BUSINESS OVERVIEW · NON-TECHNICAL SUMMARY

AI Developer Tools — Ten Workflows

This document describes ten AI-powered tools available to the development team. Each one automates a specific part of the software development process — from writing new code to checking quality, securing the application, and keeping the codebase up to date.

All tools are activated by typing a short command in the AI chat during normal working hours. No special setup is required. The tools work with any Angular project by copying a single folder.

February 2026

GROUP 1 · BUILDING NEW THINGS

TOOL 1 — FEATURE PLANNER

Activated by typing: ***/angular-feature-plan***

Before writing any code for a new feature, this tool designs the complete architecture — what screens are needed, what services will handle the data, how information flows between components, and how the feature connects to the rest of the application. The AI produces a written plan for the team to review and approve before development begins.

What the developer does

The developer describes the feature in plain language: what it should do, where it appears in the application, and any known constraints. The AI handles the rest.

> Example: “Design a user profile page where users can view and edit their name, email, and avatar. New route at /profile.”

What the AI produces

- **A component diagram** — which screens exist and how they nest inside each other
- **A service list** — what background services are needed and what they do
- **Data structures** — the shape of information the feature stores and passes around
- **A routing plan** — what URLs the feature occupies and what guards protect them
- **A written plan document** — saved to the project for team review

> **Important:** The AI does not write any code until the team explicitly approves the plan. This acts as a design review gate before any implementation work begins.

Business value

Without the tool	With the tool
Developers design as they go, sometimes discovering structural problems mid-implementation	Architecture is agreed upfront; structural problems are found in a document, not in code
Rework is expensive if the wrong approach is taken and discovered late	Course corrections happen at the planning stage, when the cost is near zero

---

TOOL 2 — CODE GENERATOR

*Activated by typing: **/angular-generate***

When a new building block is needed — a new screen, a data service, a security check, or a reusable component — this tool generates the code files automatically, already matching the team's conventions. No manual pattern-matching or boilerplate writing required.

What the developer does

The developer provides three pieces of information: the type of building block needed, a name for it, and one sentence describing its purpose.

> Example: "Create a service called *notification* that shows success and error messages across the application."

What the AI does

1. **Reads the existing codebase first.** Before writing a single line, the AI examines two or three similar existing files to understand exactly how the team structures and styles its code.
2. **Generates the files.** The AI produces ready-to-use code files that already follow the team's conventions — not a generic template that needs manual adjustment.
3. **Validates the output.** The AI automatically checks that the generated code is correct before presenting it.
4. **Explains next steps** — such as "connect this service to the following component" or "register this route."

Business value

Without the tool	With the tool
Developer spends 15–45 minutes creating boilerplate, then adjusting it to match team patterns	Developer describes what is needed in one sentence; convention-compliant code is ready in seconds
New team members may not know all conventions, producing inconsistent code	Convention compliance is automatic — every developer produces the same quality output

TOOL 3 — API CONNECTOR

*Activated by typing: **/angular-api-integration***

When the application needs to connect to a new back-end API or external service, this tool generates the entire integration layer automatically: the data models, the service that makes the network calls, the authentication headers, and the environment configuration. All of it is fully typed and matches the project's existing patterns.

What the developer does

The developer provides the API specification (a standard technical document describing available endpoints), or simply describes the endpoints in plain language. The AI generates everything needed to connect the application to that API.

What the AI produces

- **Data models** — TypeScript definitions of every request and response object
- **A typed service** — one method per API endpoint, with correct return types
- **Environment configuration** — base URLs wired to the correct config files
- **Authentication headers** — if required, an interceptor that attaches auth tokens to every request

> **Security note:** The AI is instructed never to hardcode API keys, passwords, or real server URLs into source code. These are always placed in environment configuration files that are excluded from version control.

GROUP 2 · QUALITY & REVIEW

TOOL 4 — AUTOMATED CODE REVIEW

*Activated by typing: **/ai-code-review***

Before a developer submits work for a colleague to review, this tool checks it first. The AI acts as a senior reviewer, applying a consistent checklist of quality and security standards, then producing a written report with a clear verdict.

What the developer does

The developer specifies what to review: their latest changes, a specific file, or the difference between two versions. That is the entire input required.

