Part -B MM

4-task ➜ Create Your DbContext Class

5-task ➜ Run migration + update database:

6-task ➜ Controllers department

🌺 **6.1 ➜ 1-Group — Registration Controller File**

🌺**6.2 ➜ 2-Group — Expectations Controller file**

🌺**6.3 ➜ 3 -Group —*UploadPhotos Controller***

🌺**6.4 ➜ 4 -Group —verify controller**

🌺**6.5 ➜ 5 -Group —Payment controller**

🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸

4-task ➜ Create Your DbContext Class

🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🟢

STEP 1 — Create Data Folder

In Solution Explorer, right-click on your project name — e.g. MatrimonyAPI.➜ Choose ➜ Add → New Folder ➜ Name the folder exactly: Data

---

🟢 STEP 2 — Create AppDbContext.cs File

1. Right-click the new Data folder → Add → Class…➜ Name the file: ➜ AppDbContext.cs

2. Click Add — Visual Studio will open the file automatically.

```
using Microsoft.EntityFrameworkCore;
using MatrimonyAPI.Models;

namespace MatrimonyAPI.Data
{
    public class AppDbContext : DbContext
    {
        public AppDbContext(DbContextOptions<AppDbContext> options) :
base(options)
        {
        }

        // 🔹 1. Basic Registration
        public DbSet<Register> Register { get; set; }

        // 🔹 2. Education + Career
        public DbSet<EducationCareer> EducationCareer { get; set; }

        // 🔹 3. Family Information
        public DbSet<FamilyInfo> FamilyInfo { get; set; }

        // 🔹 4. Sibling Information
        public DbSet<Sibling> Sibling { get; set; }

        // 🔹 5. Lifestyle Information
        public DbSet<LifestyleInfo> LifestyleInfo { get; set; }

        // 🔹 6. Astrology Information
        public DbSet<AstrologyInfo> AstrologyInfo { get; set; }

        // 🔹 7. Expectations
        public DbSet<Expectations> Expectations { get; set; }

        // 🔹 8. Photos Upload
        public DbSet<UploadPhotos> UploadPhotos { get; set; }

        // 🔹 9. Verification
        public DbSet<Verify> Verify { get; set; }

        // 🔹 10. Payment Information
        public DbSet<Payment> Payment { get; set; }
```

```
    }
}
```

🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸

5-task Run migration + update database:

🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸

1️⃣ Go to Tools on top menu
2️⃣ Click NuGet Package Manager
3️⃣ Click Package Manager Console

```
Add-Migration InitialCreate  ----------------Create Migration
Update-Database ------------------- Apply Migration to SQL Server
```

🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸

6-task Controllers department

🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸

Now you must create API Controllers for each section.

Each controller will handle CRUD:

- POST → Insert
- GET → Fetch
- PUT → Update
- DELETE → Delete

🟢 1-STEP: Create Controllers Automatically

For each model (Register, EducationCareer, etc.):

In Visual Studio:

1. Right-click Controllers folder ➜ Choose: Add → New Scaffolded Item ➜ Select: API Controller with actions using Entity Framework
2. ➜ Choose:
   - Model Class: Register
   - DbContext: AppDbContext
3. Click Add

Repeat this for all:

Section	Model Class	Controller Name
1	Register	RegisterController
2	EducationCareer	EducationCareerController
3	FamilyInfo	FamilyInfoController
4	Sibling	SiblingController
5	LifestyleInfo	LifestyleInfoController
6	AstrologyInfo	AstrologyInfoController
7	Expectations	ExpectationsController
8	UploadPhotos	UploadPhotosController
9	Verify	VerifyController
10	Payment	PaymentController

⭐ BEST PRACTICE FOR MATRIMONY PROJECT

Since you have multi-step profile, I recommend:

🔹 Group related tables:

1) Registration / Personal Info Controller

- Register

- **EducationCareer**
- **FamilyInfo**
- **Sibling**
- **LifestyleInfo**
- **AstrologyInfo**

2) Expectations Controller

- **Expectations**

3) Photos Controller

- **UploadPhotos**

4) Verification Controller

- **Verify**

5) Payment Controller

- **Payment**

This structure is clean and industry-standard.

✔ Group 2 — Expectations Controller

Separate controller because it is its own screen.

✔ Group 3 — UploadPhotos Controller

Separate controller because image uploading is an independent module.

✔ Group 4 — Verify Controller

Correct — verification is a separate process.

✔ Group 5 — Payment Controller

Correct — payments should always be a dedicated controller.

