обліку наукової роботи кафедри на мові C#, з використанням бази даних MySql.

Мета дослідження полягає в розробці програмного забезпечення для планування та обліку наукової роботи кафедри, яке дозволить ефективно вести облік наукових робіт, планувати їх виконання та відстежувати результати.

Для досягнення мети необхідно розв'язати наступні завдання:

- розробити структуру бази даних та визначити необхідні таблиці;
- розробити інтерфейс користувача та забезпечити його зручність;
- реалізувати можливість планування і обліку наукових робіт;
- реалізувати можливість формування і відправлення звітів про виконання

наукових робіт.

Наукова новизна дослідження полягає в розробці програмного забезпечення для планування та обліку наукової роботи кафедри, що є новим та оригінальним рішенням в даній галузі.

Практична значущість результатів дослідження полягає в можливості ефективно вести облік наукових робіт, планувати їх виконання, що дозволить краще контролювати наукову діяльність кафедри та досягати поставлених наукових цілей.

1.ПРОЕКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1 Вимоги до програмного забезпечення

Функціональні вимоги:

– автентифікація користувачів: Система повинна забезпечувати можливість створення облікових записів користувачів, а також автентифікацію за допомогою логіна та пароля;

– інтерфейс користувача: Система повинна мати інтуїтивно зрозумілий та дружній інтерфейс користувача, що дозволить легко навігувати, виконувати операції та взаємодіяти з системою;

– адміністративна панель: Система повинна мати інтерфейс для керування системою або програмою з боку завідувача кафери;

– планування наукових робіт: Система повинна надавати можливість планувати наукові роботи, включаючи створення та видалення проектів, призначення термінів, встановлення критеріїв оцінювання;

– облік наукових робіт;

– звітність: Система повинна надавати можливість звітування у форматі файлу для подальшої відправки.

Нефункціональні вимоги:

– використання мови програмування C#: Система повинна бути розроблена з використанням мови програмування C# для реалізації логіки програми, інтерфейсів користувача та інших компонентів системи;

– використання бази даних MySQL: Система повинна використовувати базу даних MySQL для зберігання і керування даними про наукові проекти, користувачів та другу інформацію;

– сумісність: Система повинна бути сумісною з різними веб-браузерами для забезпечення зручного доступу користувачів;

– тестування: Система повинна пройти модульні тести, для перевірки

функціональності, стабільності та відповідності вимогам.

Загальною метою розробки програмного забезпечення для планування та обліку наукової роботи кафедри є полегшення процесу керування науковими роботами а саме конкурсами наукових робіт студентів, підвищення ефективності роботи кафедри, забезпечення зручного доступу до інформації та покращення спільної роботи команди. Виконання вищезгаданих вимог допоможе досягти цих цілей і створити потужну та функціональну систему.

1.2 Характеристика використаних технологій для проекування і розробки ПЗ

ASP.NET Core MVC - це популярний фреймворк для розробки веб-додатків, який використовує архітектуру “Model-View-Controller” (MVC). Давайте розглянемо структуру та складові цієї архітектури:

Модель (Model):

– модель представляє бізнес-логіку та дані додатку. Вона включає в себе класи, які описують структуру та властивості даних, а також проводить валідацію даних.;

Представлення (View):

– представлення відповідає за відображення даних користувачеві. Це можуть бути HTML-сторінки, шаблони, вигляди даних або інші елементи веб-інтерфейсу;

– представлення отримують дані з моделі та відображають їх відповідно до вимог дизайну. Вони можуть також обробляти користувацький ввід.

Контролер (Controller):

– контролер приймає запити від користувача, обробляє їх та взаємодіє з моделлю та представленням. Він містить методи, які відповідають за обробку різних дій користувача, таких як створення, зчитування, оновлення та видалення даних;

– контролер також відповідає за передачу даних з моделі до представлення та обробку результатів користувацького вводу.

ASP.NET Core MVC забезпечує розділення відповідальностей між цими компонентами, що сприяє модульності та розширюваності додатку. Кожен компонент виконує свою роль у веб-додатку та взаємодіє з іншими компонентами для забезпечення повнофункціонального та ефективного виконання завдань. Розділення на модель, представлення та контролер дозволяє розробникам працювати над окремими частинами додатку незалежно один від одного. Це спрощує розробку, тестування та збереження коду.

ASP.NET Core MVC також підтримує інтеграцію з іншими фронтенд технологіями, такими як Angular, React або Vue.js, що дозволяє розробникам створювати сучасні односторінкові додатки (Single-Page Applications, SPA). За допомогою API-контролерів, можна створювати веб-сервіси для взаємодії з клієнтськими додатками на різних платформах.

А також надає багато вбудованих функцій та можливостей, які спрощують розробку веб-додатків.

Деякі з них включають:

– маршрутизація (Routing): ASP.NET Core MVC надає гнучку систему маршрутизації, яка дозволяє визначати, який контролер та дія повинні обробляти кожний запит. Це дозволяє розподілити обробку запитів між різними контролерами та діями на основі URL-шляхів;

– модель зв'язування (Model Binding): Механізм моделі зв'язування дозволяє автоматично зв'язувати дані з HTTP-запиту з параметрами методів контролера або моделями. Це спрощує отримання та обробку даних з форм, запитів POST або параметрів URL;

– пакети перегляду (View Packages): ASP.NET Core MVC підтримує використання пакетів перегляду, таких як Razor для генерації динамічного HTML-коду. Це дозволяє розробникам швидко створювати розширювані та зручні вьюшки, які можуть використовувати логіку та дані з моделей;

– підтримка залежностей (Dependency Injection): ASP.NET Core MVC надає вбудовану підтримку для контейнера ін’єкції залежностей. Це дозволяє зручно впроваджувати та керувати залежностями між компонентами додатку, полегшуючи

тестування та розширення.

В цілому, завдяки своїй архітектурі та вбудованим функціям, ASP.NET Core MVC є потужним інструментом для розробки веб-додатків з підтримкою модульності, розширюваності та ефективної організації коду. Він надає розробникам гнучкість та контроль над процесом розробки, дозволяючи створювати високоякісні та масштабовані програми.

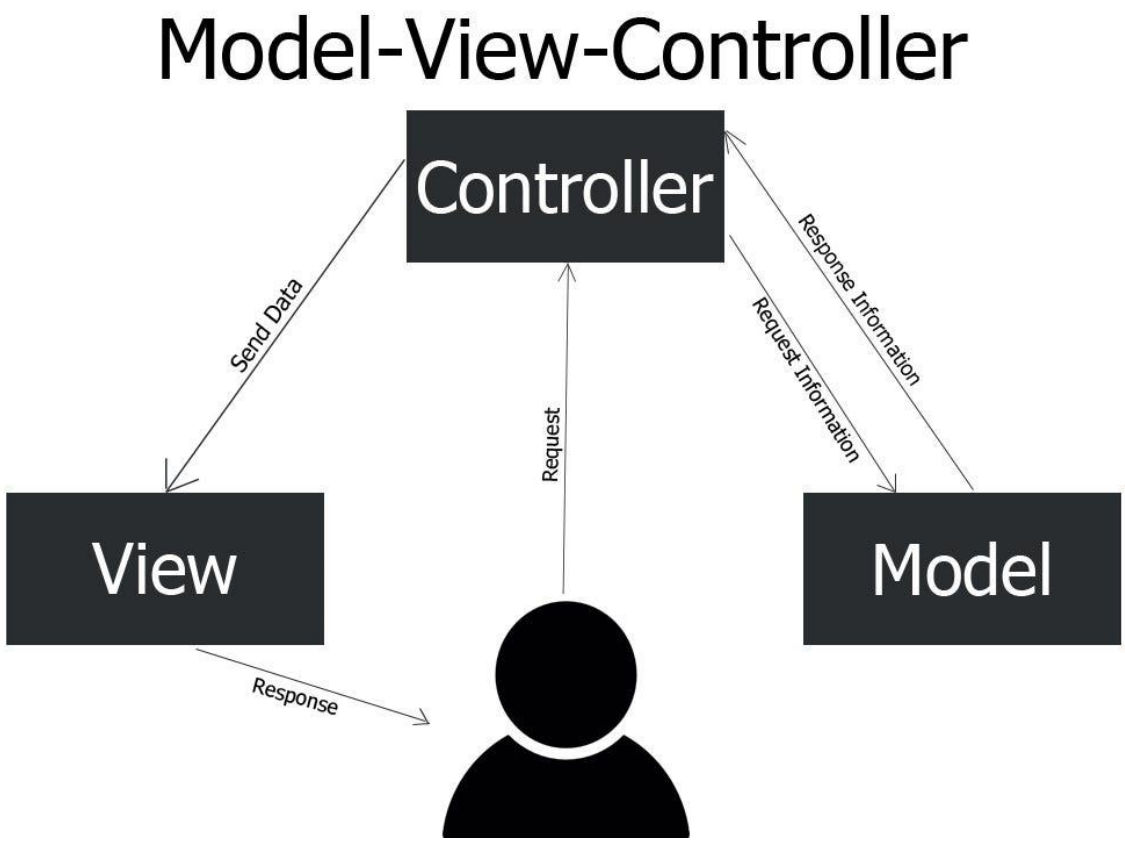


Рис 1.1. – Модель взаємодії з додатком MVC

MySQL – це популярна відкрита реляційна система управління базами даних (СУБД). Вона є однією з найбільш використовуваних СУБД у світі, особливо веб-додатками та іншими проектами, що вимагають надійного та ефективного збереження та доступу до даних.

Основні риси та переваги MySQL включають:

– надійність та швидкодію: MySQL відомий своєю стабільністю та високою швидкодією. Він добре справляється з великими обсягами даних та високими навантаженнями, що робить його популярним в різних сферах;

– Масштабованість: MySQL може працювати зі зростанням обсягів даних та кількості користувачів. Він підтримує різні режими розподіленої реплікації, кластеризацію та шардування для підвищення масштабованості додатків;

– гнучкість: MySQL підтримує багато мов та платформ, що дозволяє легко інтегрувати його з різними технологіями та розроблювати багатоплатформові додатки;

– безпека: MySQL надає широкі можливості для забезпечення безпеки даних. Це включає в себе механізми автентифікації користувачів, контроль доступу до бази даних, шифрування даних та захист від SQL-ін’єкцій та інших атак;

– відкритий код та спільнота: MySQL є відкритим програмним забезпеченням, що означає, що його вихідний код вільно доступний для перегляду, змін та розповсюдження. Це надає розробникам можливість адаптувати та налаштовувати MySQL під свої потреби.

Загально кажучи, використання MySQL дозволяє забезпечити надійне та ефективне збереження, доступ та керування даними. Він пропонує багатий функціонал, широкі можливості масштабування та гнучкість інтеграції з іншими технологіями.

DATABASE MANAGEMENT SYSTEMS COMPARISON

	Database type	Licensing	Documentation	Scalability	Data types supported	Learning curve
MySQL	SQL	GNU Generally Public License	✓✓	Vertical, complex	Structured, semi-structured	Mild
Maria DB	SQL	GNU Generally Public License	✓✓✓	Vertical	Structured, semi-structured	Mild
Oracle	Multi-model, SQL	Proprietary	✓✓✓	Vertical	Structured, semi-structured, unstructured	Hard
PostgreSQL	Object-relational, SQL	Open-source	✓✓	Vertical	Structured, semi-structured, unstructured	Hard
MSSQL Server	T-SQL	Proprietary	✓✓✓	Vertical, complex	Structured, semi-structured, unstructured	Hard
MongoDB	NoSQL, document-oriented	SSPL	✓✓✓	Horizontal	Structured, semi-structured, unstructured	Mild
Redis	NoSQL, key-value	Open-source, BSD 3-clause	✓✓✓	Horizontal	Structured, semi-structured, unstructured	Mild
Cassandra	NoSQL, wide-column	Open-source	✓✓✓	Horizontal	Structured, semi-structured, unstructured	Hard
Elasticsearch	NoSQL, document-oriented	Open-source	✓✓	Horizontal	Structured, semi-structured, unstructured	Hard
Firebase	NoSQL, real-time database	Open-source	✓✓✓	Horizontal	Structured, semi-structured, unstructured	Mild

Рис 1.2. – Коротка таблиця порівняльних характеристик СУБД

Entity Framework Core (EF Core) - це сучасний інструмент для проекування та розробки програмного забезпечення. Ось деякі характеристики, які роблять EF Core популярним у розробників:

– об'єктно-реляційне відображення (Object-Relational Mapping, ORM): EF Core забезпечує зручний мапінг між об'єктно-орієнтованою моделлю даних у додатку і реляційною базою даних. Це дозволяє розробникам працювати з об'єктами та класами, а EF Core автоматично виконує необхідні операції з базою даних за допомогою LINQ у вигляді LINQ to Entities;

– міграції бази даних: EF Core надає механізм міграцій, що дозволяє розробникам легко змінювати схему бази даних з підтримкою версіонування. Це полегшує розгортання та оновлення додатків, не втрачаючи існуючі дані.

– LINQ (Language Integrated Query): EF Core підтримує LINQ, що дозволяє розробникам писати мовою запитів, інтегрованою в мову програмування C#. Це дозволяє зручно взаємодіяти з даними, виконувати фільтрацію, сортування та інші операції над об'єктами в базі даних;

– підтримка різних баз даних: EF Core підтримує різні провайдери баз даних, включаючи MySQL, SQL Server, PostgreSQL і багато інших. Це дозволяє розробникам використовувати EF Core з базою даних, яка найкраще відповідає їх потребам.

Використання Entity Framework Core дозволяє ефективно працювати з базою даних, забезпечувати стабільність та безпеку даних, а також спрощує розробку та підтримку додатку. Він є потужним інструментом, який допомагає розробникам швидко і надійно взаємодіяти з базою даних та забезпечувати потрібну функціональність.

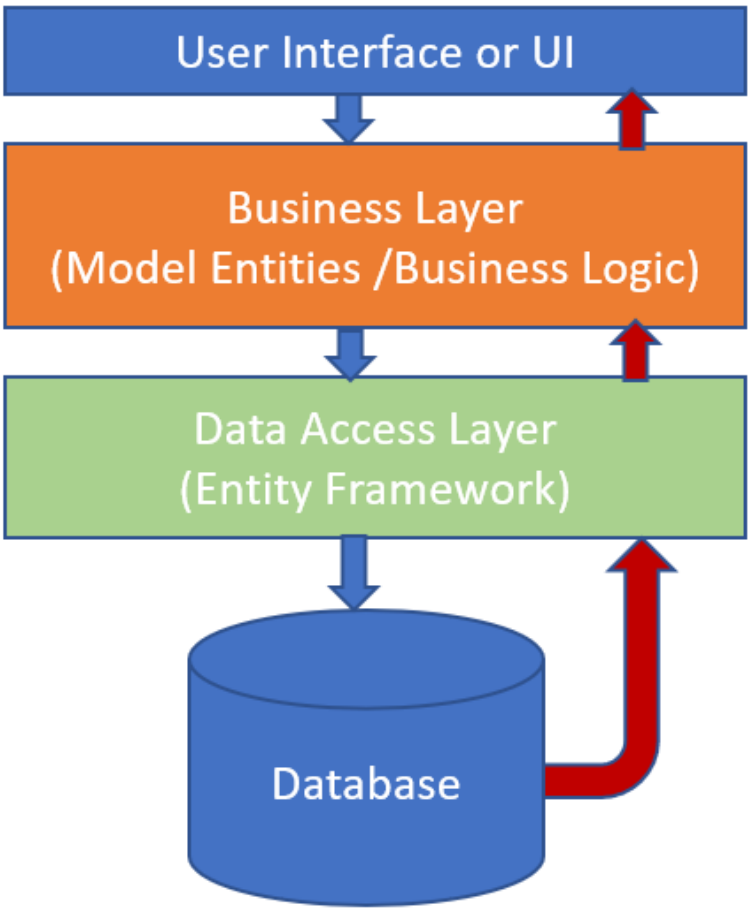


Рис 1.3. – Схема роботи Entity Framework

Gmail API - широко використовується для розширення функціональності, пов'язаної з електронною поштою. Вона надає можливість взаємодіяти зі скриньками Gmail, зчитувати та відправляти електронні листи, керувати мітками, контактами та вкладеннями. Ось деякі характеристики, які роблять Gmail API популярним у розробників:

– отримання електронних листів: За допомогою Gmail API, можливо отримувати електронні листи зі скриньок Gmail користувачів. Це дозволяє додатку аналізувати та обробляти інформацію з листів, таку як заголовок, вміст, відправник, дата тощо;

– відправлення електронних листів: За допомогою Gmail API, можливо відправляти електронні листи від імені користувачів. Це дозволяє додатку надсилати повідомлення з важливою інформацією, сповіщеннями або зворотними зв'язками користувачам;

– інтеграція контактів: За допомогою Gmail API, використовують отримувати доступ до контактів користувачів зі скриньок Gmail. Це дозволяє додатку інтегрувати контакти в систему та використовувати їх для різних цілей, наприклад, відправлення повідомлень або збагачення профілів користувачів.

– робота з вкладеннями: Gmail API використовують для зчитування та завантаження вкладень (наприклад, документів, зображень чи інших файлів), що додаються до електронних листів. Це дозволяє користувачам зберігати та керувати вкладеннями безпосередньо в додатку;

– аутентифікація та авторизація: Gmail API надає механізми для аутентифікації та авторизації користувачів. Це забезпечує безпеку та захист даних користувачів, дозволяючи додатку отримати доступ лише до необхідних облікових записів Gmail.