What the AI checks

Category	What is checked
Correctness	Are there coding errors that would cause the application to fail to build or behave incorrectly?
Code quality	Is the code following defined team standards? Are there memory leaks, leftover debug logs, or inconsistent patterns?
Modern patterns	Is the code using current Angular conventions, or older patterns that should be updated?
Security	Are there API keys accidentally left in source code? Are there patterns that could expose data to unauthorised users?
Performance	Are there patterns that could cause the application to update unnecessarily or slow down with larger data sets?

What the AI produces

A written report saved automatically to the project, containing:

- **Must Fix** — issues that must be resolved before the work is deployed
- **Should Fix** — quality issues that are not blocking but should be addressed soon
- **Suggestions** — optional improvements the developer may choose to act on
- **Positive notes** — things done well, to reinforce good practices
- **Overall verdict** — Approved / Approved with comments / Changes requested

> **Note on scope:** The AI only flags issues in the code that was just changed. Pre-existing problems elsewhere in the codebase are not raised. This keeps the report focused and actionable.

Business value

Without the tool	With the tool
Junior developers may submit work with avoidable issues, consuming senior reviewer time	Common issues are caught automatically before human review, freeing senior reviewers for higher-order decisions
Security checks depend on individual reviewer knowledge on any given day	A consistent security checklist is applied to every change, every time, without exception
Code quality standards drift as team composition changes	Standards are enforced at the point of submission, not retroactively after problems appear

---

TOOL 5 — TEST GENERATOR

*Activated by typing: **/angular-test-generate***

Writing automated tests is time-consuming but essential for catching regressions when the codebase changes. This tool reads an existing piece of code and generates a comprehensive test file for it automatically, then runs the tests to verify they pass before presenting them.

What the developer does

The developer points the tool at a file (or folder) and optionally specifies whether they want basic happy-path tests or a full suite including error conditions and edge cases.

What the AI does

1. **Reads the code under test** to understand every public method, every input, and every possible output.
2. **Generates a test file** covering: normal operation, error handling, boundary conditions, and user interaction.
3. **Runs the tests automatically.** If any tests fail, the AI fixes them before presenting the result. No failing tests are left for the developer to resolve.

Business value

Without the tool	With the tool
Writing tests for existing code is low-priority work that gets deferred indefinitely	A full test suite for any file can be generated and verified in minutes
Untested code silently breaks when the application is updated	Automated tests catch regressions immediately, before they reach production

---

TOOL 6 — SECURITY AUDITOR

*Activated by typing: **/angular-security-audit***

This tool performs a dedicated security scan of the application, checking for vulnerabilities that standard code review may not consistently catch. It examines the entire codebase (or a specific area) and produces a prioritised report of findings with concrete remediation steps.

What the AI checks

Risk area	What is checked
Cross-site scripting (XSS)	Is any user-supplied content rendered as raw HTML without sanitisation? This could allow attackers to inject malicious scripts.
Hardcoded secrets	Are any API keys, passwords, or authentication tokens present in source code? These must never be committed to a repository.
Access control	Are all routes that require authentication properly protected? Can a user reach a restricted page by manipulating the URL?
Insecure data storage	Are sensitive tokens stored in browser localStorage, where they can be accessed by any script on the page?
Dependency vulnerabilities	Are any third-party packages in use known to contain security vulnerabilities?

What the AI produces

A `SECURITY_REPORT.md` file with findings rated Critical / High / Medium / Low, each with a plain-English description of the risk and specific steps to remediate it.

TOOL 7 — DOCUMENTATION GENERATOR

*Activated by typing: **/angular-docs-generate***

This tool reads existing code and generates accurate, up-to-date documentation from it — inline code comments, component API reference tables, and feature README sections. Documentation is derived from the actual implementation, so it cannot become out of sync with the code the way manually written docs often do.

What the AI produces

- **Inline comments** — added directly to the source file, describing each public property and method
- **Component API table** — a Markdown reference listing all inputs, outputs, and public methods with types and descriptions
- **Feature README** — a plain-English summary of what a feature does, what screens it contains, and how to use it
- **Storybook stories** — interactive component previews (if Storybook is configured in the project)

GROUP 3 · MAINTENANCE & HEALTH

TOOL 8 — CODE MODERNISER

*Activated by typing: **/angular-refactor***

Angular evolves over time, and older coding patterns become outdated. This tool scans existing code for patterns that should be updated to the current standard and migrates them automatically — without changing what the application actually does from the user's perspective.

