Explicit Context-Driven Development (ECDD) Prompts

This repository contains canonical prompt templates and simple text-based workflows for **Explicit Context-Driven Development (ECDD)** – a methodology for AI-augmented software development that treats prompts and contextual information as first-class, version-controlled artifacts.

The prompts and templates in this repo are **AI-IDE agnostic** and can be used with:

- **GitHub Copilot** (VS Code, Visual Studio, JetBrains)
- **Cursor IDE**
- **Windsurf**
- **Any AI-powered IDE or chat interface**

  ECDD is a "**copilot, not autopilot**" approach: large language models help with planning, coding, and review, but humans stay firmly in the loop and all important context lives in the repository.

---

1. What is ECDD?

ECDD is built on three pillars:

1. **Prompt-Driven AI Governance** Project-wide goals, architecture, and coding standards are captured in persistent prompt files and checked into version control.
2. **Auditable Task & State Management** Work packages and their outcomes are tracked as plain-text artifacts in the `artifacts/` folder instead of ephemeral chat messages.
3. **Five-Phase Workflow** Each project follows a structured workflow:
   - **Define** the project scope and architecture.
   - **Plan** the high-level roadmap.
   - **Elaborate** specific work packages into detailed specs.
   - **Scope** the AI context with project-specific instructions.
   - **Implement** the code changes systematically.

---

2. Repository Structure

```
.
  .github/
     prompts/                    # Executable prompts for AI agents
         define.prompt.md        # 1. Define project (creates project_definition.md)
```

```
        plan.prompt.md           # 2. Plan roadmap (creates workpackage_list.md)
        elaborate.prompt.md      # 3. Elaborate WP (creates workpackage_WP-XXX.md)
        scope.prompt.md          # 4. Scope AI context (creates copilot-instructions.md)
        implement.prompt.md      # 5. Implement WP (creates todos & code)
    copilot-instructions.md      # Auto-loaded by GitHub Copilot in VS Code

templates/                       # Markdown templates used by prompts
    template_project_definition.md
    template_simple_workpackage.md     # For roadmap (Phase 2)
    template_complete_workpackage.md   # For detailed specs (Phase 3)

artifacts/                       # Generated project documents (Git-tracked)
    project_definition.md        # Project overview, tech stack, scope
    workpackage_list.md          # High-level roadmap of all WPs
    workpackage_WP-001.md        # Detailed spec for WP-001
    workpackage_WP-002.md        # Detailed spec for WP-002
    todos_WP-001.md              # Granular implementation checklist
    log.md                       # Append-only implementation log

README.md
```

Key Folders

- **`.github/prompts/`**: Contains the five core prompts that drive the ECDD workflow. These are the "entry points" you'll invoke with your AI assistant.
- **`templates/`**: Markdown templates that define the structure for project artifacts. Prompts read these templates to ensure consistent formatting.
- **`artifacts/`**: Generated files that capture project context, plans, and logs. **You can edit these files manually** at any time to refine or correct the AI's output.

---

3. Getting Started

Step 1: Clone This Repository

Clone ECDD into your project folder:

```
git clone https://github.com/kmkarakaya/ECDD.git .ecdd
cd .ecdd
```

Or, if you want to integrate ECDD into an existing project:

```
cd your-project-folder
git clone https://github.com/kmkarakaya/ECDD.git .ecdd
```

Step 2: Choose Your AI IDE

ECDD works with any AI-powered development environment:

- **GitHub Copilot (VS Code)**: The `.github/copilot-instructions.md` file (generated in Phase 4) is automatically included in the chat context.
- **Cursor / Windsurf / Other**: Manually reference the prompts and artifacts when interacting with the AI.

Step 3: Run the Five-Phase Workflow

Follow the workflow below using your AI assistant. After each phase, **review and edit** the generated artifacts in `artifacts/` before proceeding.

---

4. The Five-Phase ECDD Workflow

Phase 1: Define the Project

Goal: Establish the project's scope, goals, and technical foundation.

VS Code (GitHub Copilot):

```
/define [your project description]
```

Copy-pastable example (VS Code):

```
/define a web app named "AI Concepts Dictionary" - the user enters an AI concept or keyword
```

What to include in the project definition (the prompt will ask for these):

- **Name & purpose:** short name and 1–2 sentence mission
- **Core features:** search input, definition view, examples, resource links, small settings page
- **Tech stack:** frontend (React + Vite), styling (Tailwind), backend (Node + Express) as an API proxy
- **Environment variables:** `GEMINI_API_KEY` (for Gemini API), `PORT` for local server
- **LLM & config:** Gemini 2.5 Flash Lite, temperature 0.2–0.6, short max tokens for concise responses
- **Acceptance criteria:** returns structured JSON with `definition`, `examples`, `resources`; UI shows items clearly; runs locally with `npm run dev`

Other IDEs: Open `.github/prompts/define.prompt.md` in your AI chat and paste the example or fill the requested fields when prompted.

