- Exception Handling -

① هنروح على ال Error folder ونضيف جواه class جديد GlobalExceptionHandler

```
Public class GlobalExceptionHandler (Ilogger <GlobalExceptionHandler> logger)
                : IExceptionHandler
{
    Private readonly Ilogger <GlobalExceptionHandler> _logger = logger;

    Public async ValueTask <bool> TryHandlerAsync (HttpContext httpContext,
        Exception exception, CancellationToken cancellationToken)
    {
        // هنعمل ال logging لل exception اللي حصل عشان
        _logger.LogError (exception, "Something went wrong: {Message}",
            exception.Message);

        // هنستخدم ال ProblemDetails
        var problemDetails = new ProblemDetails
        {
            Status = StatusCodes.Status500InternalServerError,
            Title = "Internal Server Error",
            Type = " "      // لينك هنجيبه من rfc Standard
        };
        httpContext.Response.StatusCode = Status500InternalServerError;
        await httpContext.Response.WriteAsJsonAsync (problemDetails,
                    cancellationToken: cancellationToken);
        return true;
    }
}
```

② هنروح نضيف ال Middleware في ال program بعد MapControllers

```
app.UseExceptionHandler();
```

③ هنروح على ال DI

```
Services.AddExceptionHandler <GlobalExceptionHandler>();
Services.AddProblemDetails();
```

- Handle Duplicated Poll Titles -

المشكلة دي ممكن تحصل عند 2EndPoints ⇐ Add / Update

هنروح على ال PollService ونعدل ع ال **Add**

```
Public async Task<Result<PollResponse>> AddAsync ( ✓ )
{
    var isExisting = await _context.Polls.AnyAsync(x => x.Title ==
                        request.Title, cancellationToken: cancellationToken);
                        // لو نفس الشرط في ال update بس هنزود && x.id != id
    if (isExisting)
        return Result.Failure<PollResponse>(PollErrors.DuplicatedPollTitle);

    var poll = req ——
    ✓
}
```

هنزود على ال PollErrors

```
Public static readonly Error DuplicatedPollTitle =
    new ("Poll.DuplicatedTitle", "Another Poll with the Same title is
                                   already exists");
```

هنروح على ال EndPoint

```
Public Async Task<IActionResult> Add ( ✓ )
{
    ✓

    return result.Success
        ? CreatedAtAction ( ✓ )
        : result.ToProblem(StatusCodes.Status409Conflict);
}
```

Note

[illegible] ⇐ UserNotFound / Conflict > 2 Problems عندي في ال update ممكن فيه

```
return result.Error!.Equals(PollErrors.DuplicatedPollTitle)
    ? result.ToProblem(StatusCode.Status409Conflict)
    : result.ToProblem(StatusCode.Status404NotFound);
```

- working on Questions Module -

- Add Entities - 2 Entities. (Questions, Answers)

① // Questions.

```
Public Sealed class Questions AuditableEntity
{
    Public int Id {get; set;}
    Public string Content {get; set;} = String.Empty;
    Public int PollId {get; set;}
    Public bool IsActive {get; set;} = true;

    Public Poll Poll {get; Set;} = default!;
    Public ICollection<Answer> Answers {get; set;} = [];
}
```

② // Answers.

```
Public Sealed class Answer
{
    Public int Id {get; set;}
    Public string Content {get; set;} = String.Empty;
    Public int QuestionId {get; Set;}
    Public bool IsActive {get; Set;} = true;
    Public Question Question {get; set;} = default!;
}
```

③ ضروري نزود الـ Navprop الـ questions جوا الـ Poll

```
Public ICollection<Question> Questions {get; Set;} = [];
```

④ ضروري نعمل الـ Configuration لكل entity

```
Question Configuration:
{
    builder.HasIndex(x => new {x.PollId, x.Content}).IsUnique();
    builder.Property(x => x.Content).HasMaxLength(1000);
}

Answer Configuration:
{
    builder.HasIndex(x => new {x.QuestionId, x.Content}).IsUnique();
    builder.Property(x => x.Content).HasMaxValue(1000);
}
```

⑤ ضروري عمل الـ DbSets في الـ ApplicationDbContext

```
Public DbSet<Answer> Answers {get; Set;}
Public DbSet<Question> Questions {get; Set;}
```

- Change OnDelete Behavior -

المفروض هنعمل Migration لحس لو جينا علشان نعمل update-database هيطلع error multiple cascade or cause cycle ← يعني بالشكل لما نعمل delete لل Poll هيعمل delete لل Questions والعكس صحيح.