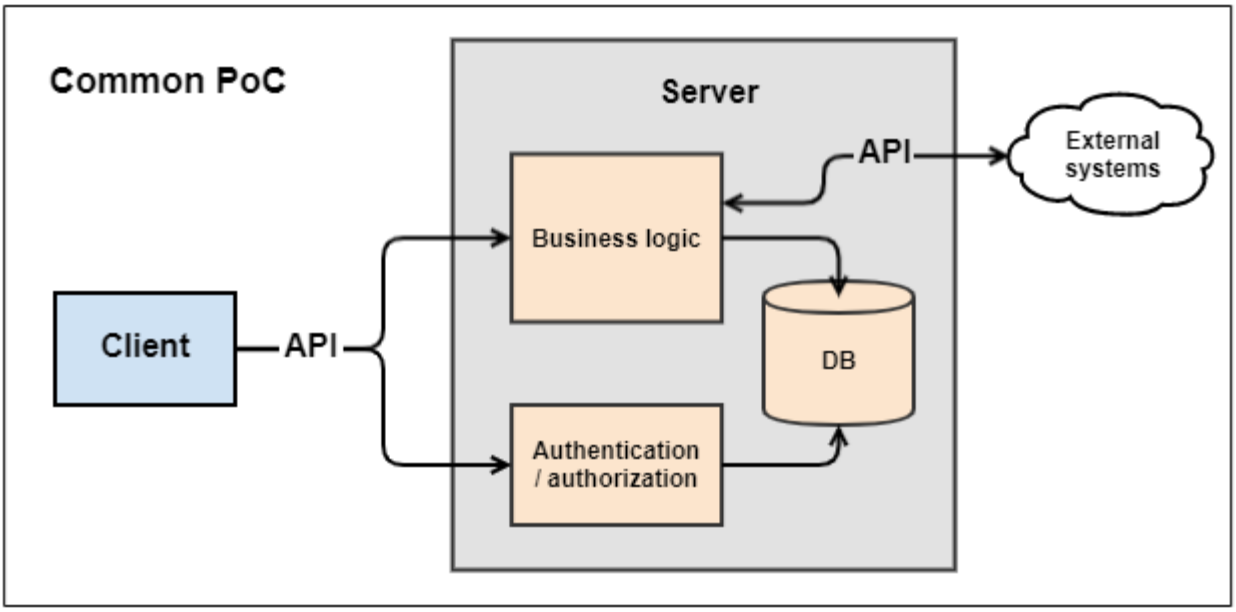


Рис 1.4. – Схема роботи автентифікації Gmail API

Particles.js - це потужна бібліотека JavaScript, яка дозволяє реалізувати захоплюючі анімаційні ефекти на веб-сторінках за допомогою частинок. Вона використовується для створення рухомих частинок, які можуть взаємодіяти між собою та з користувачем.

Основним елементом у Particle.js є HTML елемент canvas, який використовується для візуалізації частинок. Цей елемент надає можливість

створювати і керувати тисячами частинок на екрані. Частинки можуть мати різні розміри, кольори, форми і швидкості руху, що дозволяє створювати різноманітні візуальні ефекти.

Бібліотека Particle.js надає широкий спектр налаштувань, що дозволяють контролювати поведінку і зовнішній вигляд частинок. Можна налаштувати їх розташування, величину, швидкість, кут руху, кольори та інші параметри. Крім того, є можливість створювати власні анімаційні сценарії та взаємодіяти з частинками за допомогою JavaScript.

Particle.js використовується для створення захоплюючих фонових ефектів на веб-сторінках. Це можуть бути рухомі зірки, спалахи, водяні краплі, магічні частинки та багато інших візуальних ефектів. Це додає живості та динаміки до інтерфейсу сторінки, зроблюючи її більш привабливою та вражаючою для користувачів.

Завдяки великій гнучкості і налаштовуваності Particle.js, розробники можуть створювати унікальні та індивідуальні ефекти, що відповідають стилю та тематиці конкурсу наукових робіт студентів. Візуальні ефекти, створені за допомогою цієї бібліотеки, можуть покращити враження користувачів та надати особливу атмосферу веб-сайту конкурсу.

Крім того, Particle.js підтримується активною спільнотою розробників, що означає наявність постійних оновлень, виправлень помилок і нових функцій. Це дозволяє бути впевненим в стабільності та якості бібліотеки, а також отримати підтримку та допомогу у разі потреби.

Ось деякі характеристики Particle.js:

– легкість використання: Particle.js має простий та інтуїтивно зрозумілий API, що робить його доступним навіть для розробників з невеликим досвідом;

– конфігурованість: Бібліотека надає безліч параметрів налаштування, що дозволяють контролювати зовнішній вигляд та поведінку частинок. Ви можете налаштувати такі параметри, як кількість часток, швидкість руху, колір, розмір, форма та інші властивості;

– анімація: Particle.js дозволяє створювати плавні та красиві анімовані

ефекти частинок. Ви можете налаштувати ефекти руху, зміни розміру, швидкості та траєкторії частинок для створення бажаного візуального ефекту;

– крос-браузерна сумісність: Particle.js сумісний із різними веб-браузерами, включаючи Chrome, Firefox, Safari та інші популярні браузери. Це дозволяє створювати ефекти частинок, які працюють однаково добре на різних платформах;

– розширюваність: Бібліотека Particle.js надає гнучкі можливості для створення власних ефектів частинок та інтеграції з іншими бібліотеками чи фреймворками;

Узагалі, Particle.js є потужним інструментом для створення захоплюючих анімаційних ефектів на веб-сайтах конкурсу наукових робіт студентів. Використовуючи цю бібліотеку, розробники можуть створити вражаючий візуальний досвід та привернути увагу до важливих деталей та інформації, пов'язаної з конкурсом.

– Nunit - це популярний фреймворк для тестування. Основні особливості та переваги Nunit:

Простота використання: Nunit має простий та зрозумілий синтаксис, що дозволяє легко писати тести. Він підтримує атрибути, які дозволяють зазначати різноманітні умови для тестування, такі як очікувані результати, передумови та постумови;

Гнучкість: Nunit дозволяє гнучко налаштовувати тести та керувати їх виконанням. Він підтримує різні види тестів, такі як параметризовані тести, тести на виключення, ігнорування тестів та багато інших;

Розширюваність: Nunit надає можливість розширення функціональності за допомогою власних розширень та встановлення плагінів. Це дозволяє використовувати додаткові функції та інтегрувати Nunit з іншими інструментами тестування;

Підтримка різних мов та середовищ: Nunit підтримує різні мови програмування, включаючи C#, VB.NET, F# та інші. Він також може бути використаний в різних середовищах розробки, таких як Visual Studio, Xamarin, Unity та інші. Інтеграція з іншими інструментами: Nunit може бути легко

інтегрований з іншими інструментами розробки та засобами автоматизації, такими як CI/CD системи, засоби збирання коду та інші.

– HTML & CSS - Використання HTML та CSS дозволяє веб-розробникам створювати красиві, функціональні та привабливі веб-сторінки. HTML визначає структуру та семантику сторінки, вказуючи, які елементи є заголовками, абзацами, списками тощо. CSS використовується для визначення стилів, які впливають на зовнішній вигляд елементів, дозволяючи змінювати їх кольори, розміри, розташування та інші аспекти дизайну.

– C# - Є сучасною, об'єктно-орієнтованою мовою програмування, розробленою компанією Microsoft. Вона стала частиною платформи .NET і здатна працювати на різних платформах, включаючи Windows, macOS і Linux. C# має широке застосування в розробці веб-додатків, настільних програм, ігор, мобільних додатків та багатьох інших сферах. Основні особливості C# включають:

Об'єктно-орієнтованість: C# підтримує парадигму об'єктно-орієнтованого програмування, дозволяючи розробникам створювати класи, об'єкти, успадкування, поліморфізм та інші концепції ООП;

Строга типізація: C# є строго типізованою мовою, що означає, що типи змінних повинні бути визначені і зберігатись під час компіляції. Це сприяє виявленню помилок на етапі компіляції та покращує безпеку програм;

Управління пам'яттю: C# використовує автоматичне управління пам'яттю, що дозволяє автоматично відслідковувати та звільняти пам'ять, використовану об'єктами, що більше не потрібні. Це полегшує розробку програм та уникнення проблем, пов'язаних з утворенням сміття;

Широкі можливості бібліотек: Стандартна бібліотека .NET, яка включає багато компонентів і класів, дозволяє розробникам швидко та ефективно реалізовувати різноманітні функціональні можливості у своїх програмах;

Підтримка асинхронного програмування: C# надає потужні засоби для асинхронного програмування, що дозволяє створювати швидкодіючі та ефективні програми, які можуть обробляти багатопотокові завдання.

– Visual Studio - це інтегроване середовище розробки (IDE) для

програмування на різних мовах, таких як C#, C++, Visual Basic .NET, F#, JavaScript, Python та інші. Розроблене компанією Microsoft, Visual Studio надає розробникам широкий набір інструментів, сервісів і можливостей для зручної і продуктивної розробки програмного забезпечення. Особливості Visual Studio включають:

Інтегроване середовище розробки: Visual Studio надає однорідне інтерфейсне середовище для розробки програм. Воно включає редактор коду з функціями автодоповнення, форматування, перехресних посилань та іншими зручними інструментами для редагування коду. Крім того, Visual Studio має вбудовану підтримку для контролю версій, рефакторингу коду, налагодження програм та інших розробчих задач;

Підтримка різних мов програмування: Visual Studio дозволяє розробляти програми на різних мовах програмування, включаючи C#, C++, Visual Basic .NET, F#, JavaScript, Python та інші. Воно надає засоби для підсвічування синтаксису, перевірки помилок, автодоповнення та інших інструментів, специфічних для кожної мови;

Універсальність платформи .NET: Visual Studio є основним інструментом для розробки програм під платформу .NET. Воно надає доступ до широкого набору бібліотек, фреймворків і сервісів, що спрощують розробку різноманітних програм, включаючи веб-додатки, настільні програми, мобільні додатки, хмарні рішення та інші;

Розширюваність і плагіни: Visual Studio підтримує розширення та плагіни, що дозволяють розробникам розширити його функціональність та додати нові інструменти. Розширення доступні через магазин Visual Studio, де можна знайти різноманітні додатки, шаблони, компоненти та інші розширення;

Засоби для командної розробки: Visual Studio надає різні засоби для командної розробки, що дозволяють розробникам спільно працювати над проектами. Включаючи можливості контролю версій, спільного доступу до кодової бази, розподіленого відладження та інші інструменти спільної роботи;

Інструменти для тестування: Visual Studio надає вбудовані інструменти для тестування програм, включаючи модульні тести, тести одиниць, тести

продуктивності та інші. Вони допомагають забезпечити якість коду та виявляти помилки на ранніх етапах розробки.

Visual Studio також має інтеграцію з системою контролю версій Git. Ця функціональність дозволяє розробникам керувати своїми проектами та співпрацювати над ними із використанням Git, популярної системи контролю версій.

Інтеграція Git у Visual Studio дозволяє розробникам ефективно керувати версіями свого коду, відстежувати зміни, працювати в команді та використовувати всі можливості Git, не залишаючи середовище розробки.

Visual Studio надає різні інструменти для роботи з Git, включаючи можливість ініціалізації репозиторію Git, створення гілок, коммітів, злиття, вилучення та відправлення змін та перегляду історії змін. Всередині IDE також доступний візуальний інтерфейс для перегляду відмінностей між версіями файлів, вирішення конфліктів під час злиття та інших операцій, пов'язаних з Git.

Visual Studio є одним з найпопулярніших інструментів розробки програмного забезпечення та використовується розробниками по всьому світу. Воно дозволяє створювати високоякісне програмне забезпечення з підвищеною продуктивністю і забезпечує розробникам широкий спектр інструментів та сервісів для розробки різноманітних типів програм.

– Open XML SDK - це набір бібліотек і інструментів, розроблений компанією Microsoft, який дозволяє розробникам взаємодіяти з документами у форматі Office Open XML. Ця технологія надає зручні та потужні можливості для створення, зміни та читання документів у форматах DOCX, XLSX та PPTX;

Використовуючи Open XML SDK, розробники можуть програмно маніпулювати змістом документів Office, створювати нові документи або змінювати існуючі. Завдяки цьому можна автоматизувати рутинні завдання, такі як створення звітів, генерація документації, обробка даних та інше.

Open XML SDK надає розробникам доступ до структури документів Office Open XML, що дозволяє контролювати різноманітні елементи документа, такі як текстові блоки, таблиці, графіки, зображення, форми та інші. Завдяки цьому можна

змінювати вміст документів, застосовувати стилі та форматування, додавати або видаляти елементи з документа за допомогою програмного коду;

Open XML SDK є потужним інструментом для розробників, які працюють з документами у форматі Office Open XML. Вона дозволяє забезпечити автоматизовану обробку документів, спростити рутинні завдання та забезпечити більшу гнучкість і контроль над документами у форматі Office.

– Bootstrap - є потужним фреймворком для розробки веб-інтерфейсів. Використання бібліотеки Bootstrap дозволяє стилізувати інтерфейс та забезпечити його адаптивність для різних пристроїв. Це означає, що веб-сторінки, побудовані з використанням Bootstrap, коректно відображаються як на комп'ютерах, так і на мобільних пристроях, забезпечуючи зручне взаємодію для користувачів.

Одна з головних переваг використання Bootstrap полягає в його широкому спектрі готових компонентів і стилів. Це дозволяє ефективно будувати інтерфейси, використовуючи готові блоки, кнопки, форми, навігаційні панелі та інші елементи. Це спростовує процес розробки, оскільки не потрібно писати всі стилі з нуля.

Крім того, Bootstrap має широку підтримку браузерів, що забезпечує сумісність з різними веб-переглядачами. Це дає впевненість, що інтерфейс буде відображатися коректно для користувачів незалежно від того, який браузер вони використовують.

Окрім базового набору компонентів і стилів, Bootstrap також надає можливість налаштування теми за допомогою змінних CSS. Це дозволяє легко змінювати вигляд інтерфейсу, забезпечуючи його відповідність бренду кафедри чи університету.

Розробники можуть використовувати змінні для налаштування кольорів, шрифтів, розмірів елементів та інших аспектів дизайну. Крім того, Bootstrap постійно оновлюється і підтримується активною спільнотою розробників. Це означає, що ви можете розраховувати на надійність, безпеку та постійне вдосконалення фреймворку.

У підсумку, використання Bootstrap дозволяє зекономити час і зусилля при розробці інтерфейсу для конкурсу наукових робіт студентів. Він надає широкий

набір готових компонентів та стилів, що спрощує процес розробки і забезпечує сучасний та адаптивний інтерфейс. Завдяки своїй популярності і активній спільноті розробників, Bootstrap є надійним інструментом для реалізації веб-сайтів для конкурсів наукових робіт.

– Razor є потужним інструментом для розробки динамічних веб-сторінок у фреймворку ASP.NET Core. Використання Razor дозволяє розробникам комбінувати код C# з HTML, створюючи таким чином динамічний контент і реагуючи на дані на серверній стороні.

Одним з переваг використання Razor є його підтримка моделей. Це означає, що розробники можуть передавати дані з контролера до веб-сторінки і використовувати їх для генерації вмісту. Це забезпечує гнучкість та контроль над відображенням даних на сторінці, дозволяючи легко взаємодіяти з різними даними та змінювати їх відповідно до потреб.

Окрім того, Razor підтримує використання конструкцій умов, циклів та інших виразів C#. Це надає розробникам широкий спектр можливостей для реалізації складних сценаріїв та логіки на серверній стороні. Розробники можуть використовувати умовні оператори, цикли, функції та інші можливості C# для динамічного створення сторінок з різноманітними функціями і функціональністю.

Інтерфейс користувача, розроблений з використанням Razor, відповідає сучасним стандартам дизайну та веб-розробки. Він пропонує зручну навігацію, ефективний код і чисту структуру. Разом з гнучкістю та розширюваністю Razor це дозволяє розробникам легко вносити зміни та додавати нові функціональні можливості, щоб відповідати конкретним потребам користувачів. Завдяки цьому, інтерфейс стає більш користувачоорієнтованим і забезпечує більш зручний спосіб взаємодії з додатком.

– Git є розподіленою системою контролю версій, яка широко використовується в різних проектах розробки програмного забезпечення. Він надає набір інструментів та функцій для ефективного управління версіями коду, співпраці розробників та відстеження змін у проекті. Ось деякі характеристики використання Git:

Розподілена архітектура: Git використовує розподілену модель, що дозволяє кожному розробнику мати повну копію репозиторію на своєму локальному комп'ютері. Це забезпечує високу гнучкість та незалежність роботи розробників, а також стійкість до відключених мережних з'єднань.

Розгалуження та злиття: Git надає потужні можливості розгалуження та злиття, що дозволяє розробникам створювати та перемикатися між різними гілками коду для розробки нових функцій, виправлення помилок чи експериментів. Після того, як зміни у гілках були протестовані та перевірені, вони можуть бути безпечно поєднані з основною гілкою проекту.

Історія змін: Git записує повну історію змін у репозиторії, включаючи інформацію про коміти, авторів, дату та час. Це дозволяє розробникам легко відстежувати, хто і коли вносив зміни до коду, а також відновлювати попередні версії файлу за потреби.

Управління конфліктами: У випадку, якщо два або більше розробників вносять зміни до однієї частини коду, Git допомагає в управлінні конфліктами, що виникають при злитті змін. Розробники можуть вручну вирішувати конфлікти або використовувати інструменти для автоматичного злиття та вирішення конфліктів.

Резервне копіювання та відновлення: Git забезпечує можливість створення резервних копій репозиторію та його відновлення у разі втрати даних або помилкових змін. Це допомагає захистити код від непередбачених збоїв та відновити його у попередній робочий стан.

Інтеграція з іншими інструментами: Git інтегрується з різними середовищами розробки, такими як Visual Studio, IntelliJ IDEA, Eclipse та багатьма іншими. Це полегшує роботу з Git, дозволяючи розробникам використовувати звичні засоби розробки разом із потужними можливостями системи контролю версій.

Git є одним з найбільш популярних та широко використовуваних інструментів розробки програмного забезпечення та відіграє важливу роль у сучасних розробницьких процесах, забезпечуючи ефективне керування версіями коду та співробітництво розробників.

Рис 1.5. – Програмні засоби реалізації.

1.3 Архітектура програмного забезпечення

Архітектура додатку Science Planner, реалізованого на платформі ASP.NET Core MVC, включає різноманітні компоненти та підходи, що забезпечують ефективну організацію коду, логічну роздільність функцій та зручне управління додатком. Основні компоненти та принципи архітектури додатку такі:

Клієнтська частина:

– користувацький інтерфейс: Реалізований за допомогою веб-сторінок, що відображаються в браузері або мобільному додатку за допомогою HTML, CSS, JavaScript. Він взаємодіє з серверною частиною для отримання та передачі даних;

– клієнтські скрипти: JavaScript-код, який виконується на стороні клієнта;

– Bootstrap шаблони: Використовались для створення повторюваних елементів і структурування користувацького інтерфейсу.

