ДОДАТОК А

Таблиці

Таблиця А.1 – Технологічний стек застосунку

Технологія	Версія	Призначення
ASP.NET Core MVC	8.0	Серверний фреймворк, маршрутизація
Entity Framework Core	8.0	ORM, міграції схеми БД
SQLite	–	Реляційна база даних
ASP.NET Core Identity	8.0	Автентифікація, авторизація
Microsoft Playwright	1.40	Керування браузером у скрапері
AngleSharp	1.1.2	Розбір HTML-структури
Polly	8.3	Політика повторних мережевих спроб
Anthropic Claude API	claude-3-5-haiku	Аналіз відкритих відповідей
FullCalendar.js	–	Інтерактивний календар підготовки
Bootstrap	5	CSS-фреймворк, базова верстка інтерфейсу

Таблиця А.2 – Порівняння фреймворків серверної частини

Критерій	ASP.NET Core	Django	Laravel	Spring Boot
Мова програмування	C#	Python	PHP	Java
Тип типізації	Статична	Динамічна	Динамічна	Статична
Архітектурний шаблон	MVC	MTV	MVC	MVC
Упровадження залежностей	Вбудоване	Бібліотеки	Контейнер	Вбудоване
Засіб доступу до даних	EF Core	Django ORM	Eloquent	Hibernate
Рівень продуктивності	Високий	Середній	Середній	Високий
Поріг входу	Середній	Низький	Низький	Високий

Таблиця А.3 – Механізми захисту даних та загрози, яким вони протидіють

Механізм	Призначення (загроза)	Засіб реалізації
Автентифікація	Розмежування доступу	Керування обліковими записами, вхід за паролем
Хешування паролів	Компрометація сховища паролів	Одностороннє хешування із сіллю

Продовження таблиці А.3

Механізм	Призначення (загроза)	Засіб реалізації
Авторизація даних	Доступ до чужих даних	Прив’язка даних до власника та фільтрація вибірок
Захист від CSRF	Підроблені міжсайтові запити	Антипідробні токени форм
Відмовостійкість	Відмова зовнішніх служб	Резервний режим та повторні спроби

Таблиця А.4 – Основні варіанти використання системи

Варіант використання	Актор	Призначення
Зареєструватися	Гість	Реєстрація облікового запису з вибором чотирьох предметів
Увійти в систему	Гість	Автентифікація зареєстрованого користувача
Керувати профілем	Здобувач	Перегляд і редагування особистих даних, зміна пароля, вихід із системи
Налаштувати та запустити тест	Здобувач	Вибір предмета, теми, року й кількості питань тесту
Пройти тест	Здобувач	Послідовне надання відповідей на питання сесії
Оцінити відкриту відповідь	Сервіс ШІ	Оцінювання рівня розуміння відкритої відповіді, включається до варіанта «Пройти тест»

Продовження таблиці А.4

Варіант використання	Актор	Призначення
Переглянути результат і історію	Здобувач	Перегляд підсумків сесії та попередніх тестувань
Переглянути дашборд прогресу	Здобувач	Аналіз показників успішності й активності
Оновити рекомендації	Здобувач, Планувальник	Формування персональних рекомендацій засобами ШІ
Переглянути календар	Здобувач	Перегляд запланованих і рекомендованих занять
Керувати завданнями календаря	Здобувач	Створення, позначення виконаними та видалення завдань
Сформувати розклад автоматично	Здобувач	Алгоритмічне планування занять із чергуванням статей і тестів

Таблиця А.5 – Дані, що видобуваються зі сторінки завдання

Поле	Джерело на сторінці	Призначення
Зовнішній ідентифікатор	Предмет, номер тесту та ідентифікатор питання	Унікальний ключ для виявлення повторів
Текст питання	Блок формулювання завдання	Зміст питання, що показують здобувачеві
Зображення	Рисунок у блоці питання	Ілюстрація до завдання за наявності