الحل: أغير ال OnDelete من Cascade لـ Restrict

فهنخليها Global على مستوى ال Project أو ال DataBase

① هنروح على ال ApplicationDbContext في ال OnModelCreating

```
// هنعمل Select لكل ال Fks اللي ال OnDelete تاعتهم Cascade
var cascadeFks = modelBuilder.Model
                 .GetEntityTypes()
                 .SelectMany(t => t.GetForeignKeys())
                 .Where(fk => fk.DeleteBehavior == DeleteBehavior.Cascade && !fk.IsOwnership);

Foreach (var fk in cascadeFks)
    fk.DeleteBehavior = DeleteBehavior.Restrict;
```

بعد كدا هنعمل ال Migration عادي و Update Database

- Add Contracts and Validators -

هنعمل Folder اسمه Questions جوا ال Contracts هنضيف فيه QuestionRequestValidator, QuestionRequest, QuestionResponse

```
Public record QuestionRequest(
    String Content,
    List<String> Answers
);

Public Class QuestionRequestValidator : AbstractValidator<QuestionRequest>
{
    Public QuestionRequestValidator()
    {
        RuleFor(x => x.Content).NotEmpty().Length(3, 1000);
        RuleFor(x => x.Answers).NotNull();
        RuleFor(x => x.Answers).Must(x => x.Count > 1)
            .WithMessage("Question should had at least two answers")
            .When(x => x.Answers != null);
        RuleFor(x => x.Answers).Must(x => x.Distinct().Count() == x.Count)
            .WithMessage("You cannot add duplicated answers for the same question").When(x => x.Answers != null);
```

```
Public record QuestionResponse(
        int Id,
        String Content,
        IEnumerable<AnswerResponse> Answers
    );
```

هنضيف في ال Contract نروح لل Folder ال Answers هنعمل جواه ال Answerreponse

```
Public record AnswerResponse(
        int Id,
        String Content
    );
```

- Add Question Service -

هنعمل ال IQuestionService علشان ن validate عليه

```
Public interface IQuestionService
{
  Task<Result<QuestionResponse>> AddAsync(int PollId, QuestionRequest
        request, CancellationToken cancellationToken = default);
}
```

هنروح نعمل implement في ال QuestionService

```
Public class QuestionService(ApplicationDbContext context) : IQuestionService
{
  Private readonly ApplicationDbContext _context = Context;

  Public async Task<Result<QuestionResponse>> AddAsync(int PollId,
      QuestionRequest request, CancellationToken cancellationToken = default)
  {
    // هنا check على ال PollId موجود ولا
    var PollIsExist = await _context.Polls.AnyAsync(x => x.Id == PollId,
                          CancellationToken);
    if (!PollIsExist)
        return Result.Failure<PollResponse>(PollError.PollNOTFound);
    // هنشوف لو فيه سؤال موجود بنفس ال content في نفس ال Poll
    var questionIsExist = await _context.Questions.AnyAsync(x => x.Content ==
          request.Content && x.PollId == PollId, cancellationToken);
    if (questionIsExist)
        return Result.Failure<QuestionResponse>(QuestionErrors.Duplicated
                                       QuestionContent);
    // هنبدأ نضيف ال question
    var question = request.Adapt<Question>();
    question.PollId = PollId;
```

→

```
request.Answers.ForEach(answer => question.Answers.Add
                  (new Answer { Content = answer }));

await _context.AddAsync(question, cancellationToken);
await _context.SaveChangesAsync(cancellationToken);

return Result.Success(question.Adapt<QuestionResponse>());

}
}
```

أيضًا هنضيف ال QuestionErrors

```
Public Static Class QuestionErrors
{
  Public Static readOnly Error QuestionNotFound =
      new("Question.NotFound", "No Question was Found with given id

  Public Static readonly Error DuplicatedQuestionContent =
  new("Question.DuplicatedContent", "Another question with the Same
                  title is already exists");

}
```

