Right — the reason you can’t just keep using:

routes.MapPost("/upload", async (IFormFile file, FileUploadService service) => ...

…is because IFormFile represents **one single uploaded file**.  
If the client sends multiple files (<input name="files" multiple>), ASP.NET Core **will not** wrap them into one IFormFile — instead, it will populate HttpRequest.Form.Files with **a collection of IFormFile objects**.

That means:

* IFormFile file → works only if **one** file is uploaded with that field name.
* IFormFile[] files or List<IFormFile> files → works for multiple uploads, but only if the field names match (name="files" in HTML).

So in your case, if you want something shorter and avoid calling request.Form.Files.GetMultiple("files"), you **can** do this instead:

csharp

CopyEdit

routes.MapPost("/upload", async (List<IFormFile> files, FileUploadService service) =>

{

if (files == null || files.Count == 0)

return Results.BadRequest(new { error = "No files uploaded" });

var savedPaths = new List<string>();

foreach (var file in files)

{

var savedPath = await service.SaveFileAsync(file);

savedPaths.Add(savedPath);

}

return Results.Ok(new { paths = savedPaths });

});

**Conditions for this to work:**

1. Client <input> must have name="files".
2. All files must be sent under that same field name.
3. Request must be multipart/form-data.

If you want, I can rewrite your UploadController so it handles **both single and multiple uploads** automatically without changing the client. That way you can still accept one file or many.  
Want me to do that?