UNIVERSITETI I EVROPËS JUGLINDORE
УНИВЕРЗИТЕТ НА ЈУГОИСТОЧНА ЕВРОПА
SOUTH EAST EUROPEAN UNIVERSITY

Study program	**Computer Sciences (2026/2027)**
Faculty	Contemporary Sciences and Technologies
Study Cycle	Second Cycle (Postgraduate)
ECTS	90
Code	CS-90
Title	Master in Computer Sciences
Accreditation date	01.10.2026

Description of the program

The introduction of this study programme is based on the strategic academic cooperation between South EastEuropean University (SEEU), North Macedonia, and Offenburg University of Applied Sciences (HSO), Germany,formalised through a Double Degree Cooperation Agreement. The programme has been jointly developed tocombine the academic strengths, educational resources, research capacities, and international expertise of bothinstitutions, offering students the opportunity to pursue high-quality education in Computer Science within aninternational and interdisciplinary academic environment.The study programme responds to the growing demand for highly qualified professionals in Computer Science,Software Engineering, Artificial Intelligence, Machine Learning, Computer Vision, Distributed Systems, CloudComputing, and related fields of Information and Communication Technologies (ICT). Through a moderncurriculum aligned with European higher education standards and labour market needs, the programme aims toequip students with advanced theoretical knowledge, practical skills, research competencies, and professionalexpertise required for successful careers in industry, research, and academia.A key objective of the programme is to promote internationalisation and academic mobility in accordance with theprinciples of the Bologna Process and the European Credit Transfer and Accumulation System (ECTS). Within theDouble Degree framework, students have the opportunity to study at both partner institutions, gain internationalacademic experience, and benefit from the complementary expertise of academic staff at SEEU and OffenburgUniversity of Applied Sciences. This cooperation fosters intercultural understanding, academic exchange, and thedevelopment of competencies essential for professional engagement in an increasingly interconnected globalenvironment.In accordance with the Double Degree Cooperation Agreement, students enrolled in the programme may obtaindegrees awarded by both institutions, provided that they fulfil the academic requirements jointly established bythe partner universities. The award of the Double Degree is conditional upon the successful completion of theprescribed mobility period, fulfilment of the curriculum requirements of both institutions, and preparation anddefence of a Master’s Thesis under the joint supervision of academic staff appointed by the partner universities.The Master’s Thesis shall be written and defended in English before a joint evaluation committee comprisingrepresentatives of both institutions, in accordance with the applicable academic regulations and procedures of thepartner universities.The programme also contributes to strengthening scientific cooperation, joint research activities, academic staffmobility, and collaboration with industry partners in North Macedonia, Germany, and the wider European HigherEducation Area (EHEA). Furthermore, it supports the strategic objectives of both institutions in the areas ofinternationalisation, educational excellence, innovation, and the development of highly qualified ICT professionalscapable of addressing contemporary technological and societal challenges.The curriculum has been designed to ensure compatibility and comparability with similar master’s programmesoffered by leading European higher education institutions, while providing graduates with a solid foundation forfurther academic advancement through doctoral studies and participation in international research and innovationprojects. By educating competent, innovative, and professionally responsible graduates, the programmecontributes to the development of a knowledge-based economy, the digital transformation of society, and thesustainable growth of the ICT sector at national, regional, and international levels.

Career

Upon completion of their studies, graduates may pursue careers as software engineers, systems developers,artificial intelligence specialists, data scientists, cybersecurity experts, or researchers in academic and researchinstitutions. They may find employment in information technology companies, public institutions, internationalorganisations, and various industrial sectors, as well as pursue doctoral studies.

Learning outcomes

Knowledge and understanding

Possession of knowledge and understanding of Computer Sciences areas and information architectures, networksocieties and internet cultures, Internet and web technologies proportionally expanded in comparison with secondcycle studies.Ability to develop and implement original and creative ideas in environments where overlapping or related fieldsof Information Technology occur.Ability to apply interdisciplinary knowledge and demonstration of specialist competencies in InformationTechnology.

Applying knowledge and understanding

Ability to critically, independently and creatively solve problems in new, previously not encountered orenvironments for which has no prior experience in a multidisciplinary context of real organizational environment.Planning, managing and evaluation of independent research in the field of Computer Sciences implementingappropriate Calculator tools, environments and technologies.Creativity and originality in the interpretation of the knowledge of e-technological processes and appropriate useof computer-based tools and environments based on defined techniques for research and investigation.