أيضًا متنساش تضيف ال Service في ال DI

```
Services.AddScoped<IQuestionService, QuestionService>();
```

- Add Create QuestionEndPoint -

هنعمل Controller ← QuestionsController

لكن لو نظرنا خد ال PollId من ال Route هغير ال Route {[Route
Polls ... Question ال QuestionController لأن مش هيبقى ال ...
الأساسي ال PollId ... ال question

```
[Route("api/Polls/{pollId}/[Controller]")]
[ApiController]
[Authorize]

Public class QuestionController(IQuestionService QuesionService)
                   : ControllerBase
{

  Private readonly IQuestionService _questionService = questionService;
```

```
[HttpPost("")]
Public async Task<IActionResult> Add([fromRoute] int PollId,
    [fromBody] QuestionRequest request, CancellationToken cancellationToken)
{
  var result = await _questionService.AddAsync(PollId, request,
                                  cancellationToken);

  if (result.IsSuccess)
      return CreatedAtAction(nameof(Get), new { PollId,
              result.Value.Id }, result.Value);

  return result.Error.Equals(QuestionErrors.DuplicatedQuestionContent)
         ? result.ToProblem(StatusCode.Status409Conflict)
         : result.ToProblem(StatusCode.Status404NotFound);
}
```

فيه مشكلة بقا الوقتي عن أن ال Answers اللي موجودة في ال Question ال Type بتاعها ICollection<Answer> وال Answers اللي موجودة في ال QuestionRequest ال Type بتاعها List<String> فلو كتبنا كده request.Adapt<Quest>() ال Mapster كده هيحصل مشكلة في إنه يعمل map ال Answers مكان بعض

فعندك حلين ① إنك تغير ال Answers اللي موجودة في ال QuestionRequest لـ Answerquest (مش أحسن حاجة)

② الحل التاني إنك تستخدم ال MappingConfiguration

```
Public Class MappingConfigurations : IRegister
{
  Public void Register(TypeAdapterConfig config)
  {
                        Src              Dest
    Config.NewConfig<QuestionRequest, Question>()
        .Map(dest => dest.Answers, src => src.Answers.Select
                (answer => new answer{ Content = answer }));
  }
```

وبعد نشيل السطر بتاع ال loop (foreach) ال موجودة في ال QuestionService

- Add GetAll Questions EndPoint -

هنروح على ال IQuestionService نضيف الميثود

```
Task <Result<IEnumerable<QuestionResponse>>> GetAllAsyn (
    int PollId, CancellationToken cancellationtoken = default);
```

هنروح على ال QuestionService

```
Public async Task <Result<IEnumerable<QuestionResponse>>> GetAllAsync (
    int PollId, CancellationToken Cancellationtoken = default)
{
    var PollIsExist = await _Context.Polls.AnyAsync (x => x.Id == PollId,
                    cancellationToken);
    if (! PollIsExist)
        return Result.Failure <IEnumerable<QuestionResponse>>
                                    (PollErrors.PollNotfound);

    var questions = await _Context.Questions
            .Where (x => x.PollId == PollId)
            .Include (x => x.Answers)
            .Select (q => new QuestionResponse (
                        q.Id,
                        q.Content,
                        q.Answers.Select(a => new AnswerResponse (a.Id,
                                                    a.Content))
                        )).AsNoTracking()
                        .ToListAsync(cancellationToken);

    return Result.Success <IEnumerable<QuestionResponse>> (questions);
```

→ ممكن نستغنى عن ده ونستخدم ميثود في Mapster اسمها ProjectToType<>() ← نبعتلها النوع اللي هحول ليه

```
.ProjectToType<QuestionResponse>()
```

هنروح على ال EndPoint

```
[HttpGet(" ")]
Public async Task <IActionResult> GetAll ([FromRoute] int PollId,
            CancellationToken cancellationToken)
{
    var result = await _questionService.GetAllAsync (PollId, CancellationToken);
    return result.IsSuccess ? Ok (result.value) : result.ToProblem (StatusCodes.
                                                Status404NotFound);
}
```

Add Get Question EndPoint -

① في الـ IQuestionService

```
Task<Result<QuestionResponse>> GetAsync(int pollId, int id, CancellationToken
    cancellationToken = default);
```

