Report for judges

TL;DR:
*Below is the report for “AI in Software Development” prediction tournament. Five features are already in the wild — project-wide context awareness, Sketch2Code, multi-file generation, AI-code annotation, and live refactoring, we suggest marking them **YES**. Daily ecosystem summaries and business-context awareness are plausible but unproven (**Probably**). Truly proactive automation, documentation, optimization, debugging, and BCI-based coding remain absent (**NO**). Among these features, JetBrains AI tools currently offer project-wide context awareness and multi-file generation.*

The present research report addresses the questions posed in the JetPredict AI in Software Development tournament. The tournament's rules stipulated that participants would be provided with a set of features and their descriptions, along with questions concerning the availability of these features in AI for coding tools by July 2025. We are currently in the process of resolving these questions. The questions were framed as descriptions of potential AI features, derived from the HAX design space study, with the following resolution criteria: "The question will be resolved as YES if the feature is available in a publicly accessible coding tool before July 1, 2025," or "The question will be resolved as YES if the feature is available in a publicly accessible JetBrains IDE before July 1, 2025."

It is improbable that features appearing in the market will be named or described precisely as we have formulated them. Consequently, we require expert judgment to assess the existence of these features based on their functional essence rather than their literal description. We request that you review the following report and, for each feature, provide a comment indicating the appropriate resolution (YES or NO) based on your assessment.

Any coding tools AI solutions

Suggested YES

The initial set of questions presented are those we believe warrant a "YES" resolution, indicating that the features described therein have appeared in at least some tools currently available in the market. These features and their descriptions are as follows:

Project-wide context awareness	Project-wide context awareness for an AI assistant in coding ensures that the AI understands the entire scope, structure, and history of a project, leading to more accurate and relevant code suggestions, bug fixes, and optimizations. This comprehensive understanding enhances productivity by providing context-sensitive assistance, improving code quality, reducing errors, and facilitating seamless collaboration among team members.
Sketch2Code	Sketch2Code for an AI coding assistant involves the AI autonomously converting sketch design files into production-ready code. By analyzing the visual elements, layouts, and interactions specified in the Figma design, the AI generates corresponding HTML, CSS, and JavaScript, or other relevant code.

Multi-file generation	Multi-file generation for an AI coding assistant involves the AI autonomously creating and managing multiple interconnected files within a project. By understanding the project structure and dependencies, the AI generates all necessary files such as configuration files, modules, components, and documentation. This capability ensures coherence, reduces manual setup effort, and accelerates the development process, allowing developers to focus on higher-level coding tasks and functionality.
Annotating AI-code	Annotating AI-generated code involves the AI automatically marking or tagging the sections of code it has produced. By clearly indicating which parts of the codebase were generated by the AI, this feature ensures transparency and helps developers quickly identify and review AI contributions. This capability enhances code accountability, makes debugging easier, and fosters better collaboration by clearly distinguishing between human-written and AI-generated code.
AI-Driven Real-Time Code Refactoring	AI-Driven Real-Time Code Refactoring utilizes AI to anticipate code refactoring opportunities as developers write and even as they pause to think about potential solutions. By continuously analyzing the code structure and developer's patterns, the AI dynamically suggests refactoring strategies such as modularization, method extraction, and code optimization

- **Annotation of AI-generated code** has emerged as a possibility. For example, the ChatGPT Code Detector or CodeSpy projects to analyze and identify code produced by AI models such as ChatGPT, offering insights based on indicators like coding style, structure, and syntax. CopyLeaks further expands on this, providing content authenticity assessment for both code and various forms of text. However, the reliability of such annotation remains contentious. While vendors often claim high detection accuracies (95-99%), these figures are typically based on small, proprietary datasets used for marketing. Independent, peer-reviewed benchmarks consistently demonstrate significant drops in accuracy under more realistic conditions, including diverse domains, newer large language models, minor paraphrasing, and non-native English writing. Furthermore, this feature does not appear to be of significant interest within the developer community.

- Currently, most market tools, such as for example Windsurf, Cursor or GitHub Copilot, incorporate **project-wide context awareness.** Similarly, the management of multiple files has been integrated into various existing tools, with GitHub Copilot providing multi-file editing capabilities for extensive changes across numerous files.

