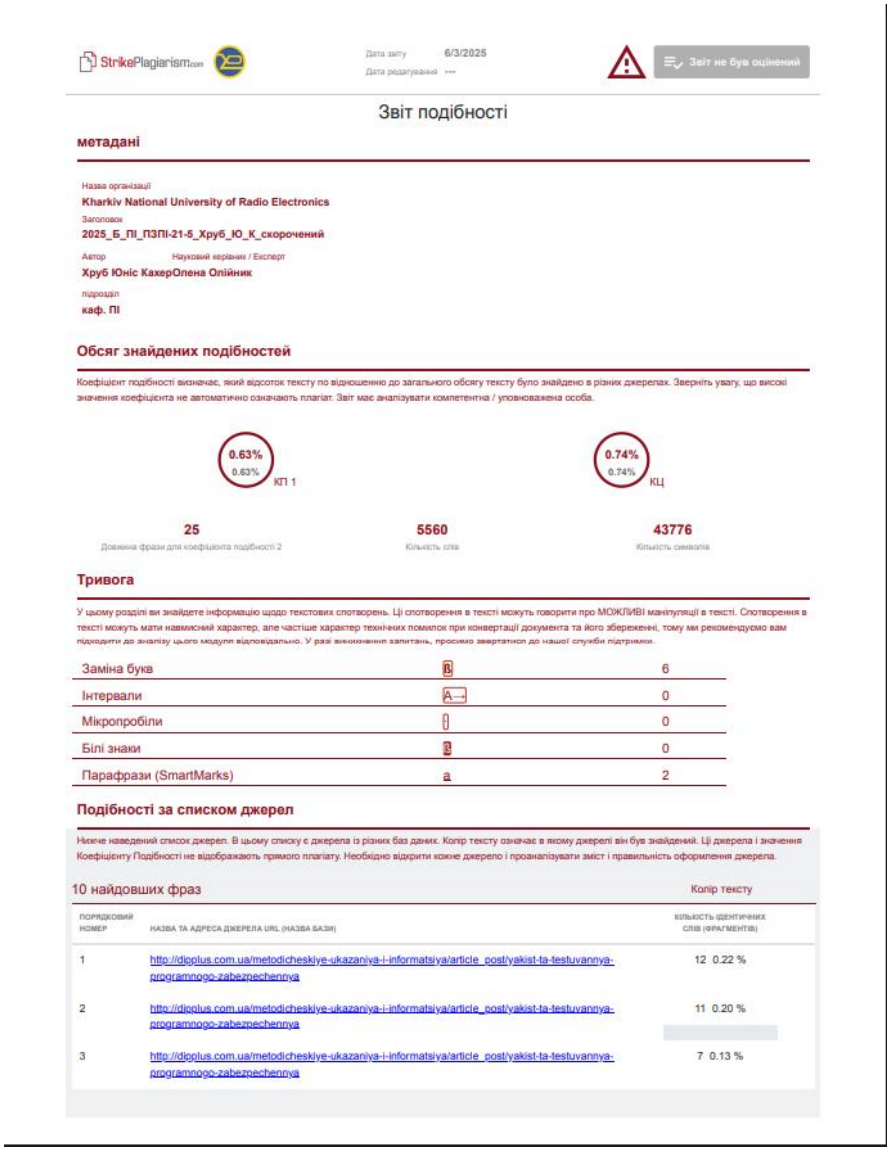
ДОДАТОК А

Звіт результатів перевірки на унікальність тексту в базі ХНУРЕ



StrikePlagiarism.com

Дата звіту 6/3/2025
Дата редагування ---

Звіт не був оцінений

Звіт подібності

метадані

Назва організації
Kharkiv National University of Radio Electronics
Заголовок
2025_Б_ПІ_ПЗПІ-21-5_Хруб_Ю_К_скорочений
Автор Науковий керівник / Експерт
Хруб Юніс КахерОлена Олійник
підрозділ
каф. ПІ

Обсяг знайдених подібностей

Коефіцієнт подібності визначає, який відсоток тексту по відношенню до загального обсягу тексту було знайдено в різних джерелах. Зверніть увагу, що високі значення коефіцієнта не автоматично означають плагіат. Звіт має аналізувати компетентна / уповноважена особа.