What happens:

1. The AI asks you clarifying questions about your project (one at a time).

2. It fills out the `template_project_definition.md` template.
3. It creates `artifacts/project_definition.md`.

Action: Review and edit `artifacts/project_definition.md` to ensure it accurately reflects your vision.

Phase 2: Plan the Roadmap

Goal: Break the project into manageable Work Packages (WPs).

VS Code (GitHub Copilot):

`/plan`

Other IDEs: Open `.github/prompts/plan.prompt.md` in your AI chat.

What happens:

1. The AI reads `artifacts/project_definition.md`.
2. It decomposes the project into a list of Work Packages.
3. It creates `artifacts/workpackage_list.md` using the `template_simple_workpackage.md` format.

Action: Review the roadmap. Adjust priorities, dependencies, or descriptions as needed.

Phase 3: Elaborate a Work Package

Goal: Expand one Work Package into a detailed, implementable specification.

VS Code (GitHub Copilot):

`/elaborate WP-001`

Other IDEs: Open `.github/prompts/elaborate.prompt.md` and specify the WP ID (e.g., `WP-001`).

What happens:

1. The AI reads the roadmap and selects the specified WP.
2. It creates a detailed spec with acceptance criteria, technical specs, implementation steps, and testing requirements.
3. It creates `artifacts/workpackage_WP-001.md` using the `template_complete_workpackage.md` format.

Action: Review the spec. Adjust implementation steps, tech choices, or acceptance criteria.

Note: Repeat this phase for each WP you want to implement (e.g., `WP-002`, `WP-003`, etc.).

Phase 4: Scope the AI Context

Goal: Generate a `copilot-instructions.md` file so GitHub Copilot knows your project's coding standards and architecture.

VS Code (GitHub Copilot):

```
/scope
```

Other IDEs: Open `.github/prompts/scope.prompt.md`.

What happens:

1. The AI reads all artifacts (`project_definition.md`, `workpackage_list.md`, detailed WPs).
2. It generates `.github/copilot-instructions.md` with sections for:
   - Project overview
   - Tech stack
   - Coding guidelines
   - Project structure
   - Available resources
3. **In VS Code with GitHub Copilot**, this file is automatically included in the chat context.

Action: Review the instructions file. Add any project-specific rules or conventions.

Phase 5: Implement Code

Goal: Execute the implementation for a specific Work Package.

VS Code (GitHub Copilot):

```
/implement WP-001
```

Other IDEs: Open `.github/prompts/implement.prompt.md` and specify the WP ID.

What happens:

1. The AI reads the detailed spec (`workpackage_WP-001.md`).
2. It checks if `artifacts/todos_WP-001.md` exists:
   - If **yes**, it resumes from the first unchecked item.
   - If **no**, it creates a new granular todo list.
3. It iterates through the todos, writing code and verifying each step.
4. It updates `artifacts/log.md` with a detailed summary of changes.

Action:

- Review the generated code.
- Run tests to verify the implementation.
- If needed, re-run `/implement WP-001` to resume work.

---

5. Human in the Loop

ECDD is designed for **continuous human oversight**:

- **Review artifacts between phases**: After each prompt execution, check the generated files in `artifacts/` and edit them if needed.
- **Approve plans before coding**: Review the todo list (`todos_WP-XXX.md`) before letting the AI write code.
- **Verify implementations**: Always run tests and manually check the application after implementation.
- **Iterate as needed**: You can re-run any phase or manually edit any artifact at any time.

This repository is not an autonomous agent system; it is a **structured collaboration pattern** between humans and AI tools.

---

6. Tips for Success

1. **Don't skip the Define phase**: A clear `project_definition.md` is the foundation for everything.
2. **Iterate on the roadmap**: The `workpackage_list.md` doesn't have to be perfect on the first try.
3. **Elaborate one WP at a time**: Avoid overwhelming the AI (or yourself) by trying to detail everything at once.
4. **Use the log**: The `artifacts/log.md` file is crucial for tracking what's been built and avoiding conflicts.
5. **Customize templates**: The `templates/` folder is yours to modify. Tailor the structure to your project's needs.

---

7. License