1. **Create Domain Model**

namespace NZWalksAPI.Models.Domain

{

public class Image

{

public Guid Id { get; set; }

[NotMapped]

public IFormFile File { get; set; }

public string FileName { get; set; }

public string? Description { get; set; }

public string FileExtension { get; set; }

public long FileSizeInBytes { get; set; }

public string FilePath { get; set; }

}

}

1. **Add this Domain inside Dbcontext file**

public NZWalksDbContext(DbContextOptions<NZWalksDbContext> options) : base(options)

{

}

public DbSet<Walk> Walks { get; set; }

public DbSet<Region> Regions { get; set; }

public DbSet<Difficulty> Difficulties { get; set; }

public DbSet<Image> Images { get; set; }

1. **Add mirgaran and run the migration**

PM> add-migration createImageTable -context NZWalksDbContext -outputDir Migrations

PM> update-database -context NZWalksDbContext

**Create DTO**

namespace NZWalksAPI.Models.DTO

{

public class ImageUploadRequestDTO

{

[Required]

public IFormFile File { get; set; }

[Required]

public string FileName { get; set; }

public string? Description { get; set; }

}

}

**ImageController**

namespace NZWalksAPI.Controllers

{

[Route("api/[controller]")]

[ApiController]

public class ImagesController : ControllerBase

{

private readonly IImageRepository \_imageRepository;

public ImagesController(IImageRepository imageRepository)

{

\_imageRepository = imageRepository;

}

[HttpPost]

[Route("Upload")]

public async Task<IActionResult> UploadImage([FromForm] ImageUploadRequestDTO imgRequest)

{

ValidateFileUpload(imgRequest);

if (!ModelState.IsValid)

{

return BadRequest(ModelState);

}

//convert dto to domain model since repository layer works with domain models

var imageDomainModel = new Image

{

File = imgRequest.File,//Iform file

FileExtension = Path.GetExtension(imgRequest.File.FileName).ToLowerInvariant(),

FileSizeInBytes = imgRequest.File.Length,

FileName = imgRequest.FileName,

Description = imgRequest.Description,

//FilePath will be set later in the repository

};

// Call the repository to save the image

await \_imageRepository.Upload(imageDomainModel);

return Ok(imageDomainModel);

}

//for image validation extension and size

private void ValidateFileUpload(ImageUploadRequestDTO imgRequest)

{

var allowedExtensions = new string[] { ".jpg", ".jpeg", ".png" };

var fileExtension = Path.GetExtension(imgRequest.File.FileName).ToLowerInvariant();

if (!allowedExtensions.Contains(fileExtension))

{

//throw new Exception("Invalid file type. Only .jpg, .jpeg, and .png files are allowed.");

ModelState.AddModelError("File", "Invalid file type. Only .jpg, .jpeg, and .png files are allowed.");

}

if (imgRequest.File.Length > 5 \* 1024 \* 1024) // 5 MB limit

{

//throw new Exception("File size exceeds the maximum limit of 5 MB.");

ModelState.AddModelError("File", "File size exceeds the maximum limit of 5 MB.");

}

}

}}

**Repository**

namespace NZWalksAPI.Repositories

{

public interface IImageRepository

{

Task<Image> Upload(Image image);

}

}

public class LocalImageRepository : IImageRepository

{

private readonly IWebHostEnvironment \_evm;

private readonly NZWalksDbContext \_dbContext;

private readonly IHttpContextAccessor \_httpContextAccessor;

public LocalImageRepository(IWebHostEnvironment evm, NZWalksDbContext dbContext, IHttpContextAccessor httpContextAccessor)

{

\_evm = evm;

\_dbContext = dbContext;

\_httpContextAccessor = httpContextAccessor;

}

public async Task<Image> Upload(Image image)

{

//combine given file name with extension

//var fileNameWithExtension = $"{image.FileName}{image.FileExtension}";

var fileNameWithExtension = Guid.NewGuid().ToString()+"\_"+$"{image.FileName}{image.FileExtension}";

var localFilePath = Path.Combine(\_evm.ContentRootPath, "Images", fileNameWithExtension);

//upload image to local path

using var stream = new FileStream(localFilePath, FileMode.Create);

await image.File.CopyToAsync(stream);

//create path - https://localhost:5000/Images/abc.jpg

//var urlFilePath = $"{\_httpContextAccessor.HttpContext.Request.Scheme}://{\_httpContextAccessor.HttpContext.Request.Host} /Images/{fileNameWithExtension}";

var urlFilePath = $"{\_httpContextAccessor.HttpContext.Request.Scheme}://{\_httpContextAccessor.HttpContext.Request.Host}{\_httpContextAccessor.HttpContext.Request.PathBase}/Images/{fileNameWithExtension}";

image.FilePath = urlFilePath; //set the file path to the image object

//add the image to the database table

await \_dbContext.Images.AddAsync(image);

await \_dbContext.SaveChangesAsync();

return image;

}

}