Продовження таблиці А.5

Поле	Джерело на сторінці	Призначення
Варіанти відповіді	Перелік варіантів (А–Г)	До чотирьох варіантів вибору
Правильна відповідь	Позначка правильного варіанта	Еталон для автоматичного оцінювання
Пояснення	Блок розбору відповіді	Розгорнутий коментар до правильної відповіді
Тема	Рубрикатор завдання	Віднесення питання до теми предмета
Рік і сесія	Метадані сторінки (адреса, заголовок)	Належність завдання до року та сесії тестування

Таблиця А.6 – Показники аналітики прогресу

Показник	Спосіб обчислення	Призначення
Успішність за предметом	Частка правильних відповідей серед усіх відповідей предмета	Оцінка рівня підготовки з кожного предмета
Щоденна активність	Кількість відповідей і частка правильних за кожен з останніх днів	Відстеження регулярності занять
Слабкі теми	Теми з принаймні двома спробами та часткою помилок від 40%	Визначення тем для повторення
Рекомендовані тести	Добір тестів і статей за найслабшими темами	Підказка наступних кроків підготовки

Продовження таблиці А.6

Показник	Спосіб обчислення	Призначення
Середній бал	Середнє значення результатів завершених сесій	Узагальнена оцінка успішності
Серія активності	Кількість поспіль днів із завершеними сесіями	Підтримання мотивації до регулярних занять

Таблиця А.7 – Результати функціонального тестування

Сценарій	Очікуваний результат	Стан
Реєстрація з вибором чотирьох предметів	Обліковий запис створено, виконано автоматичний вхід	Пройдено
Реєстрація з кількістю предметів, відмінною від чотирьох	Відмова з повідомленням про помилку	Пройдено
Вхід із правильними обліковими даними	Доступ до дашборду	Пройдено
Вхід із хибним паролем	Повідомлення про помилку, доступ заборонено	Пройдено
Проходження тесту із закритими питаннями	Підрахунок правильних відповідей, перехід до результату	Пройдено
Повна відповідь на відкрите питання	Відповідь зараховано, показано аналіз ШІ	Пройдено

Продовження таблиці А.7

Сценарій	Очікуваний результат	Стан
Неповна відповідь на відкрите питання	Відповідь не зараховано, аналіз указує на недоліки	Пройдено
Перегляд дашборду	Показники прогресу й слабкі теми відповідають історії	Пройдено
Автоматичне планування занять	Сформовано пари «стаття – тест» за темами	Пройдено
Звернення до чужої сесії тестування	Відмова в доступі	Пройдено
Робота за відсутності ключа доступу до ШІ	Резервний оцінювач повертає оцінку, застосунок працює	Пройдено

ДОДАТОК Б

Лістинги

Лістинг Б.1 – Модель сутності Answer

```
public class Answer
{
    [Key]
    public int Id { get; set; }

    public int TestSessionId { get; set; }
    public int QuestionId { get; set; }

    [MaxLength(5)]
    public string? SelectedOption { get; set; }
    public string? OpenText { get; set; }

    public bool IsCorrect { get; set; }
    public bool IsOpenAnswer { get; set; } = false;
    public DateTime AnsweredAt { get; set; } =
DateTime.UtcNow;
    public int TimeSpentSeconds { get; set; }

    public string? AiAnalysisJson { get; set; }

    [ForeignKey(nameof(TestSessionId))]
    public TestSession? TestSession { get; set; }

    [ForeignKey(nameof(QuestionId))]
    public Question? Question { get; set; }
}
```