② في الـ Service

```
Public async Task<Result<QuestionResponse>> GetAsync(↵)
{
    var question = await _context.Questions
        .Where(x => x.PollId == pollId && x.Id == id)
        .Include(x => x.Answers)
        .Project To<QuestionResponse>()
        .AsNoTracking()
        .SingleOrDefaultAsync(~~QuestionResponse~~ cancellationToken);

    if (question is null)
        return Result.Failure<QuestionResponse>(QuestionErrors.
                                                QuestionNotFound);

    return Result.Success(question);
```

③ في الـ Controller الـ EndPoint

```
[HttpGet("{id}")]
Public async Task<IActionResult> Get([FromRoute] int pollId,
    [FromRoute] int id, CancellationToken cancellationToken = default)
{
    var result = await _questionService.GetAsync(pollId, id,
                                cancellationToken);

    return result.IsSuccess ? Ok(result.Value) :
        result.ToProblem(statusCodes.Status404NotFound);
```

Add Toggle Question Status EndPoint -

Answer الـ بغير Question الـ المتغير الـ IsActive

① في الـ IQuestionService

```
Task<Result> ToggleStatusAsync(int pollId, int id, CancellationToken
                                cancellationToken = default);
```

② في الـ QuestionService

```
Public async Task<Result> ToggleStatusAsync(↵)
{
```

→

```
var question = await _context.Questions.SingleOrDefaultAsync(
        x => x.PollId == PollId && x.Id == id, cancellationToken);

if (question is null)
    return Result.Failure(QuestionErrors.QuestionNotFound);

question.IsActive = !question.IsActive

await _context.SaveChangesAsync(cancellationToken);

return Result.Success();
}
```

③ هنروح نعمل ال EndPoint ↓

```
[HttpPut("{id}/toggleStatus")]
public async Task<IActionResult> ToggleStatus([FromRoute] int PollId,
        [FromRoute] int id, CancellationToken cancellationToken)
{
 var result = await _questionService.ToggleStatusAsync(PollId,
          id, cancellationToken);

 return result.IsSuccess ? NoContent()
                 : result.ToProblem(StatusCodes.Status404NotFound);
}
```

- Add Update Question EndPoint -

① هنروح ال IQuestionService

```
Task<Result> UpdateAsync(int PollId, int id, QuestionRequest request
        CancellationToken cancellationToken = default);
```

② هنروح ال QuestionService

```
public async Task<Result> UpdateAsync( ← )
{
```

أول حاجة هنـ check على الـ تبع ال Index اللي عملناه يعني لو في انا عاوز أعدل ال q2 لـ نفس ال Poll مثلا في q1 هيرجع أجيب نفس ال Content اللي موجود في error

```
var questionIsExists = await _context.Questions
        .AnyAsync(x => x.PollId == PollId &&
                x.Id != id &&
                x.Content == request.Content,
                cancellationToken
        );
```

```
if (questionIsExist)
    return Result.Failure(QuestionErrors.DuplicatedQuestionContent);
```

هنشتغل على ال question بقا

```
var question = await Context.Questions
    .Include(x => x.Answers)
    .SingleOrDefaultAsync(x => x.PollId == PollId
        && x.Id == id, cancellationToken);

if (question is null)
    return Result.Failure(QuestionErrors.QuestionNotFound);
```

هنقدر نغير ال Content

```
question.Content = request.Content;
```

علشان نشتغل على ال Answers ف أنا عندي 3 احتمالات لل answer

① ال answers اللي موجودة already في الداتا بيز

```
var CurrentAnswers = question.Answers.Select(x => x.Conten)
                                      .ToList();
```

② هنضيف ال answers الجديدة اللي مش موجودة في الداتا بيز

```
var newAnswers = request.Answers.Except(CurrentAnswers).ToList();
```

هو كدا جاب ال answers اللي جديدة موجودة في ال request بس مش موجودة في ال DB

③ الوقتي هنضيف ال Answers الجديدة جوا ال question

```
newAnswers.ForEach(answer =>
{
    question.Answer.Add(new Answer { Content = answer });
});
```