0.63% 0.63% КП 1

0.74% 0.74% КЦ

25 Довжина фрази для коефіцієнта подібності 2

5560 Кількість слів

43776 Кількість символів

Тривога

У цьому розділі ви знайдете інформацію щодо текстових спотворень. Ці спотворення в тексті можуть говорити про МОЖЛИВІ маніпуляції в тексті. Спотворення в тексті можуть мати навмисний характер, але частіше характер технічних помилок при конвертації документа та його збереженні, тому ми рекомендуємо вам підходити до аналізу цього модуля відповідально. У разі виникнення запитань, просимо звертатися до нашої служби підтримки.

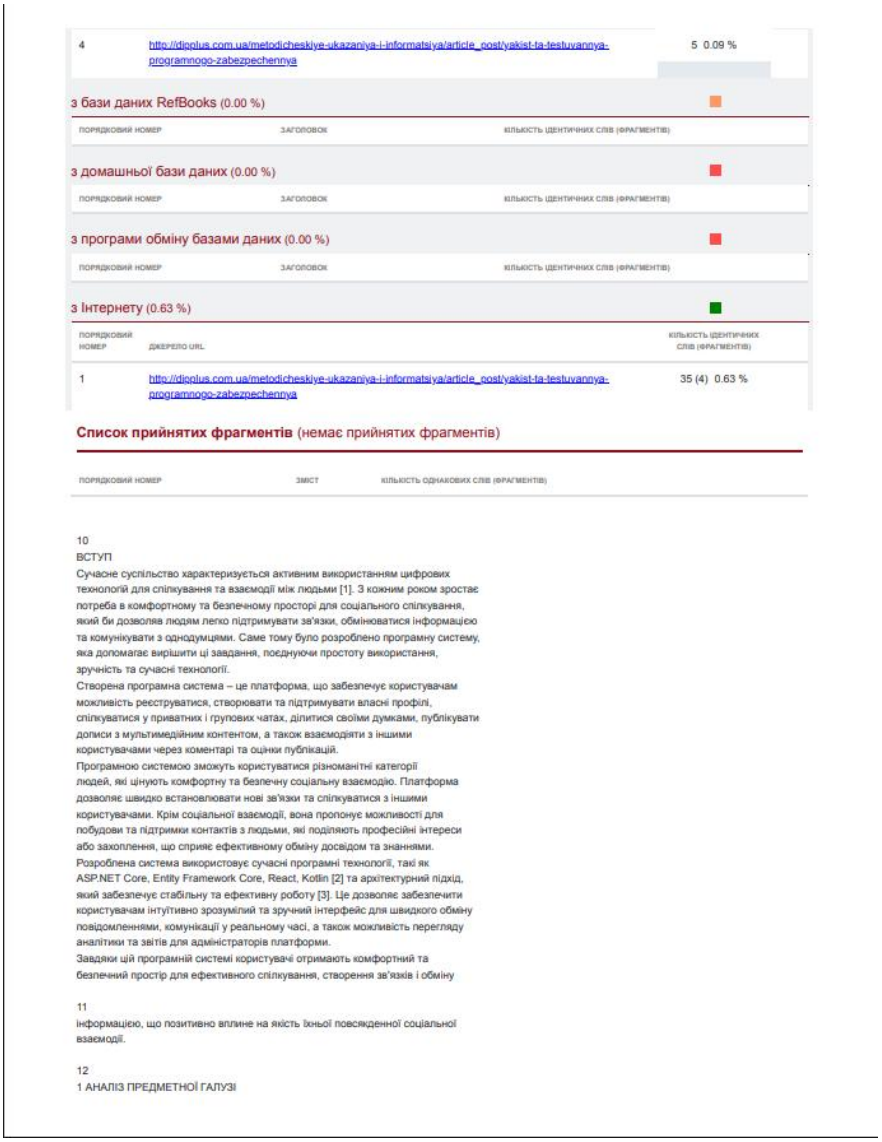
Заміна букв	В	6
Інтервали	A	0
Мікропробіли	;	0
Білі знаки	;	0
Парафрази (SmartMarks)	a	2

Подібності за списком джерел

Нижче наведений список джерел. В цьому списку є джерела із різних баз даних. Колір тексту означає в якому джерелі він був знайдений. Ці джерела і значення Коефіцієнту Подібності не відображають прямого плагіату. Необхідно відкрити кожне джерело і проаналізувати зміст і правильність оформлення джерела.

