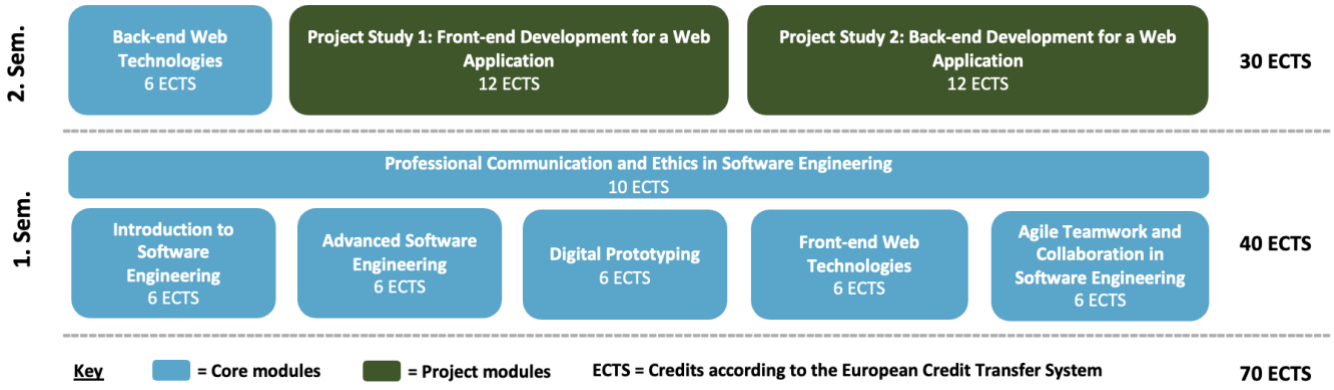
Syllabi
KG 7 Avenue
Ministry of Education (MINEDUC)
Kigali, RW

Module Handbook for the Study Program
Applied Software Engineering Foundations - “The Gym”

30.04.2025

Curriculum overview



ECTS overview

Type of Module	Module	ECTS	Semester 1	Semester 2
Core Module	Introduction to Software Engineering	6	x	
Core Module	Advanced Software Engineering	6	x	
Core Module	Digital Prototyping	6	x	
Core Module	Front-end Web Technologies	6	x	
Core Module	Agile Teamwork and Collaboration in Software Engineering	6	x	
Core Module	Professional Communication and Ethics in Software Engineering	10	x	
Core Module	Back-end Web Technologies	6		x
Project Study Module	Project Study 1: Front-end Development of a Web-Based Application	12		x
Project Study Module	Project Study 2: Back-end Development of a Web-Based Application	12		x
		70	40	30

Type of Module	ECTS	Semester 1	Semester 2
Core Modules	46	40	6
Project Study Modules	24	0	24
	70	40	30

1st Semester

Winter Term 2024/2025

<table>
<tr><td>Module ID
ASEF1</td><td colspan="2">Module name
Introduction to Software Engineering</td></tr>
<tr><td colspan="2">Type of course
Core Module</td><td>Semester / Rotation
Winter Term</td><td>Student capacity: 45</td></tr>
<tr><td colspan="2">Teaching methods:
Lecture, seminar, course</td><td>Prerequisites for attendance
None</td><td>Language
English</td></tr>
<tr><td colspan="3">Type of examination (Final Grade Composition)
Oral exam (100%), 20 minutes</td><td>ECTS (+Workload in hrs)
6 (180 hours, of this 135 self-study hours + 45 instruction hours)</td></tr>
<tr><td colspan="3">Module coordinator
Prof. Dr. Adam Roe</td><td>Weekly workload in hours:
30</td></tr>
<tr><td colspan="4">Additional instructor(s) involved:
Yannick Musafiri, BSc</td></tr>
<tr><td colspan="4">Syllabus
Software plays an essential role in our world, from daily life applications to complex systems. Software engineering forms the backbone of modern technology, enabling the creation of reliable, scalable, and efficient software systems. This module introduces students to the principles, processes, and methodologies that underpin the discipline. Additionally, students will develop foundational programming skills, preparing them for further study in software engineering.

