SYLABUS

DOTYCZY CYKLU KSZTAŁCENIA 2026/2027 - 2029/2030

Rok akademicki 2028/2029

1. PODSTAWOWE INFORMACJE O PRZEDMIOCIE

Nazwa przedmiotu	*programowanie zespołowe*
Kod przedmiotu*	
Nazwa jednostki prowadzącej kierunek	*Instytut Informatyki, Wydział Nauk Ścisłych i Technicznych*
Nazwa jednostki realizującej przedmiot	*Instytut Informatyki, Wydział Nauk Ścisłych i Technicznych*
Kierunek studiów	*sztuczna inteligencja*
Poziom studiów	*studia I stopnia*
Profil	*ogólnoakademicki*
Forma studiów	*stacjonarne*
Rok i semestr/y studiów	*rok III, semestr 6*
Rodzaj przedmiotu	*przedmiot kierunkowy*
Język wykładowy	*polski*
Koordynator	*dr inż. Przemysław Pardel*
Imię i nazwisko osoby prowadzącej / osób prowadzących	*dr inż. Przemysław Pardel*

***** - *opcjonalnie, zgodnie z ustaleniami w Jednostce*

1.1.Formy zajęć dydaktycznych, wymiar godzin i punktów ECTS

Semestr (nr)	Wykł.	Ćw.	Konw.	Lab.	Sem.	ZP	Prakt.	Inne (jakie?)	**Liczba pkt. ECTS**
6	15			30					3

1.2. Sposób realizacji zajęć

zajęcia w formie tradycyjnej

1.3 Forma zaliczenia przedmiotu (z toku)

wykład – zaliczenie bez oceny, laboratoria – zaliczenie z oceną

2.WYMAGANIA WSTĘPNE

Znajomość zagadnień z przedmiotów: programowanie obiektowe, inżynieria oprogramowania, algorytmy i struktury danych, bazy danych, programowanie urządzeń mobilnych

3. CELE, EFEKTY UCZENIA SIĘ, TREŚCI PROGRAMOWE I STOSOWANE METODY DYDAKTYCZNE

3.1 Cele przedmiotu

C1	Poznanie najnowszych narzędzi tworzenia oprogramowania (które mają być używane przez studentów na zajęciach laboratoryjnych)
C2	Nauka organizacji pracy zespołu informatycznego w procesie tworzenia, serwisowania i wdrażania oprogramowania
C3	Wykonanie przez studentów złożonego, praktycznego projektu informatycznego w grupie

3.2 Efekty uczenia się dla przedmiotu

EK (efekt uczenia się)	Treść efektu uczenia się zdefiniowanego dla przedmiotu	Odniesienie do efektów kierunkowych [1]
EK_01	Zna w zaawansowanym stopniu techniki oraz popularne i specjalistyczne narzędzia stosowane przy rozwiązywaniu złożonych zadań inżynierskich z zakresu wytwarzania oprogramowania oraz wybranych zastosowań informatyki	K_W07
EK_02	Zna zasady funkcjonowania oraz narzędzia wspierające realizację interdyscyplinarnych projektów wdrożeniowych i badawczych w obszarach wybranych zastosowań sztucznej inteligencji	K_W08
EK_03	Zna zagadnienia cyklu życia systemów informatycznych i modeli AI, obejmujące przetwarzanie danych w chmurze, wdrażanie i utrzymanie modeli (MLOps) oraz zapewnianie bezpieczeństwa i niezawodności	K_W09
EK_04	Potrafi właściwie zaprojektować oraz zrealizować obiekty informatyczne o komponencie programistycznym, a następnie dokonać weryfikacji i interpretacji rezultatów, oraz sporządzić dokumentację	K_U09
EK_05	Potrafi opracować w języku polskim i angielskim raporty, prezentacje wyników i dokumentację techniczną dla odbiorcy naukowego lub biznesowego, dyskutować posługując się specjalistyczną terminologią	K_U11
EK_06	Potrafi przyjmować różne role w zespole interdyscyplinarnym i kierować pracą projektową, odpowiedzialnie planować i realizować zadania projektowe, dbając o jakość i wiarygodność, przy wykorzystaniu odpowiednich narzędzi wspierających pracę zespołową	K_U12

3.3 Treści programowe

A. Problematyka wykładu

Modele i metodyki wytwarzania oprogramowania

[1] W przypadku ścieżki kształcenia prowadzącej do uzyskania kwalifikacji nauczycielskich uwzględnić również efekty uczenia się ze standardów kształcenia przygotowującego do wykonywania zawodu nauczyciela.