ال CurrentAnswer لو كانت موجودة ضمن ال newAnswer ف ال IsActive بتاعتها هتفضل ب True غير كدا هتكون ب False

```
question.Answers.ToList().ForEach(answer =>
{
    answer.IsActive = request.Answers.Contains(answer.Content);
});

await Context.SaveChangesAsync(cancellationToken);
return Result.Success([illegible]);
}
```

EndPoint لل Update

```
[HttpPut("{id}")]
Public Async Task<IActionResult> Update([FromRoute] int PollId,
        [FromRoute] int id, [FromBody] QuestionRequest,
        CancellationToken cancellationToken = default)
{
  var result = await _questionService.UpdateAsync(PollId, id,
                request, CancellationToken);

  if(result.IsSuccess)
      return NoContent();

  return result.Error!.Equals(QuestionErrors.DuplicatedQuestionCon
        ? result.ToProblem(StatusCodes.Status409Conflict)
        : result.ToProblem(StatusCodes.Status404NotFound);
}
```

- Working On Votes Module -

- Add Domain Models - (vote), (VoteAnswer).

```
Public Sealed class vote
{
    Public int Id {get; Set;}
    Public int PollId {get; Set;}
    Public String UserId {get; Set;}
    Public DateTime SubmittedOn {get; Set;}

    Public Poll Poll {get; Set;} = default! ;
    Public ApplicationUser User {get; Set;} = default!;
    Public ICollection <VoteAnswer> VoteAnswers {get; Set;} = [];
}

Public sealed class VoteAnswer
{
    Public int Id {get; Set;}
    Public int VoteId {get; Set;}
    Public int QuestionId {get; Set;}
    Public int AnswerId {get; Set;}

    Public Vote vote {get; Set;} = default!;
    Public Question Question {get; Set;} = default! ;
    Public Answer Answer {get; Set;} default! ;
}
```

هنروح على ال Poll نضيف فيها New Prop

```
Public ICollection <Vote> Votes {get; Set;} = [];
```

هنروح نضيف ال DbSets في ال ApplicationDbContext

```
Public DbSet <Vote> Votes {get; Set;}
Public DbSet <VoteAnswers> VoteAnswers {get; Set;}
```

هنروح نضيف ال Configuration

```
Public class VoteConfiguration : IEntityTypeConfiguration <Vote>
{
    Public void Configure (EntityTypeBuilder <Answer> builder)
    {
        builder.HasIndex (x => new {x.PollId , x.UserId }).IsUnique();
    }
}
```

```
Public class VoteAnswerConfiguration : IEntityTypeConfiguration<
                                         VoteAnswer>
{
  Public void Configure ( EntityTypeBuilder <VoteAnswer> builder)
  {
    builder. HasIndex (x => new { x. VoteId, x.QuestionId }).IsUnique
  }
}
```

هنضيف ال Migration و Update-DataBase

- Add Get Current Polls EndPoint -

هنضيفها ال Polls

① IPollService

```
Task <IEnumerable <PollResponse>> GetCurrentAsync (CancellationToken
                                   CancellationToken=default);
```

② PollService

```
Public async Task <IEnumerable <PollResponse>> GetCurrentAsync ( )
{
return
  await _Context. Polls.
        .Where (x => x.IsPublish &&
              x.StartsAt <= DateOnly.FromDateTime (DateTime.UtcNow)
          && x.EndsAt >= DateOnly.FromDateTime (DateTime.UtcNow)
        .AsNoTracking()
        .ProjectToType<PollResponse>()
        .ToListAsync (CancellationToken) ;
}
```

الفكرة هنا انى عاوزة أ Select ال Polls اللي ال Publish بتاعها ب True وكمان بوقت
الكاين بين ال Start وال End

③ PollController

```
[HttpGet("current")]
Public async Task <IActionResult> GetCurrent (CancellationToken
                                    CancellationToken)
{
 return Ok( await _PollService. GetCurrentAsync (CancellationToken));
}
```

- Add Get Available Questions Method -

Questions ال تعديلات

IQuestionService ①

```
Task <Result<IEnumerable<QuestionResponse>>> GetAvailableAsync(int PollId,
        String UserId, CancellationToken cancellationToken = default);
```