Making judgement

Ability for creative integration and synthesis of knowledge from many areas related to media processes and use ofcomputer tools and techniques.Ability to deal with complex situations related to process-specific technologies, the identification of appropriatespecialized domain instances in the internet and informatics and making sound judgments in situations lackingcomplete information or data based on personal, social and ethical principles and responsibilities related to theapplication of knowledge and understanding.

Communication skills

Ability to clearly and unambiguously communicate conclusions, results, studies and knowledge of ComputerSciences specialists' areas with the ability to adapt to the style and form of expression for non-specialists.Competence for critical, independent and creative argumentative research, evaluation methodologies andproposing and defending new hypotheses.Ability to initiate, conduct, and taking responsibility for individuals and groups in cases where communication,organizational and informatics competencies are of essential importance.

Learning skills

Ability to identify personal needs and directions for individual and autonomous additional education and itsperformance independently and autonomously in the Computer Science areas.Ability to assume responsibility for continuous individual learning in specialized and new e- technologies.

List of courses

Semester 1

- [CM362] [5.0 ECTS] **Project Management**
- [CM353] [5.0 ECTS] **Software Architectures**
- [CM351] [5.0 ECTS] **Advanced Programming**
- [CM306] [5.0 ECTS] **Model Driven Software Development**
- [CM307] [5.0 ECTS] **Parallel Computing**
- [5.0 ECTS] **Elective course**

Semester 2

- [CM354] [5.0 ECTS] **Computer Vision**
- [CM355] [5.0 ECTS] **Current Methods of Machine Learning and Applications**
- [CM356] [5.0 ECTS] **Compilers and Language Technologies**
- [CM357] [5.0 ECTS] **Team Project**
- [5.0 ECTS] **Elective course**
- [5.0 ECTS] **Elective course**

Semester 3

- [CST-MA-THESIS] [30.0 ECTS] **Master Thesis**

Description of courses

Core courses

- **Project Management**
  The course aims to develop students' knowledge and skills in applying contemporary project management principles, methodologies, and techniques. It prepares students to initiate, plan, execute, monitor, control, and close projects, covering key areas such as scope, schedule, cost, resource, quality, and risk management. Particular emphasis is placed on effective communication, leadership, stakeholder management, and the application of best practices to support successful project delivery.

- **Software Architectures**
  The course covers the fundamental principles, styles, and patterns used in designing software system architectures, including quality attributes, architectural trade-offs, documentation, and architecture evaluation. Students explore domain-driven design, modularity, microservices, cloud-native architectures, and distributed systems, as well as mechanisms for integration, security, resilience, and production deployment. Through requirements analysis, prototype development, and practical projects, students develop the skills needed to design, document, and evaluate software architectures while considering stakeholder needs, technical constraints, risks, and the requirements of complex systems.

- **Advanced Programming**
  The topics covered in this course include advanced functions, higher-order functions, lambda functions, modular programming, code organization, regular expressions, iterators, generators, decorators, context managers, exception handling, file processing, serialization, automated testing, debugging, logging, and code quality. The course also covers software design principles, selected design patterns, database connectivity, API development, multithreading, multiprocessing, coroutines, asynchronous programming, dependency management, version control, documentation, and project structuring. The course concludes with a practical project in which students apply advanced programming concepts to design, implement, test, and present a complete software solution.

- **Model Driven Software Development**
  The course covers Model-Driven Software Development (MDSD), including the fundamental concepts of modeling, metamodels, Model-Driven Architecture, UML and OCL, as well as the design and implementation of Domain-Specific Languages (DSLs). Through the use of technologies and tools such as EMF/Ecore, Xtext, Sirius/GMF, ATL, QVT, Xtend and Acceleo, students develop theoretical knowledge and practical skills in model creation, transformation and validation, automatic code generation, and model evolution and migration. The course also explores contemporary topics, including low-code/no-code platforms and AI-assisted modeling and code generation. Teaching is delivered through practical workshops, collaborative activities, discussions, idea development and problem-solving in a safe and inclusive learning environment that promotes active participation, communication, joint research and equal involvement in decision-making.