Zarządzanie projektem programistycznym: role w zespole, planowanie sprintów, estymacja, zarządzanie backlogiem i ryzykiem
Współpraca i komunikacja w zespole developerskim: code review, pair programming, narzędzia komunikacji (Slack, Teams, Discord, itp.), kultura feedbacku.
Systemy kontroli wersji i zarządzanie repozytorium: Git, GitHub/GitLab, strategie pracy z gałęziami (Git Flow, trunk-based development).
Jakość oprogramowania i testowanie: testy jednostkowe, integracyjne, automatyzacja testów, code coverage, CI pipelines.
Narzędzia wspomagające rozwój oprogramowania: IDE, systemy do zarządzania zadaniami (Jira, Trello), CI/CD (GitHub Actions, Jenkins), konteneryzacja (Docker).
Wdrażanie, utrzymanie i rozwój oprogramowania: deployment, monitoring, dokumentacja techniczna, DevOps w praktyce, feedback od użytkowników i iteracyjny rozwój.

B. Problematyka laboratoriów

Ustalenie 3–5-osobowych zespołów projektowych, wybór liderów oraz tematów projektów.
Praca z systemami kontroli wersji (Git, GitHub/GitLab) – tworzenie i utrzymanie repozytorium.
Organizacja projektu zespołowego w metodyce Scrum – backlog, sprinty, role, tablica zadań.
Określenie architektury systemu i podziału ról oraz zadań w zespole.
Tworzenie dokumentacji projektowej (README, wiki, backlog techniczny).
Konfiguracja środowiska programistycznego i współdzielenie ustawień w zespole.
Praca z gałęziami (branch), merge requestami i code review.
Projektowanie interfejsu użytkownika i/lub API systemu.
Implementacja modułów projektu zgodnie z przyjętą architekturą.
Tworzenie testów jednostkowych i integracyjnych.
Integracja modułów, rozwiązywanie konfliktów kodu, analiza jakości.
Przygotowanie wersji demonstracyjnej aplikacji i wdrożenie środowiska testowego.
Publiczna prezentacja projektów – pokaz działania, omówienie rozwiązań technicznych.
Retrospektywa projektu – analiza sukcesów, błędów i doświadczeń zespołu.
Zaliczenie i prezentacja projektów – ocena aplikacji, dokumentacji i pracy zespołowej.

3.4 Metody dydaktyczne

Wykład: prezentacja multimedialna, dyskusja
Laboratorium: wykonywanie ćwiczeń tablicowych i programistycznych, warsztaty, dyskusja

4. METODY I KRYTERIA OCENY

4.1 Sposoby weryfikacji efektów uczenia się

Symbol efektu	Metody oceny efektów uczenia się (np.: kolokwium, egzamin ustny, egzamin pisemny, projekt, sprawozdanie, obserwacja w trakcie zajęć)	Forma zajęć dydaktycznych (w, ćw, ...)
EK_01	kolokwium zaliczeniowe, aktywny udział w dyskusji	wykład
EK_02	kolokwium zaliczeniowe, aktywny udział w dyskusji	wykład
EK_03	kolokwium zaliczeniowe, aktywny udział w dyskusji	wykład
EK_04	obserwacja w trakcie zajęć / projekt / kolokwium	laboratorium
EK_05	obserwacja w trakcie zajęć / projekt / kolokwium	laboratorium
EK_06	obserwacja w trakcie zajęć / projekt / kolokwium	laboratorium