Серверна частина:

– реалізована на основі фреймворка ASP.NET Core MVC, який обробляє запити від клієнта, взаємодіє з базою даних та надає відповіді клієнту;

– контролери: Відповідають за обробку запитів, взаємодію з моделями та передачу даних у відповідні представлення. Моделі: Представляють структуру та логіку даних додатку, включаючи взаємодію з базою даних.

Допоміжні функції:

– організація писем: Додаток Science Planner включає функціональність організації писем. Користувачі можуть отримувати сповіщення електронною поштою про додавання наукових робіт. Для цього використовуються SMTP-сервери та бібліотеки для надсилання та прийому електронних листів;

– генерація звіту - Додаток Science Planner включає функціональність генерація звіту. Вона відіграє важливу роль у забезпеченні ефективності та зручності роботи з науковими дослідженнями. Вона надає можливість автоматично створювати докладні звіти про виконану роботу. Яка значно спрощує процес створення та подання результатів наукових досліджень. Вона дозволяє користувачам зосередитись на самому дослідженні, не витрачаючи час на ручне складання звітів, та забезпечує чітке та професійне подання отриманих даних.

Ці додаткові функції допомагають покращити ефективність та зручність використання додатку Science Planner, роблять його більш гнучким та інтегрованим з іншими сервісами, що сприяє поліпшенню роботи з науковою роботою кафедр.

База даних:

– додаток Science Planner використовує базу даних MySql для зберігання всієї необхідної інформації. База даних забезпечує постійне збереження та доступ до даних, що стосуються користувачів, наукових робіт та інших важливих елементів системи. Кожна таблиця містить колонки, які відповідають конкретним атрибутам цих сутностей. Зв'язки між таблицями визначаються за допомогою зовнішніх ключів, що дозволяє встановлювати залежності між сутностями та забезпечує цілісність даних.

Middleware: Це компоненти, які обробляють HTTP-запити та відповіді між вхідним і вихідним потоками додатку. Існуючі Middleware компоненти в додатку Science Planner:

– app.UseDeveloperExceptionPage() - Цей middleware компонент використовується в режимі розробки і надає сторінку з детальною інформацією про виниклі виключення;

– app.UseExceptionHandler("/Home/Error") - Якщо виключення сталося в

пізніших middleware, цей middleware компонент перехоплює його і перенаправляє на сторінку /Home/Error, де можна обробити виключення і показати відповідну сторінку помилки;

– app.UseHsts() - Цей middleware компонент додає Strict-Transport-Security (HSTS), який повідомляє браузерам використовувати тільки HTTPS для всіх наступних запитів;

– app.UseHttpsRedirection() - Цей middleware компонент перенаправляє HTTP-запити на HTTPS-версію додатку, забезпечуючи безпеку передачі даних;

– app.UseStaticFiles() - Цей middleware компонент дозволяє обслуговувати статичні файли, такі як зображення, CSS-файли та JavaScript-скрипти, з папки wwwroot;

– app.UseRouting() - Цей middleware компонент визначає маршрутизацію запитів і визначає, який обробник (controller) має бути викликаний для конкретного запиту;

– app.UseAuthentication() - Цей middleware компонент додає автентифікацію до додатку, що дозволяє перевіряти та ідентифікувати користувачів на основі їх облікових записів;

– app.UseAuthorization() - Цей middleware компонент додає авторизацію до додатку, що дозволяє контролювати доступ користувачів до певних ресурсів або функціональності на основі їх прав доступу;

– app.UseEndpoints() - Цей middleware компонент визначає конфігурацію кінцевих точок додатку, тобто вказує, які URL-шляхи пов'язуються з відповідними обробниками (controllers) та діями (actions).

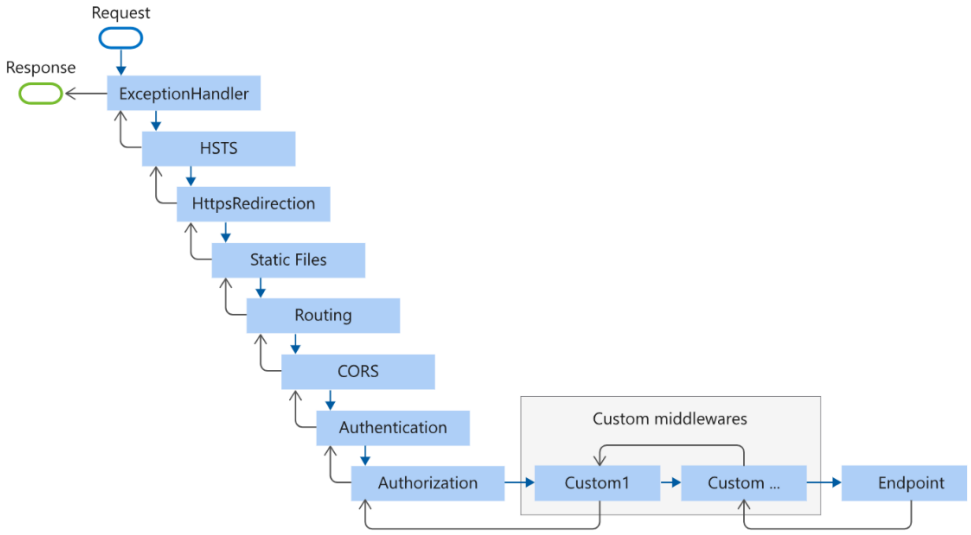


Рис 1.6. – Схема роботи Middleware компонентів

Маршрутизація:

– Science Planner використовує систему маршрутизації для визначення, який контролер та дія повинні обробити конкретний HTTP-запит. Маршрутизація здійснюється на основі шаблонів маршрутів, які вказують шляхи URL та параметри, які необхідно передати до контролера та дії.

Конфігураційний файл:

– є прикладом конфігурації для додатка у форматі json. Він містить кілька секцій з різними налаштуваннями, які використовуються в цьому додатку:

– секція "AllowedHosts" визначає список дозволених хостів, до яких дозволяється доступ до додатка;

– секція "ConnectionStrings" містить рядок підключення до бази даних. В додатку використовується база даних MySql з параметрами сервера, порту, назвою бази даних та обліковими даними користувача;

– секція "EmailConfiguration" містить конфігурацію для відправки електронних листів. Вона включає адресу відправника (SenderEmail) та пароль відправника (SenderPassword);

– секція "BaseUrl" визначає базовий URL для додатка, який може використовуватися, наприклад, для побудови посилань;

Цей конфігураційний файл дозволяє настроїти різні параметри додатка зручним способом і звертатися до них в коді додатка.

2. РЕАЛІЗАЦІЯ ФУНКЦІЙ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Розробка інтерфейсу користувача

Інтерфейс користувача для Science Planner був реалізований з використанням технологій ASP.NET Core та HTML/CSS. Структура HTML-сторінки складається з таких елементів:

– <head>: У цьому розділі задаються метатеги, такі як кодування символів та налаштування відображення на мобільних пристроях. Також підключаються зовнішні таблиці стилів (CSS) та бібліотеки, такі як Bootstrap;

– <body>: У цій частині розміщується весь вміст сторінки, включаючи заголовок, навігаційне меню, основний вміст;

– <header>: Заголовок сторінки містить навігаційне меню з логотипом та кнопкою розгортання;

– <main>: Основна частина сторінки, де розміщується зміст, що динамічно залежить від контексту. Зміст рендериться за допомогою методу RenderBody();

– Модальні вікна є важливою складовою інтерфейсу користувача веб-сайту конкурсу наукових робіт студентів. Вони забезпечують зручний спосіб взаємодії з додатковими функціями та дозволяють користувачам вводити необхідну інформацію.

Перше модальне вікно, "Створення наукової роботи", відкривається при натисканні на відповідний елемент навігаційного меню. Це вікно містить форму, де користувач може ввести необхідні дані про нову наукову роботу, такі як назва, опис, кафедра та інші важливі відомості. Після заповнення форми користувач має можливість створити нову наукову роботу, натиснувши відповідну кнопку. Це забезпечує зручний та швидкий спосіб додавання нових робіт до системи.

Друге модальне вікно, "Реєстрація або видалення адміністратора", служить для керування адміністраторськими правами. Це вікно відкривається також за допомогою натискання відповідного елемента навігаційного меню. У цьому вікні

користувач може ввести необхідні дані для реєстрації нового адміністратора або видалення існуючого. Після введення необхідних даних користувач має можливість зберегти зміни або видалити адміністратора з системи.

Використання модальних вікон значно полегшує процес взаємодії користувачів з веб-сайтом конкурсу наукових робіт студентів. Вони дозволяють зосередитись на важливих завданнях, швидко та зручно вводити необхідну інформацію і забезпечують ефективне управління роботами та адміністраторами.

– Скрипти: У кінці сторінки підключаються скрипти, такі як jQuery, Particle.js та власний файл site.js. Вони допомагають забезпечити взаємодію та функціональність на стороні клієнта;

Для забезпечення додаткової функціональності та зручного взаємодії з користувачем використовуються такі технології:

– Bootstrap: Використовувався на сторінках проекту для досягнення зручної і привабливої візуальної представленості, а також для реалізації додаткової функціональності. Нижче наведено кілька прикладів використання Бутстрапу на різних сторінках проекту:

Навігаційне меню: Бутстрап надавав готові компоненти для створення респонсивного навігаційного меню. Застосування класів Бутстрапу, таких як navbar і nav, дозволило створити стильне та функціональне меню, яке добре пристосовувалося до різних розмірів екрану;

Форми: Для стилізації форм використовувалися класи Бутстрапу, такі як form-control, form-group і btn. Це дозволило швидко створювати естетично збалансовані та легкі використовувати форми для введення даних та взаємодії з користувачем;

Картки (кардс): Бутстрап надав готові стилі для створення карточок з інформацією. Застосування класу card дозволяло легко створювати контейнери для вміщення важливої інформації, включаючи зображення, заголовки, текст та кнопки;

Модальні вікна: Для реалізації взаємодії з користувачем через модальні вікна використовувався функціонал Бутстрапу. Застосування класу modal дозволяло

легко створювати вікна з контентом, які відкривалися при натисканні на певні елементи або за певних умов;

Кнопки: Бутстрап надавав різноманітні стилізовані кнопки, які використовувалися для виконання різних дій на сторінках проекту. Застосування класу btn дозволяло швидко створювати кнопки з різними стилями та ефектами;

Таблиці: Для створення елегантних та добре оформлених таблиць використовувалися класи Бутстрапу, такі як table, table-striped та table-responsive. Це дозволяло легко відображати дані в табличному форматі зі стильним оформленням і можливістю адаптації до різних розмірів екрану.

Для забезпечення додаткової динамічності та взаємодії з користувачем використовувалась бібліотека jQuery.

– jQuery є потужним інструментом, який спрощує взаємодію з HTML-документами, анімацію, обробку подій та виконання AJAX-запитів.

Обробка подій: jQuery надавав потужні функції для обробки різних подій, таких як клік, наведення, фокус, втрата фокусу тощо. Ми могли легко призначати обробники подій за допомогою функції on(), що дозволяло реагувати на взаємодію користувача з елементами сторінки і виконувати певні дії.

Маніпуляція DOM-елементами: З jQuery ми могли легко маніпулювати DOM-елементами сторінки. Ми використовували функції, такі як addClass(), removeClass() або toggleClass(), для зміни класів елементів і зміни їх стилів. Також, ми могли динамічно додавати, видаляти або змінювати елементи на сторінці за допомогою функцій, таких як append(), remove() або html().

Валідація форм: З використанням додаткових плагінів jQuery, ми забезпечували валідацію форм на сторінках. Це дозволяло нам перевіряти правильність введених користувачем даних перед їх відправкою на сервер. Ми використовували плагіни, такі як jQuery Validation, для створення правил валідації та відображення повідомлень про помилки.

– Particle.js – Був активно використаний на веб-сторінках для створення захоплюючих фонових анімацій. Ця бібліотека надає простий та ефективний спосіб додати жвавість та інтерактивність до веб-дизайну.

Основним елементом Particle.js був файл particles.json, в якому налаштовані параметри анімації та візуального представлення частинок. Цей файл дозволяє контролювати зовнішній вигляд та поведінку частинок, визначати їх кольори, розміри, форми та швидкості руху.

Particles.json надав гнучкість налаштування анімації, дозволяючи створювати різноманітні ефекти, такі як рух частинок у заданих напрямках, обертання, зміна прозорості та багато іншого. А також можливість визначити положення та розмір області, де частинки можуть вільно переміщатися.

– Razor було використано для створення динамічних веб-сторінок з інтеграцією логіки та даних безпосередньо у HTML-код. Синтаксис C# використовувався для виконання умовних операторів, циклів і звернень до бази даних у представленнях Razor.

Моделі даних передавалися з контролера в уявлення, що дозволяло звертатися до їх властивостей та відображати значення на сторінці. Часткові уявлення використовувалися виділення загальних елементів інтерфейсу та його перевикористання різних сторінках.

Макети сторінок визначали основну структуру та елементи інтерфейсу, які використовувалися у кожному поданні. Це забезпечувало одноманітний дизайн та спрощує зміни в макеті.

Синтаксис Razor інтегрувався з HTML та CSS, що дозволяло використовувати звичні теги та стилі разом із Razor синтаксисом. Це робило код більш читаним та зрозумілим.

Razor забезпечував зручні інструменти для роботи з формами та взаємодії з користувачем. Форми створювалися, дані оброблялися, і додаткова обробка виконувалася на сервері за допомогою C#.

Таким чином, використання Razor у веб-сторінках дозволяло створювати динамічні сторінки з інтегрованою логікою та даними, полегшуючи розробку та підтримку коду.

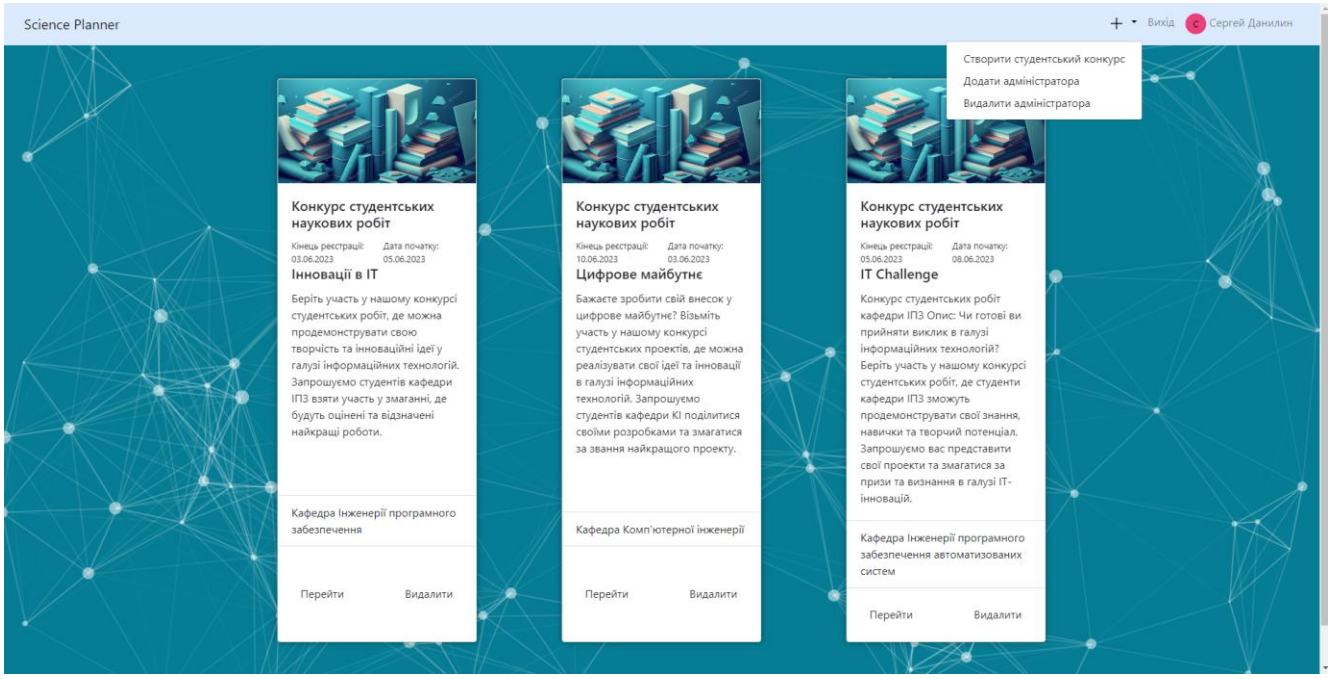


Рис 2.1. – Головна сторінка додатку

2.2 Авторизація користувача

Мета цього розділу полягає в описі процесу авторизації користувачів у додатку, а також представленні використовуваних інструментів та механізмів для забезпечення безпеки та зручності користувачів.

Авторизація користувача є важливою складовою будь-якої сучасної веб-платформи. Вона дозволяє користувачам автентифікуватися та отримувати доступ до різних функцій та ресурсів системи з урахуванням їх прав доступу.

2.2.1 Інтеграція з Gmail API та бібліотеками автентифікації

Для реалізації авторизації користувачів у програмі використовується інтеграція з Gmail API та наступні бібліотеки автентифікації:

- Microsoft.AspNetCore.Authentication.Google.
- Microsoft.AspNetCore.Authentication.Cookies.

Ці інструменти надають потужні та надійні механізми для автентифікації користувачів.

Бібліотека Microsoft.AspNetCore.Authentication.Google дозволяє інтегрувати авторизацію через облікові записи Google у додаток. З її допомогою можна

налаштувати інтеграцію зі службами Google, такими як Gmail, Google Drive тощо. Це дає користувачам зручну можливість використовувати свої існуючі облікові записи Google для авторизації в додатку, що спрощує процес входу та забезпечує високий рівень безпеки.

Бібліотека Microsoft.AspNetCore.Authentication.Cookies забезпечує механізми авторизації на основі кукі-файлів. Після успішної авторизації, користувач отримує спеціальний токен, який зберігається в кукі-файлі на боці клієнта. Цей токен використовується для ідентифікації та автентифікації користувача під час подальшого взаємодії з додатком. Використання кукі-файлів дозволяє зберігати стан авторизації між різними запитами користувача, що забезпечує зручність в роботі з додатком.