- **Parallel Computing**
  The course covers the fundamental and advanced principles of parallel computing, parallel computer architectures, performance analysis and scalability, as well as the design and optimization of parallel algorithms. Students are introduced to shared-memory programming using OpenMP, distributed-memory programming using MPI, and GPU computing with CUDA, developing practical skills in synchronization, inter-process communication, load balancing, and performance optimization. The course also explores the application of parallel computing in artificial intelligence and data science, along with emerging topics such as cloud-based high-performance computing, heterogeneous systems, Edge and Fog computing, exascale systems, and the fundamentals of quantum computing.

- **Computer Vision**
  The course aims to provide students with theoretical and practical knowledge of methods and algorithms for analyzing and interpreting digital images. The course covers image feature detection and description, image transformations and matching, motion analysis and visual odometry, as well as the application of machine learning and deep learning techniques for object classification, detection, and segmentation. Students will also develop practical skills in using modern tools and technologies, including MATLAB, Python, Keras, and TensorFlow, to implement and test computer vision algorithms.

- **Current Methods of Machine Learning and Applications**
  The course aims to provide students with knowledge of state-of-the-art (SOTA) machine learning algorithms and their applications across various fields of artificial intelligence. The course covers image generation methods, diffusion models, 3D scene reconstruction, multimodal learning, large language models (LLMs), and natural language processing (NLP). Students will also develop practical skills in implementing and evaluating models using PyTorch, critically analyzing scientific literature, conducting experiments, and presenting research findings.

- **Compilers and Language Technologies**
  The course provides an introduction to the fundamental principles and techniques of compiler design and implementation. Students will explore how compilers and interpreters work, the differences between them, and the use of Ahead-of-Time (AOT) and Just-in-Time (JIT) compilation strategies. Topics include variables, arithmetic expressions, type systems, programming language semantics, lexical analysis, and parsing, helping students understand how programs are analyzed and transformed during compilation. The course also introduces low-level programming through MIPS assembly and WebAssembly, as well as control flow, memory organization, functions, calling conventions, and object-oriented programming concepts. Advanced topics cover code optimization, register allocation, liveness analysis, and memory management techniques such as garbage collection. Through practical exercises and the implementation of basic components of a compiler pipeline, students will develop the skills needed to analyze, transform, and optimize programs and gain a deeper understanding of how programming languages are translated into executable code.

- **Team Project**
  The course aims to develop students' skills in planning, designing, and implementing practical projects in software engineering and artificial intelligence. Students will engage in identifying real-world problems, analyzing requirements, designing system architectures, developing and integrating software components, and applying Agile methodologies and DevOps practices. The course also emphasizes teamwork, testing, system evaluation, technical documentation, and the professional presentation of project outcomes.

- **Master Thesis**
  Master's thesis aims to: ⍰ Develop research skills: It shows that the student can formulate a research problem, design a methodology, collect and analyze data, and interpret results. ⍰ Contribute to knowledge: It should provide new insights, findings, or a deeper understanding of a topic within the chosen field. ⍰ Apply theoretical knowledge: It connects academic theories and concepts learned during the study program with real or theoretical research problems. ⍰ Demonstrate critical thinking: The student must critically evaluate literature, methods, and findings rather than only describing them. ⍰ Prepare for professional or academic careers: It builds competencies useful for further research (PhD studies) or advanced professional practice. In short, a master's thesis is both a learning process and a scientific output that proves the student's readiness to work independently at an advanced academic level. Furthermore, the activity seeks to improve research techniques and the writing style of the master's thesis, while strictly respecting the principles of academic integrity and avoiding the use of unauthorized means, such as plagiarism and copyright infringement, which are prohibited by the SEEU Statute.

Elective courses

- **Embedded and Industrial Networks**
  The course aims to provide theoretical and practical knowledge of the design, configuration, integration, and maintenance of Embedded Linux systems and industrial networks. Through seven modules, students will explore Embedded Linux system architecture, the boot process and bootloaders, root file system creation and kernel configuration, device driver and application development, system programming, debugging and troubleshooting, as well as system deployment and maintenance. Particular emphasis is placed on the interaction between embedded platforms, electronic devices, and industrial networks, enabling students to develop the skills required for device

integration, reliable communication, fault diagnosis, and ensuring the stable operation of systems in industrial environments.