What the tool modernises

- Old-style module declarations → modern standalone components
- Verbose observable subscription patterns → simpler reactive signals
- Legacy template syntax (`*ngIf`, `*ngFor`) → current control-flow syntax
- Class-based security guards → lightweight functional guards
- Deprecated utility methods → their modern replacements

> **Safety first:** Before making any changes, the AI scans the codebase and presents a full list of what it intends to modify for developer approval. After each change, it runs automated checks to confirm nothing was broken.

Business value

Without the tool	With the tool
Modernisation backlog grows silently; outdated patterns accumulate over years	Entire codebases can be modernised in a single session, with full verification
Manual refactoring is time-consuming and carries a risk of introducing bugs	Each change is verified by automated type-checking before the next change is made

---

TOOL 9 — VERSION UPGRADER

*Activated by typing: **/angular-migration***

Upgrading Angular to a new major version is a complex process involving package updates, code changes, configuration adjustments, and thorough testing. This tool automates the entire process end-to-end, with a mandatory human approval gate before any changes are made.

How the upgrade process works

Step	Who	What happens
1. Plan	AI	Audits the codebase, researches breaking changes in the target version, and writes a detailed migration plan
Gate	**You**	**Review and approve the plan before anything is changed**
2. Backup	AI	Creates a git snapshot so the upgrade can be fully reversed if needed
3. Upgrade	AI	Runs upgrade commands; automatically resolves common errors; escalates to developer if a problem cannot be resolved automatically
4. Verify	AI	Runs type checks, development build, production build, linting, and a live server check
5. Recommend	AI	Produces an optional list of follow-up improvements (e.g. “run the Code Moderniser next”)

> **Error escalation policy:** If fewer than 10 errors remain after the upgrade, the AI fixes them automatically. If 10–50 errors remain, it fixes recognisable patterns and reports the rest. If more than 50 errors remain, it stops and escalates — this typically indicates a third-party library incompatibility requiring human judgement.

TOOL 10 — DEBUGGER

*Activated by typing: **/angular-debug***

When something goes wrong — an error in the browser, a failing build, a test that will not pass, or a page that displays incorrectly — this tool systematically diagnoses the root cause and applies a fix. It does not guess; it reads the relevant code, applies a structured diagnostic checklist, and identifies the exact cause before changing anything.

What the developer does

The developer pastes the error message (or describes the unexpected behaviour) and where it appears. The AI handles the investigation.

What the AI does

1. **Classifies the error** into one of eight categories (dependency, template, routing, build, test, etc.) to narrow down the investigation.
2. **Reads the relevant files** identified in the error trace.
3. **Applies a diagnostic checklist** of the most common causes for that error type.
4. **States the root cause** in plain English, applies the fix, and explains why the fix resolves the problem.
5. **Verifies the fix** by re-running the failing command or test to confirm the problem is resolved.

SUMMARY — ALL TEN TOOLS

Tool	Command	What it does	Who benefits
Feature Planner	/angular-feature-plan	Designs feature architecture and produces an approval-gated plan before any code is written	Tech leads, developers, project managers
Code Generator	/angular-generate	Reads existing conventions and produces new code files that already match the team's patterns	All developers, especially those new to the project
API Connector	/angular-api-integration	Generates a complete, typed integration layer from an API specification	Developers integrating with back-end services
Code Review	/ai-code-review	Checks code changes for correctness, quality, and security; produces a written verdict	All developers; frees up senior reviewer capacity
Test Generator	/angular-test-generate	Generates and verifies a full automated test suite for any existing file	All developers; QA teams
Security Auditor	/angular-security-audit	Scans for XSS, hardcoded secrets, broken access control, and vulnerable dependencies	Security teams; developers before a release
Documentation Generator	/angular-docs-generate	Generates accurate inline comments, API tables, and feature README files from existing code	All developers; onboarding new team members
Code Moderniser	/angular-refactor	Migrates outdated patterns to current Angular standards without changing application behaviour	All developers; reduces technical debt
Version Upgrader	/angular-migration	Runs end-to-end Angular version upgrades with human approval gate, auto-fix, and full verification	Tech leads; reduces upgrade risk and effort
Debugger	/angular-debug	Diagnoses errors, build failures, and failing tests; identifies root cause and applies a verified fix	All developers; reduces time lost to debugging