1 -Group — Registration Controller File

In Visual Studio:

Controllers → Add → New Item → API Controller Empty Name it:
RegistrationController.cs

```
using Microsoft.AspNetCore.Mvc;
using MatrimonyAPI.Data;
using MatrimonyAPI.Models;
using Microsoft.EntityFrameworkCore;

namespace MatrimonyAPI.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class RegistrationController : ControllerBase
    {
        private readonly AppDbContext _context;

        public RegistrationController(AppDbContext context)
        {
            _context = context;
        }

        // ------------------------------------------------------------
        //                    REGISTER CRUD
        // ------------------------------------------------------------

        // CREATE
        [HttpPost("register")]
        public async Task<IActionResult> CreateRegister(Register model)
        {
            _context.Register.Add(model);
            await _context.SaveChangesAsync();
            return Ok(model);
        }
```

```
        // GET ALL
        [HttpGet("register")]
        public async Task<IActionResult> GetAllRegister()
        {
            return Ok(await _context.Register.ToListAsync());
        }

        // GET BY ID
        [HttpGet("register/{id}")]
        public async Task<IActionResult> GetRegisterById(int id)
        {
            var data = await _context.Register.FindAsync(id);
            if (data == null) return NotFound();
            return Ok(data);
        }

        // UPDATE
        [HttpPut("register/{id}")]
        public async Task<IActionResult> UpdateRegister(int id, Register
model)
        {
            if (id != model.UserID) return BadRequest("UserID mismatch");

            _context.Entry(model).State = EntityState.Modified;
            await _context.SaveChangesAsync();
            return Ok("Updated Successfully");
        }

        // DELETE
        [HttpDelete("register/{id}")]
        public async Task<IActionResult> DeleteRegister(int id)
        {
            var data = await _context.Register.FindAsync(id);
            if (data == null) return NotFound();

            _context.Register.Remove(data);
            await _context.SaveChangesAsync();
            return Ok("Deleted Successfully");
        }

        // ------------------------------------------------------------
        //              EDUCATION CAREER CRUD
        // ------------------------------------------------------------

        [HttpPost("education-career")]
```

```
        public async Task<IActionResult>
AddEducationCareer(EducationCareer model)
        {
            _context.EducationCareer.Add(model);
            await _context.SaveChangesAsync();
            return Ok(model);
        }

        [HttpGet("education-career/{userId}")]
        public async Task<IActionResult> GetEducationCareer(int userId)
        {
            var data = await _context.EducationCareer
                .FirstOrDefaultAsync(x => x.UserID == userId);

            if (data == null) return NotFound();
            return Ok(data);
        }

        [HttpPut("education-career/{id}")]
        public async Task<IActionResult> UpdateEducationCareer(int id,
EducationCareer model)
        {
            if (id != model.EducationCareerID) return BadRequest("ID
mismatch");

            _context.Entry(model).State = EntityState.Modified;
            await _context.SaveChangesAsync();
            return Ok("Updated");
        }

        [HttpDelete("education-career/{id}")]
        public async Task<IActionResult> DeleteEducationCareer(int id)
        {
            var data = await _context.EducationCareer.FindAsync(id);
            if (data == null) return NotFound();

            _context.EducationCareer.Remove(data);
            await _context.SaveChangesAsync();
            return Ok("Deleted Successfully");
        }

        // ------------------------------------------------------------
        //                      FAMILY INFO CRUD
        // ------------------------------------------------------------

        [HttpPost("family-info")]
        public async Task<IActionResult> AddFamilyInfo(FamilyInfo model)
```

```
        {
            _context.FamilyInfo.Add(model);
            await _context.SaveChangesAsync();
            return Ok(model);
        }

        [HttpGet("family-info/{userId}")]
        public async Task<IActionResult> GetFamilyInfo(int userId)
        {
            var data = await _context.FamilyInfo
                .FirstOrDefaultAsync(x => x.UserID == userId);

            if (data == null) return NotFound();
            return Ok(data);
        }

        [HttpPut("family-info/{id}")]
        public async Task<IActionResult> UpdateFamilyInfo(int id,
FamilyInfo model)
        {
            if (id != model.FamilyID) return BadRequest("ID mismatch");

            _context.Entry(model).State = EntityState.Modified;
            await _context.SaveChangesAsync();