Ці бібліотеки надають гнучкі налаштування, які дозволяють визначити правила авторизації, перенаправлення після успішної авторизації або невдачі, налаштування зберігання токенів та багато іншого. Вони також підтримують безпеку шляхом шифрування та підпису токенів, що забезпечує захист від підробки та зловживань.

Загальна інтеграція з Gmail API та використання бібліотек автентифікації Microsoft.AspNetCore.Authentication.Google та Microsoft.AspNetCore.Authentication.Cookies дозволяє програмному забезпеченню системи обліку наукової роботи забезпечити ефективну та безпечну авторизацію користувачів, що в свою чергу забезпечує зручність та захищеність їхньої роботи з додатком.

2.2.2 Налаштування авторизації

У класі Startup програми виконуються важливі налаштування, пов'язані з авторизацією користувачів. Цей клас відіграє ключову роль у конфігурації додатку та визначенні необхідних компонентів для забезпечення аутентифікації.

У методі ConfigureServices, який є частиною класу Startup, відбувається ініціалізація та конфігурація різних компонентів, які необхідні для забезпечення аутентифікації користувачів. В цьому методі визначаються схеми авторизації, налаштовуються параметри і проводяться необхідні реєстрації сервісів.

Один з основних кроків - це встановлення схеми авторизації, яка буде використовуватись у додатку. Це може бути схема авторизації на основі Google або схема авторизації на основі cookie. Визначення схеми авторизації дозволяє програмі знати, який механізм використовувати для перевірки ідентифікаційних даних користувача.

Правильна конфігурація авторизації у методі ConfigureServices дозволяє програмному забезпеченню додатку належним чином працювати з авторизованими користувачами, забезпечуючи їм доступ до відповідних функцій та ресурсів системи.

Всі ці дії в класі Startup є важливими кроками у процесі реалізації авторизації користувачів у програмі та забезпечення безпеки та зручності для користувачів.

2.2.3 Схема авторизації на основі Google

За допомогою схеми на основі Google користувачі можуть використовувати свої існуючі облікові записи Google для входу до програми, що спрощує процес авторизації та . Крім того, ця схема забезпечує високий рівень безпеки, оскільки процес автентифікації здійснюється через офіційні сервіси Google.

Використання схеми на основі Google дозволяє програмі отримувати різні дані про користувача, що надаються Google, такі як електронна пошта, ім'я та інша інформація, яка може бути корисною для подальшої персоналізації та керування обліковим записом користувача. Все це робить схему авторизації на основі Google

важливою та корисною складовою програми, забезпечуючи зручність, безпеку та розширені можливості для користувачів.

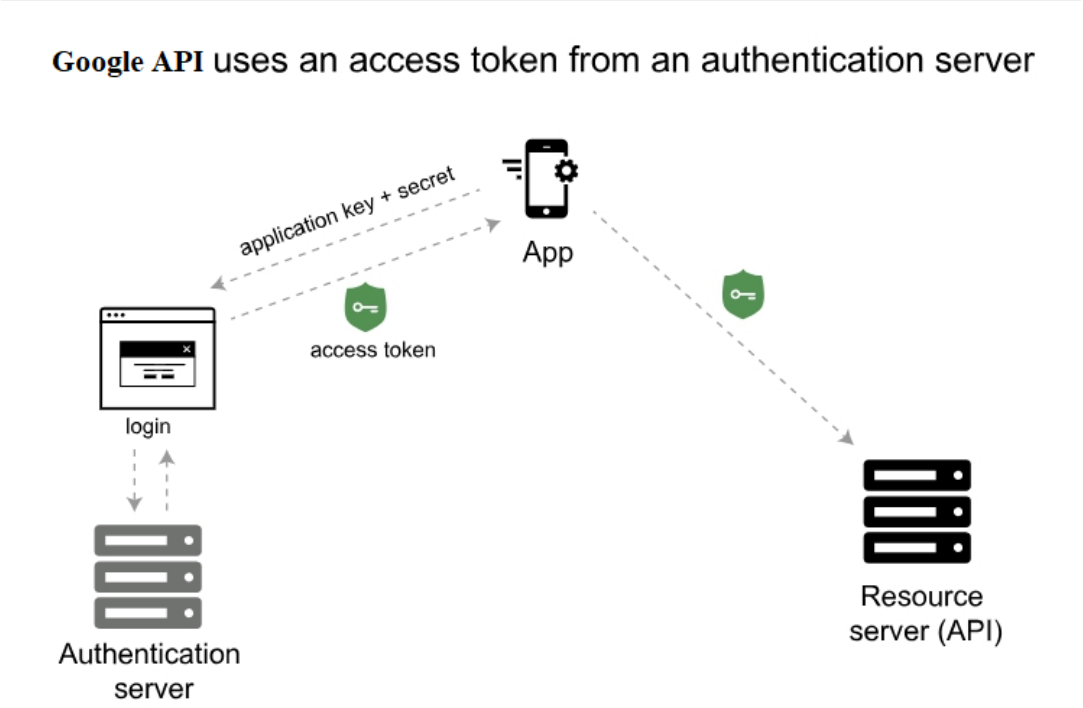


Рис 2.2. – Схема авторизації Google.

2.2.4 Аутентифікація на основі cookie

Крім схеми авторизації на основі Google, в даному додатку також використовується схема аутентифікації на основі cookie (CookieAuthenticationDefaults.AuthenticationScheme). Ця схема дозволяє зберігати сеанс автентифікації користувача, що дозволяє йому залишатися в системі після успішної автентифікації.

Схема автентифікації на основі cookie працює наступним чином: після успішної автентифікації користувача, сервер створює та відправляє клієнту спеціальний автентифікаційний cookie. Цей cookie містить унікальний ідентифікатор сеансу, який використовується для ідентифікації автентифікованого користувача.

Після цього, кожен наступний запит клієнта на сервер включає це автентифікаційне cookie, яке сервер використовує для перевірки сеансу авторизації користувача. Завдяки цьому механізму, користувач може залишатися автентифікованим під час взаємодії з різними сторінками та функціями системи, не потребуючи повторно вводити свої облікові дані.

Схема автентифікації на основі cookie є дуже поширеним і ефективним механізмом, який забезпечує зручну автентифікацію та збереження сеансу користувача. Вона дозволяє підтримувати стан автентифікації протягом тривалого періоду, що забезпечує зручну роботу з системою для користувачів.

2.2.5 Інтеграція з Google Developer Console

Для використання Gmail API та автентифікації через облікові записи Google у програмі потрібно виконати певні кроки в Google Developer Console. Створити проект, що включає Gmail API, а потім отримати клієнтський ідентифікатор (Client ID) і клієнтський секрет (Client Secret), необхідні для процесу авторизації. Ось як це можна зробити:

Створення проекту: Необхідно створити проект у Google Developer Console. Це можна зробити, перейшовши за адресою https://console.developers.google.com та дотримуючись інструкцій щодо створення нового проекту;

Увімкнення Gmail API: Після створення проекту потрібно увімкнути Gmail API для проекту. Для цього потрібно перейти до розділу "API та сервіси" у Google Developer Console, знайти "Gmail API" та активувати його для проекту;

Створення облікових даних: Щоб отримати клієнтський ідентифікатор (Client ID) та клієнтський секрет (Client Secret), необхідні для процесу авторизації, потрібно створити облікові дані OAuth 2.0 для проекту. У Google Developer Console перейти в розділ "API та сервіси" -> "Облікові дані" та натиснути на кнопку "Створити облікові дані". Потім виберіть тип облікових даних "Ідентифікатор клієнта OAuth";

Налаштування облікових даних: При створенні облікових даних OAuth 2.0 може знадобитися деякі параметри, такі як дозволені URI перенаправлення та допустимі області доступу. Необхідно вказати URI перенаправлення, куди буде

відправлено користувач після успішної авторизації. Також потрібно вказати необхідні області доступу, пов'язані з Gmail API, наприклад, "https://www.googleapis.com/auth/gmail.readonly" для читання електронної пошти;

Отримання клієнтського ідентифікатора та клієнтського секрету: Після завершення налаштування облікових даних OAuth 2.0 ви отримаєте клієнтський ідентифікатор та клієнтський секрет. Ці дані будуть використовуватися програмою для автентифікації через облікові записи Google.

2.2.6 Контролер авторизації

Для забезпечення автентифікації та управління процесом авторизації було створено спеціальний контролер - AuthenticationController. Цей контролер містить ряд методів, які відповідають за різні аспекти автентифікації користувачів.

Один з основних методів цього контролера - метод Login. Він обробляє GET-запит та ініціює автентифікацію через схему Google. Після того, як користувач натисне кнопку "Увійти через Google", він буде перенаправлений на сторінку авторизації Google, де введе свої облікові дані. Після успішної авторизації, Google надішле відповідь з автентифікаційними даними на сторону сервера. В нашому випадку, користувач буде перенаправлений на метод GoogleResponse для подальшої обробки цих даних.

У методі GoogleResponse відповідь від Google перевіряється на наявність тверджень (claims), які містять інформацію про користувача, таку як його електронна пошта та ім'я. Далі, ми перевіряємо, чи міститься користувач із зазначеною електронною поштою у нашій базі даних. Якщо такого користувача немає, він додається до бази даних. Після автентифікації, користувачеві надається певна роль, яка визначає його права доступу до системи. Залежно від цієї ролі, система визначає, до яких даних користувач має доступ для читання та запису. Важливою частиною цього процесу є відображення даних користувача у верхній частині веб-програми, де можуть бути відображені його ім'я, прізвище та аватар.

Також, в контролері AuthenticationController присутній метод Logout. Цей метод відповідає за вихід користувача з програми. При виклику цього методу, користувач виходить із системи, його автентифікаційні cookie очищаються, і він

повертається на головну сторінку або на сторінку авторизації для подальшої роботи з програмою.

Таким чином, завдяки контролеру AuthenticationController та його методам Login, GoogleResponse та Logout, ми можемо забезпечити відповідну аутентифікацію та керування процесом авторизації в нашій програмі. Ці методи дозволяють взаємодіяти з Google API, перевіряти та зберігати дані користувача, а також забезпечувати правильність доступу до функцій та даних системи.

2.3 Планування наукових робіт

За планування наукових робіт відповідає розроблений контролер "ScientificWorkController" в додатку Science Planner. Цей контролер містить декілька дій, які дозволяють додавати нові наукові роботи та видаляти існуючі. У конструкторі контролера передаються дві залежності: "MyDbContext" і "IConfiguration". "MyDbContext" є об'єктом доступу до бази даних, де зберігаються дані про наукові роботи. Це дозволяє контролеру звертатися до бази даних для отримання, збереження та видалення робіт. "IConfiguration" використовується для отримання конфігураційних налаштувань додатку, таких як параметри підключення до бази даних або налаштування електронної пошти.

Перша дія контролера, "AddScientificWork", є POST-запитом, який додає нову наукову роботу до бази даних. При отриманні запиту, контролер встановлює дату створення роботи на поточну дату, заповнює необхідні поля та зберігає роботу в базі даних за допомогою "MyDbContext". Після збереження наукової роботи в базі даних, контролер створює екземпляр сервісу "EmailSender" і викликає метод для відправки електронних листів всім користувачам, які підписалися на сповіщення про наукові роботи. Це забезпечує інформування користувачів про нові роботи та покращує спілкування між системою та користувачами.

Друга дія контролера, "RemoveScientificWork", призначена для видалення наукової роботи з бази даних. При отриманні запиту, контролер спочатку видаляє всі пов'язані записи, які стосуються даної наукової роботи, наприклад, коментарі

чи оцінки. Після цього контролер видаляє саму роботу з бази даних за допомогою "MyDbContext". Ця дія забезпечує можливість користувачам видаляти зайві або невірні роботи з системи. Контролер "ScientificWorkController" управляє додаванням та видаленням наукових робіт в додатку Science Planner.

Він використовується для управління роботами та забезпечення взаємодії з базою даних, а також для сповіщення користувачів про нові роботи через електронну пошту. Це важлива функціональність, яка допомагає покращити ефективність та зручність використання системи Science Planner для організації та управління науковими дослідженнями.

2.4 Звіти

Сервіс CompetitionReportGenerator виконує генерацію звітів для проведених конкурсів у форматі DOCX. Він є важливою складовою системи управління конкурсами і надає можливість автоматично створювати детальні звіти для кожного проведеного конкурсу. Усередині сервісу використовуються наступні простори імен:

- DocumentFormat.OpenXml;
- DocumentFormat.OpenXml.Packaging;
- DocumentFormat.OpenXml.Wordprocessing.

Сервіс містить два методи: GenerateWinnerReport та DownloadWinnerReport. Ці методи гарантують зручний і простий спосіб генерації та завантаження звітів для проведених конкурсів.

Метод GenerateWinnerReport отримує дані про конкурс, такі як назва, категорія, критерії оцінювання, дати та інші вхідні параметри. Він відповідає за створення нового файлу DOCX з назвою "WinnerReport.docx" та заповнення його вмістом. Всередині методу використовуються об'єкти з бібліотеки Open XML SDK для створення розділів, абзаців та тексту у документі. Заголовки, інформація про конкурс, інформація про переможця та інші деталі заносяться до документа з

використанням різних елементів Open XML. Наприкінці методу, після успішного створення звіту, документ зберігається, а потім викликається метод DownloadWinnerReport для завантаження згенерованого звіту.

Метод DownloadWinnerReport повертає фізичний файл "WinnerReport.docx", який можна легко завантажити для подальшого використання або поширення.

Сервіс CompetitionReportGenerator використовується в різних сферах, де проводяться конкурси, наприклад, в освітніх установах, наукових організаціях, спортивних подіях та багатьох інших. Генерація звітів у форматі DOCX дозволяє зручно представляти інформацію про проведений конкурс, його результати та переможців. Завдяки сервісу CompetitionReportGenerator організатори конкурсів отримують потужний інструмент для автоматизації процесу генерації звітів, що сприяє підвищенню ефективності та точності управління конкурсами.

2.5 Сповіщення

За сповіщення у додатку відповідає сервіс EmailSender, який відповідає за надсилання повідомлень електронною поштою у разі успішного створення наукової роботи або відправлення студентом роботи до конкурсу. Цей сервіс має наступні основні функції:

Відправка електронних листів до всіх користувачів з бази даних з використанням SMTP-сервера та налаштувань, збережених у файлі конфігурації appsettings.json.

Це стандартний протокол, який використовується для відправлення електронної пошти в Інтернеті. Він забезпечує надійну доставку повідомлень від відправника до отримувача, а також обробку помилок та повідомлень про помилки.

SMTP працює на основі клієнт-серверної моделі. Коли відправник відправляє лист, його поштова програма (клієнт) встановлює з'єднання з поштовим сервером відправника. Потім клієнт передає серверу інформацію про повідомлення, включаючи адресу відправника, адресу отримувача, тему листа та його зміст.

Після отримання цих даних поштовий сервер відправника використовує протокол SMTP для передачі повідомлення до поштового сервера отримувача. Для цього він використовує команди та відповіді, визначені в протоколі. Поштовий сервер отримувача приймає повідомлення та зберігає його в поштовій скринці отримувача.

SMTP також підтримує додаткові функції, такі як аутентифікація відправника, можливість передачі вкладень та шифрування повідомлень.

У даному додатку використовується порт 587 для встановлення з'єднання з SMTP-сервером. Використання порту 587 зазвичай означає, що з'єднання з SMTP-сервером буде виконуватись через TLS.

Це протокол шифрування, який забезпечує захищене з'єднання між клієнтом (наприклад, додатком, який відправляє електронні листи) та сервером (SMTP-сервером).

TLS шифрує дані, які передаються між клієнтом та сервером, тому навіть якщо хтось перехопить ці дані, вони залишатимуться незрозумілими для зловмисника. Це забезпечує конфіденційність, оскільки лише передавач та отримувач можуть розшифрувати та прочитати дані.

Крім того, TLS також забезпечує цілісність даних, оскільки воно використовує механізми хешування та перевірки цифрових підписів для виявлення будь-яких змін або модифікацій даних під час передачі.

Використання TLS з портом 587 є важливим для забезпечення безпеки під час відправки електронних листів через SMTP-сервер. Воно гарантує, що дані, включаючи будь-яку чутливу інформацію, залишаються захищеними під час транспорту в мережі Інтернет.

При створенні нової наукової роботи викликається метод SendEmailToAllUsers. Цей метод використовує бібліотеку MailKit, яка є потужною та поширеною бібліотекою для роботи з протоколами електронної пошти, такими як SMTP, POP3 та IMAP.

Кроки, виконувані методом SendEmailToAllUsers:

– встановлення адреси електронної пошти завідуючого кафедри та адреси відображуваного імені. Отримання списку користувачів з бази даних.

– для кожного користувача створюється об'єкт MailMessage, в якому встановлюється адреса отримувача, тема та тіло повідомлення. Тіло повідомлення містить посилання на роботу, оформлене як гіперпосилання з відповідним стилем CSS, що містить дані про назву, опис, кафедру та інші деталі.

– встановлюється властивість IsBodyHtml для відображення повідомлення як HTML. Встановлюються налаштування SMTP-сервера та облікові дані для входу в обліковий запис відправника.

– використовуючи SmtpClient, відправляється повідомлення до отримувача.

метод SendEmailToAdmin відповідає за надсилання повідомлення електронною поштою адміністратору у разі успішного надсилання роботи студента для участі в конкурсі. Цей метод включає наступні кроки:

– отримання адреси електронної пошти завідуючого кафедри з налаштувань додатку;

– створення об'єкту MailMessage, в якому встановлюється адреса отримувача, тема та тіло повідомлення;

– тіло повідомлення містить інформацію про роботу, яка була успішно надіслана для участі в конкурсі, включаючи назву роботи, автора, категорію тощо;

– встановлення властивості IsBodyHtml для відображення повідомлення як HTML.

Надсилання повідомлення електронною поштою дозволяє забезпечити швидку та надійну комунікацію між студентами та завідуючим кафедри, сприяючи ефективному взаємодії та обробці наукових робіт у конкурсі.