- The capability of **Sketch2Code** has also been introduced, notably by Figma through a community plugin. Coding tools like Tabnine also offer such features, enabling the generation of actionable code from UI mockups, flowcharts, database diagrams, or annotated screenshots via their AI Chat.

- Real-time code refactoring is also asserted as possible. Sourcery, for example, offers a code reviewer tool that reportedly refactors code, identifies bugs, and enhances quality in real time. Nevertheless, the objective assessment of the dynamism of its suggestions remains challenging.

Probably YES

The second group consists of two questions concerning features for which the optimal resolution is more difficult to determine. These features are listed below.

in-IDE daily ecosystem summarizer	A Daily Ecosystem Summarizer is a tool or function designed to provide concise, actionable summaries of key activities, metrics, and significant events within a given ecosystem on a daily basis. The summarizer aims to aggregate and synthesize relevant information to help users stay informed and make informed decisions without needing to sift through large volumes of data.
Business-context awareness	Business-context awareness for an AI coding assistant allows the AI to tailor its suggestions and solutions based on the specific business goals, industry regulations, and organizational workflows. This targeted assistance ensures that the generated code aligns with the company's strategic objectives, compliance requirements, and operational practices, thereby enhancing relevance, accelerating development processes, and supporting better business outcomes.

We were unable to locate tools that directly offer **Business-context awareness** or **in-IDE daily ecosystem summarizer** functionalities, particularly concerning coding. The closest equivalent we identified is likely Tabnine, which features direct integration with the Jira system. However, while the daily ecosystem and business context are not explicitly displayed, these features could potentially be developed by users through their own pipelines.

Projects of this nature appear entirely feasible given the current technological landscape. An illustrative example is Gentlemanager, an internal initiative stemming from JetBrains Hackathon and Innovation Jams. This project aimed to develop a proactive, AI-powered assistant seamlessly integrated across a user's daily applications and platforms, including Notes, Slack, YouTrack, Google Docs, Meetings, and web browsers. The assistant's functionality relies on analyzing user activity to discern patterns and personal preferences, subsequently offering proactive suggestions and maintaining contextual awareness. Through intelligent reminders, it facilitates adherence to personal tasks and external commitments. The objective of this application is to augment productivity, alleviate the cognitive burden associated with context-switching and task tracking, and ensure comprehensive task management. Preliminary solutions were showcased during JetBrains' internal hackathon.

Suggested NO

The final significant category of inquiries within our prediction market pertained to potential proactive features for AI tools in software development. These tools encompassed proactive documentation, proactive automation, and proactive optimization. The descriptions of these features were derived from developer interviews conducted during the preparation of the design space study. The definitions are presented below:

Brain-computer interface integrated coding	Coding with BCI for an AI coding assistant involves the AI assisting in code generation and development through direct brain-computer interface inputs provided by any BCI technology. By translating neural signals into actionable coding commands and suggestions, the AI enables developers to interact with their coding environment more intuitively and efficiently. It involves both direct control of IDE with BCI-derived commands and brain-based adjustments of the workflow.

Proactive automation of repetitive tasks	Proactive automation of repetitive tasks for a coding assistant involves the AI autonomously identifying and executing routine coding activities that do not require unique human insight. This includes tasks such as formatting code, generating boilerplate code, running routine tests, and managing code documentation.
Proactive documentation	Proactive documentation for an AI coding assistant involves the AI automatically generating and suggesting relevant documentation as developers write or modify code. By understanding the code's context, purpose, and structure, the AI provides timely documentation, such as comments, function descriptions, and usage examples. This ensures that the codebase remains well-documented without extra manual effort, improving code readability, maintainability, and easing collaboration among team members.
Proactive optimization	Proactive optimization for an AI coding assistant involves the AI automatically identifying potential improvements in the code for performance, efficiency, and best practices. By continuously analyzing the codebase, the AI offers real-time suggestions for optimizing algorithms, reducing resource consumption, and enhancing code quality. This proactive approach helps developers maintain high performance and efficient code, reducing technical debt and improving overall application performance without waiting for manual reviews or refactoring sessions.
Proactive debugging	Proactive debugging for an AI coding assistant involves the AI automatically detecting and diagnosing potential bugs and issues in real-time as developers write code. By continuously analyzing the code for common errors, logical inconsistencies, and potential runtime issues, the AI provides immediate feedback and suggests fixes.