10 найдовших фраз — Колір тексту

ПОРЯДКОВИЙ НОМЕР	НАЗВА ТА АДРЕСА ДЖЕРЕЛА URL (НАЗВА БАЗИ)	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	http://dipplus.com.ua/metodicheskiye-ukazaniya-i-informatsiya/article_post/yakist-ta-testuvannya-programnogo-zabezpechennya	12 0.22 %
2	http://dipplus.com.ua/metodicheskiye-ukazaniya-i-informatsiya/article_post/yakist-ta-testuvannya-programnogo-zabezpechennya	11 0.20 %
3	http://dipplus.com.ua/metodicheskiye-ukazaniya-i-informatsiya/article_post/yakist-ta-testuvannya-programnogo-zabezpechennya	7 0.13 %



4	http://djoplus.com.ua/metodicheskiye-ukazaniya-i-informatsiya/article_post/yakist-ta-testuvannya-programnogo-zabezpechennya	5 0.09 %

з бази даних RefBooks (0.00 %)

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)

з домашньої бази даних (0.00 %)

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)

з програми обміну базами даних (0.00 %)

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)

з Інтернету (0.63 %)

ПОРЯДКОВИЙ НОМЕР	ДЖЕРЕЛО URL	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	http://djoplus.com.ua/metodicheskiye-ukazaniya-i-informatsiya/article_post/yakist-ta-testuvannya-programnogo-zabezpechennya	35 (4) 0.63 %

Список прийнятих фрагментів (немає прийнятих фрагментів)

ПОРЯДКОВИЙ НОМЕР	ЗМІСТ	КІЛЬКІСТЬ ОДНАКОВИХ СЛІВ (ФРАГМЕНТІВ)

10
ВСТУП
Сучасне суспільство характеризується активним використанням цифрових технологій для спілкування та взаємодії між людьми [1]. З кожним роком зростає потреба в комфортному та безпечному просторі для соціального спілкування, який би дозволяв людям легко підтримувати зв'язки, обмінюватися інформацією та комунікувати з однодумцями. Саме тому було розроблено програмну систему, яка допомагає вирішити ці завдання, поєднуючи простоту використання, зручність та сучасні технології.
Створена програмна система – це платформа, що забезпечує користувачам можливість реєструватися, створювати та підтримувати власні профілі, спілкуватися у приватних і групових чатах, ділитися своїми думками, публікувати дописи з мультимедійним контентом, а також взаємодіяти з іншими користувачами через коментарі та оцінки публікацій.
Програмною системою зможуть користуватися різноманітні категорії людей, які цінують комфортну та безпечну соціальну взаємодію. Платформа дозволяє швидко встановлювати нові зв'язки та спілкуватися з іншими користувачами. Крім соціальної взаємодії, вона пропонує можливості для побудови та підтримки контактів з людьми, які поділяють професійні інтереси або захоплення, що сприяє ефективному обміну досвідом та знаннями.
Розроблена система використовує сучасні програмні технології, такі як ASP.NET Core, Entity Framework Core, React, Kotlin [2] та архітектурний підхід, який забезпечує стабільну та ефективну роботу [3]. Це дозволяє забезпечити користувачам інтуїтивно зрозумілий та зручний інтерфейс для швидкого обміну повідомленнями, комунікації у реальному часі, а також можливість перегляду аналітики та звітів для адміністраторів платформи.
Завдяки цій програмній системі користувачі отримають комфортний та безпечний простір для ефективного спілкування, створення зв'язків і обміну

11
інформацією, що позитивно вплине на якість їхньої повсякденної соціальної взаємодії.