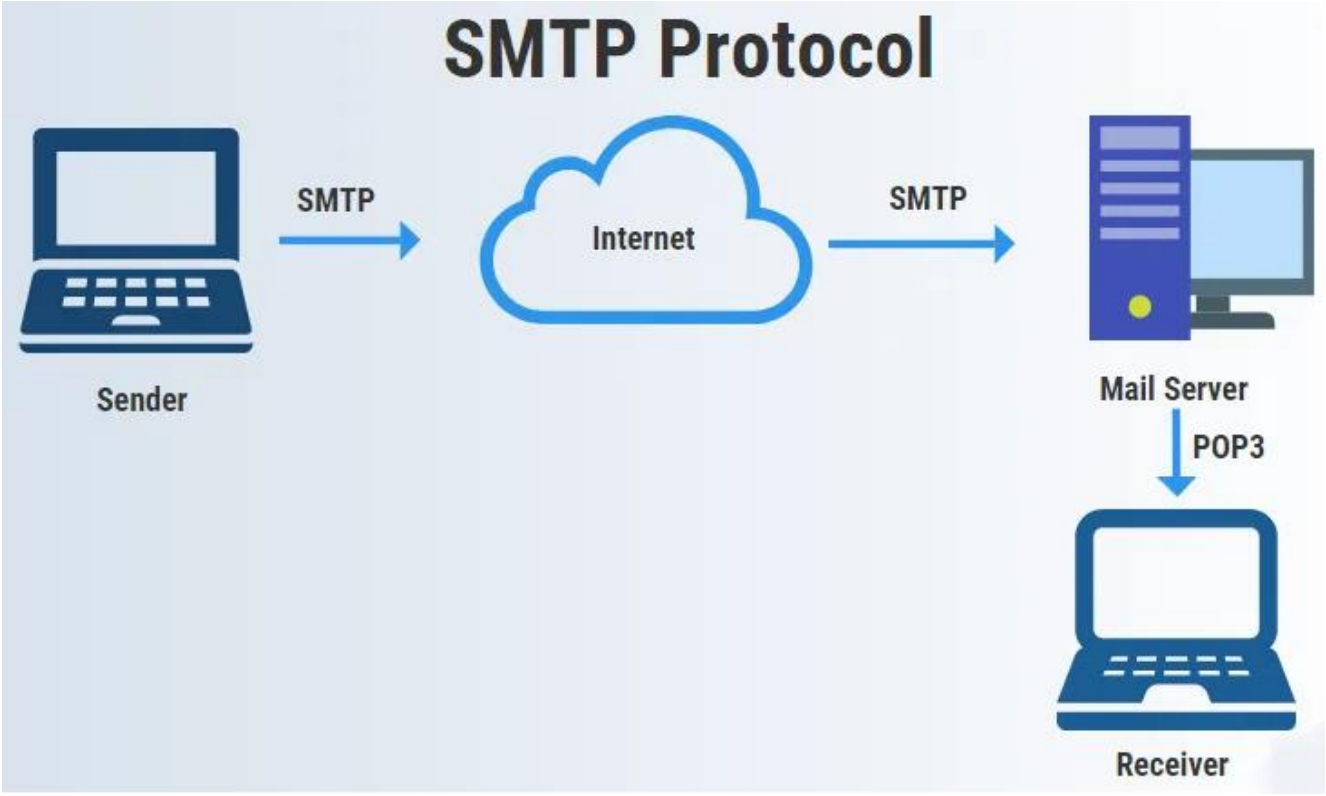


Рис 2.3. – Схема роботи SMPT Сервера

2.6 Облік наукових робіт

У додатку є реалізовані наступні можливості запису даних:

– додавання наукової роботи: У класі ScientificWorkController визначено метод AddScientificWork, який приймає об'єкт типу ScientificWork як параметр. Після деяких перетворень та перевірок об'єкт наукової роботи зберігається в базі даних з використанням контексту MyDbContext;

– додавання адміністратора: У контролері HomeController реалізовано метод AddAdmin, що дозволяє додавати нових користувачів з роллю адміністратора в систему. Дані користувача вводяться за допомогою модального вікна з полями ім’я і електронна адреса та зберігаються у базі даних;

– видалення адміністратора: У контролері HomeController реалізовано метод RemoveAdmin, що дозволяє видаляти користувачів з роллю адміністратора в систему. Метод приймає як параметр рядок email, який отримує з модального вікна, який представляє адресу електронної пошти адміністратора. Якщо користувач

знайдено за допомогою LINQ і його роль (Role) дорівнює "Admin", він видаляється з контексту бази даних;

– додавання студентом роботи на конкурс: У класі CompetitionWorkController визначено метод AddCompetitionWork, який приймає ідентифікатори працівника та конкурсу як параметри. Потім, на основі отриманих даних, створюється об'єкт CompetitionWork і зберігається у базі даних через контекст MyDbContext;

– додавання конкурсу: У класі CompetitionController визначено метод AddCompetition, який приймає об'єкт типу Competition як параметр. При додаванні конкурсу використовуються дані наукової роботи, яка прив'язана до конкурсу. Крім того, додаються оціночні критерії для конкурсу. Усі дані зберігаються у базі даних через контекст MyDbContext;

– реєстрація учасника на курс: У класі CompetitionController визначено метод RegisterOnCource, який приймає об'єкт RegisteredMember, файл та об'єкт Competition як параметри. Зареєстрований учасник зберігається у базі даних разом із прикріпленим файлом через контекст MyDbContext.

Таким чином, у додатку реалізовано різні функції запису даних, що дозволяють додавати наукові роботи, роботи на конкурси, конкурси, а також реєструвати учасників.

2.7 Організація бази даних

У веб-додатку Science Planner, MySQL використовується як система управління базами даних для зберігання та управління науковими роботами. Контекст бази даних, представлений класом MyDbContext, успадковується від базового класу DbContext, який надає доступ до функцій та можливостей роботи з базою даних.

Схема бази даних, що використовується в додатку Science Planner (показана на малюнку 2.4.), включає таблиці, такі як "ScientificWorks" для зберігання інформації про наукові роботи та їх атрибути, "Users" для зберігання даних

користувачів, "Competition" для зберігання інформації про конкурс та "CompetitionWorks" для зберігання інформації про відправлені наукові роботи студентів до конкурсу.

У конфігураційному файлі програми (appsettings.json) є розділ ConnectionStrings, де визначено рядок підключення з ім'ям "MySqlConnection". Цей рядок підключення вказує на локальний сервер MySQL з використанням порту 3306 та базою даних з відповідним ім'ям. Також вказані облікові дані користувача, включаючи пароль, для забезпечення автентифікації під час підключення до бази даних.

Організація бази даних у програмі Science Planner також включає визначення моделей даних, які відображають сутності, що зберігаються в базі даних, такі як наукові роботи та користувачі. Крім того, встановлюються відносини між таблицями для забезпечення цілісності даних та зв'язків між об'єктами.

Організація бази даних також включає налаштування автентифікації та авторизації для забезпечення безпеки даних та контролю доступу до функціонала програми. Конфігурація конвеєра обробки HTTP-запитів також виконується для обробки та маршрутизації запитів до відповідних контролерів.

Таким чином, організація бази даних у додатку Science Planner забезпечує надійне зберігання та управління науковими роботами, забезпечує безпеку даних та контроль доступу, а також забезпечує ефективну обробку запитів користувачів.

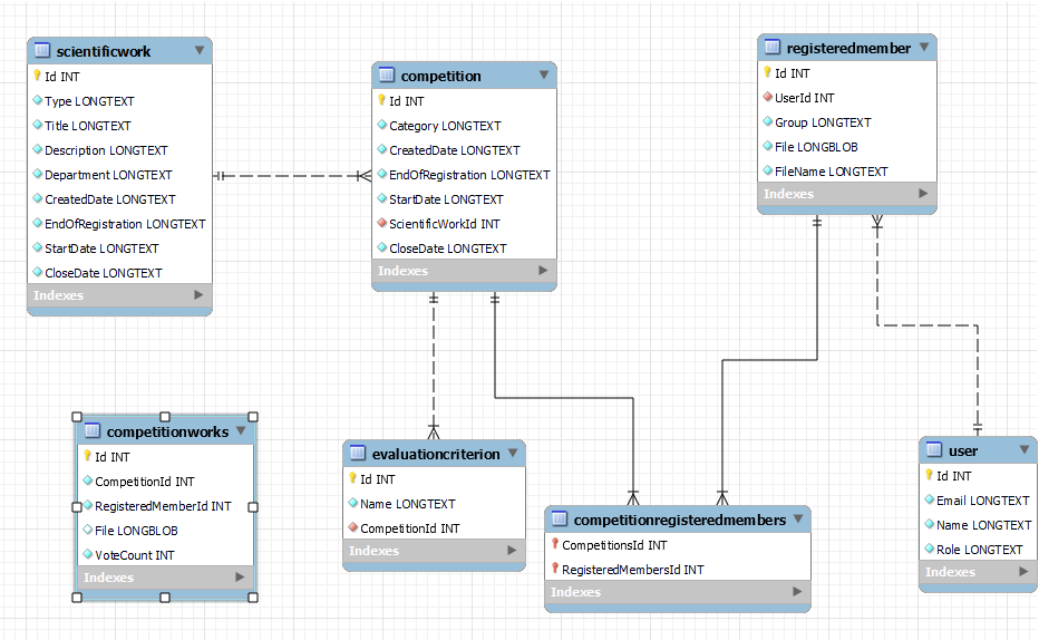


Рис 2.4.– Схема бази даних

3.ТЕСТУВАННЯ ТА ОЦІНКА ЕФЕКТИВНОСТІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Методологія тестування

У розробці веб-додатку Science Planner використовується поєднання різних методологій тестування, включаючи модульне тестування та тестування продуктивності. Цей підхід допомагає забезпечити якість, надійність та оптимальну продуктивність системи, забезпечуючи задоволення користувачів від використання додатку.

Одним з аспектів тестування є методологія модульного тестування для забезпечення коректності та працездатності модулів та компонентів. Методологія модульного тестування впроваджується у розробку з метою виявлення помилок та перевірки окремих модулів програмного забезпечення в ізоляції, що допомагає покращити якість та надійність програми.

Інструмент, що використовуються в Science Planner для модульного тестування, є NUnit – популярний фреймворк для тестування .NET-додатків. NUnit надає потужний та гнучкий набір функцій для написання та запуску модульних тестів, включаючи безліч атрибутів та тверджень для перевірки результатів тестів.

Для забезпечення надійності та ізоляції тестів, у Science Planner використовувався механізм вбудованої бази даних у пам'яті. Це дозволяє створювати та керувати тестовою базою даних усередині програми без необхідності підключення до реальної бази даних. Кожен тест у Science Planner був розроблений відповідно до принципів модульного тестування, включаючи підготовку тестового оточення (Arrange), виконання необхідних дій (Act) та перевірку результатів (Assert). Тести покривали різні методи застосування та перевіряли його функціональність, коректність обробки даних, взаємодію з базою даних та інші аспекти роботи.

У додатку Science Planner були створені спеціальні класи тестів, наприклад:

HomeControllerTests – це тестовий клас, який містить набір модульних тестів для класу HomeController. Цей клас розроблений з використанням фреймворку NUnit, який надає зручні засоби для написання та виконання модульних тестів.

Структура класу: Тестовий клас HomeControllerTests складається з кількох методів, кожен із яких представляє окремий модульний тест. Нижче наведено опис основних методів класу:

Метод Index_WithAuthenticatedUser_ReturnsViewResultWithScientificWorks:
Цей метод перевіряє, що під час виклику методу Index контролера HomeController з автентифікованим користувачем повертається результат типу ViewResult.

Для цього створюється об'єкт ClaimsPrincipal, що представляє автентифікованого користувача. Потім контекст контролера встановлюється для імітації автентифікації користувача. Очікується, що результатом виклику методу Index буде об'єкт ViewResult, який містить список наукових робіт.

Метод GetUsers_ReturnsJsonResultWithUserList:
Цей метод перевіряє, що під час виклику методу GetUsers контролера HomeController повертається результат типу JsonResult. Очікується, що результат міститиме список користувачів у форматі JSON.

Метод CreateCompetition_ReturnsViewResult:
Цей метод перевіряє, що під час виклику методу CreateCompetition з передачею ідентифікатора змагання повертається результат типу ViewResult. У методі створюється новий об'єкт контролера HomeController із використанням нового контексту бази даних. Потім створюється імітація автентифікованого користувача та встановлюється контекст контролера. Очікується, що результатом виклику методу CreateCompetition буде об'єкт ViewResult.

Метод RegisterView_ReturnsViewResult:
Цей метод перевіряє, що під час виклику методу RegisterView контролера HomeController з передачею ідентифікатора подання повертається результат типу ViewResult. Метод аналогічний попередньому методу CreateCompetition_ReturnsViewResult, але перевіряє виклик іншого методу контролера.

Метод Competition_ReturnsViewResultWithCompetitionWorks:

Цей метод перевіряє, що при виклику Competition контролера HomeController з передачею ідентифікатора змагання повертається результат типу ViewResult.

Метод аналогічний попереднім методам CreateCompetition_ReturnsViewResult та RegisterView_ReturnsViewResult, але перевіряє виклик іншого методу контролера та зміст списку змагальних робіт.

Метод AddAdmin_RedirectsToIndex:

Цей метод перевіряє, що під час виклику методу AddAdmin з передачею об'єкта користувача відбувається перенаправлення на метод Index контролера. Очікується, що результатом виклику методу AddAdmin буде об'єкт RedirectToActionResult, який вказує на метод Index контролера Home.

Метод RemoveAdmin_WithExistingAdminEmail_RedirectsToIndex:

Цей метод перевіряє, що під час виклику методу RemoveAdmin з передачею існуючої адреси електронної пошти адміністратора відбувається перенаправлення на метод Index контролера.

Метод створює нового адміністратора в контексті бази даних та зберігає його.

Потім викликається метод RemoveAdmin із адресою електронної пошти існуючого адміністратора. Очікується, що результатом виклику методу RemoveAdmin буде об'єкт RedirectToActionResult, що вказує на метод Index контролера Home.

Метод RemoveAdmin_WithNonExistingAdminEmail_RedirectsToIndex:

Цей метод перевіряє, що при виклику методу RemoveAdmin контролера HomeController з передачею неіснуючої адреси електронної пошти адміністратора відбувається перенаправлення на метод Index контролера. Метод викликає метод RemoveAdmin з неіснуючою адресою електронної пошти адміністратора.

Очікується, що результатом виклику методу RemoveAdmin буде об'єкт RedirectToActionResult, що вказує на метод Index контролера Home.

Клас CompetitionWorkControllerTests містить модульні тести для класу CompetitionWorkController. Який містить наступні методи:

Метод AddCompetitionWork_ValidData_RedirectsToCompetition():

Цей метод тестує метод AddCompetitionWork контролера CompetitionWorkController.

Створюється об'єкт зареєстрованого учасника та зберігається у контексті бази даних.

Викликається метод AddCompetitionWork контролера та результат перевіряється на відповідність очікуваним значенням.

Метод Vote_CompetitionWorkExistsAndHasNoVotes_RedirectsToIndex():

Цей метод тестує метод контролера Vote CompetitionWorkController.

Створюється об'єкт CompetitionWork з кількістю голосів та зберігається у контексті бази даних.

Встановлюється автентифікований користувач за допомогою об'єкта ClaimsPrincipal та контексту контролера.

Викликається метод Vote контролера та результат перевіряється на відповідність очікуваним значенням.

Клас CompetitionControllerTests є набором модульних тестів для перевірки функціональності класу CompetitionController. Який містить наступні методи:

AddCompetition_ValidCompetition_RedirectsToIndex() - Цей метод тестує функціональність методу AddCompetition. Він створює екземпляр класу MyDbContext та IConfiguration, а також екземпляр CompetitionController. Потім створюється об'єкт наукової роботи і зберігається у базі даних. Далі створюється об'єкт конкурсу з певними значеннями та викликається метод AddCompetition контролера. Після цього перевіряється, що результат є перенаправленням на метод Index контролера Home.

AddCategory_ValidEvaluationCriterion_RedirectsToIndex() - Цей метод перевіряє функціональність методу AddCategory. Він створює список критерій оцінювання з певними значеннями. Потім викликається метод AddCategory із цим списком. Після цього перевіряється, що результат є перенаправленням на метод Index контролера Home.

RemoveCompetition_ExistingCompetition_RemovesCompetitionAndRegistered Members_RedirectsToIndex() - Цей метод тестує функціональність методу

RemoveCompetition. Всередині методу створюється об'єкт конкурсу з певним ідентифікатором та пов'язаним об'єктом зареєстрованого учасника. Потім цей об'єкт додається до бази даних. Метод RemoveCompetition контролера викликається із передачею ідентифікатора існуючого конкурсу. Після цього перевіряється, що результат є перенаправленням на метод Index контролера Home, а також перевіряється, що об'єкти конкурсу і зареєстрованого учасника видалені з бази даних.

Клас ScientificWorkControllerTests містить тести для перевірки функціональності методів класу ScientificWorkController.

RemoveScientificWork_RemovesScientificWorkAndRegisteredMembers_ReturnsRedirectToActionResult(): Цей метод тестує функціональність методу RemoveScientificWork() контролера ScientificWorkController. Він створює об'єкт наукової роботи, додає його в контекст бази даних та викликає метод RemoveScientificWork() контролера. Потім перевіряє що об'єкт наукової роботи та пов'язані з ним зареєстровані учасники були успішно видалені з бази даних.

AddScientificWork_AddedScientificWorkAndRegisteredMembers_ReturnsRedirectToActionResult() тестує функціональність методу AddScientificWork() контролера ScientificWorkController. Цей метод відповідає за додавання нової наукової роботи та пов'язаних з нею зареєстрованих учасників до бази даних.

Другим з аспектві тестування є тестування продуктивності та ефективності. Оцінка продуктивності та ефективності є важливим етапом у процесі тестування додатку Science Planner і має на меті виявлення потенційних проблем, що можуть вплинути на швидкодію та роботу програми. Додаток, який працює швидко та ефективно, забезпечує задоволення користувачів та позитивне враження від його використання.

Додатково, тестування продуктивності включає оцінку часу відгуку системи на користувацькі дії, швидкість завантаження даних, час виконання операцій та реакцію додатку на навантаження. Ці метрики дозволяють виявити можливі проблеми з продуктивністю та швидкодією додатку, такі як затримки в роботі, низька швидкість відгуку або відсутність реагування на користувацькі дії.