QuestionService ②

```
Public async Task<Result<IEnumerable<QuestionResponse>>> GetAvailableAsync
        ( ← )
```

① هنا هنعمل check نقطة هل ال user عمل vote قبل كدا على نفس ال Poll ولا لا

```
{
var hasVote = await _context.Votes.AnyAsync(x => x.PollId == PollId
        && x.UserId == UserId, cancellationToken);

if (hasVote)
    return Result.Failure<IEnumerable<QuestionResponse>>(VoteErrors
                                        .DuplicatedVote);
```

② هنا check على ال Poll ان اكون موجود و IsPublish و اوقات تاريخه صالح ولا لا

```
var PollIsExist = await _context.Polls.AnyAsync(x => x.Id == PollId &&
    x.IsPublished && x.StartsAt <= DateOnly.FromDateTime(DateTime.UtcNow)
    && x.EndsAt >= DateOnly.FromDateTime(DateTime.UtcNow));

if (!PollIsExist)
    return Result.Failure<IEnumerable<QuestionResponse>>(PollErrors.
                                        PollNotFound);
```

③ هنا هنجيب الاسئلة اللي موجودة في ال Poll و IsActive و هنجيب ال Answers اللي IsActive برضه ولا لا

```
var questions = await _context.Questions
        .Where(x => x.PollId == PollId && x.IsActive)
        .Include(x => x.Answers)
        .Select(q => new QuestionResponse(
                q.Id,
                q.Content,
                q.Answers.Where(a => a.IsActive)
                        .Select(a => new AnswerResponse
                                (a.Id, a.Content))
        ))
        .AsNoTracking()
        .ToListAsync(cancellationToken);

return Result.Success<IEnumerable<QuestionResponse>(questions);
}
```

VoteErrors class لإنشاء

```
Public Static class VoteErrors
{
    Public Static readonly Error DuplicatedVote =
        new ("vote.DuplicatedVote", "this User is already voted
                before for this Poll");
}
```

- Add StartVote EndPoint -

VoteController لإنشاء

```
[Route ("api/Polls/{PollId}/vote")]
[ApiController]
[Authorize]
Public class VotesController (IQuestionService questionService)
                : ControllerBase
{
    Private readonly IQuestionService _questionService = questionService;

    [HttpGet ("")]
    Public async Task<IActionResult> Start ([FromRoute] int PollId,
                CancellationToken cancellationToken)
    {
        var userId = User.FindFirstValue (ClaimTypes.NameIdentifier);
        var result = await _questionService.GetAvailableAsync (PollId,
                userId!, cancellationToken);

        if (result.IsSuccess)
            return Ok (result.Value);

        return result.Error!.Equals (VoteErrors.DuplicatedVote)
                ? result.Problem (StatusCode.Conflict)
                : result.Problem (StatusCodes.Status404NotFound);
    }
```

- Add GetUserId Extension Method -

هنضيف Folder جديد اسمه Extensions هنضيف فيه class اسمه UserExtensions

```
public static class UserExtensions
{
    public static string? GetUserId (this ClaimsPrincipal user)
        => user.FindFirstValue(ClaimTypes.NameIdentifier);
}
```

هنروح نستعملها في ال EndPoint الـ Start ← ApplicationDbContext ال

```
User.GetUserId();
```

- Add Save Vote Method -

هنضيف في ال Contracts Folder جديد اسمه Votes هنضيف فيه 2 records

voteRequest ⟷ voteAnswerRequest

```
public record VoteRequest (
    IEnumerable<VoteAnswerRequest> Answers
    );

public record VoteAnswerRequest (
    int QuestionId,
    int AnswerId
    );
```

هنضيف VoteRequestValidator ←

```
public class VoteRequestValidator : AbstractValidator<VoteRequest>
{
    public VoteRequestValidator()
    {
        RuleFor(x => x.Answers)
            .NotEmpty();

        RuleForEach(x => x.Answers)
            .SetInheritanceValidator(v =>
                v.Add(new VoteAnswerRequestValidator())
            );
    }
}
```

بس علشان كده ال 9 علشان ال voteRequest عنده ال Answers من نوع VoteAnswerRequest يعني مش نقدر نعمل inherit من voteRequest فا هنحتاج برده ال validation بتاعت voteAnswerRequest اللي عملناه