12
1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

ДОДАТОК Б

Слайди презентації

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

Кваліфікаційна робота бакалавра

Програмна система для соціальної взаємодії та комунікації

Виконав: Хруб Ю.К., ПЗПІ-21-5
Науковий керівник: к.т.н., доц. Дмитро Колесников

13 червня 2025

Мета роботи

Створити програмну систему, яка забезпечує ефективну комунікацію та соціальну взаємодію користувачів у межах єдиної платформи, з можливістю зручного обміну повідомленнями, формування контактів і підтримання дружнього спілкування.

2

Завдання

- Провести аналіз аналогічних програмних рішень для онлайн-комунікації та соціальної взаємодії.
- Спроєктувати архітектуру програмної системи.
- Реалізувати модулі:
  a) реєстрації та авторизації користувачів;
  b) редагування профілю;
  c) приватні повідомлення;
  d) публікації дописів із мультимедійним контентом;
  e) коментування та оцінювання публікацій;
  f) адміністрування.
- Здійснити тестування працездатності основних компонентів системи.

3

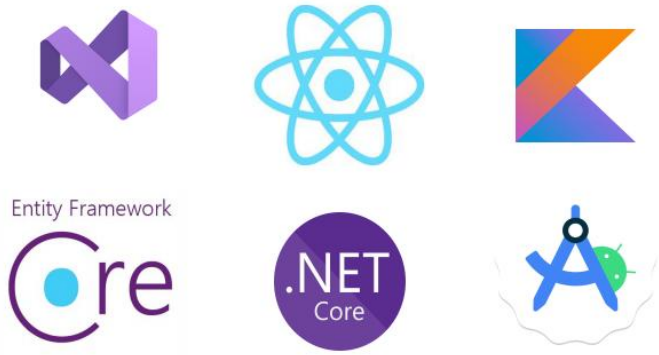
Аналіз існуючих рішень

- Slack
- Discord
- WhatsApp
- Telegram
- MS Teams

Параметр	Discord	Telegram	WhatsApp	Slack	MS Teams
Соціальна взаємодія	Висока	Висока	Середня	Середня	Середня
Складність використання	Легке	Низька	Низька	Низька	Середня
Інтеграції	Середньо	Середньо	Дуже мало	Найбільше	Багато
Нетворкінг	Низький	Середній	Низький	Найвищий	Високий
Інформаційне переванта ження	Середнє	Середнє	Середнє	Велике	Велике
Рівень безпеки	Середній	Середній	Середній	Дуже високий	Високий

