Tytuł szkolenia: AI-Driven Development: Wykorzystanie narzędzi AI w pracy programisty i testera

Kod szkolenia: AI-Dev-Practice

Wprowadzenie

Szkolenie ma charakter warsztatowy – 70% czasu to praktyczne ćwiczenia (live coding) w środowiskach programistycznych (IDE), a 30% to demonstracja narzędzi i omówienie dobrych praktyk. Uczestnicy pracują na własnym kodzie lub na przygotowanych repozytoriach, ucząc się integrować asystentów AI z codziennym cyklem wytwarzania oprogramowania (SDLC).

Adresaci szkolenia

Szkolenie przeznaczone jest dla personelu technicznego: programistów (Backend, Frontend, Fullstack) chcących przyspieszyć pisanie kodu, testerów (QA) szukających sposobów na automatyzację scenariuszy oraz architektów systemowych zainteresowanych wsparciem AI w projektowaniu i pracy z kodem legacy.

Wymagania:

Od uczestników wymagana jest dobra znajomość przynajmniej jednego języka programowania (np. Java, C#, C++, JavaScript/TypeScript), znajomość obsługi wybranego IDE (IntelliJ IDEA, Visual Studio Code) oraz podstawowa znajomość systemu kontroli wersji Git.

Cel szkolenia

Celem szkolenia jest nabycie praktycznej umiejętności obsługi asystentów kodowania (AI Coding Assistants) w celu zwiększenia efektywności pracy, poprawy jakości kodu oraz automatyzacji powtarzalnych zadań programistycznych, przy zachowaniu standardów bezpieczeństwa kodu.

Czas i forma szkolenia

- 7 godzin (1 dni x 7 godzin), w tym wykłady i warsztaty praktyczne.

Plan szkolenia

1. Wprowadzenie do AI w inżynierii oprogramowania
- Modele LLM w kontekście kodu – jak działają tokeny i okno kontekstowe?
- Przegląd ekosystemu narzędzi: Copilot, Chat vs Autocomplete
- Konfiguracja środowiska – instalacja wtyczek do VS Code / IntelliJ

2. GitHub Copilot i alternatywy w praktyce (Coding Assistants & LLMs)
- Ghost Text: Efektywne wykorzystanie autouzupełniania kodu
- Generowanie kodu z komentarzy: Zamiana języka naturalnego na logikę biznesową
- Interaktywny Chat w IDE: Praca z panelem bocznym (wyjaśnianie błędów, pytania o składnię)
- Cursor IDE: Przegląd edytora natywnie wspieranego przez AI
- Claude 3.5 Sonnet vs Google Gemini 1.5 Pro w pracy z kodem
- Porównanie: GitHub Copilot vs Tabnine vs Amazon CodeWhisperer

3. Refaktoryzacja i praca z kodem Legacy
- Automatyczne wyjaśnianie niezrozumiałych fragmentów kodu
- Translacja kodu między językami (np. Java -> Kotlin)
- Optymalizacja złożoności cyklomatycznej z pomocą sugestii AI
- Dokumentowanie kodu – automatyczne generowanie Javadoc/DocBlock

4. AI w zapewnieniu jakości (Quality Assurance)
- SonarLint + AI: Analiza statyczna kodu wspierana sztuczną inteligencją
- Generowanie testów jednostkowych (Unit Tests) – tworzenie asercji
- Generowanie przypadków brzegowych (Edge cases)
- Testim.io / Mabl: Wstęp do samonaprawiających się testów E2E

5. Bezpieczeństwo i poufność kodu (Security)
- Ryzyko wycieku własności intelektualnej – jak nie trenować publicznych modeli na kodzie firmowym
- Halucynacje AI – wykrywanie nieistniejących bibliotek i błędnych importów
- Analiza podatności – wykorzystanie AI do wstępnego audytu bezpieczeństwa

6. Lokalne modele AI (Prywatność i Offline)
- Wprowadzenie do Ollama i LM Studio
- Uruchamianie modeli open-source (np. Llama 3, Mistral) lokalnie
- Porównanie modeli chmurowych z lokalnymi pod kątem bezpieczeństwa danych

7. Podsumowanie i sesja Q&A
- Kiedy NIE używać AI? Pułapki "ślepego zaufania"
- Przyszłość roli programisty w erze AI