Regarding **Brain-computer interface (BCI) integrated coding**, no real-world solutions currently exist, nor are they anticipated in the near future. Further details concerning the state of the BCI industry are discussed in this recent JetTalk.

Proactivity in coding can be described as taking initiative and anticipating potential issues or needs in the development process rather than simply reacting to problems as they arise. It involves planning, preparing, and acting to prevent future issues and improve the overall quality and efficiency of the code and development workflow. According to our preliminary market research there are currently no AI tools in the market that offers truly proactive solutions. However, there are several tools that have at least some features that can be viewed as proactive.

- For the **proactive documentation** feature such a tool can be Swimm coding AI assistant. Swimm's proactivity is triggered by code changes detected within the CI pipeline. Its patented "Auto-sync" algorithm continuously checks if the code snippets embedded in the documentation still match the source code in the repository. If a discrepancy is found, Swimm flags the documentation as outdated and can even gate the CI process, forcing an update. Another project for AI-backed documentation is Mintify. It analyzes existing documentation and proactively suggests improvements for clarity and completeness (e.g., with CI checks feature)

- For the **proactive automation** Renovate Bot is the a noticeable project, working on operations-level infrastructure monitoring. Renovate's proactivity is schedule-driven. The hosted application scans repositories approximately every three hours, while self-hosted instances can be configured with a custom schedule. When the bot runs, it checks for available updates that match the configured policies and initiates action. One more example: the Codefresh CI/CD

platform proactively automates the build and testing process. It is automatically triggered by code commits to run a pipeline that enforces quality and security checks, preventing faulty code from being merged.

- There are several projects potentially related to **proactive debugging**. GitHub Copilot Autofix: can detect a vulnerability with a security scan (CodeQL) in a pull request. This feature automatically generates a potential fix and presents it as a suggestion within the pull request. CodeRabbit & CodeAnt AI automatically review new pull requests. They provide line-by-line feedback, identify bugs and security issues, and suggest one-click fixes directly in the PR comments. Workik AI offers automated AI Debugging pipelines that proactively scan code commits to find and suggest fixes for bugs. Snyk (formerly DeepCode) can scan code in real-time as it's being written in the IDE or upon commit. It proactively provides immediate feedback and AI-powered quick fixes for security vulnerabilities. Seer from Sentry offers Automate Issue Scans and Automate Issue Fixes features. "Issue Scan" can be configured to automatically invoke Seer agent on newly ingested issues to assess their fixability, making the process fully autonomous from detection to solution proposal.

- Concerning **proactive optimization** a project to be mentioned is again Workik AI that analyzes code to identify performance bottlenecks, suggests ways to optimize memory and CPU usage, and provides recommendations for improving database query speed. It can also suggest compiler settings to produce more efficient code. Another project is already mentioned Sourcery, the tool that claims to specialize in AI-powered real-time code refactoring. Codiga offers real-time and automate refactoring and vulnerability fixes. Moderne focuses on automation of large-scale code refactoring. Its core capability can be potentially applied to proactively optimize an organization's tech stack and architecture across thousands of repositories.

JetBrains AI solutions

A distinct set of inquiries pertained exclusively to JetBrains AI Products. Our current analysis indicates that only two of the features discussed—project-wide context awareness and multifile generations—have been incorporated directly into JetBrains AI products. At the same time, it is important to acknowledge that automation, optimization, and debugging have been central to JetBrains' offerings even before the recent advancements in AI tools for developers. There are also intensive research for proactive features for currently developed JetBrains AI tools, such as for example building Proactive and Smart AI Chat prototype based on this architecture.

The table detailing all features and their suggested resolutions for both JetBrains products and “any coding tools” is below

FEATURE	**Any coding tool** Suggested:	**JetBrains IDEs** Suggested:
Proactive automation of repetitive tasks	No	No
Proactive documentation	No	No
Proactive optimization	No	No

Brain-computer interface coding	No	No
Proactive debugging	No	No
in-IDE daily ecosystem summarizer	Probably	No
AI-driven real-time code refactoring	Probably	No
Business-context awareness	Probably	No
Project-wide context awareness	Yes	Yes
Sketch2Code	Yes	No
Multi-file generation	Yes	Yes
Annotating AI-code	Yes	No