            return Ok("Updated");
        }

        [HttpDelete("family-info/{id}")]
        public async Task<IActionResult> DeleteFamilyInfo(int id)
        {
            var data = await _context.FamilyInfo.FindAsync(id);
            if (data == null) return NotFound();

            _context.FamilyInfo.Remove(data);
            await _context.SaveChangesAsync();
            return Ok("Deleted Successfully");
        }

        // ------------------------------------------------------------
        //                      SIBLING CRUD
        // ------------------------------------------------------------

        [HttpPost("sibling")]
        public async Task<IActionResult> AddSibling(Sibling model)
        {
            _context.Sibling.Add(model);
            await _context.SaveChangesAsync();
```

```
            return Ok(model);
        }

        [HttpGet("sibling/{userId}")]
        public async Task<IActionResult> GetSibling(int userId)
        {
            var data = await _context.Sibling
                .Where(x => x.UserID == userId)
                .ToListAsync();

            return Ok(data);
        }

        [HttpPut("sibling/{id}")]
        public async Task<IActionResult> UpdateSibling(int id, Sibling
model)
        {
            if (id != model.SiblingID) return BadRequest("ID mismatch");

            _context.Entry(model).State = EntityState.Modified;
            await _context.SaveChangesAsync();
            return Ok("Updated");
        }

        [HttpDelete("sibling/{id}")]
        public async Task<IActionResult> DeleteSibling(int id)
        {
            var data = await _context.Sibling.FindAsync(id);
            if (data == null) return NotFound();

            _context.Sibling.Remove(data);
            await _context.SaveChangesAsync();
            return Ok("Deleted Successfully");
        }

        // ------------------------------------------------------------
        //                  LIFESTYLE INFO CRUD
        // ------------------------------------------------------------

        [HttpPost("lifestyle")]
        public async Task<IActionResult> AddLifestyle(LifestyleInfo model)
        {
            _context.LifestyleInfo.Add(model);
            await _context.SaveChangesAsync();
            return Ok(model);
        }
```

```
        [HttpGet("lifestyle/{userId}")]
        public async Task<IActionResult> GetLifestyle(int userId)
        {
            var data = await _context.LifestyleInfo
                .FirstOrDefaultAsync(x => x.UserID == userId);

            if (data == null) return NotFound();
            return Ok(data);
        }

        [HttpPut("lifestyle/{id}")]
        public async Task<IActionResult> UpdateLifestyle(int id,
LifestyleInfo model)
        {
            if (id != model.LifestyleID) return BadRequest("ID mismatch");

            _context.Entry(model).State = EntityState.Modified;
            await _context.SaveChangesAsync();
            return Ok("Updated");
        }

        [HttpDelete("lifestyle/{id}")]
        public async Task<IActionResult> DeleteLifestyle(int id)
        {
            var data = await _context.LifestyleInfo.FindAsync(id);
            if (data == null) return NotFound();

            _context.LifestyleInfo.Remove(data);
            await _context.SaveChangesAsync();
            return Ok("Deleted Successfully");
        }

        // ------------------------------------------------------------
        //                    ASTROLOGY INFO CRUD
        // ------------------------------------------------------------

        [HttpPost("astrology")]
        public async Task<IActionResult> AddAstrology(AstrologyInfo model)
        {
            _context.AstrologyInfo.Add(model);
            await _context.SaveChangesAsync();
            return Ok(model);
        }

        [HttpGet("astrology/{userId}")]
        public async Task<IActionResult> GetAstrology(int userId)
        {
```

```
            var data = await _context.AstrologyInfo
                .FirstOrDefaultAsync(x => x.UserID == userId);

            if (data == null) return NotFound();
            return Ok(data);
        }

        [HttpPut("astrology/{id}")]
        public async Task<IActionResult> UpdateAstrology(int id,
AstrologyInfo model)
        {
            if (id != model.AstrologyID) return BadRequest("ID mismatch");

            _context.Entry(model).State = EntityState.Modified;
            await _context.SaveChangesAsync();
            return Ok("Updated");
        }

        [HttpDelete("astrology/{id}")]
        public async Task<IActionResult> DeleteAstrology(int id)
        {
            var data = await _context.AstrologyInfo.FindAsync(id);
            if (data == null) return NotFound();

            _context.AstrologyInfo.Remove(data);
            await _context.SaveChangesAsync();
            return Ok("Deleted Successfully");
        }
    }
}
```