Key topics covered:
• Overview of Software Engineering: Understanding the discipline's scope, ethical considerations, and societal impact.
• Software Development Processes: Comparative analysis of linear, iterative, and agile methodologies.
• Modeling and Design: Introduction to Unified Modeling Language (UML) for capturing requirements and system design.
• Requirements Engineering: Eliciting, documenting, and analyzing functional and non-functional requirements.
• Quality Assurance and Testing: Introduction to software quality assurance and the role of different testing strategies, including unit, integration, and system testing.
• Programming Basics: Introduction to programming concepts using JavaScript, including variables, data types, loops, functions, and basic algorithms.
• Version Control Systems: Using Git for version control in individual and team projects.</td></tr>
</table>

Learning goals and qualifications:

Upon successful completion of this module, students will be able to:

1. Recall foundational principles, terminology, and the scope of software engineering. *(1)*
2. Understand and compare software development methodologies to identify the most suitable approach for specific project contexts. *(2)*
3. Apply UML diagrams to model requirements and system designs effectively. *(3)*
4. Analyze software requirements to identify key functional and non-functional needs. *(4)*
5. Explain fundamental software testing concepts and their role in ensuring software quality. (2)
6. Apply programming concepts to solve fundamental problems. *(3)*
7. Use Git for version control in individual and team-based projects. *(3)*
8. Reflect on their learning process and formulate strategies for continuous professional growth. *(6)*

(Numbers reflect Bloom's cognitive skill classification.)

Classification of cognitive skills following Bloom (1956):

1 = *Knowledge*: recalling facts, terms, basic concepts and answers; 2 = *Comprehension*: understanding something; 3 = *Application*: using a general concept to solve problems in a particular situation; 4 = *Analysis*: breaking something down into its parts; 5 = *Synthesis*: creating something new by putting parts of different ideas together to make a whole; 6 = *Evaluation*: judging the value of material or methods.

Core readings:

- Sommerville, I. (2015). Software engineering (10th ed.). Pearson Education.
- Bruegge B., Dutoit, A. (2010). Object-Oriented Software Engineering: Using UML, Design Patterns and Java, 3rd Edition, Pearson Education.
- Ferguson, R. (2019). Beginning JavaScript: The ultimate guide to modern JavaScript development (3rd ed.). Apress.

Module ID ASEF2	Module name Advanced Software Engineering	
Type of course Core Module	Semester / Rotation Winter Term	Student capacity: 45
Teaching methods: Lecture, seminar, course	Prerequisites for attendance Introduction to Software Development	Language English
Type of examination (Final Grade Composition) Oral exam (100%), 20 minutes		**ECTS (+Workload in hrs)** 6 (180 hours, of this 135 self-study hours + 45 instruction hours)
Module coordinator Prof. Dr. Adam Roe		Weekly workload in hours: 30
Additional instructor(s) involved: Yannick Musafiri, BSc		

Syllabus

Building on the foundations laid in Introduction to Software Engineering, this module explores advanced programming concepts and tools. Students will develop proficiency in writing maintainable and scalable code while learning advanced JavaScript (ES6+), Git workflows, and foundational software engineering concepts such as design patterns and testing frameworks. Furthermore, the module will cover the application of Artificial Intelligence in software development, including the practical use of modern AI-assisted tools.

Key topics covered:

- **Modern JavaScript (ES6+)**:
  - Advanced syntax: Arrow functions, destructuring, template literals, and modules.
  - Object-oriented programming: Classes, inheritance, and encapsulation.
  - Introduction to design patterns: Singleton, observer, and factory patterns.
- **Version Control Advanced**:
  - Git: Advanced workflows (e.g., rebasing, resolving merge conflicts).
  - Collaboration using pull requests, code reviews, and GitHub/GitLab tools.
- **Introduction to Testing**:
  - Unit testing using Jest or similar frameworks.
  - Writing and executing structured test cases, mocking dependencies, and measuring code coverage.
- **Debugging and Performance**:
  - Efficient debugging techniques and basic performance optimization.
  - Profiling code for bottlenecks.
- **Artificial Intelligence in Software Development**
  - Leveraging AI to enhance productivity in coding, debugging, and testing.
  - AI-assisted development tools (e.g., GitHub Copilot, ChatGPT, AI-driven debugging).
  - Introduction to machine learning in applications (e.g., TensorFlow.js, ML5.js).