Лістинг Б.2 – Конфігурування моделі даних у методі OnModelCreating

```
protected override void OnModelCreating(ModelBuilder builder)
{
    base.OnModelCreating(builder);

    builder.Entity<Question>(e =>
    {
        e.HasIndex(q => q.ExternalId).IsUnique();
        e.HasIndex(q => new { q.Subject, q.Year });
        e.HasIndex(q => q.TestId);
    });

    builder.Entity<TestSession>(e =>
    {
        e.HasOne(t => t.User)
         .WithMany(u => u.TestSessions)
         .HasForeignKey(t => t.UserId)
         .OnDelete(DeleteBehavior.Cascade);

        e.HasIndex(t => new { t.UserId, t.Subject });
    });

    builder.Entity<Answer>(e =>
    {
        e.HasOne(a => a.TestSession)
         .WithMany(t => t.Answers)
         .HasForeignKey(a => a.TestSessionId)
         .OnDelete(DeleteBehavior.Cascade);

        e.HasOne(a => a.Question)
         .WithMany(q => q.Answers)
         .HasForeignKey(a => a.QuestionId)
         .OnDelete(DeleteBehavior.Restrict);
    });
```

Продовження лістингу Б.2

```
    builder.Entity<CalendarTask>(e =>
    {
        e.HasOne(c => c.User)
         .WithMany(u => u.CalendarTasks)
         .HasForeignKey(c => c.UserId)
         .OnDelete(DeleteBehavior.Cascade);

        e.HasIndex(c => new { c.UserId, c.ScheduledDate });
    });

    builder.Entity<ApplicationUser>().ToTable("Users");
}
```

Лістинг Б.3 – Фрагмент репозиторію завдань

```
public class QuestionRepository : IQuestionRepository
{
    private readonly NmtDbContext _ctx;
    public QuestionRepository(NmtDbContext ctx) => _ctx = ctx;

    public async Task<IEnumerable<Question>>
GetRandomBySubjectAsync(
        string subject, int count, IEnumerable<int>?
excludeIds)
    {
        var query = _ctx.Questions.Where(q => q.Subject ==
subject);
        if (excludeIds != null && excludeIds.Any())
            query = query.Where(q =>
!excludeIds.Contains(q.Id));
        return await query.OrderBy(_ => EF.Functions.Random())
                          .Take(count).ToListAsync();
    }

    public async Task<IEnumerable<Question>> GetByThemeAsync(
```

Продовження лістингу Б.3

```
        string subject, string theme, int count) =>
        await _ctx.Questions
            .Where(q => q.Subject == subject && q.Theme ==
theme)
            .OrderBy(_ => EF.Functions.Random())
            .Take(count).ToListAsync();

    public Task<bool> ExistsAsync(string externalId) =>
        _ctx.Questions.AnyAsync(q => q.ExternalId ==
externalId);

    public async Task AddRangeAsync(IEnumerable<Question>
questions) =>
        await _ctx.Questions.AddRangeAsync(questions);

    public Task SaveChangesAsync() => _ctx.SaveChangesAsync();
}
```

Лістинг Б.4 – Перевірка відповіді у сервісі сесій

```
public async Task<AnswerResultDto>
SubmitAnswerAsync(SubmitAnswerDto dto)
{
    var question = await
_questions.GetByIdAsync(dto.QuestionId)
        ?? throw new InvalidOperationException("Question not
found");

    var correctText = ResolveCorrectAnswerText(question);

    bool isCorrect = !dto.IsOpenAnswer &&
        string.Equals(dto.SelectedOption,
question.CorrectAnswer,
                      StringComparison.OrdinalIgnoreCase);
```

Продовження лістингу Б.4

```
    AiAnalysisDto? aiAnalysis = null;
    if (dto.IsOpenAnswer &&
!string.IsNullOrWhiteSpace(dto.OpenText))
    {
        isCorrect = AnswersMatch(dto.OpenText, correctText);
        aiAnalysis = await _ai.AnalyzeOpenAnswerAsync(
            question.Text, correctText, question.Explanation,
dto.OpenText);
        if (!isCorrect && aiAnalysis != null)
            isCorrect = aiAnalysis.UnderstandingScore >= 60;
    }

    var answer = new Answer
    {
        TestSessionId = dto.TestSessionId,
        QuestionId = dto.QuestionId,
        SelectedOption = dto.SelectedOption,
        OpenText = dto.OpenText,
        IsCorrect = isCorrect,
        IsOpenAnswer = dto.IsOpenAnswer,
        TimeSpentSeconds = dto.TimeSpentSeconds,
        AiAnalysisJson = aiAnalysis != null
            ? JsonSerializer.Serialize(aiAnalysis) : null
    };

    await _ctx.Answers.AddAsync(answer);
    await _ctx.SaveChangesAsync();

    return new AnswerResultDto
    {
        IsCorrect = isCorrect,
        CorrectAnswer = dto.IsOpenAnswer ? correctText :
question.CorrectAnswer,
        Explanation = question.Explanation,
```