- **Artificial Intelligence**
  The course provides a comprehensive overview of the fundamental principles, methods, and modern applications of Artificial Intelligence (AI). Students will explore the main AI paradigms, intelligent agents, search and optimization techniques, as well as knowledge representation and reasoning. The course covers machine learning methods, including supervised, unsupervised, and reinforcement learning, together with advanced techniques for model selection, evaluation, and optimization. Particular emphasis is placed on deep learning and neural networks, natural language processing, large language models (LLMs), and generative AI, including prompt engineering and Retrieval-Augmented Generation (RAG). Students will also examine the principles of explainable and responsible AI, including fairness, bias mitigation, ethics, and AI governance. Through practical exercises, projects, and case studies, students will develop the skills needed to design, implement, and evaluate intelligent solutions to complex real-world problems and integrate modern AI techniques into industrial, healthcare, financial, and research applications.

- **Cloud Infrastructures and Technologies**
  The course aims to equip students with theoretical and practical knowledge of contemporary cloud computing concepts, architectures, and technologies, covering cloud service and deployment models, virtualization, containers and cloud-native technologies, leading platforms such as AWS, Microsoft Azure, and Google Cloud Platform, as well as cloud infrastructure management, monitoring, and security. Through seven modules, students will develop skills in configuring and administering cloud resources, orchestrating applications with Kubernetes, implementing security and identity management mechanisms, automating infrastructure through DevOps, CI/CD, and Infrastructure as Code, and designing advanced hybrid and multi-cloud architectures. Particular emphasis is placed on serverless technologies, edge and fog computing, cloud cost optimization, the integration of artificial intelligence services, and sustainable cloud solutions. Upon completion, students will be able to design, deploy, secure, optimize, and efficiently manage modern cloud infrastructures, addressing the scalability, reliability, security, and operational requirements of contemporary digital environments.

- **Containerized Architectures**
  The course aims to provide students with theoretical and practical knowledge of contemporary containerization concepts, technologies, and architectures, covering the fundamental principles of containers, their comparison with virtual machines, Docker architecture and usage, container image, network, and volume management, as well as application orchestration using Kubernetes. The course also addresses networking in virtualized environments, VLAN and SDN technologies, container security and isolation, and best practices for building, deploying, and managing containerized applications. Through laboratory exercises, practical projects, and industry-based case studies, students will develop the skills required to design, implement, secure, and administer containerized architectures, enabling the efficient, reliable, and scalable development and execution of applications in modern computing environments.

- **Human–Computer Interaction.**
  The course aims to provide students with theoretical and practical knowledge of the principles, methods, and techniques used to design and evaluate interactions between people and computer systems. The course covers the fundamental concepts and historical development of HCI, the analysis of user characteristics, needs, and contexts of use, the application of design, usability, and accessibility principles, requirements gathering and task analysis, and the design of user-centered interfaces. Through the development of wireframes and low- and high- fidelity prototypes, usability testing, and evaluation, students will acquire the skills needed to analyze user experience, identify usability issues, and improve interactive solutions. Through a final course project, students will demonstrate their ability to design and present functional, intuitive, inclusive, and user-centered digital solutions that effectively address users' needs and expectations.

- **Web Services and Applications**
  The course aims to provide students with theoretical and practical knowledge of designing, developing, integrating, securing, and managing modern web services and applications, with particular emphasis on distributed systems and microservices architectures. Through seven modules, students will explore web architectures, HTTP/HTTPS and WebSocket protocols, RESTful API design and documentation, inter-service communication patterns, data management in distributed systems, and authentication, authorization, and security mechanisms. The course also covers cloud-native technologies, Docker containerization, Kubernetes orchestration, CI/CD automation, and service integration through APIs and messaging systems such as Kafka and RabbitMQ. Particular emphasis is placed on monitoring, observability, performance optimization, scalability, and system resilience, as well as emerging technologies, including serverless computing, edge computing, GraphQL, and AI-powered APIs. Upon completion,

students will be equipped to design, implement, integrate, and maintain secure, efficient, reliable, and scalable web applications and distributed service-based systems.