2 Group— Create ExpectationsController

In Visual Studio: Right-click Controllers → Add → Controller → API Controller – Empty Name it:ExpectationsController

```
using Microsoft.AspNetCore.Mvc;

using Microsoft.EntityFrameworkCore;

using MatrimonyAPI.Data;

using MatrimonyAPI.Models;


namespace MatrimonyAPI.Controllers

{

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

    [ApiController]

    public class ExpectationsController : ControllerBase

    {

        private readonly AppDbContext _context;


        public ExpectationsController(AppDbContext context)

        {

            _context = context;

        }


        // GET: api/Expectations

        [HttpGet]

        public async Task<IActionResult> GetAll()

        {

            var list = await _context.Expectations.ToListAsync();

            return Ok(list);
```

```
        }

        // GET: api/Expectations/5
        [HttpGet("{id}")]
        public async Task<IActionResult> GetById(int id)
        {
            var expectation = await _context.Expectations.FindAsync(id);

            if (expectation == null)
                return NotFound();

            return Ok(expectation);
        }

        // POST: api/Expectations
        [HttpPost]
        public async Task<IActionResult> Create([FromBody] Expectations
model)
        {
            if (!ModelState.IsValid)
                return BadRequest(ModelState);

            await _context.Expectations.AddAsync(model);
            await _context.SaveChangesAsync();

            return Ok(model);
```

```
        }

        // PUT: api/Expectations/5
        [HttpPut("{id}")]
        public async Task<IActionResult> Update(int id, [FromBody]
Expectations model)
        {
            var expectation = await _context.Expectations.FindAsync(id);

            if (expectation == null)
                return NotFound();

            _context.Entry(expectation).CurrentValues.SetValues(model);
            await _context.SaveChangesAsync();

            return Ok(expectation);
        }

        // DELETE: api/Expectations/5
        [HttpDelete("{id}")]
        public async Task<IActionResult> Delete(int id)
        {
            var expectation = await _context.Expectations.FindAsync(id);

            if (expectation == null)
                return NotFound();
```

```
            _context.Expectations.Remove(expectation);
            await _context.SaveChangesAsync();

            return Ok("Deleted successfully");
        }
    }
}
```

🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸

3rd Group — UploadPhotos Controller

🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸🌸✅ **STEP 1 — In Visual Studio**

Go to:

Right-click on Controllers folder→ Add → Controller

✅ STEP 2 — Choose Template

Select:API Controller - Empty

✅ STEP 3 — Name Your Controller

Enter this name:UploadPhotosController - UploadPhotosController Click Add.

```
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using MatrimonyAPI.Data;
using MatrimonyAPI.Models;
```

```
namespace MatrimonyAPI.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class UploadPhotosController : ControllerBase
    {
        private readonly AppDbContext _context;

        public UploadPhotosController(AppDbContext context)
        {
            _context = context;
        }

        // GET: api/UploadPhotos
        [HttpGet]
        public async Task<IActionResult> GetAll()
        {
            var photos = await _context.UploadPhotos.ToListAsync();
            return Ok(photos);
        }

        // GET: api/UploadPhotos/5
        [HttpGet("{id}")]
        public async Task<IActionResult> GetById(int id)
        {
```

```
            var photo = await _context.UploadPhotos.FindAsync(id);

            if (photo == null)
                return NotFound();

            return Ok(photo);
        }

        // POST: api/UploadPhotos
        [HttpPost]
        public async Task<IActionResult> Create([FromBody] UploadPhotos
model)
        {
            if (!ModelState.IsValid)
                return BadRequest(ModelState);

            await _context.UploadPhotos.AddAsync(model);
            await _context.SaveChangesAsync();

            return Ok(model);
        }

        // PUT: api/UploadPhotos/5
        [HttpPut("{id}")]
        public async Task<IActionResult> Update(int id, [FromBody]
UploadPhotos model)
```

```
{
    var photo = await _context.UploadPhotos.FindAsync(id);

    if (photo == null)
        return NotFound();

    _context.Entry(photo).CurrentValues.SetValues(model);
    await _context.SaveChangesAsync();

    return Ok(photo);
}

// DELETE: api/UploadPhotos/5
[HttpDelete("{id}")]
public async Task<IActionResult> Delete(int id)
{
    var photo = await _context.UploadPhotos.FindAsync(id);

    if (photo == null)
        return NotFound();

    _context.UploadPhotos.Remove(photo);
    await _context.SaveChangesAsync();

    return Ok("Deleted successfully");
```

```
        }
    }
}
```