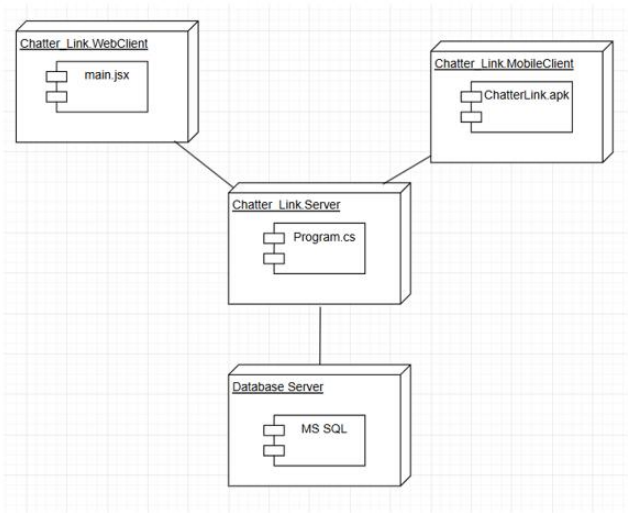
4

Вибір технологій розробки



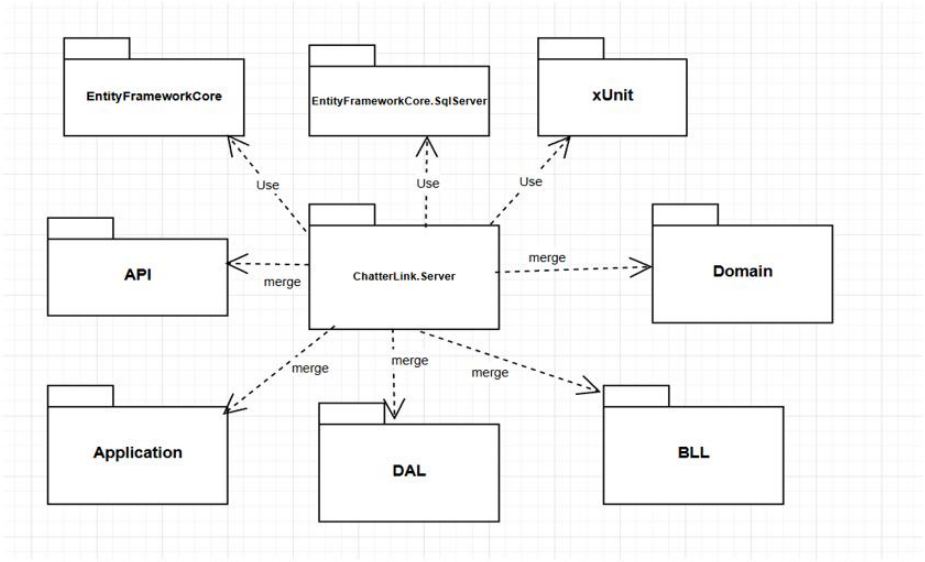
5

Архітектура програмної системи



6

Серверна частина



7

Опис процесу розробки та опис ПЗ

Опис процесу розробки

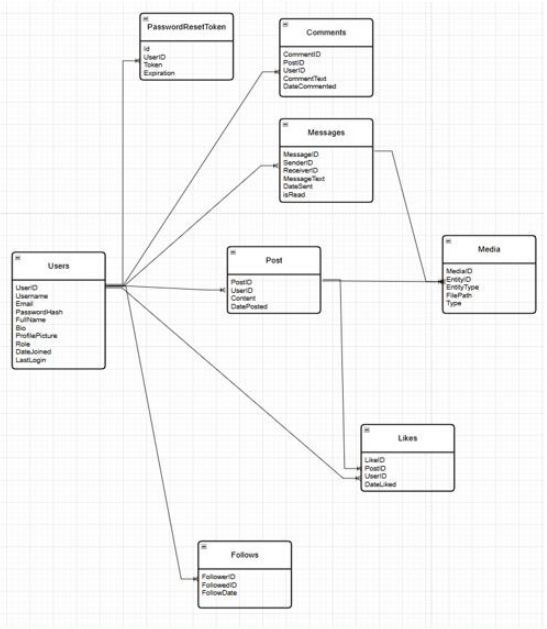
- Аналіз вимог користувача
- Проєктування архітектури та бази даних
- Розробка серверної та клієнтської частини
- Виконання тестування

Опис ПЗ:

- Front-end: React (JavaScript)
- Back-end: ASP.NET Core (C#)
- ORM: Entity Framework Core
- Тестування: xUnit, Moq
- БД: MS SQL Server