Тестування ефективності додатку Science Planner включає оцінку його ресурсоємності та оптимального використання системних ресурсів. Це включає аналіз використання оперативної пам'яті, процесора та інших системних ресурсів під час роботи програми. Прикладом тестування ефективності є вимірювання часу, необхідного для виконання певних операцій, та порівняння його з прийнятними стандартами продуктивності.

Для забезпечення успішного тестування продуктивності Science Planner використовує консоль розробника. Консоль розробника надає можливість моніторити роботу програми в режимі реального часу та аналізувати її продуктивність.

Після проведення тестування продуктивності додатку Science Planner були отримані наступні результати:

- Loading (завантаження): 149 мс.
- Scripting (скрипти): 490 мс.
- Rendering (відображення): 138 мс.
- Painting (малювання): 14 мс.
- System (система) 198 мс.

Результати тестування свідчать про швидку реакцію додатку на користувацькі дії. Значення в мілісекундах (мс) показують час, який витрачається на виконання певних операцій.

За результатами тестування, час завантаження додатку становить 149 мс. Це вказує на швидку швидкість завантаження програми, що сприяє позитивному враженню користувачів та забезпечує ефективне використання додатку.

Час виконання скриптів становить 490 мс. Це означає, що обробка логіки програми відбувається досить швидко, що сприяє плавному функціонуванню додатку.

Значення 14 мс для відображення та 138 мс для малювання свідчать про швидку реакцію графічного інтерфейсу додатку на користувацькі взаємодії. Це означає, що елементи інтерфейсу швидко відображаються та малюються на екрані, що сприяє зручному користуванню та задоволенню користувачів.

Значення 193 мс для System (система) вказує на час, який витрачається на системні операції під час роботи додатку Science Planner. Системні операції можуть включати взаємодію з оперативною пам'яттю, процесором та іншими системними ресурсами.

У підсумку, під час тестування продуктивності Science Planner були отримані гарні результати, що свідчать про його швидку роботу та високу ефективність. При виконанні більшості операцій програма працює миттєво, без помітних затримок чи перебоїв у функціонуванні.

Зазначено, що деякі системні операції можуть призводити до незначних часових затримок. Однак, ці затримки є малозначними та не мають істотного впливу на загальний досвід користувачів. Science Planner все ще забезпечує зручне та ефективне використання, що робить його привабливим для широкого кола користувачів.

Це особливо важливо для науковців, дослідників та студентів, які використовують Science Planner у своїй роботі щодня. Швидка реакція програми на їхні запити та висока продуктивність дозволяють їм зосередитися на своїх наукових завданнях, не витрачаючи зайвий час на очікування відповіді від програми.

Таким чином, результати тестування продуктивності Science Planner переконливо підтверджують його швидку роботу з незначними часовими затримками лише у випадку деяких системних операцій. Це робить додаток зручним та ефективним використанням для користувачів.

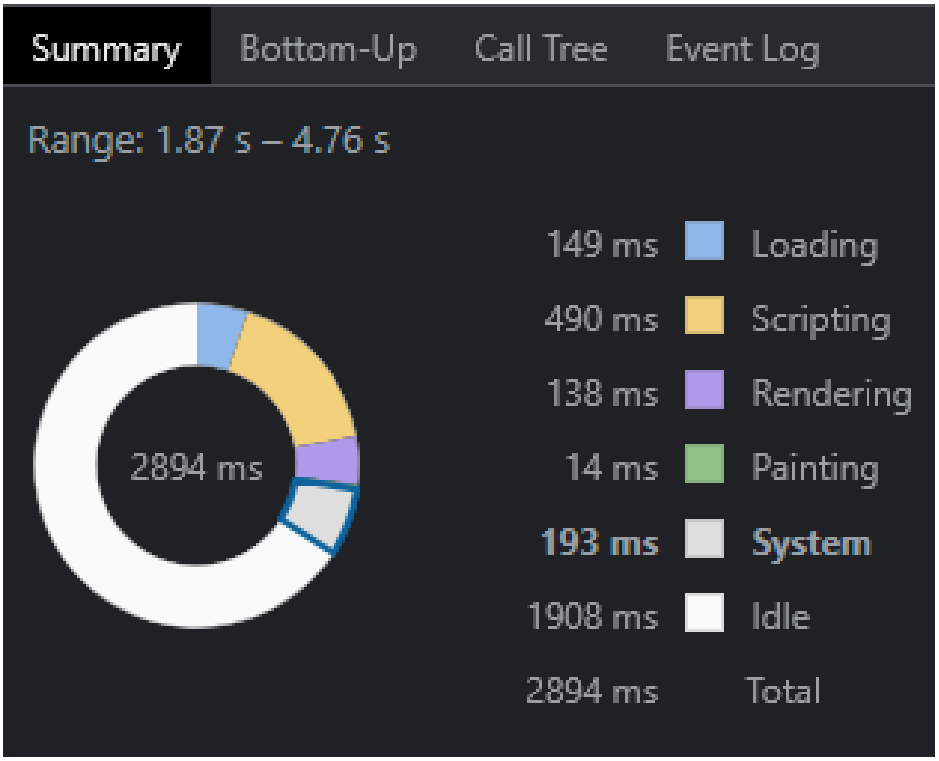


Рис 3.1. – Діаграма продуктивності

Висновки

У рамках даної дипломної роботи було розроблено програмне забезпечення під назвою "Science Planner" для системи планування та обліку наукової роботи кафедри. Під час реалізації проекту були використані сучасні технології та методології розробки, що дозволило створити функціональну та ефективну систему.

У процесі розробки були визначені вимоги до програмного забезпечення, включаючи можливості планування наукових робіт, авторизації користувачів, звітності та сповіщень. Було розроблено інтерфейс користувача, що дозволяє зручно та ефективно взаємодіяти з системою.

Для забезпечення надійності та якості програмного забезпечення, була використана методологія модульного тестування. Застосування фреймворку NUnit дозволило написати та виконати модульні тести, перевіряючи правильність роботи окремих модулів системи.

Також було звернуто увагу на безпеку та конфіденційність даних. В системі були використані методи контролю доступу, щоб забезпечити захищений доступ до інформації.

Після завершення розробки, проведення тестування та оцінки ефективності програмного забезпечення, виявилося, що Science Planner є функціональним та надійним рішенням для системи обліку та планування наукової роботи кафедри. Впровадження системи в організації сприятиме покращенню організації роботи, ефективному плануванню наукових робіт.

На основі проведеної роботи можна зробити наступні висновки:

- Було розроблено структуру бази даних та визначено необхідні таблиці для збереження інформації про наукові роботи.
- Реалізовано інтерфейс користувача, забезпечивши зручну та ефективну взаємодію з системою.
- Реалізовано можливість планування і обліку наукових робіт.

– Реалізовано можливість формування і відправлення звітів про виконання наукових робіт, що дозволяє зручно відстежувати результати роботи.

На основі проведеної роботи, можна зробити висновок, що розроблене програмне забезпечення Science Planner відповідає поставленим вимогам і володіє потрібними функціональними можливостями.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Visual Studio 2022: https://visualstudio.microsoft.com/
2. Загальні відомості ASP.NET Core MVC: https://learn.microsoft.com/en-us/aspnet/core/mvc/overview?view=aspnetcore-5.0
3. Gmail API Overview: https://developers.google.com/gmail/api/guides?hl=en
4. Git Documentation: https://git-scm.com/about
5. NUnit Documentation: https://docs.nunit.org/
6. MySQL Documentation https://dev.mysql.com/doc/
7. Bootstrap Documentation: https://getbootstrap.com/docs/5.3/getting-started/introduction/
8. Using MailKit To Send And Receive Email In ASP.NET Core: https://dotnetcoretutorials.com/using-mailkit-send-receive-email-asp-net-core/
9. Open XML SDK: https://learn.microsoft.com/en-us/office/open-xml/open-xml-sdk
10. SMTP: https://www.hostinger.com.ua/rukovodstva/kak-ispolzovat-smtp-server

Додаток А

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Розробка програмного забезпечення для планування та обліку наукової роботи кафедри засобами MySql, C#

Виконав студент 4 курсу
групи ПД-43
Данилін Сергій Олександрович
Керівник роботи
К.ф-м.н., доцент кафедри ІПЗ Садовенко Володимир Сергійович

Київ – 2023

МЕТА, ОБ’ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

- **Мета** - Надання можливості планувати наукові роботи та вести їх облік для ефективної роботи кафедри.
- **Об’єкт** - Процес планування та обліку наукової роботи кафедри.
- **Предмет** - Програмне забезпечення для планування та обліку наукової роботи кафедри на мові C#, з використанням бази даних MySql.

ЗАДАЧІ ДИПЛОМНОЇ РОБОТИ

1. Розробити структуру бази даних та визначити необхідні таблиці;
2. Розробити інтерфейс користувача та забезпечити його зручність;
3. Реалізувати можливість планування і обліку наукових робіт;
4. Реалізувати можливість формування і відправлення звітів про виконання наукових робіт.

АНАЛІЗ АНАЛОГІВ

Показник	Microsoft To-Do	Todoist	Trello	Google Keep	Розроблений додаток
Підтримка спільної роботи	-	-	+	-	+
Підтримка різних медіафайлів	-	-	-	-	+
Інтеграція з іншими інструментами	+	+	+	+	+
Спрямованість на наукові проекти.	-	-	-	-	+
Зручність інтерфейсу	+	+	+	+	+

ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Функціональні вимоги:
1.Автентифікація користувачів.
2.Інтерфейс користувача.
3.Адміністративна панель.
4.Планування наукових робіт.
5.Облік наукових робіт
6.Звітність.

Нефункціональні вимоги:
1.Використання мови програмування C#.
2.Використання бази даних MySQL.
3.Сумісність.
4.Тестування.

ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ

Діаграма класів

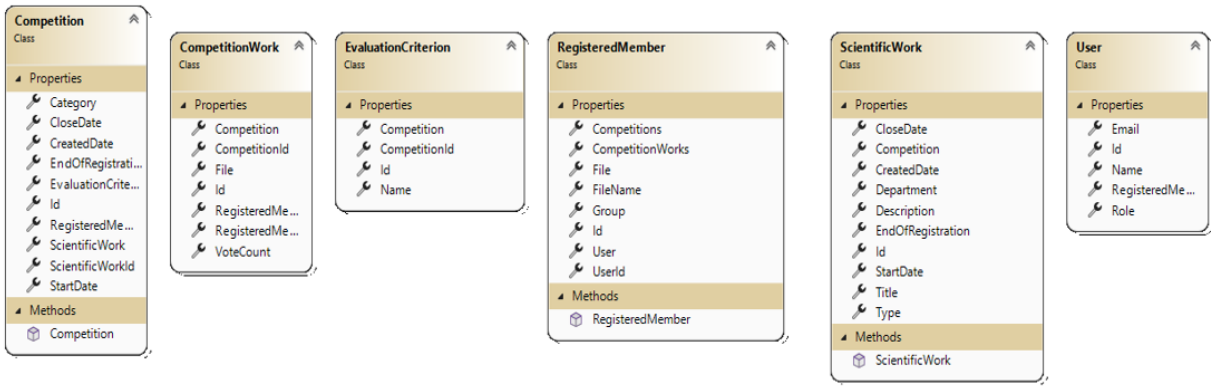
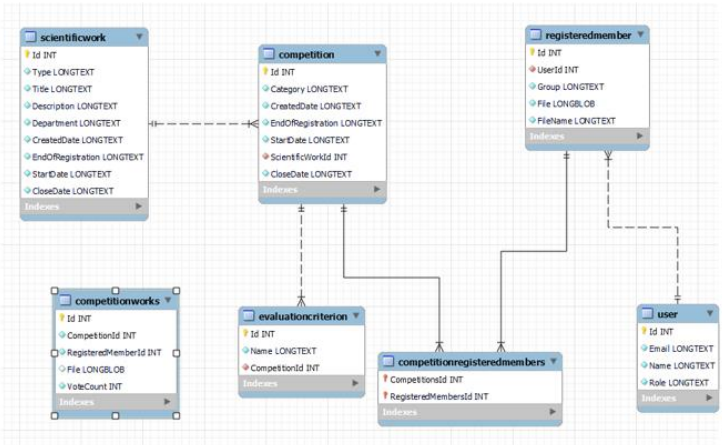
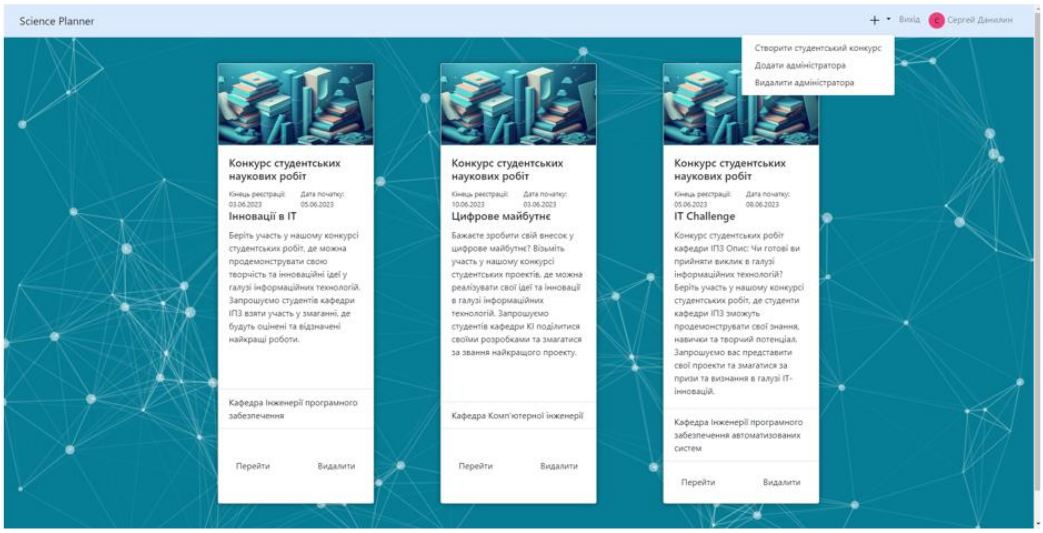


Схема бази даних

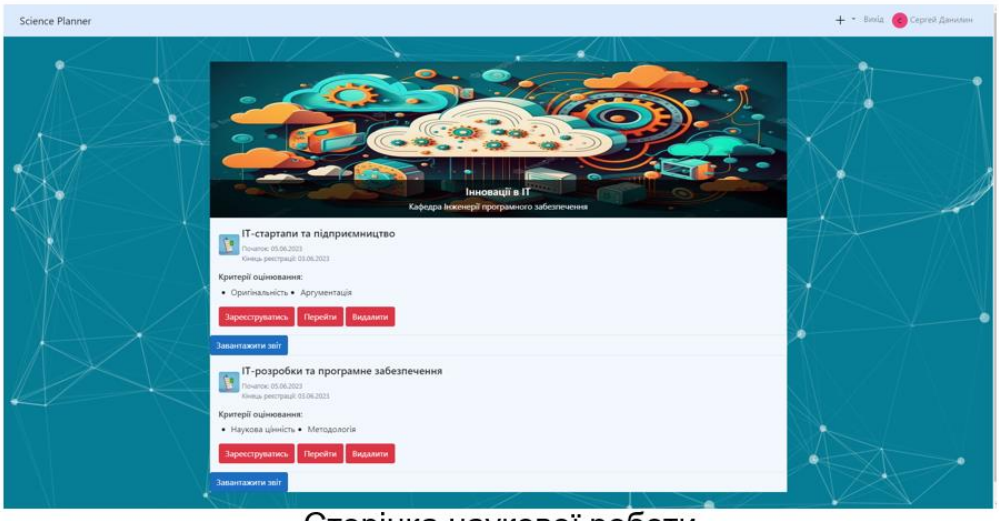


ЕКРАННІ ФОРМИ



Головна сторінка додатку

ЕКРАННІ ФОРМИ



Сторінка наукової роботи

ВИСНОВКИ

Розроблений додаток "Science Planner" успішно відповідає, описаним вимогам:

1. Розроблено структуру бази даних та визначено необхідні таблиці;
2. Розроблено інтерфейс користувача та забезпечено його зручність;
3. Реалізовано можливість планування і обліку наукових робіт;
4. Реалізовано можливість формування і відправлення звітів про виконання наукових робіт.

ДЯКУЮ ЗА УВАГУ!