- **Advanced Algorithms and Data Structures**
  The course aims to develop students' theoretical knowledge and practical skills in the analysis, design, and optimization of algorithms and complex data structures. The course covers asymptotic analysis, recurrence relations, amortized and probabilistic analysis, balanced trees, heaps and priority queues, advanced hashing techniques, dynamic programming, graph algorithms, advanced data structures, and string-processing algorithms. It also addresses randomized algorithms, computational complexity theory, NP-complete problems, approximation algorithms, and emerging topics, including streaming algorithms, parallel algorithms, and external-memory algorithms. Through practical exercises, case studies, and presentations of research papers, students will develop the ability to evaluate the efficiency of algorithmic solutions and select and implement appropriate algorithms and data structures to solve complex computational problems.

- **Usability Testing**
  The course aims to provide students with theoretical and practical knowledge of methods, techniques, and tools for evaluating usability and user experience (UX) in digital systems. The course covers user profiling and context-of- use analysis, test planning, participant recruitment and ethical considerations, the development of task scenarios and testing materials, and the implementation of moderated, remote, and unmoderated usability tests. Through techniques such as think-aloud protocols, systematic observation, usability metrics, and data analysis, students will develop the skills needed to identify interaction problems, interpret findings, and prepare professional reports containing evidence-based recommendations for improvement. Through a practical project, students will apply all stages of usability testing to a website, application, or prototype, contributing to the development of more effective, intuitive, and user-centered digital products.

- **Requirements Engineering**
  The course aims to provide students with theoretical knowledge and practical skills in identifying, eliciting, analyzing, modeling, specifying, validating, and managing requirements throughout the software development lifecycle. The course covers functional and non-functional requirements, requirements elicitation techniques, use case modeling and UML diagrams, feasibility analysis, prioritization, and requirements documentation in accordance with relevant standards. It also addresses change management and requirements traceability, the use of tools such as Jira, DOORS, and Azure DevOps, agile requirements engineering, requirements for distributed and cloud-based systems, formal methods, and emerging trends involving artificial intelligence and DevOps. Through practical exercises and case studies, students will develop the ability to define clear, verifiable, and testable requirements, ensuring that software solutions meet user needs, stakeholder expectations, and project objectives.

- **Computer Security**
  The course aims to provide students with theoretical and practical knowledge of contemporary principles, methods, and technologies for protecting computer systems, networks, applications, and data against cyber threats. The course covers security principles, cryptography, identity and access management, threat modeling, risk assessment and management, vulnerability management, and the protection of networks, Linux and Windows operating systems, Active Directory, cloud environments, virtualized platforms, and containers. Particular emphasis is placed on software security, the Secure Software Development Lifecycle (SSDLC), Static and Dynamic Application Security Testing (SAST/DAST), DevSecOps practices, and OWASP standards, as well as security auditing, monitoring, and incident management. Through practical scenarios and applied security techniques, students will develop the skills required to identify and prevent attacks, detect and remediate vulnerabilities, respond effectively to cyber incidents, and design secure and resilient systems while addressing relevant governance, compliance, and regulatory requirements.

- **Formal Methods in Computer Science**
  The course aims to provide students with theoretical and practical knowledge of mathematical and logical techniques for modeling, specifying, analyzing, and verifying computer systems. The course covers the foundations of computation theory, finite-state automata, nondeterminism, regular expressions, and context-free grammars, as well as propositional, predicate, and temporal logic. Through practical exercises using tools such as Z3 and NuSMV, students will develop skills in formalizing logical statements, constructing and analyzing models, specifying system properties, and performing formal verification through model checking. The course also explores applications of computation theory and case studies involving real-world computer science problems, enabling students to apply rigorous reasoning, identify logical errors, and assess the correctness and reliability of software systems.

- **Software Testing and Quality Assurance**

The course aims to provide students with theoretical and practical knowledge of the principles, methods, and techniques of software testing and quality assurance throughout the software development lifecycle. The course covers software quality concepts and standards, verification and validation, white-box and black-box testing, static testing techniques for defect detection, test-case design, unit and integration testing, usability evaluation, and defect management. Through practical exercises in Java and Python, students will develop skills in test automation for web and mobile applications, the use of testing tools, performance and load testing, and the application of AI- assisted testing techniques within agile development environments. Upon completion, students will be able to design effective testing strategies, identify and document defects, evaluate application reliability and functionality, and contribute to the development of high-quality, secure, and reliable software products.