8

Структура зберігання даних



9

Приклад реалізації

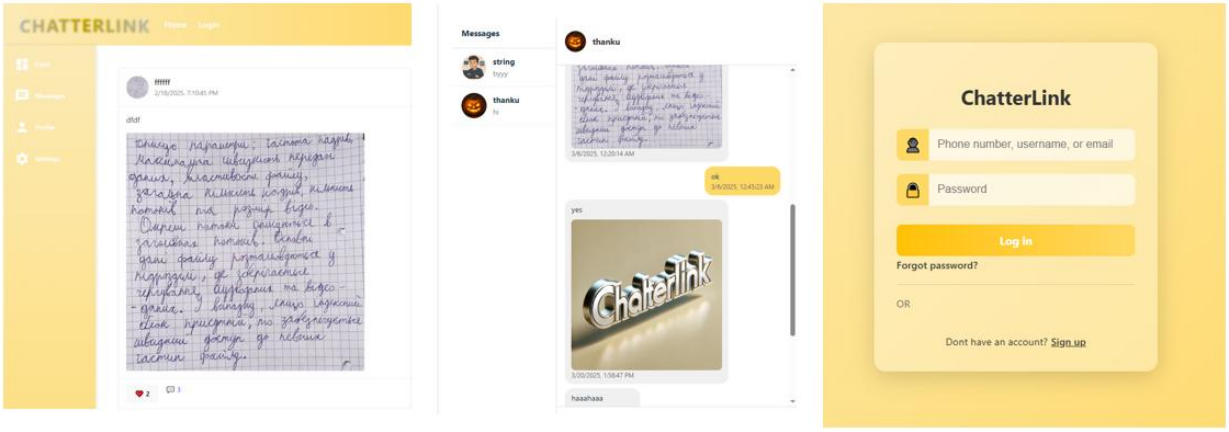
```
private async Task<string> SaveFileAsync(IFormFile file, string subfolder)
{
    var rootPath = _env.WebRootPath ?? Path.Combine(Directory.GetCurrentDirectory(), "wwwroot");
    var uploadsFolder = Path.Combine(rootPath, "uploads", subfolder);
    if (!Directory.Exists(uploadsFolder))
    {
        Directory.CreateDirectory(uploadsFolder);
    }
    var fileName = $"{Guid.NewGuid()}_{file.FileName}";
    var filePath = Path.Combine(uploadsFolder, fileName);

    using (var stream = new FileStream(filePath, FileMode.Create))
    {
        await file.CopyToAsync(stream);
    }


    return Path.Combine("uploads", subfolder, fileName).Replace("\\", "/");
}
```

```
public async Task AddCommentAsync(int userId, AddCommentRequest request)
{

    bool postExists = await _commentRepository.PostExistsAsync(request.PostID);
    if (!postExists)
    {
        throw new Exception("Post not found.");
    }


    var comment = new Comment
    {
        PostID = request.PostID,
        UserID = userId,
        CommentText = request.CommentText,
        DateCommented = DateTime.UtcNow
    };

    await _commentRepository.AddCommentAsync(comment);
}
```

```
public class Media
{
    [Key]
    8 references
    public int MediaID { get; set; }

    [Required]
    10 references
    public int EntityID { get; set; }

    [Required]
    10 references
    public string EntityType { get; set; }

    [Required]
    17 references
    public string FilePath { get; set; }

    [Required]
    11 references
    public MediaType Type { get; set; }
}
```

10

Інтерфейс користувача



Тестування

- xUnit
- Moq

```
[Fact]
public async Task ToggleLikeAsync_ExistingLike_CallsRemoveLike()
{
    int postId = 1, userId = 2;
    var existing = new Like { PostID = postId, UserID = userId };

    _repoMock.Setup(r => r.PostExistsAsync(postId))
             .ReturnsAsync(true);
    _repoMock.Setup(r => r.GetLikeAsync(postId, userId))
             .ReturnsAsync(existing);

    await _service.ToggleLikeAsync(postId, userId);

    _repoMock.Verify(r => r.PostExistsAsync(postId), Times.Once);
    _repoMock.Verify(r => r.GetLikeAsync(postId, userId), Times.Once);
    _repoMock.Verify(r => r.RemoveLikeAsync(existing), Times.Once);
    _repoMock.Verify(r => r.AddLikeAsync(It.IsAny<Like>()), Times.Never);
}
```

```
[Fact]
public async Task ToggleLikeAsync_NoExistingLike_CallsAddLike()
{
    int postId = 3, userId = 4;

    _repoMock.Setup(r => r.PostExistsAsync(postId))
             .ReturnsAsync(true);
    _repoMock.Setup(r => r.GetLikeAsync(postId, userId))
             .ReturnsAsync((Like)null);

    await _service.ToggleLikeAsync(postId, userId);

    _repoMock.Verify(r => r.PostExistsAsync(postId), Times.Once);
    _repoMock.Verify(r => r.GetLikeAsync(postId, userId), Times.Once);
    _repoMock.Verify(r =>
        r.AddLikeAsync(It.Is<Like>(l =>
            l.PostID == postId &&
            l.UserID == userId
        )),
        Times.Once);
    _repoMock.Verify(r => r.RemoveLikeAsync(It.IsAny<Like>()), Times.Never);
}
```

```
CommentServiceTests (4)    285 ms
LikeServiceTests (5)       289 ms
MessageServiceTests (5)    289 ms
```