Додаток Б

Сервер:

```
using Microsoft.AspNetCore.Authentication;
using Microsoft.AspNetCore.Authentication.Cookies;
using Microsoft.AspNetCore.Authentication.Google;
using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Hosting;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.Razor;
using Microsoft.EntityFrameworkCore;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;

namespace SciencePlanner
{
    public class Startup
    {
        public Startup(IConfiguration configuration)
        {
            Configuration = configuration;
        }

        public IConfiguration Configuration { get; }

        // This method gets called by the runtime. Use this method to add services to the
container.
        public void ConfigureServices(IServiceCollection services)
        {
            services.AddControllersWithViews();
            services.AddDbContext<MyDbContext>(options =>
options.UseMySql(Configuration.GetConnectionString("MySqlConnection"), new
MySqlServerVersion("8.0.33")));
            services.AddSession();
            services.AddAuthentication(options =>
             {
                 options.DefaultScheme =
CookieAuthenticationDefaults.AuthenticationScheme;
                 options.DefaultChallengeScheme = GoogleDefaults.AuthenticationScheme;
             })
            .AddCookie(options =>
            {
                options.LoginPath = "/Authentication/Login";
                options.LogoutPath = "/Authentication/Logout";
            })
             .AddGoogle(options =>
             {
                options.ClientId = "824720956549-
de8or77ekrnoau7f5f35h5b24qb9486o.apps.googleusercontent.com";
                options.ClientSecret = "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";
                 options.ClaimActions.MapJsonKey("urn:google:picture", "picture", "url");
             });
            services.AddMvc().SetCompatibilityVersion(CompatibilityVersion.Version_2_1);
            services.Configure<RazorViewEngineOptions>(options =>
            {
                options.ViewLocationFormats.Clear();
                options.ViewLocationFormats.Add("~/UI/Views/{1}/{0}.cshtml");
                options.ViewLocationFormats.Add("~/UI/Views/Shared/{0}.cshtml");
            });
            services.AddSingleton<IConfiguration>(Configuration);
            services.AddHttpContextAccessor();
```

```
services.AddScoped<SciencePlanner.ReportGenerator.CompetitionReportGenerator>();

        }

        // This method gets called by the runtime. Use this method to configure the HTTP
request pipeline.
        public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
        {
            if (env.IsDevelopment())
            {
                app.UseDeveloperExceptionPage();
            }
            else
            {
                app.UseExceptionHandler("/Home/Error");
                app.UseHsts();
            }
            //app.UseSession();
            app.UseHttpsRedirection();
            app.UseStaticFiles();

            app.UseRouting();
            app.UseAuthentication();
            app.UseAuthorization();

            app.UseEndpoints(endpoints =>
            {
                endpoints.MapControllerRoute(
                    name: "default",
                    pattern: "{controller=Home}/{action=Index}/{id?}");
            });
        }
    }
}
```

AuthenticationController.cs:

```
using Microsoft.AspNetCore.Authentication;
using Microsoft.AspNetCore.Authentication.Cookies;
using Microsoft.AspNetCore.Authentication.Google;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using SciencePlanner.DAL.Models;
using System.Linq;
using System.Security.Claims;
using System.Threading.Tasks;

namespace SciencePlanner.BL.Controllers
{
    public class AuthenticationController : Controller
    {
        private readonly MyDbContext _dbContext;

        private IAuthenticationSchemeProvider _schemeProvider;
        public AuthenticationController(IAuthenticationSchemeProvider schemeProvider,
MyDbContext dbContext)
        {
            _schemeProvider = schemeProvider;
            _dbContext = dbContext;
        }
        public IActionResult SignIn()
        {
            return View();
        }
        [HttpGet]
```

```
        [AllowAnonymous]
        public async Task Login()
        {
            await HttpContext.ChallengeAsync(GoogleDefaults.AuthenticationScheme, new
AuthenticationProperties()
            {
                RedirectUri = Url.Action("GoogleResponse")
            });
        }

        public async Task<IActionResult> GoogleResponse()
        {
            var result = await
HttpContext.AuthenticateAsync(CookieAuthenticationDefaults.AuthenticationScheme);
            var claims = result.Principal.Identities
                .FirstOrDefault().Claims.Select(claim => new
                {
                    claim.Issuer,
                    claim.OriginalIssuer,
                    claim.Type,
                    claim.Value
                });
            string userEmail = claims.FirstOrDefault(c => c.Type ==
ClaimTypes.Email)?.Value;
            bool isContained = _dbContext.User.FirstOrDefault(x => x.Email == userEmail)
!= null;
            if (!isContained)
            {
                _dbContext.User.Add(new User
                {
                    Name = claims.FirstOrDefault(c => c.Type == ClaimTypes.Name)?.Value,
                    Email = claims.FirstOrDefault(c => c.Type ==
ClaimTypes.Email)?.Value,
                    Role = "User"
                });
                _dbContext.SaveChanges();
            }
            return RedirectToAction("Index", "Home");
        }

        [Authorize]
        public async Task<IActionResult> Logout()
        {
            await
HttpContext.SignOutAsync(CookieAuthenticationDefaults.AuthenticationScheme);

            HttpContext.Response.Cookies.Delete("ExternalCookie");

            // Clear the existing authentication cookie
            HttpContext.Response.Cookies.Delete(".AspNetCore.Cookies");

            HttpContext.User = new ClaimsPrincipal(new ClaimsIdentity());
            return RedirectToAction("Index", "Home");
        }
    }
}
```

CompetitionController.cs:

```
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using Microsoft.Extensions.Configuration;
using SciencePlanner.BL.Services;
```

```
using SciencePlanner.DAL.Models;
using System.Collections.Generic;
using System.IO;
using System.Linq;

namespace SciencePlanner.BL.Controllers
{
    public class CompetitionController : Controller
    {
        private readonly MyDbContext _dbContext;
        private readonly IConfiguration _configuration;


        public CompetitionController(MyDbContext dbContext, IConfiguration configuration)
        {
            _dbContext = dbContext;
            _configuration = configuration;
        }
        [Authorize]
        public IActionResult AddCompetition(Competition competition)
        {
            if (competition != null)
            {
                competition.ScientificWork = _dbContext.ScientificWork.FirstOrDefault(x
=> x.Id == competition.ScientificWorkId);
                competition.CreatedDate = competition.ScientificWork.CreatedDate;
                competition.EndOfRegistration =
competition.ScientificWork.EndOfRegistration;
                competition.StartDate = competition.ScientificWork.StartDate;
                competition.CloseDate = competition.ScientificWork.CloseDate;

                foreach (var criterion in competition.EvaluationCriteria)
                {
                    _dbContext.EvaluationCriterion.Add(criterion);
                }

                _dbContext.Competition.Add(competition);
                _dbContext.SaveChanges();
            }

            return RedirectToAction("Index", "Home");
        }


        [Authorize]
        public IActionResult AddCategory(List<string> evaluationCriterion)
        {
            foreach (var ev in evaluationCriterion)
            {
                var criterion = new EvaluationCriterion { Name = ev };
                _dbContext.EvaluationCriterion.Add(criterion);
            }
            _dbContext.SaveChanges();

            return RedirectToAction("Index", "Home");
        }

        [Authorize]
        public IActionResult RemoveCompetition(int id)
        {
            var competition = _dbContext.Competition.Include(c =>
c.RegisteredMembers).FirstOrDefault(x => x.Id == id);

            if (competition != null)
            {
```

```
                var registeredMembers = competition.RegisteredMembers.ToList();

                _dbContext.RegisteredMember.RemoveRange(registeredMembers);
                _dbContext.Competition.Remove(competition);
                _dbContext.SaveChanges();
            }

            return RedirectToAction("Index", "Home");
        }

        [Authorize]
        public IActionResult RegisterOnCource(RegisteredMember registeredMember,
IFormFile fileUpload, Competition competition)
        {
            using (var binaryReader = new BinaryReader(fileUpload.OpenReadStream()))
            {
                registeredMember.File = binaryReader.ReadBytes((int)fileUpload.Length);
                registeredMember.User = _dbContext.User.FirstOrDefault(x => x.Id ==
registeredMember.UserId);
                var comp = _dbContext.Competition.FirstOrDefault(x => x.Id ==
competition.Id);
                var scientifcWork = _dbContext.ScientificWork.FirstOrDefault(x => x.Id ==
comp.ScientificWorkId);
                EmailSendler emailSendler = new EmailSendler(_dbContext, _configuration);
                comp.RegisteredMembers.Add(registeredMember);
                _dbContext.SaveChanges();
                emailSendler.SendEmailToAdmin(registeredMember, scientifcWork, comp);
            }
            return RedirectToAction("Index", "Home");
        }

    }
}
```

CompetitionWorkController.cs:

```
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using SciencePlanner.DAL.Models;
using System.Linq;
using System.Security.Claims;

namespace SciencePlanner.BL.Controllers
{
    public class CompetitionWorkController : Controller
    {
        private readonly MyDbContext _dbContext;

        public CompetitionWorkController(MyDbContext dbContext)
        {
            _dbContext = dbContext;
        }

        public IActionResult AddCompetitionWork(int id, int competitionId)
        {
            RegisteredMember registeredMember =
_dbContext.RegisteredMember.FirstOrDefault(x => x.Id == id);
            CompetitionWork competitionWork = new CompetitionWork();
            competitionWork.File = registeredMember.File;
            competitionWork.CompetitionId = competitionId;
            competitionWork.RegisteredMemberId = registeredMember.Id;

            _dbContext.CompetitionWorks.Add(competitionWork);
            _dbContext.SaveChanges();

            return RedirectToAction("Competition", "Home");
```

```
        }

        [Authorize]
        [HttpPost]
        public IActionResult Vote(int workId)
        {
            var userId = User.FindFirst(ClaimTypes.NameIdentifier)?.Value;

            var competitionWork = _dbContext.CompetitionWorks.FirstOrDefault(cw => cw.Id
== workId);

            if (competitionWork != null && competitionWork.VoteCount == 0)
            {
                competitionWork.VoteCount = 1;

                _dbContext.SaveChanges();
            }

            return RedirectToAction("Index", "Home");
        }
    }
}
```

HomeController.cs:

```
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using SciencePlanner.DAL.Models;
using System.Collections.Generic;
using System.Linq;
using System.Security.Claims;

namespace SciencePlanner.BL.Controllers
{
    public class HomeController : Controller
    {
        private readonly MyDbContext _dbContext;

        public HomeController(MyDbContext dbContext)
        {
            _dbContext = dbContext;
        }
        public IActionResult Index()
        {
            if (User.Identity.IsAuthenticated)
            {
                var name = User.FindFirstValue(ClaimTypes.Name);
                ViewBag.Name = name;
                var initials = GetInitials(name);

                // Создание URL-адреса аватара с использованием ui-avatars.com
                var avatarUrl = $"https://ui-
avatars.com/api/?name={initials}&size=48&background=random&rounded=true";

                ViewBag.Avatar = avatarUrl;
            }
            return View(_dbContext.ScientificWork.ToList());
        }

        private string GetInitials(string name)
        {
            if (!string.IsNullOrEmpty(name))
            {
                return name[0].ToString();
```

```
            }
            return string.Empty;
        }
        [HttpGet]
        public ActionResult GetUsers()
        {
            List<string> userList = new List<string>();

            userList = _dbContext.User.Select(x => x.Email).ToList();

            return Json(userList);
        }

        [Authorize]
        public IActionResult ScientificWork(ScientificWork scientificWork)
        {
            if (User.Identity.IsAuthenticated)
            {
                var name = User.FindFirstValue(ClaimTypes.Name);

                ViewBag.Name = name;
                var initials = GetInitials(name);


                var avatarUrl = $"https://ui-
avatars.com/api/?name={initials}&size=48&background=random&rounded=true";

                ViewBag.Avatar = avatarUrl;
            }

            if (scientificWork != null)
            {
                ViewBag.ScientificWorkId = scientificWork.Id;
                ViewBag.ScientificWorkTitle = scientificWork.Title;
                ViewBag.ScientificWorkDescription = scientificWork.Description;
                ViewBag.ScientificWorkDepartament = scientificWork.Department;
                var competitions = _dbContext.Competition.Include(c =>
c.EvaluationCriteria).Include(x => x.RegisteredMembers).Include(a =>
a.ScientificWork).ToList();

                return View(competitions);
            }
            return RedirectToAction("Index");
        }
        [Authorize]
        public IActionResult CreateCompetition(int id)
        {
            if (User.Identity.IsAuthenticated)
            {
                var name = User.FindFirstValue(ClaimTypes.Name);

                ViewBag.Name = name;
                var initials = GetInitials(name);

                // Создание URL-адреса аватара с использованием ui-avatars.com
                var avatarUrl = $"https://ui-
avatars.com/api/?name={initials}&size=48&background=random&rounded=true";

                ViewBag.Avatar = avatarUrl;
            }
            return View();
        }
        [Authorize]
        public IActionResult RegisterView(int id)
        {
```

```
            if (User.Identity.IsAuthenticated)
            {
                var name = User.FindFirstValue(ClaimTypes.Name);

                ViewBag.Name = name;
                ViewBag.CompetitionId = id;
                var initials = GetInitials(name);

                // Создание URL-адреса аватара с использованием ui-avatars.com
                var avatarUrl = $"https://ui-
avatars.com/api/?name={initials}&size=48&background=random&rounded=true";

                ViewBag.Avatar = avatarUrl;
            }
            return View();
        }
        [Authorize]
        public IActionResult Competition(int id)
        {
            if (User.Identity.IsAuthenticated)
            {
                var name = User.FindFirstValue(ClaimTypes.Name);
                ViewBag.Name = name;
                var initials = GetInitials(name);

                // Создание URL-адреса аватара с использованием ui-avatars.com
                var avatarUrl = $"https://ui-
avatars.com/api/?name={initials}&size=48&background=random&rounded=true";

                ViewBag.Avatar = avatarUrl;
            }
            ViewBag.CompetitionId = id;
            var competitionWorks = _dbContext.CompetitionWorks.ToList();
            return View(competitionWorks);
        }

        [Authorize]
        public IActionResult AddAdmin(User user)
        {
            _dbContext.User.Add(user);
            _dbContext.SaveChanges();
            return RedirectToAction("Index", "Home");
        }

        [Authorize]
        public IActionResult RemoveAdmin(string email)
        {
            var user = _dbContext.User.FirstOrDefault(x => x.Email == email);
            if (user != null && user.Role == "Admin")
            {
                _dbContext.User.Remove(user);
                _dbContext.SaveChanges();
            }

            return RedirectToAction("Index", "Home");
        }
    }
}
```

ScientificWorkController.cs:

```
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using Microsoft.Extensions.Configuration;
using SciencePlanner.BL.Services;
```

```
using SciencePlanner.DAL.Models;
using System;
using System.Globalization;
using System.Linq;

namespace SciencePlanner.BL.Controllers
{
    public class ScientificWorkController : Controller
    {
        private readonly MyDbContext _dbContext;
        private readonly IConfiguration _configuration;

        public ScientificWorkController(MyDbContext dbContext, IConfiguration
configuration)
        {
            _dbContext = dbContext;
            _configuration = configuration;
        }
        [Authorize]
        [HttpPost]
        public IActionResult AddScientificWork(ScientificWork scientificWork)
        {
            scientificWork.CreatedDate = DateTime.Today.Date.ToShortDateString();
            scientificWork.EndOfRegistration =
scientificWork.EndOfRegistration.Replace("-", ".");
            string endDateString = scientificWork.EndOfRegistration;
            DateTime originalDate = DateTime.ParseExact(endDateString, "yyyy.MM.dd",
CultureInfo.InvariantCulture);
            scientificWork.EndOfRegistration = originalDate.ToString("dd.MM.yyyy");

            scientificWork.StartDate = scientificWork.StartDate.Replace("-", ".");
            string todayDateString = scientificWork.StartDate;
            DateTime originalsDate = DateTime.ParseExact(todayDateString, "yyyy.MM.dd",
CultureInfo.InvariantCulture);
            scientificWork.StartDate = originalsDate.ToString("dd.MM.yyyy");

            scientificWork.CloseDate = scientificWork.CloseDate.Replace("-", ".");
            DateTime originalDate1 = DateTime.ParseExact(scientificWork.CloseDate,
"yyyy.MM.dd", CultureInfo.InvariantCulture);
            scientificWork.CloseDate = originalDate1.ToString("dd.MM.yyyy");

            _dbContext.ScientificWork.Add(scientificWork);
            _dbContext.SaveChanges();
            EmailSendler emailSendler = new EmailSendler(_dbContext, _configuration);
            emailSendler.SendEmailToAllUsers(scientificWork);
            return RedirectToAction("Index", "Home");
        }
        [Authorize]
        public IActionResult RemoveScientificWork(ScientificWork experiment)
        {
            if (experiment != null)
            {
                // Удалить связанных RegisteredMember
                var registeredMembers = _dbContext.RegisteredMember
                    .Where(rm => rm.Competitions.Any(c => c.ScientificWorkId ==
experiment.Id))
                    .ToList();

                _dbContext.RegisteredMember.RemoveRange(registeredMembers);
                _dbContext.ScientificWork.Remove(experiment);
                _dbContext.SaveChanges();
            }

            return RedirectToAction("Index", "Home");
        }
```

```
    }
}
```

Сервіси:

CompetitionReportGenerator.cs:

```
using System.Collections.Generic;
using System.Linq;
using DocumentFormat.OpenXml;
using DocumentFormat.OpenXml.Packaging;
using DocumentFormat.OpenXml.Wordprocessing;
using Microsoft.AspNetCore.Mvc;
using SciencePlanner.DAL.Models;
using Document = DocumentFormat.OpenXml.Wordprocessing.Document;

namespace SciencePlanner.ReportGenerator
{
    public class CompetitionReportGenerator : Controller
    {
        public IActionResult GenerateWinnerReport(Competition competition, string Group,
string Name, string ScientificWork, string Department, List<string> EvaluationCriteria,
string CloseDate, string EndOfRegistration)
        {
                using (WordprocessingDocument document =
WordprocessingDocument.Create("WinnerReport.docx", WordprocessingDocumentType.Document))
                {
                    MainDocumentPart mainPart = document.AddMainDocumentPart();
                    mainPart.Document = new Document();
                    Body body = mainPart.Document.AppendChild(new Body());

                    Paragraph titleParagraph = body.AppendChild(new Paragraph());
    Run titleRun = titleParagraph.AppendChild(new Run());
    titleRun.AppendChild(new Text("ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ"));

    titleParagraph.ParagraphProperties = new ParagraphProperties(new Justification() {
Val = JustificationValues.Center });

    titleRun.RunProperties = new RunProperties(new Bold(), new FontSize() { Val = "36"
});
    titleParagraph.AppendChild(new Break());

    titleParagraph = body.AppendChild(new Paragraph());
    titleRun = titleParagraph.AppendChild(new Run());
    titleRun.AppendChild(new Text("Звіт з проведеного конкурсу"));

    titleParagraph.ParagraphProperties = new ParagraphProperties(new Justification() {
Val = JustificationValues.Center });

    titleRun.RunProperties = new RunProperties(new FontSize() { Val = "28" });

    titleParagraph = body.AppendChild(new Paragraph());
    titleRun = titleParagraph.AppendChild(new Run());
    titleRun.AppendChild(new Text("Кафедра: " + Department));

    titleParagraph.ParagraphProperties = new ParagraphProperties(new Justification() {
Val = JustificationValues.Center });

    titleRun.RunProperties = new RunProperties(new FontSize() { Val = "28" });

                    Paragraph competitionInfoParagraph = body.AppendChild(new
Paragraph());
```

```
                    competitionInfoParagraph.AppendChild(new Run(new Text("Назва
конкурсу: " + ScientificWork)));

                    competitionInfoParagraph = body.AppendChild(new Paragraph());
                    competitionInfoParagraph.AppendChild(new Run(new Text("Категорія: " +
competition.Category)));

                    competitionInfoParagraph = body.AppendChild(new Paragraph());
                    competitionInfoParagraph.AppendChild(new Run(new Text("Критерії
оцінювання: ")));

                    Run criteriaRun = competitionInfoParagraph.AppendChild(new Run());
                    foreach (var criterion in EvaluationCriteria)
                    {
                        criteriaRun.AppendChild(new Text(criterion));

                        if (criterion != EvaluationCriteria.Last())
                        {
                            criteriaRun.AppendChild(new Text(", "));
                        }
                    }

                    competitionInfoParagraph = body.AppendChild(new Paragraph());
                    competitionInfoParagraph.AppendChild(new Run(new Text("Дата
створення: " + competition.CreatedDate)));

                    competitionInfoParagraph = body.AppendChild(new Paragraph());
                    competitionInfoParagraph.AppendChild(new Run(new Text("Кінець
реєстрації: " + EndOfRegistration)));

                    competitionInfoParagraph = body.AppendChild(new Paragraph());
                    competitionInfoParagraph.AppendChild(new Run(new Text("Дата
закінчення: " + CloseDate)));

                    Paragraph winnerInfoParagraph = body.AppendChild(new Paragraph());
                    winnerInfoParagraph.AppendChild(new Run(new Text("Переможець: " +
Name)));

                    winnerInfoParagraph = body.AppendChild(new Paragraph());
                    winnerInfoParagraph.AppendChild(new Run(new Text("Група: " +
Group)));

                    mainPart.Document.Save();

                    return DownloadWinnerReport();
                }

        }
        public IActionResult DownloadWinnerReport()
        {
            string path = "C:\\Users\\danil\\Downloads\\SciencePlanner-
master\\SciencePlanner-master\\SciencePlanner\\WinnerReport.docx";
            return PhysicalFile(path, "application/vnd.openxmlformats-
officedocument.wordprocessingml.document", "WinnerReport");
        }

    }
}
```

