COURSE DESCRIPTION CARD - SYLLABUS

Course name
Frontend Development

Course

Field of study
Computing
Area of study (specialization)
Software Engineering
Level of study
Second-cycle studies
Form of study
full-time

Year/Semester
2/3
Profile of study
general academic
Course offered in
English
Requirements
compulsory

Number of hours

Lecture
30
Laboratory classes
30

Number of credit points

5

Lecturers

Responsible for the course/lecturer:
Marcin Borowski, BEng, PhD
email: mborowski@cs.put.poznan.pl
phone: 61 665 3032
faculty: Faculty of Computing and Telecommunications
address: Piotrowo 2, 60-965 Poznań

Responsible for the course/lecturer:
Marcin Borowski, BEng, PhD
email: mborowski@cs.put.poznan.pl
phone: +48 61 665 30 32
faculty: Faculty of Computing and Telecommunications
address: Piotrowo 2, 60-965 Poznań

Prerequisites

The student starting this course should have basic knowledge of structured and object-oriented programming, programming using the MVC scheme, basic knowledge of internet technologies (HTML, CSS, JS), and basic knowledge of database design.

Should have the ability to solve basic problems related to the process of designing IT systems and the ability to obtain information from the indicated sources.

Should also understand the need to expand their competences / be ready to cooperate within the team. Moreover, in terms of social competences, the student must present such attitudes as honesty, responsibility, perseverance, cognitive curiosity, creativity, personal culture, respect for other people.

Course objective

1. Provide students with basic knowledge about technologies used in the construction of web applications, in particular frontend techniques, in the field of design approaches, technology selection and implementation (including solutions for mobile devices).
2. Developing students' skills in solving problems related to the design of web applications also operating in real time (reactivity), the use of frameworks, libraries and other tools supporting the construction of websites and web applications.
3. Shaping students' teamwork skills as well as independence in solving problems.

Course-related learning outcomes

Knowledge

Student:

- has advanced and in-depth knowledge of Internet technologies, the theoretical foundations of their building, as well as the methods, tools, and programming environments used to implement them
- has advanced detailed knowledge of front-end technologies
- knows advanced methods, techniques, and tools used to solve complex engineering tasks while building web applications

Skills

Student:

- can assess the usefulness and the possibility of using new achievements and new IT products (dedicated tools, dedicated languages, etc.)
- can - when formulating and solving engineering tasks - integrate knowledge from various areas of computer science and apply a system approach, also taking into account non-technical aspects
- can use information and communication techniques used in the implementation of front-end applications
- can assess the usefulness of methods and tools for solving an engineering task consisting in the construction or evaluation of an IT system or its components, including the limitations of these methods and tools - as a result, can choose the appropriate application development technology depending on the requirements

- can - following the given specification - design and implement complex internet applications - at least in part, using appropriate methods, techniques, and tools, including adapting existing or developing new tools for this purpose

Social competences

Student:

- understands that in computer science knowledge and skills very quickly become obsolete, especially internet technologies
- understands the need to use the latest technology achievements and knows examples and understands the causes of malfunctioning IT systems that may lead to serious financial, image, or social losses

Methods for verifying learning outcomes and assessment criteria

Learning outcomes presented above are verified as follows:

Formative assessment

a. Lecture: based on activity during the interactive parts of the lectures;

b. laboratory: based on the assessment of the current progress in the implementation of tasks;

Summative assessment

a. Lecture:

- assessment of the presentation prepared by the student on the chosen technique, library, or framework used in building web applications;

b. Laboratory:

- verification of the assumed learning outcomes realized by
- students' assessment and defense of the prepared tasks - 5 small projects;

When assigning the final grade, the student may obtain an increase in grade for:

- discussing additional aspects of the presented issues, not presented during classes;
- using skills and knowledge from outside the study program to solve the tasks performed;
- help in improving teaching materials related to the subject;

Programme content

Lecture:

The lecture program covers the following topics: the HTTP communication protocol. Introduction to node.js technology Building simple servers of popular network services (echo, chat, HTTP). Introduction to the Express.js framework. Introduction to the AngularJS framework. Introduction to the ReactJS framework. Introduction to the Meteor.js framework. Introduction to the Svelte library. Overview of supporting tools such as Grunt, Gulp, Webpack, Rollup, SASS, Less, Postcss. Languages for defining application templates and components EJS, Jade, HAML, JSX.

Laboratory:

Laboratory classes are conducted in the form of fifteen 2-hour exercises, held in the laboratory. Classes are carried out independently by students. The laboratory program covers the following topics: Preparation of page templates and component views using HTML5, CSS, LESS, SASS, and the use of frameworks and component libraries (including Bootstrap, SemanticUI, Tailwindcss). Installation and configuration of the node.js environment. Running applications are written in node.js. Simple service servers. Implementation of simple applications in the Express.js framework with AngularJS and MongoDB, ReactJS, Meteor.js, Svelte / Sapper. Examples of the use of supporting tools and modules for node.js: Gulp, Webpack, Nodemon, Rollup, etc.

Teaching methods

Lecture: multimedia presentation, illustrated with examples given on the board.

Laboratory exercises: multimedia presentation, presentation illustrated with examples given on the whiteboard, live coding, and carrying out the tasks given by the teacher - practical exercises.

Bibliography

Basic

Technical documentation of the mentioned tools available on the Internet

Additional

Breakdown of average student's workload

	Hours	ECTS
Total workload	125	5,0
Classes requiring direct contact with the teacher	60	2,5
Student's work (literature studies; preparation for laboratory classes; participation in consultations; preparation of programs, launching, and testing; preparation of the final presentation and its presentation)	65	2,5