Продовження лістингу Б.4

```
        AiAnalysis = aiAnalysis
    };
}
```

Лістинг Б.5 – Основний цикл збору завдань

```
foreach (var subject in _config.Subjects)
    foreach (var range in ranges)
        for (int testId = range.From; testId <= range.To;
testId++)
        {
            var url =
$"{_config.BaseUrl}/{subject}/test/{testId}/";

            var questions = await _retryPolicy.ExecuteAsync(()
=>
                ScrapeTestPageAsync(browser, url, subject,
testId, ct));
            if (questions == null || !questions.Any())
continue;

            int saved = 0;
            foreach (var q in questions)
            {
                if (await _repo.ExistsAsync(q.ExternalId)) {
skipped++; continue; }
                await _repo.AddAsync(q);
                saved++; total++;
            }
            if (saved > 0) await _repo.SaveChangesAsync();

            await Task.Delay(_rng.Next(_config.DelayMinMs,
_config.DelayMaxMs), ct);
        }
```

Лістинг Б.6 – Розбір окремого завдання

```
var idAttr = node.GetAttribute("data-id") ??
node.GetAttribute("id") ?? "";
if (string.IsNullOrEmpty(idAttr)) return null;
var externalId = $"{subject}-{testId}-{idAttr}";

var textEl = node.QuerySelector(".question-text, .task-text,
p:first-child");
var questionText = textEl?.TextContent.Trim() ??
node.TextContent.Trim();
if (string.IsNullOrWhiteSpace(questionText) ||
questionText.Length < 5) return null;

var options = new List<object>();
var optionKeys = new[] { "А", "Б", "В", "Г" };
var optionNodes = node.QuerySelectorAll(".answer-option,
.variant, li[class*='answer']");
for (int i = 0; i < Math.Min(optionNodes.Length, 4); i++)
    options.Add(new { key = optionKeys[i], text =
optionNodes[i].TextContent.Trim() });
if (!options.Any()) return null;

var correctEl = node.QuerySelector(".correct-answer, [data-
correct]");
var correctAnswer = correctEl?.TextContent.Trim() ?? "A";

return new Question
{
    ExternalId = externalId, TestId = testId, Subject =
subject,
    Year = year, Session = session, Theme = theme, Text =
questionText,
    OptionsJson = JsonSerializer.Serialize(options),
    CorrectAnswer = NormalizeAnswer(correctAnswer),
    QuestionType = "multiple_choice"
```

Лістинг Б.7 – Оцінювання відкритої відповіді засобами ШІ

```
var prompt = $$"""
    Ти — досвідчений вчитель, що перевіряє відповідь учня на
тест НМТ.

    ПИТАННЯ ТЕСТУ:
    {{questionText}}
    ПРАВИЛЬНА ВІДПОВІДЬ: {{correctAnswer}}
    ВІДПОВІДЬ УЧНЯ:
    {{studentAnswer}}

    Поверни ТІЛЬКИ JSON у форматі:
    {
      "understandingScore": <число від 0 до 100>,
      "whatUnderstood": "<що учень зрозумів>",
      "whatMissed": "<що пропустив>",
      "recommendation": "<що повторити>"
    }
    """;

var response = await CallClaudeAsync(prompt, maxTokens: 500);
if (response == null) return null;

var cleaned = response.Trim().TrimStart('').TrimEnd('');
if (cleaned.StartsWith("json")) cleaned = cleaned[4..];
var data = JsonSerializer.Deserialize<JsonElement>(cleaned);

return new AiAnalysisDto
{
    UnderstandingScore =
data.GetProperty("understandingScore").GetInt32(),
    WhatUnderstood =
data.GetProperty("whatUnderstood").GetString() ?? "",
    WhatMissed = data.GetProperty("whatMissed").GetString() ??
"",
```