EmailSendler.cs:

```
using SciencePlanner.DAL.Models;
using System.Net.Mail;
using System.Linq;
using System.Net;
```

```
using Microsoft.Extensions.Configuration;
using System.Security.Claims;
using System;
using System.IO;

namespace SciencePlanner.BL.Services
{
    public  class EmailSendler
    {
        private readonly MyDbContext _dbContext;
        private readonly IConfiguration _configuration;

        public EmailSendler(MyDbContext dbContext, IConfiguration configuration)
        {
            _dbContext = dbContext;
            _configuration = configuration;
        }

        public void SendEmailToAllUsers(ScientificWork scientificWork)
        {
            MailAddress from = new
MailAddress(_configuration["EmailConfiguration:SenderEmail"], "Сергій Данилін");
            var users = _dbContext.User.ToList();
            foreach (var user in users)
            {
                var to = new MailAddress(user.Email);
                MailMessage message = new MailMessage(from, to);
                message.Subject = scientificWork.Type;
                string scientificWorkUrl =
$"{_configuration["BaseUrl"]}Home/ScientificWork/{scientificWork.Id}?name={scientificWork
.Title}&description={scientificWork.Description}&department={scientificWork.Department}";
                message.Body = $"Було створенно нову наукову роботу:
{scientificWork.Title}<p><a href='{scientificWorkUrl}' style='display: inline-block;
padding: 10px 20px; background-color: #007bff; color: #fff;" +
                    $" text-decoration: none;'>Перейти</a></p>";

                message.IsBodyHtml = true;
                SmtpClient smtpClient = new SmtpClient("smtp.gmail.com", 587);
                smtpClient.Credentials = new
NetworkCredential(_configuration["EmailConfiguration:SenderEmail"],
_configuration["EmailConfiguration:SenderPassword"]);
                smtpClient.EnableSsl = true;
                smtpClient.Send(message);
            }
        }
        public void SendEmailToAdmin(RegisteredMember registeredMember, ScientificWork
scientificWork, Competition competition)
        {
            var from = new MailAddress(registeredMember.User.Email,
registeredMember.User.Name);
            var to = new MailAddress(_configuration["EmailConfiguration:SenderEmail"]);
            var message = new MailMessage(from, to);

            message.Subject = "Відправлено наукову роботу";

            // Добавление информации об отправителе в теле сообщения
            var body = $"Відправник: {registeredMember.User.Name}<br>";
            body += $"№ Відправника: {registeredMember.Id}<br>";
            body += $"Email Відправника: {registeredMember.User.Email}<br>";

            // Создание вложения с научной работой
            var attachment = new Attachment(new MemoryStream(registeredMember.File),
"Научная работа.pdf");

            // Добавление вложения к сообщению
```

```
            message.Attachments.Add(attachment);

            // Добавление текста и ссылки в тело сообщения
            var scientificWorkUrl =
$"{_configuration["BaseUrl"]}Home/ScientificWork/{scientificWork.Id}?name={scientificWork
.Title}&description={scientificWork.Description}&department={scientificWork.Department}";
            body += $"<br>Відправлено наукову роботу до участі у конкурсі наукових робіт
студентів {scientificWork.Title} у категорії {competition.Category}<br>";
            body += $"<a href='{scientificWorkUrl}' style='display: inline-block;
padding: 10px 20px; background-color: #007bff; color: #fff; text-decoration:
none;'>Перейти</a>";

            message.Body = body;
            message.IsBodyHtml = true;

            // Отправка сообщения
            using (var smtpClient = new SmtpClient("smtp.gmail.com", 587))
            {
                smtpClient.Credentials = new
NetworkCredential(_configuration["EmailConfiguration:SenderEmail"],
_configuration["EmailConfiguration:SenderPassword"]);
                smtpClient.EnableSsl = true;
                smtpClient.Send(message);
            }
        }
    }
}
```

Клієнт:

Competition.cshtml:

```
@inject MyDbContext myDbContext

@model IEnumerable<CompetitionWork>

@using System.Linq;
@using System.Security.Claims;
<script>
    var elements = document.getElementsByClassName("dropdown-item");
    var element = Array.from(elements).filter((e) => e.innerHTML === "Створити
студентський конкурс")
    element[0].innerText = "Додати роботу"
    function openFullscreen(fileData) {
        const newWindow = window.open('', '_blank');
        const embedCode = `<embed src="${fileData}" type="application/pdf" style="width:
100%; height: 100vh;">`;

        newWindow.document.write(embedCode);
        newWindow.document.close();
    }
</script>

<div class="container">
    <h1>Конкурсні работи</h1>

    <table class="table table-striped">
        <thead>
            <tr>
                <th>Назва роботи</th>
                <th>Автор</th>
                <th>Група</th>
                <th>Робота</th>
```

```
            </tr>
        </thead>
        <tbody>
            @foreach (var work in Model)
            {
                if (work.CompetitionId == ViewBag.CompetitionId)
                {
                    var registeredMember = myDbContext.RegisteredMember.FirstOrDefault(rm
=> rm.Id == work.RegisteredMemberId);
                    registeredMember.User = myDbContext.User.FirstOrDefault(x => x.Id ==
registeredMember.UserId);
                    var user = registeredMember.User;

                    <tr>
                        <td>@registeredMember.FileName</td>
                        <td>@user.Name</td>
                        <td>@registeredMember.Group</td>
                        <td>
                            @{
                                var fileData = "data:application/pdf;base64," +
Convert.ToBase64String(work.File);
                                <a href="#" class="btn btn-light"
onclick="openFullscreen('@fileData')">Відкрити роботу</a>
                            }
                        </td>
                        <td>
                            <form method="post" asp-action="Vote" asp-
controller="CompetitionWork">
                                <input type="hidden" name="workId" value="@work.Id" />
                                <button type="submit" class="btn btn-
primary">Проголосувати</button>
                            </form>
                        </td>
                    </tr>
                }
            }
        </tbody>
    </table>
</div>
<div class="modal fade" id="createCourseModal" tabindex="-1" aria-
labelledby="createCourseModalLabel" aria-hidden="true">
    <div class="modal-dialog">
        <div class="modal-content">
            <div class="modal-header">
                <h5 class="modal-title" id="createCourseModalLabel">Реєстрація на
конкурс</h5>
                <button type="button" class="close" data-dismiss="modal" aria-
label="Close">
                    <span aria-hidden="true">&times;</span>
                </button>
            </div>
            <div class="modal-body">
                <form method="post" asp-action="AddCompetitionWork" asp-
controller="CompetitionWork" enctype="multipart/form-data">
                    <input type="hidden" name="competitionId"
value="@ViewBag.CompetitionId">
                    <div class="form-group">
                        <label for="RegisteredMemberId" style="color: #000; font-weight:
bold;">№ Учасника</label>
                        <input type="number" class="form-control" id="RegisteredMemberId"
name="id">
                    </div>
                    <div class="form-group text-center">
                        <button type="submit" class="btn btn-danger">Додати</button>
                    </div>
```

```
                </form>
            </div>
        </div>
    </div>
</div>
```

Index.cshtml:

```
@using System.Security.Claims;
@inject MyDbContext myDbContext
@model IEnumerable<ScientificWork>

    @{
        ViewData["Title"] = "Наукові роботи";
    }
    @if (Model != null)
    {
    @foreach (var scientificWork in Model)
    {
            <form method="post" class="card" style="width: 18rem;">
            <input type="hidden" name="Id" value="@scientificWork.Id" />
            <input type="hidden" name="Title" value="@scientificWork.Title" />
            <input type="hidden" name="Description" value="@scientificWork.Description"
/>
            <input type="hidden" name="Department" value="@scientificWork.Department" />
            <img src="/img/Experiment/rsz_11blue-background-with-books_872147-1357.jpg"
class="card-img-top" alt="img" style="height: 150px;">
                <div class="card-body">
                <h5 class="card-title">@scientificWork.Type</h5>
                <div class="d-flex justify-content-between">
                    <small>Кінець реєстрації: @scientificWork.EndOfRegistration</small>
                    <small>Дата початку: @scientificWork.StartDate</small>
                </div>
                <h5 class="card-title">@scientificWork.Title</h5>
                <p class="card-text">@scientificWork.Description</p>
                </div>
                <ul class="list-group list-group-flush">
                <li class="list-group-item">@scientificWork.Department</li>
                </ul>
            <div class="card-body d-flex justify-content-between">
                <button asp-controller="Home" asp-action="ScientificWork" class="btn mr-
2" type="submit">Перейти</button>
                @{
                    string email = User.FindFirstValue(ClaimTypes.Email);
                    bool userExistsWithRole = myDbContext.User.Any(u => u.Email == email
&& u.Role == "Admin");
                    if (userExistsWithRole)
                    {   <!--
                        <button asp-controller="Experiment" asp-action="EditExperiment"
class="btn" type="submit">Змінити</button>
                        -->
                        <button asp-controller="ScientificWork" asp-
action="RemoveScientificWork" class="btn" type="submit">Видалити</button>
                    }
                }
             </div>
            </form>
        }
    }
```

ScientificWork.cshtml:

```
@model IEnumerable<Competition>
@using Microsoft.AspNetCore.Http;
@using System.Security.Claims;
@using System.Globalization;
@inject MyDbContext myDbContext

@{
    ViewData["Title"] = @ViewBag.ScientificWorkTitle;
}

<script>
    var elements = document.getElementsByClassName("dropdown-item");
    var element = Array.from(elements).filter((e) => e.innerHTML === "Створити
студентський конкурс")
    element[0].innerText = "Додати категорію"

    var currentIndex = -1;

    function setIndex(checkbox) {
        if (checkbox.checked) {
            currentIndex++;
            checkbox.name = "EvaluationCriteria[" + currentIndex + "].Name";
            document.getElementById("selectedIndex").value = currentIndex;
        } else {
            checkbox.name = "EvaluationCriteria[]";
            if (currentIndex === checkbox.value) {
                currentIndex--;
                document.getElementById("selectedIndex").value = currentIndex;
            }
        }
    }
</script>

<div class="mb-3 w-100">
    <div style="position: relative;">
        <img src="/img/Experiment/rsz_Безымянный1.png" class="card-img-top" alt="img"
style="height: 300px;">
        <div style="position: absolute; bottom: 0; left: 0; width: 100%; background-
color: rgba(0, 0, 0, 0.7); text-align: center; padding: 10px;">
            <h5 class="card-title" style="color: white; margin-bottom:
0;">@ViewBag.ScientificWorkTitle</h5>
            <p class="card-text" style="color: white; margin-top:
5px;">@ViewBag.ScientificWorkDepartament</p>
        </div>
    </div>
    <div class="card-body d-flex p-0" style="background-color: #F3F8FE;">
        <div class="flex-grow-1" style="width: 85%; height: 100%;">
            @if (Model != null)
            {
                @foreach (var competition in Model.Where(x => x.ScientificWorkId ==
ViewBag.ScientificWorkId))
                {
                        <div class="p-3 border rounded" role="alert">
                            <div class="d-flex align-items-center">
                                <img src="/img/Work/3d-clipboard-with-todo-checklist-
paper-green-tick.png" class="rounded me-2" alt="work" style="width:40px; height: 40px;
margin-right: 5px;">
                                <div>
                                    <h5>@competition.Category</h5>
                                    <small class="text-muted d-block">Початок:
@competition.StartDate</small>
                                    <small class="text-muted d-block">Кінець реєстрації:
@competition.EndOfRegistration</small>
                                </div>
```

```
                            </div>
                            <div class="mt-3">
                                <h6>Критерії оцінювання:</h6>
                                <ul class="d-flex" style="width: 100%;">
                                    @foreach (var criterion in
competition.EvaluationCriteria.Where(x => x.CompetitionId == competition.Id))
                                    {
                                        <li class="ml-4">@criterion.Name</li>
                                    }
                                </ul>
                                @{
                                    DateTime currentDate = DateTime.Now;
                                    DateTime endOfRegistrationDate =
DateTime.ParseExact(competition.EndOfRegistration, "dd.MM.yyyy",
CultureInfo.InvariantCulture);
                                    DateTime startDate =
DateTime.ParseExact(competition.StartDate, "dd.MM.yyyy", CultureInfo.InvariantCulture);
                                }

                                @if (currentDate < endOfRegistrationDate)
                                {
                                    <a id="registerButton-@competition.Id" asp-
controller="Home" asp-action="RegisterView" asp-route-id="@competition.Id" class="btn
btn-danger">Зареєструватись</a>
                                }
                                else if (currentDate > startDate)
                                {
                                <a id="registerButton-@competition.Id" asp-
controller="Home" asp-action="Competition" asp-route-id="@competition.Id" asp-route-
competition="@competition" class="btn btn-danger">Перейти</a>
                                }
                                else
                                {
                                    <p>Реєстрація закрита</p>
                                }
                            <a id="registerButton-@competition.Id" asp-controller="Home"
asp-action="Competition" asp-route-id="@competition.Id" asp-route-
competition="@competition" class="btn btn-danger">Перейти</a>
                                @{
                                    string email = User.FindFirstValue(ClaimTypes.Email);
                                    bool userExistsWithRole = myDbContext.User.Any(u =>
u.Email == email && u.Role == "Admin");
                                    if (userExistsWithRole)
                                    {
                                    <a asp-controller="Competition" asp-
action="RemoveCompetition" asp-route-id="@competition.Id" type="submit" class="btn btn-
danger">Видалити</a>
                                }
                            }
                        </div>
                    </div>

                    <div>
                        <form method="post" enctype="multipart/form-data">
                            <input type="hidden" name="Category"
value="@competition.Category"></input>
                            <input type="hidden" name="CreatedDate"
value="@competition.CreatedDate"></input>
                            <input type="hidden" name="CloseDate"
value="@competition.CloseDate"></input>
                            <input type="hidden" name="StartDate"
value="@competition.StartDate"></input>
                            <input type="hidden" name="EndOfRegistration"
value="@competition.EndOfRegistration"></input>
```

```
                            <input type="hidden" name="ScientificWork"
value="@competition.ScientificWork.Title"></input>
                            <input type="hidden" name="Department"
value="@competition.ScientificWork.Department"></input>
                            @foreach (var criterion in competition.EvaluationCriteria)
                            {
                            <input type="hidden" name="EvaluationCriteria"
value="@criterion.Name"></input>

                            }
                            @{
                                var winner = competition.RegisteredMembers
                                                        .OrderByDescending(m =>
m.CompetitionWorks.Sum(cw => cw.VoteCount))
                                                        .FirstOrDefault();
                                <!-- Тут ошибка var user =
myDbContext.User.FirstOrDefault(x => x.Id == winner.UserId);
                                var Name = user.Name;-->


                                if (winner != null)
                                {
                                    <input type="hidden" name="Group"
value="@winner.Group"></input>
                                    <!-- Тут ошибка<input type="hidden" name="Name"
value="Name"></input> -->

                                }
                            }

                            <button type="submit"
formaction="@Url.Action("GenerateWinnerReport", "CompetitionReportGenerator")" class="btn
btn-primary">Завантажити звіи</button>
                        </form>
                    </div>
                }
            }
        </div>
    </div>
</div>

<div class="modal fade" id="createCourseModal" tabindex="-1" aria-
labelledby="createCourseModalLabel" aria-hidden="true">
    <div class="modal-dialog">
        <div class="modal-content">
            <div class="modal-header">
                <h5 class="modal-title" id="createCourseModalLabel">Створення
студентського конкурсу</h5>
                <button type="button" class="close" data-dismiss="modal" aria-
label="Close">
                    <span aria-hidden="true">&times;</span>
                </button>
            </div>
            <div class="modal-body">
                <form method="post" asp-controller="Competition" asp-
action="AddCompetition" enctype="multipart/form-data">
                    <input type="hidden" name="ScientificWorkId"
value="@ViewBag.ScientificWorkId">
                    <h1 class="text-center">Створення категорії конкурсу</h1>
                    <div class="mb-3">
                        <label for="title" class="form-label">Категорія</label>
                        <input type="text" class="form-control" id="title"
name="Category" required>
                    </div>
                    @{
```

```
                        string[] evaluationCriteria = new string[]
                        {
                    "Наукова цінність",
                    "Оригінальність",
                    "Методологія",
                    "Структура",
                    "Аргументація",
                    "Рівень антиплагіату"
                        };
                    }
                    <div class="mb-3">
                        <h5>Критерії оцінювання</h5>
                        @for (int i = 0; i < evaluationCriteria.Length; i++)
                        {
                            <label>
                                <input type="checkbox" name="EvaluationCriteria[]"
value="@evaluationCriteria[i]" onclick="setIndex(this)" />
                                @evaluationCriteria[i]
                            </label>
                        }
                        <input type="hidden" id="selectedIndex" name="SelectedIndex"
value="-1" />
                    </div>
                    <button type="submit" class="btn btn-primary">Додати</button>
                </form>
            </div>
        </div>
    </div>
</div>
```