COURSE DESCRIPTION CARD - SYLLABUS

Course name
Frontend Development [S2Inf1E-IO>FDEV]

Course

Field of study
Computing

Year/Semester
2/3

Area of study (specialization)
Software Engineering

Profile of study
general academic

Level of study
second-cycle

Course offered in
English

Form of study
full-time

Requirements
compulsory

Number of hours

Lecture
30

Laboratory classes
30

Other (e.g. online)
0

Tutorials
0

Projects/seminars
0

Number of credit points

4,00

Coordinators

dr inż. Marcin Borowski
marcin.borowski@put.poznan.pl

Lecturers

Prerequisites

A student starting this subject should have basic knowledge of programming structured and object-oriented programming, programming using the MVC scheme, basic knowledge of web technologies (HTML, CSS, JS), and basic knowledge of database design. He/she should have the ability to solve basic problems related to the information systems design process and the ability to obtain information from indicated sources. He/she should also understand the necessity of extending his/her competences / have the readiness to cooperate within a team. Moreover, in terms of social competence, the student must present such attitudes as honesty, responsibility, perseverance, cognitive curiosity, creativity, personal culture, respect for other people.

Course objective

1. to provide students with basic knowledge of technologies used in the construction of web applications, in particular front-end techniques, in terms of design approaches, technology selection and implementation (including solutions for mobile devices). 2) Developing in students the ability to solve problems related to the design of web applications, including those operating in real time (responsiveness), the use of frameworks, libraries and other tools supporting the construction of services. 3. shaping in students the ability to work in a team as well as independently 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
1. lecture - on the basis of activity during the interactive parts of the lectures;
2. laboratory - on the basis of the evaluation of the ongoing progress of the assignments;
Summative assessment
1. lecture - evaluation of a self-prepared presentation by the student on a selected technology, library or framework used in the construction of web applications;
2. laboratory - evaluation and defence by students of prepared tasks - 4 mini-projects;

When calculating the final mark, the student may obtain a mark increase for:
- discussion of additional aspects of the presented issues, not presented in class;
- the use of skills and knowledge from outside the curriculum to solve ongoing tasks;
- assistance in improving teaching materials related to the subject;

Programme content

Lecture:
The lecture programme covers the following topics:
HTTP communication protocol. Introduction to node.js technology. Building simple servers popular web services (echo, chat, http). Introduction to the Express.js framework. Introduction to the Angular framework. Introduction to the React framework. Introduction to the SvelteKit framework. Discussion of supporting tools such as Grunt, Gulp, Webpack, Vite, SASS, Less and the TailwindCSS, Bootstrap libraries. Application template and component definition languages EJS, HAML, JSX.
Lab:

The laboratory classes are conducted in the form of fifteen 2-hour exercises, taking place in the laboratory. The exercises are carried out independently by the students. The lab programme covers the following topics:
Preparation of page templates and component views using HTML5, CSS, LESS, SASS and the use of frameworks and component libraries (e.g. Bootstrap, SemanticUI, Tailwindcss). Installing and configuring the node.js environment. Running applications written in node.js. Simple service servers. Implementation of simple applications realized in the Express.js framework with Angular and MongoDB, React, Svelte. Examples of using supporting tools and modules for node.js: Vite, Nodemon, Skeleton, etc.

Course topics

Lecture:
- HTTP communication protocol.
- Introduction to Node.js technology.
- Building simple servers and popular web services (echo, chat, HTTP).
- Introduction to the Express.js framework.
- Introduction to the Angular, React and SvelteKit frameworks.
- Support tools: Grunt, Gulp, Webpack, Vite, SASS, LESS, TailwindCSS, Bootstrap.
- Template and component definition languages: EJS, HAML, JSX.
Lab:
- Fifteen 2-hour lab activities, completed independently by students.
Topics:
- Preparation of page templates and component views using HTML5, CSS, LESS, SASS, and libraries and frameworks (e.g. Bootstrap, SemanticUI, TailwindCSS).
- Installation and configuration of the Node.js environment.
- Running applications written in Node.js.
- Building simple service servers.
- Implementation of simple applications using Express.js, Angular, MongoDB, React, Svelte.
- Examples of using tools and modules supporting Node.js: Vite, Nodemon, Skeleton, etc.

Teaching methods

Lecture: multimedia presentation, illustrated by examples given on the blackboard.
Laboratory exercises: multimedia presentation, illustrated by examples given on the blackboard, live coding and performance of tasks given by the instructor - practical exercises.

Bibliography

Technical documentation for these tools available on the internet

Breakdown of average student's workload

	Hours	ECTS
Total workload	100	4,00
Classes requiring direct contact with the teacher	60	2,50
Student's own work (literature studies, preparation for laboratory classes/ tutorials, preparation for tests/exam, project preparation)	40	1,50