Продовження лістингу Б.7

```
    Recommendation =
data.GetProperty("recommendation").GetString() ?? ""
};
```

Лістинг Б.8 – Формування запиту та виклик API мовної моделі

```
private async Task<string?> CallClaudeAsync(string prompt, int
maxTokens = 1000)
{
    if (string.IsNullOrEmpty(_apiKey)) return null;
    try
    {
        var request = new HttpRequestMessage(HttpMethod.Post,
            "https://api.anthropic.com/v1/messages");
        request.Headers.Add("x-api-key", _apiKey);
        request.Headers.Add("anthropic-version", "2023-06-
01");

        var body = JsonSerializer.Serialize(new
        {
            model = _model,
            max_tokens = maxTokens,
            messages = new[] { new { role = "user", content =
prompt } }
        });
        request.Content = new StringContent(body,
Encoding.UTF8, "application/json");

        var response = await _http.SendAsync(request);
        response.EnsureSuccessStatusCode();

        var json = await response.Content.ReadAsStringAsync();
        var doc = JsonDocument.Parse(json);
        return doc.RootElement.GetProperty("content")[0]
                  .GetProperty("text").GetString();
```

Продовження лістингу Б.8

```
    }
    catch (Exception ex)
    {
        _logger.LogError(ex, "Anthropic API call failed");
        return null;
    }
}
```

Лістинг Б.9 – Резервне локальне оцінювання відповіді

```
private static int ScoreAnswer(string correct, string student)
{
    var norm = Normalize(student);
    if (norm.Length == 0) return 0;

    var correctTokens = Tokenize(correct);
    var studentTokens = Tokenize(student).ToHashSet();
    if (correctTokens.Count == 0) return
Math.Clamp(norm.Length / 2, 20, 70);

    double overlap = (double)correctTokens.Count(t =>
studentTokens.Contains(t))
                     / correctTokens.Count;
    var nums = correctTokens.Where(IsNumber).ToList();
    double numHit = nums.Count == 0 ? 1.0
        : (double)nums.Count(n => studentTokens.Contains(n)) /
nums.Count;

    int score = (int)Math.Round((0.55 * overlap + 0.45 *
numHit) * 100);
    return Math.Clamp(score, 5, 100);
}
```

Лістинг Б.10 – Дії контролера для проходження тесту

```
[HttpPost, ValidateAntiForgeryToken]
public async Task<IActionResult> Begin(string subject, int?
testId,
    int questionCount = 10, string? theme = null, int? year =
null)
{
    var userId = _userManager.GetUserId(User)!;
    var questions = await
_questions.GetTestQuestionsAsync(subject, testId,
questionCount);
    if (!questions.Any())
    {
        TempData["Error"] = "Питань для обраного фільтру не
знайдено.";
        return RedirectToAction("Start", new { subject });
    }

    var session = await _sessions.StartSessionAsync(userId,
subject, testId,
        questions.Select(q => q.Id).ToList());
    return RedirectToAction("Question", new { sessionId =
session.Id, num = 1 });
}
[HttpPost, ValidateAntiForgeryToken]
public async Task<IActionResult> Answer([FromBody]
SubmitAnswerDto dto)
{
    if (dto == null) return BadRequest(new { error = "Invalid
request" });
    var result = await _sessions.SubmitAnswerAsync(dto);
    return Json(result);
}
```