3

4-Group – verify controller

In Visual Studio: Right-click Controllers → Add → Controller → API Controller – Empty Name it:VerifyController.cs

```
using Microsoft.AspNetCore.Mvc;
using MatrimonyAPI.Data;
using MatrimonyAPI.Models;
using Microsoft.EntityFrameworkCore;

namespace MatrimonyAPI.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class VerifyController : ControllerBase
    {
        private readonly AppDbContext _context;

        public VerifyController(AppDbContext context)
        {
            _context = context;
        }
```

```
// ============================================
// GET ALL VERIFICATION RECORDS
// ============================================
[HttpGet]
public async Task<ActionResult<IEnumerable<Verify>>> GetAll()
{
    return await _context.Verify.ToListAsync();
}


// ============================================
// GET SINGLE VERIFICATION BY ID
// ============================================
[HttpGet("{id}")]
public async Task<ActionResult<Verify>> GetById(int id)
{
    var verify = await _context.Verify.FindAsync(id);

    if (verify == null)
        return NotFound();

    return verify;
}


// ============================================
// CREATE NEW VERIFICATION ENTRY
```

```
        // ============================================

        [HttpPost]

        public async Task<ActionResult<Verify>> Create(Verify model)

        {

            _context.Verify.Add(model);

            await _context.SaveChangesAsync();


            return CreatedAtAction(nameof(GetById), new { id =
model.VerifyID }, model);

        }


        // ============================================

        // UPDATE VERIFICATION ENTRY

        // ============================================

        [HttpPut("{id}")]

        public async Task<IActionResult> Update(int id, Verify model)

        {

            if (id != model.VerifyID)

                return BadRequest("VerifyID mismatch");


            _context.Entry(model).State = EntityState.Modified;


            try

            {

                await _context.SaveChangesAsync();

            }
```

```
            catch (DbUpdateConcurrencyException)
            {
                if (!_context.Verify.Any(e => e.VerifyID == id))
                    return NotFound();
            }

            return NoContent();
        }

        // ============================================
        // DELETE VERIFICATION ENTRY
        // ============================================
        [HttpDelete("{id}")]
        public async Task<IActionResult> Delete(int id)
        {
            var verify = await _context.Verify.FindAsync(id);
            if (verify == null)
                return NotFound();

            _context.Verify.Remove(verify);
            await _context.SaveChangesAsync();

            return NoContent();
        }
    }
```

```
}
```

5-Group – Payment Controller

In Visual Studio: Right-click Controllers → Add → Controller → API Controller – Empty Name it:PaymentController.cs

```
using Microsoft.AspNetCore.Mvc;

using Microsoft.EntityFrameworkCore;

using MatrimonyAPI.Data;

using MatrimonyAPI.Models;


namespace MatrimonyAPI.Controllers

{

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

    [ApiController]

    public class PaymentController : ControllerBase

    {

        private readonly AppDbContext _context;


        public PaymentController(AppDbContext context)

        {

            _context = context;

        }
```

```
// ============================================
// GET ALL PAYMENTS
// ============================================
[HttpGet]
public async Task<ActionResult<IEnumerable<Payment>>> GetAll()
{
    return await _context.Payment.ToListAsync();
}

// ============================================
// GET PAYMENT BY ID
// ============================================
[HttpGet("{id}")]
public async Task<ActionResult<Payment>> GetById(int id)
{
    var payment = await _context.Payment.FindAsync(id);

    if (payment == null)
        return NotFound();

    return payment;
}

// ============================================
// CREATE NEW PAYMENT
```

```
        // ============================================

        [HttpPost]

        public async Task<ActionResult<Payment>> Create(Payment model)

        {

            _context.Payment.Add(model);

            await _context.SaveChangesAsync();


            return CreatedAtAction(nameof(GetById), new { id =
model.PaymentID }, model);

        }


        // ============================================

        // UPDATE PAYMENT

        // ============================================

        [HttpPut("{id}")]

        public async Task<IActionResult> Update(int id, Payment model)

        {

            if (id != model.PaymentID)

                return BadRequest("PaymentID mismatch");


            _context.Entry(model).State = EntityState.Modified;


            try

            {

                await _context.SaveChangesAsync();

            }
```

```
        catch (DbUpdateConcurrencyException)
        {
            if (!_context.Payment.Any(e => e.PaymentID == id))
                return NotFound();
        }

        return NoContent();
    }

    // ============================================
    // DELETE PAYMENT
    // ============================================
    [HttpDelete("{id}")]
    public async Task<IActionResult> Delete(int id)
    {
        var payment = await _context.Payment.FindAsync(id);
        if (payment == null)
            return NotFound();

        _context.Payment.Remove(payment);
        await _context.SaveChangesAsync();

        return NoContent();
    }
}
```

```
}
```