Підсумки ✓

У результаті виконання роботи було створено програмну систему для соціальної комунікації – ChatterLink, яка інтегрує соціальні взаємодії та формує комфортний цифровий простір. Програмне забезпечення успішно реалізує необхідні функції з використанням сучасних технологій і фреймворків, таких як ASP.NET Core, React, Kotlin та Entity Framework Core, що забезпечило високу продуктивність, стабільність, масштабованість і простоту підтримки системи. Завдяки архітектурі REST API функціональність чітко розподілена між серверною та клієнтською частинами, що значно спростило процес інтеграції та тестування.

CHATTERLINK

13

Дякую за увагу!

ДОДАТОК В

Специфікація REST

- /api/auth/register (POST):
  реєстрація нового користувача. потрібно передати дані: email, password, username, fullName;
- /api/auth/login (POST): вхід за email і паролем;
- /api/auth/forgot-password (POST): запит посилання для скидання пароля. Потрібно передати дані: email;
- /api/auth/reset-password (POST): Встановлення нового пароля. Потрібно передати дані: token (із листа), newPassword;
- /api/users/me (GET): Отримати дані поточного користувача;
- /api/users/{username} (GET): Отримати публічний профіль за іменем. Потрібно передати параметр username;
- /api/users/me (PUT): Оновити свій профіль. Потрібно передати будь-які з полів: fullName, bio;
- /api/users/me/avatar (POST): Завантажити або оновити аватар.
- /api/posts (GET): Отримати стрічку всіх постів;
- /api/posts/{postId} (GET): Отримати деталі одного поста. потрібно передати параметр postId;
- /api/posts/user/{userId} (GET): Отримати пости конкретного користувача. Потрібно передати параметр userId;
- /api/posts (POST): Створити новий пост. Потрібно відправити multipart/form-data з полями content, images?[], videos?[];
- /api/posts/{postId} (PUT): Редагувати свій пост. Потрібно передати параметр postId та в тілі JSON поле content;
- /api/posts/{postId} (DELETE): Видалити свій пост;
- /api/posts/{postId}/like (POST): Поставити лайк;

- /api/posts/{postId}/like (DELETE): Прибрати лайк;
- /api/comments/post/{postId} (GET): Отримати список коментарів до поста. Потрібно передати параметр postId;
- /api/comments/count/{postId}(GET): Отримати кількість коментарів. Потрібно передати параметр postId;
- /api/comments (POST): Додати коментар. Потрібно передати дані: postId, commentText;
- /api/comments/{commentId} (DELETE): Видалити свій коментар. Потрібно передати параметр commentId;
- /api/follows/{username} (POST): Підписатися на користувача. Потрібно передати параметр username;
- /api/follows/{username} (DELETE): Відписатися від користувача. Потрібно передати параметр username;
- /api/messages/dialogs (GET): Отримати список діалогів;
- /api/messages/{username} (GET): Отримати повідомлення з конкретним користувачем. Потрібно передати параметр username;
- /api/messages/send (POST): Надіслати повідомлення. Потрібно передати дані: toUsername, text;
- /api/admin/users (GET): Отримати список усіх користувачів. Потрібно мати роль Admin в JWT;
- /api/admin/users/{userId} (DELETE): Видалити користувача. Потрібно передати параметр userId та мати роль Admin;
- /api/admin/posts (GET): Отримати список усіх постів. Потрібно мати роль Admin;
- /api/admin/comments/{commentId} (DELETE): Видалити будь-який коментар. Потрібно передати параметр commentId та мати роль Admin.