Лістинг Б.11 – Надсилання відповіді без перезавантаження сторінки

```
async submitAnswer() {
    const isOpen = this.forceOpen  this.mode === 'open';
    if (!this.selectedKey && (!isOpen  !this.openText.trim()))
return;
    this.submitting = true;

    const dto = {
        testSessionId: this.sessionId, questionId:
this.questionId,
        selectedOption: isOpen ? null : this.selectedKey,
        openText: isOpen ? this.openText : null,
        isOpenAnswer: isOpen, timeSpentSeconds: this.elapsed
    };

    const token =
document.getElementById('globalCsrfToken')?.value || '';
    const res = await fetch('/Test/Answer', {
        method: 'POST',
        headers: { 'Content-Type': 'application/json',
                   'RequestVerificationToken': token },
        body: JSON.stringify(dto)
    });
    const result = await res.json();

    this.answered = true;
    this.isCorrect = result.isCorrect ?? false;
    this.correctAnswer = result.correctAnswer ?? '';
    this.explanation = result.explanation ?? '';
    this.aiAnalysis = result.aiAnalysis ?? null;
}
```

Лістинг Б.12 – Реєстрація користувача з вибором предметів

```
[HttpPost, ValidateAntiForgeryToken]
public async Task<IActionResult> Register(RegisterViewModel
model)
{
    if (!ModelState.IsValid) return View(model);
    if (model.SelectedSubjects == null ||
model.SelectedSubjects.Count != 4)
    {
        ModelState.AddModelError("", "Будь ласка, оберіть
рівно 4 предмети.");
        return View(model);
    }

    var user = new ApplicationUser
    {
        UserName = model.Email, Email = model.Email,
        FirstName = model.FirstName, LastName =
model.LastName,
        SelectedSubjects = string.Join(",",
model.SelectedSubjects)
    };

    var result = await _userManager.CreateAsync(user,
model.Password);
    if (result.Succeeded)
    {
        await _userManager.AddToRoleAsync(user, "Student");
        await _signInManager.SignInAsync(user, isPersistent:
false);
        return RedirectToAction("Index", "Dashboard");
    }
    foreach (var error in result.Errors)
        ModelState.AddModelError("", error.Description);
    return View(model);
```

Лістинг Б.13 – Ініціалізація календаря занять

```
calendar = new FullCalendar.Calendar(el, {
    locale: 'uk',
    initialView: 'dayGridMonth',
    headerToolbar: {
        left: 'prev,next today', center: 'title',
        right: 'dayGridMonth,timeGridWeek,listWeek'
    },
    events: '/Calendar/GetTasks',
    eventClick: info => showTaskDetail(info.event),
    eventClassNames: info =>
        info.event.extendedProps.isCompleted ? ['fc-event-
completed'] : []
});
calendar.render();
```

Лістинг Б.14 – Налаштування автентифікації та політики паролів

```
builder.Services.AddIdentity<ApplicationUser, IdentityRole>(opt
=>
{
    opt.Password.RequiredLength = 6;
    opt.Password.RequireDigit = false;
    opt.Password.RequireUppercase = false;
    opt.SignIn.RequireConfirmedEmail = false;
})
.AddEntityFrameworkStores<NmtDbContext>()
.AddDefaultTokenProviders();

builder.Services.ConfigureApplicationCookie(opt =>
{
    opt.LoginPath = "/Account/Login";
    opt.AccessDeniedPath = "/Account/AccessDenied";
    opt.Cookie.Name = "NMT.Auth";
    opt.ExpireTimeSpan = TimeSpan.FromDays(14);
    opt.SlidingExpiration = true;
```

ДОДАТОК В

Відомість кваліфікаційної роботи

Позначення	Найменування	Дод. відомості
	Текстові документи	
1.	Пояснювальна записка	114 с.
	Інші документи	
2.	Презентаційні матеріали	__ плакатів

.	.	Прізвище та ініціали.	Підп.	Дата			
Розробив		Хамінова О.О.			Розробка вебзастосунку для підготовки випускників шкіл до складання Національно Мультидисциплінарного іспиту із підсистемою перевірки результатів тестів	Шифр групи	Код напр./спец.
Перевірив		Бібічков І.Є.				ІТШІ-22-1	122
Н.контр.		Малєєва І.А.				ХНУРЕ кафедра ШІ	
Затв.		Чала Л.Е.					