4.2 Warunki zaliczenia przedmiotu (kryteria oceniania)

1. Zaliczenie przedmiotu odbywa się na podstawie dwóch składowych:
    a. zaliczenia wykładu,
    b. oceny z laboratorium.
2. Zaliczenie laboratorium obejmuje:
    a. projekt grupowy,
    b. publiczną prezentację projektu grupowego,
    c. kolokwium zaliczeniowe
3. Projekt grupowy polega na opracowaniu rozwiązania konkretnego problemu poprzez przygotowanie i wdrożenie aplikacji oraz publiczną prezentację wyników projektu.
4. Student zalicza laboratorium, jeżeli:
    a. uzyska ocenę co najmniej 3.0 z projektu,
    b. ocenę co najmniej 3.0 z kolokwium zaliczeniowego.
5. Student zalicza przedmiot, jeśli:
    a. Zaliczył wykłady,
    b. otrzymał z laboratorium ocenę co najmniej 3.0.

Ocenę dst. z kolokwium/projektu student uzyskuje w przypadku uzyskania minimum połowy możliwych do zdobycia punktów. Kolejne oceny równomiernie pokrywają skalę punktową.

5. CAŁKOWITY NAKŁAD PRACY STUDENTA POTRZEBNY DO OSIĄGNIĘCIA ZAŁOŻONYCH EFEKTÓW W GODZINACH ORAZ PUNKTACH ECTS

Forma aktywności	Średnia liczba godzin na zrealizowanie aktywności
Godziny z harmonogramu studiów	45
Inne z udziałem nauczyciela akademickiego (udział w konsultacjach, egzaminie)	–
Godziny niekontaktowe – praca własna studenta (przygotowanie do zajęć, egzaminu, napisanie referatu itp.)	45
SUMA GODZIN	90
SUMARYCZNA LICZBA PUNKTÓW ECTS	**3**

** Należy uwzględnić, że 1 pkt ECTS odpowiada 25-30 godzin całkowitego nakładu pracy studenta.*

6. PRAKTYKI ZAWODOWE W RAMACH PRZEDMIOTU

wymiar godzinowy	–
zasady i formy odbywania praktyk	–

7. LITERATURA

Literatura podstawowa:

1. S. McConnell, Code Complete. A Practical Handbook of Software Construction, Microsoft Press, 2nd ed., 2004.
2. R. C. Martin, Clean Code. A Handbook of Agile Software Craftsmanship, Prentice Hall, 2008.
3. K. Schwaber, J. Sutherland, Scrum Guide. Przewodnik po Scrumie, najnowsza edycja online (dostępna po polsku i angielsku na scrumguides.org).
4. M. Fowler, Refactoring. Improving the Design of Existing Code, Addison-Wesley, 2nd ed., 2018.
5. M. Białas, Inżynieria oprogramowania. Aspekty praktyczne, PWN, Warszawa 2020.

Literatura uzupełniająca:

1. A. Cockburn, Agile Software Development: The Cooperative Game, Addison-Wesley, 2nd ed., 2006.
2. R. C. Martin, Clean Architecture. A Craftsman's Guide to Software Structure and Design, Prentice Hall, 2017.
3. J. Hunt, Advanced Guide to Git, Apress, 2020.
4. J. Humble, D. Farley, Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation, Addison-Wesley, 2010.
5. R. Pressman, B. Maxim, Inżynieria oprogramowania. Podejście praktyczne, PWN, Warszawa 2022 (tłum. Software Engineering: A Practitioner's Approach).
6. The Scrum Guide, Ken Schwaber, Jeff Sutherland. Dostęp: https://scrumguides.org
   Oficjalny przewodnik po metodyce Scrum – dostępny w języku polskim i angielskim. Fundamentalne źródło wiedzy o pracy zespołowej w projektach IT.
7. Pro Git (Scott Chacon, Ben Straub). Wydanie online: https://git-scm.com/book/en/v2
   Darmowa książka o Git. Zawiera praktyczne przykłady pracy zespołowej, branchy, merge requestów i CI/CD.
8. Google for Developers – Software Engineering Best Practices. Dostęp: https://developers.google.com/tech-writing
   Zbiór krótkich kursów o tworzeniu dokumentacji technicznej i komunikacji w zespołach developerskich. Bardzo przydatny przy nauce pisania README i wiki projektowych.