COURSE DESCRIPTION CARD - SYLLABUS

Course name
Ubiquitous systems and applications [S1Inf1>UBI]

Course

Field of study
Computing

Year/Semester
2/4

Area of study (specialization)
–

Profile of study
general academic

Level of study
first-cycle

Course offered in
polish

Form of study
full-time

Requirements
elective

Number of hours

Lecture
16

Laboratory classes
16

Other (e.g. online)
0

Tutorials
0

Projects/seminars
0

Number of credit points

3,00

Coordinators

dr inż. Bartłomiej Prędki
bartlomiej.predki@put.poznan.pl

Lecturers

Prerequisites

Student should have knowledge concerning the way the computer works, imperative programming (obtained in earlier courses) and the basics of computer networks. Should be able to solve basic problems in computing, especially in user interface design and application of specific algorithms. Student should understand the need to expand his competence and be ready to partake in group activities. Besides, student should have basic social competence like honesty, responsibility, persistence, curiosity and creativity, respect for others.

Course objective

1. Students should obtain knowledge concerning the history of mobile and ubiquitous computer systems. 2. Students should be able to design and programme the ubiquitous system and process data in cloud. 3. Students should have knowledge about different forms of wireless communication. 4. Students should enhance their abiliti to work in teams.

Course-related learning outcomes

Knowledge:
1. Student has a structured and well grounded knwoledge of ubiquitous systems.

2. Student has konwledge of current developments in ubiquitous systems.
3. Student knows basic techniques, methods and tools used to solve problems associated with ubiqitous systems.
4. Student has a structured knowledge of computer architectures and operating systems.

Skills:
1. Student can search for information concerning ubiqitous systems in literature, data bases and other sources (in Polish and English languages), integrate it and formulate opinion.
2. Student is able to use information-communication techniques while solving problems in system design, especially in ubiqitous systems.
3. Student is able to choose and apply adequate methods considering ubiquitous systems.
4. Student can design an ubiquitous system, choose an appropriate programming language and methodology.
5. Student can formulate algorithms and implement them using one of the ubiquitous associated languages.
6. Student can plan his/her own development and can see need for constant dicscovery of new knowledge.

Social competences:
1. Student knows, that skills and knowledge can quickly become obsolete.
2. Student is aware of knowledge importance in solving of engineering problems and knows the dangers of bad design and computer system malfunctions.

Methods for verifying learning outcomes and assessment criteria

Learning outcomes presented above are verified as follows:

Presented outomes are verified as follows:
Forming degree:
a) on lectures - based on the answers concerning material presented on previous lectures;
b) on laboratories - based on the fullfilment of current tasks,
Summary degree:
- verification of skills used in laboratory excersises,
- constant verification in classes - verification of knowledge and skill acquisition,
- written test consisting of 10-15 questions; to pass the test studet has to obtain at least 50% of correct answers.
Additional point obtained in classes, especially:
- demonstration of interesting extraculicular competences,
- presentation of additional problem aspects,
- doing a presentation on interesting subject concerning ubiqituous systems,
- efficacy of obtained knowledge use while solving a problem,
- ability to work in team,
- useful remarks concerning teaching materials.

Programme content

Following subjects are presented on lectures:
- programming for iOS and iPadOS using Swift with some elements of Objective-C,
- programming using different frameworks: UIKit, SwiftUI, SpriteKit,
- programming using different API's, e.g.: CoreData, SwiftData, CoreLocation,
- using Cloud services,
- data exchange protocols, e.g. JSON, REST,
- In laboratories student are trying to solve in practice tasks presented in lectures as a series of mini projects usually in a single class. Laboratory classes are designed so a student has no need to own/have access to any iOS or macOS device. During classes students work on mac mini computers; entire curriculum is done entirelly in the laboratory - no need for any work outside of the classes.

Teaching methods

1. Lecture: multimedia prsentation, discussion, demonstration.
2. Laboratories: doing tasks, team work, design and implementation of sample problems.

Bibliography

1. Podstawy języka Swift : programowanie aplikacji dla platformy iOS, Mark A. Lassoff, Helion 2016
2. iOS 12 : wprowadzenie do programowania w Swifcie, Matt Neuburg, Helion 2019
3. Objective-C : praktyczny podręcznik tworzenia aplikacji na systemy iOS i Mac OS X!,Stephen G. Kochan, Helion 2012
4. Poznaj Swifta, tworząc aplikacje : profesjonalne projekty dla systemu iOS, Emil Atanasov, Helion 2019
5. Service design patterns: fundamental design solutions for SOAP/WSDL and RESTful Web services, Robert Daigneau, Addison-Wesley, 2012
6. Inteligentny dom: automatyzacja mieszkania za pomocą platformy Arduino, systemu Android i zwykłego komputera / Mike Riley, Helion 2013
7. Android : programowanie aplikacji / Dawn Griffiths, David Griffiths, Helion 2016

Breakdown of average student's workload

	Hours	ECTS
Total workload	75	3,00
Classes requiring direct contact with the teacher	32	1,50
Student's own work (literature studies, preparation for laboratory classes/ tutorials, preparation for tests/exam, project preparation)	43	1,50