Setup Project Skills — MCP Server

A **Model Context Protocol (MCP) server** that auto-detects a project's technology stack and installs the matching GitHub Copilot skill folders into `<project>/skills/`, then updates `.vscode/settings.json` so Copilot discovers them.

Runs as a Docker container (recommended) or directly via `dotnet run`.
Communicates over the MCP stdio protocol — no network ports required.

Tools Exposed

Tool	Description
`detect_project_type`	Scans a directory and returns the detected technology type(s). **Read-only.**
`check_project_skills`	Dry-run: shows which skills would be added or are already present. **No changes.**
`setup_project_skills`	**Incremental** — copies new skill folders; leaves existing ones untouched.
`refresh_project_skills`	**Full refresh** — replaces all skill folders with the latest from source.

All tools accept these optional parameters:

Parameter	Description
`targetProject`	Absolute path to the project root. Defaults to the container's working directory.
`skillSourcePath`	Path to a local `awesome-copilot` clone. Skips auto-discovery.
`skillRepoUrl`	Git URL to clone if the repo isn't found locally. Overrides `AWESOME_COPILOT_REPO_URL`.

Detected Project Types

Detection scans all files (excluding `.git/`, `node_modules/`, `bin/`, `obj/`) and is non-exclusive — a project can match multiple types.

Type	Key indicators
`Blazor`	`.razor` files + `.csproj` (also receives `Frontend` skills)
`AspNetCoreApi`	`.csproj` + `Program.cs`, no Razor / MAUI / Designer files
`MAUI`	`Microsoft.NET.Sdk.Maui` SDK or `<UseMaui>true` in `.csproj`
`WinForms`	`.csproj` + `.designer.cs` / `.designer.vb`

Type	Key indicators
Android	AndroidManifest.xml, .kt, or .java files
Frontend	package.json + .ts/.tsx, or package.json without .csproj
CppCMake	.cpp, .vcxproj, or CMakeLists.txt
Unknown	Nothing matched — common skills only

Skills Installed

Common skills (every project)

git-commit · conventional-commit · create-readme · folder-structure-blueprint-generator · technology-stack-blueprint-generator · context-map · what-context-needed

Type-specific skills

Type	Skills
Blazor	fluentui-blazor · aspnet-minimal-api-openapi · ef-core · dotnet-best-practices · dotnet-design-pattern-review · csharp-async · csharp-docs · csharp-xunit · containerize-aspnetcore · multi-stage-dockerfile · refactor · create-specification · sql-optimization · sql-code-review
AspNetCoreApi	aspnet-minimal-api-openapi · ef-core · dotnet-best-practices · dotnet-design-pattern-review · csharp-async · csharp-docs · csharp-xunit · sql-optimization · sql-code-review · containerize-aspnetcore · multi-stage-dockerfile · openapi-to-application-code · create-specification
MAUI	dotnet-best-practices · dotnet-design-pattern-review · csharp-async · csharp-docs · csharp-xunit · dotnet-upgrade · multi-stage-dockerfile · refactor
WinForms	dotnet-best-practices · dotnet-upgrade · containerize-aspnet-framework · dotnet-design-pattern-review · refactor · csharp-docs
Android	kotlin-springboot · java-springboot · java-docs · java-junit · java-refactoring-extract-method · java-refactoring-remove-parameter · java-add-graalvm-native-image-support
Frontend	create-web-form · javascript-typescript-jest · markdown-to-html · multi-stage-dockerfile · refactor
CppCMake	multi-stage-dockerfile · refactor · refactor-method-complexity-reduce · sql-code-review

Any skill folder present in the awesome-copilot/skills/ source but not in the map above is also auto-discovered and installed when its name contains a keyword matching the detected type.

Skill Source Resolution

The server looks for `awesome-copilot` in this order:

1. `skillSourcePath` parameter (if provided)
2. `AWESOME_COPILOT_REPO_URL` environment variable (sets the clone URL)
3. `G:\Repos\frontlook-admin\AI_HELPERS\awesome-copilot`
4. `G:\Repos\frontlook-admin\awesome-copilot`
5. `%USERPROFILE%\repos\awesome-copilot`
6. `%USERPROFILE%\awesome-copilot`
7. `C:\src\awesome-copilot`
8. **Auto-clone** from `https://github.com/github/awesome-copilot`

When running in Docker the host paths (3–7) are not visible; the server will auto-clone on first use unless you mount a local clone (see Docker usage below).

Prerequisites

- .NET 10 SDK — for running locally
- Docker — for container usage
- VS Code with GitHub Copilot extension (for MCP discovery)
- `git` in PATH — required only if `awesome-copilot` must be auto-cloned

Running Locally (dotnet)

```
cd SkillMcp
dotnet run
```

The server reads MCP messages from stdin and writes responses to stdout.
All logs go to stderr.

Docker

Build locally

```
docker build -t skillmcp:local .
```

Run — mount your repos drive so the server can read project files

Windows (PowerShell):

```
docker run --rm -i -v "G:\Repos:/g/Repos" skillmcp:local
```

Linux / macOS:

```
docker run --rm -i -v "$HOME/repos:/repos" skillmcp:local
```

Then pass the **container-side** path as `targetProject`:

```
targetProject = /g/Repos/your-project          # Windows mount
targetProject = /repos/your-project            # Linux/macOS mount
```

Mount a local awesome-copilot clone (optional, avoids auto-clone)

```
docker run --rm -i \
  -v "G:\Repos:/g/Repos" \
  -v "G:\Repos\frontlook-admin\awesome-copilot:/opt/awesome-copilot:ro" \
  skillmcp:local
```

Pass `skillSourcePath = /opt/awesome-copilot` as a tool argument.

Override the clone URL via environment variable

```
docker run --rm -i \
  -e AWESOME_COPILOT_REPO_URL=https://github.com/your-fork/awesome-copilot \
  skillmcp:local
```

Pull from GitHub Container Registry

```
docker pull ghcr.io/frontlook-admin/skillmcp:latest
```

VS Code MCP Configuration

The `.vscode/mcp.json` in this repo registers the server automatically.
Open the workspace in VS Code and GitHub Copilot will discover and list the tools.

Example configuration:

```
```

```
{
  "servers": {
    "setup-project-skills": {
      "type": "stdio",
      "command": "docker",
      "args": [
        "run", "--rm", "-i",
        "-v", "G:\\Repos:/g/Repos",
        "ghcr.io/frontlook-admin/skillmcp:latest"
      ]
    }
  }
}
```

> **Windows path translation** — the server automatically converts `G:\Repos\foo` to `/g/Repos/foo` to match the volume mount above.
> Pass either form as `targetProject`; both are handled correctly.

GitHub Actions — Automatic Docker publish

The workflow at `.github/workflows/docker-publish.yml` builds and pushes to **GitHub Container Registry (GHCR)** on every push.
No secrets need to be configured — it uses the built-in `GITHUB_TOKEN`.

Trigger	Published tags
Push to `main`	`latest`, `main`, `sha-<short>`
Tag `v1.2.3`	`1.2.3`, `1.2`, `sha-<short>`, `latest`
Pull request	Build only, not pushed

Output: What Gets Written

After `setup_project_skills` or `refresh_project_skills` runs successfully:

```
<project>/
  skills/
    git-commit/
    conventional-commit/
    csharp-async/            ← type-specific skills
    ...
    skills.json              ← manifest: detected type + list of installed
skills
  .vscode/
    settings.json            ← chat.promptFilesLocations entry added/updated
```