VoteAnswerRequestValidator ال تضيف

```
Public class VoteAnswerRequestValidator : AbstractValidator<
                                          VoteAnswerRequest>
{
  Public voteAnswerRequestValidator()
  {
    RuleFor(x => x.QuestionId)
              .GreaterThan(0);

    RuleFor(x => x.AnswerId)
              .GreaterThan(0);
  }
}
```

صنوع على ال Services تضيف VoteService وال IVoteService
ونعملهم inject في ال DI ✓

```
Services.AddScoped<IVoteService, VoteService>();
```

في ال IVoteService {

```
Task<Result> AddAsync(int PollId, String userId, VoteRequest request,
            CancellationToken cancellationToken = default);
```

صنوع على ال VoteService {

```
Public class VoteService(ApplicationDbContext Context) : IVoteService
{
  Private readonly ApplicationDbContext _context = context;

  Public async Task<Result> AddAsync(✓)
  {
    var hasVote = await _context.Votes.AnyAsync(x => x.PollId == PollId
        && x.UserId == userId, cancellationToken);
    if(hasVote)
      return Result.Failure(VoteErrors.DuplicatedVote);

    var PollIsExist = await _context.Polls.AnyAsync(x => x.Id == PollId
        && x.IsPublished && x.StartsAt <= DateOnly.FromDateTime(
        DateTime.UtcNow) && x.EndsAt >= DateOnly.FromDateTime(
        DateTime.UtcNow), cancellationToken);

    if(! PollIsExist)
      return Result.Failure(PollErrors.PollNotFound);

    // نتأكد ان ال request في ال Questions التي في ال Check الوقت هنا ان ال
    // active و Poll الي موجودة عندي في الداتا بيز في نفس ال
```

```
var availableQuestions = await _context.Questions
    .Where(x => x.PollId == pollId && x.IsActive)
    .Select(x => x.Id)
    .ToListAsync(cancellationToken);

if (!request.Answers.Select(x => x.QuestionId)
    .SequenceEqual(availableQuestions))
    return Result.Failure(VoteErrors.InvalidQuestions);
```

vote هنبدأ هنا بإنشاء

```
var vote = new Vote
{
    PollId = pollId,
    UserId = userId,
    VoteAnswers = request.Answers.Adapt<IEnumerable<VoteAnswer>>()
        .ToList()
};

await _context.AddAsync(vote, cancellationToken);
await _context.SaveChangesAsync(cancellationToken);
return Result.Success();
```

}

VoteErrors هنروح في الـ

```
public static readonly Error InvalidQuestions =
    new("Vote.InvalidQuestions", "Invalid Questions");
```

VoteController في الـ EndPoint هنعمل

IVoteService الأول هنعمل inject

```
[HttpPost("")]
public async Task<IActionResult> Vote([FromRoute] int pollId,
    [FromBody] VoteRequest request, CancellationToken cancellationToken)
{
    var result = await _voteService.AddAsync(pollId, User.GetUserId()!,
        request, cancellationToken);

    if (result.IsSuccess)
        return Created();
    return result.Error.Equals(VoteErrors.DuplicatedVote)
        ? result.ToProblem(StatusCodes.Status409Conflict)
        : result.ToProblem(404);
}
```

Code Refactoring:

هنعمل refactor للجزء بتاع الـ error بحيث لو حابب أشوف أرجع الـ StatusCode الـ ToProblem وأستخدمه كتير في الـ controller

هنروح للـ record اللي اسمه Error وهنضيف parameter جديد

```
Public record Error (String Code, String Description, int? StatusCode)
{
    Public Static readonly Error None = new (String.Empty,
                                   String.Empty, null);
```

وهنروح على كل error موجود عندي ونبعتله الـ StatusCode ✓

وهنعدل على الـ ToProblem ونشيل الـ Parameter بتاع الـ StatusCode

وهنقوله

```
var Problem = Results.Problem
  (StatusCode = result.Error.StatusCode);
```

وفي الـ ProblemDetails

```
"errors", new[]
  {
      result.Error.Code,
      result.Error.Description
```

ونروح على الـ Controllers اللي عندي ونشيل الـ Parameter لأن خلاص الـ ToProblem بتجيبه