Learning goals and qualifications:

Upon successful completion of this module, students will be able to:

1. Write advanced JavaScript code using ES6+ syntax and concepts. *(3)*
2. Collaborate effectively on projects using advanced Git workflows. *(4)*
3. Apply object-oriented and functional programming techniques to build scalable software. *(3)*
4. Utilize design patterns to address recurring software design challenges. *(3)*
5. Debug and optimize complex software systems. *(3)*
6. Design, implement, and evaluate unit tests to ensure code quality and reliability. (6)
7. Leverage AI-assisted development tools to improve coding efficiency and debugging processes. (3, 5)
8. Apply AI-driven automation techniques to streamline testing and performance optimization. (3, 5)
9. Integrate machine learning components into JavaScript applications using TensorFlow.js or ML5.js. (3, 5)

(Numbers reflect Bloom's cognitive skill classification.)

Classification of cognitive skills following Bloom (1956):

1 = *Knowledge*: recalling facts, terms, basic concepts and answers; 2 = *Comprehension*: understanding something; 3 = *Application*: using a general concept to solve problems in a particular situation; 4 = *Analysis*: breaking something down into its parts; 5 = *Synthesis*: creating something new by putting parts of different ideas together to make a whole; 6 = *Evaluation*: judging the value of material or methods.

Core readings:

- Ferguson, R. (2019). Beginning JavaScript: The ultimate guide to modern JavaScript development (3rd ed.). Apress.
- Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). Design patterns: Elements of reusable object-oriented software. Addison-Wesley Professional.

Module ID ASEF3	Module name Digital Prototyping	
Type of course Core Module	Semester / Rotation Winter Term	Student capacity: 45
Teaching methods: Lecture, seminar, course, project work	Prerequisites for attendance None	Language English
Type of examination (Final Grade Composition) Oral exam (30%), 20 min Project Work (40%) Presentation (30%)		**ECTS (+Workload in hrs)** 6 (180 hours, of this 135 project-work hours + 45 instruction hours)
Module coordinator Prof. Dr. Adam Roe		Weekly workload in hours: 30
Additional instructor(s) involved: Yannick Musafiri, BSc		

Syllabus

This module introduces students to the fundamental principles and methods of digital prototyping, emphasizing the integration of interaction design theory, cognitive psychology, and software ergonomics to create user-centered digital products. Students will learn to translate ideas into interactive prototypes through a structured design process, starting with rigorous problem definition and user research. They will follow the entire prototyping lifecycle, from low-fidelity to high-fidelity prototyping, using modern tools and incorporating iterative user feedback. This process emulates product development in innovative companies, with a strong focus on usability, accessibility, and ethical design practices. Teams of five students will collaborate to deliver a functional, interactive prototype by the end of the course.

Key topics covered:

- **Problem Definition and User Persona Development**
  - Identifying and defining problem statements.
  - Creating user personas based on target audience research.
  - Understanding the needs, goals, and pain points of the users.
- **Introduction to Prototyping**
  - Definition and importance of digital prototyping.
  - Theories of design: distributed cognition, activity theory, structuration theory.
  - Understanding user needs: principles from cognitive and perception psychology.
- **Low-Fidelity Prototyping**
  - Manual prototyping using pen and paper to create user flows and interfaces.
  - Affordances in interaction design: how media elements (text, image, video, animation) influence user experience.
  - Translating concepts into digital wireframes.
- **High-Fidelity Prototyping**
  - Principles of user-oriented interaction design.
  - Rapid prototyping methods and tools: Using Figma and Thunkable for interactive prototypes.
  - Advanced design elements: animations, transitions, and responsive layouts.
- **Usability Testing and Iterative Design**
  - Conducting task and work analysis for usability.
  - Software and media ergonomics: evaluating usability and cognitive load.
  - Gathering and integrating stakeholder feedback.
- **Ethical Considerations in Prototyping and UX Design**

- Ensuring accessibility, privacy, and inclusivity in prototyping decisions.
- Addressing bias and designing for diverse users.

