POZNAN UNIVERSITY OF TECHNOLOGY

EUROPEAN CREDIT TRANSFER AND ACCUMULATION SYSTEM (ECTS)

COURSE DESCRIPTION CARD - SYLLABUS

Course name
IT tools [S1Cybez1>NINF]

Course

Field of study Cybersecurity	Year/Semester 2/3
Area of study (specialization) –	Profile of study general academic
Level of study first-cycle	Course offered in Polish
Form of study full-time	Requirements compulsory

Number of hours

Lecture 16	Laboratory classes 16	Other 0
Tutorials 0	Projects/seminars 0	

Number of credit points

2,00

Coordinators	Lecturers
dr inż. Łukasz Kułacz lukasz.kulacz@put.poznan.pl	

Prerequisites

The student should have a basic knowledge of programming, especially the concepts of variables, classes and functions. He or she should have the ability to implement simple programs and recognize the risks associated with creating underdeveloped software.

Course objective

The purpose of the course is to provide students with basic knowledge of the tools used during software development. These tools are primarily concerned with storing and version control of code in local and remote repositories, collaboration on the same code, quality assurance techniques through tests of varying levels of detail and complexity, the basics of continuous integration and continuous deployment methods, and extensions to code editors that facilitate the creation of good quality software.

Course-related learning outcomes

Knowledge:
The student has basic theoretical and practical knowledge of software repositories, software testing, automation of the testing process and principles of creating correct and readable code. The student is familiar with basic tools to facilitate software development.

Skills:
The student is able to effectively use a local software repository for his/her work, as well as sync efef his/her work through a remote repository. The student is able to collaborate with other software developers through remote repositories and link their software to them. The student is able to prepare basic tests for code and test their code both locally and through automation linked to a remote software repository. The student is also able to use basic tools to facilitate the work of software development and thus improve the quality of the code being developed.

Social competences:
The student is aware of the possibilities and limitations during software development, especially the need to ensure good code quality. Understands the potential impact of incorrectly prepared software. Is aware of the challenges and risks associated with multiple programmers working together on the same code.

Methods for verifying learning outcomes and assessment criteria

Learning outcomes presented above are verified as follows:

In terms of the laboratory, verification of the established learning outcomes is carried out by: substantive evaluation of individual or group tasks performed during the class or in the form of tasks to be performed after the class, activity during the class.
In the field of lectures, verification of the established learning outcomes is realized by a written test, which can include single-choice, multiple-choice and open-ended questions.
In each form of the course assessment, the grade depends on the number of points the student earns relative to the maximum number of required points. Earning at least 50% of the possible points is a prerequisite for passing. The relationship between the grade and the number of points is defined by the Study Regulations. Additionally, the course completion rules and the exact passing thresholds will be communicated to students at the beginning of the semester through the university's electronic systems and during the first class meeting (in each form of classes).

Programme content

1. Code repository.
2. Unit, integration, functional and performance testing.
3. Test automation and software development cycle (CI/CD).
4. Software quality control tools.
5. Tools to facilitate work during software development.

Course topics

1. Basics of the git system.
2. Collaboration using the git system.
3. Preparation of unit and integration tests.
4. Preparation of functional and performance tests.
5. Test automation on the GitHub platform.
6. Code quality control.
7. Tools to facilitate work during software development.

Teaching methods

Lecture: multimedia presentation, supplemented by examples and additional explanations based on code snippets.
Laboratories: working at the computer, performing individual tasks, performing group tasks.

Bibliography

Basic:
1. Jakość oprogramowania. Podręcznik dla profesjonalistów. Michał Sobczak
2. TDD w praktyce. Niezawodny kod w języku Python. Harry Percival
3. Git i GitHub. Kontrola wersji, zarządzanie projektami i zasady pracy zespołowej. Mariot Tsitoara

Additional:
1. Testowanie i jakość oprogramowania. Modele, techniki, narzędzia. Adam Roman

Breakdown of average student's workload

	Hours	ECTS
Total workload	57	2,00
Classes requiring direct contact with the teacher	32	1,00
Student's own work (literature studies, preparation for laboratory classes/ tutorials, preparation for tests/exam, project preparation)	25	1,00