- **Final Prototype Delivery**
  - Delivering a fully functional prototype that meets project requirements.
  - Preparing a presentation and demo for showcasing the prototype.
  - Pitching solutions to industry experts.

Learning goals and qualifications:

Upon successful completion of this module, students will:

1. Apply design thinking principles to define user problems and create user personas based on research. (3)
2. Apply design thinking principles and theories of interaction design to identify user needs and generate innovative solutions. *(3, 5)*
3. Create low-fidelity paper prototypes and transition to high-fidelity digital prototypes using user-oriented design principles. *(3, 4)*
4. Use Figma and Thunkable to develop interactive prototypes, incorporating principles of usability and software ergonomics. *(3, 5)*
5. Conduct usability testing and integrate feedback based on cognitive and work psychology. *(4, 6)*
6. Collaborate effectively in teams to develop, refine, and present prototypes. *(5, 6)*
7. Identify and address ethical considerations in prototyping and UX design, including accessibility, privacy, and bias. (3, 6)
8. Present and justify design choices using theoretical frameworks from Human-Computer Interaction and interaction design. *(6)*

(Numbers reflect Bloom's cognitive skill classification.)

Classification of cognitive skills following Bloom (1956):

1 = *Knowledge*: recalling facts, terms, basic concepts and answers; 2 = *Comprehension*: understanding something; 3 = *Application*: using a general concept to solve problems in a particular situation; 4 = *Analysis*: breaking something down into its parts; 5 = *Synthesis*: creating something new by putting parts of different ideas together to make a whole; 6 = *Evaluation*: judging the value of material or methods.

Core readings:

- Buxton, B. (2010). Sketching user experiences: Getting the design right and the right design. Morgan Kaufmann.
- Norman, D. A. (2013). The design of everyday things: Revised and expanded edition. Basic Books.
- Preece, J., Rogers, Y., & Sharp, H. (2002). Interaction Design. Wiley & Sons.

Module ID ASEF4	Module name Front-end Web Technologies	
Type of course Core Module	Semester / Rotation Winter Term	Student capacity: 45
Teaching methods: Lecture, seminar, course, project work	Prerequisites for attendance Introduction to Software Development	Language English
Type of examination (Final Grade Composition) Oral exam (100%), 20 minutes		**ECTS (+Workload in hrs)** 6 (180 hours, of this 135 self-study hours + 45 instruction hours)
Module coordinator Prof. Dr. Adam Roe		Weekly workload in hours: 30
Additional instructor(s) involved: Yannick Musafiri, BSc		

Syllabus

This module focuses on creating dynamic, responsive, and user-centric web applications using modern front-end technologies. Students will deepen their understanding of JavaScript by learning TypeScript, explore asynchronous and event-based programming, and work with frameworks such as React. Emphasis is placed on developing scalable and maintainable front-end systems with a focus on accessibility, usability, and performance.

Key topics covered:

- **Front-end Development Tools**:
  - HTML and CSS: Advanced techniques including animations, selectors, and responsive layouts using Grid and Flexbox.
  - JavaScript for front-end: DOM events, asynchronous programming, and error handling.
- **TypeScript**: Static typing, interfaces, and generics.
- **React**:
  - Component-based architecture and state management (Redux and Context API).
  - React hooks: useState, useEffect, and custom hooks.
  - Performance optimization and debugging React applications.
- **User Experience (UX) Design**:
  - Responsive web design principles and accessibility best practices.
  - Creating intuitive and interactive user interfaces.
- **Front-end Testing**:
  - Unit testing with tools like Jest and React Testing Library.
- **Deployment Tools**:
  - Basics of front-end deployment using platforms like Netlify or Vercel.

Learning goals and qualifications:

Upon successful completion of this module, students will be able to:

1. Build interactive, responsive web applications using React and TypeScript. *(3)*
2. Manage application state effectively with Redux and Context API. *(4)*
3. Design web interfaces that align with UX principles and accessibility standards. *(5)*
4. Debug and test front-end applications to ensure quality assurance. *(3)*
5. Deploy front-end applications using modern tools and platforms. *(3)*

(Numbers reflect Bloom's cognitive skill classification.)

Classification of cognitive skills following Bloom (1956):

1 = *Knowledge*: recalling facts, terms, basic concepts and answers; 2 = *Comprehension*: understanding something; 3 = *Application*: using a general concept to solve problems in a particular situation; 4 = *Analysis*: breaking something down into its parts; 5 = *Synthesis*: creating something new by putting parts of different ideas together to make a whole; 6 = *Evaluation*: judging the value of material or methods.

Core readings:

- Banks, A., & Porcello, E. (2020). Learning React: Modern patterns for developing React apps (2nd ed.). O'Reilly Media.
- Cherny, B. (2019). TypeScript: Programming for JavaScript developers. O'Reilly Media.
- Sunyaev, A. (2020). Internet computing: Principles of distributed systems and emerging internet-based technologies. Springer.
- Ackermann, P. (2023). Full Stack Web Development: The Comprehensive Guide (Rheinwerk Computing). Rheinwerk Computing.

<table>
<tr><td>Module ID
ASEF5</td><td colspan="2">Module name
Agile Teamwork and Collaboration in Software Engineering</td></tr>
<tr><td colspan="2">Type of course
Core Module</td><td>Semester / Rotation
Winter Term</td><td>Student capacity: 45</td></tr>
<tr><td colspan="2">Teaching methods:
Lecture, seminar, course, project work</td><td>Prerequisites for attendance
Introduction to Software Development</td><td>Language
English</td></tr>
<tr><td colspan="3">Type of examination (Final Grade Composition)
Oral exam (30%), 20 min
Project Work (40%)
Presentation (30%)</td><td>ECTS (+Workload in hrs)
6 (180 hours, of this 135 project-work hours + 45 instruction hours)</td></tr>
<tr><td colspan="3">Module coordinator
Dr. Lenz Belzner</td><td>Weekly workload in hours:
30</td></tr>
<tr><td colspan="4">Additional instructor(s) involved:
Angel Gwiza, BSc</td></tr>
</table>

Syllabus

This module focuses on teamwork and collaboration in agile software engineering, preparing students for the dynamics of working in multidisciplinary teams. Students will engage in a practical, hands-on approach to understanding agile principles through project-based learning. The module emphasizes the fundamentals of agile workflows, such as Scrum, and their application in real-world scenarios.

Working in teams of five, students will build a fully functional website based on an example provided. Throughout the project, students will learn to divide tasks, align on project goals, manage dependencies, and communicate effectively. Key skills such as conflict resolution, intercultural collaboration, and providing constructive peer feedback are embedded in the learning process. Additionally, students will explore fundamental project management theories and principles, comparing agile methodologies with traditional approaches such as PMI's PMBOK, PRINCE2, and ISO 21500. This comparative analysis will provide a deeper understanding of how agile project management differs from conventional models in terms of planning, risk management, and decision-making. By the end of the module, students will be well-equipped with the teamwork and collaboration skills required for professional software development.

The syllabus is designed to combine theoretical understanding with practical application:

- **Introduction to Agile Methodologies**:
  - Principles of agile software development.
  - Overview of Scrum and Kanban workflows.
  - Comparison of agile with traditional project management frameworks (e.g., Waterfall, PMBOK, PRINCE2).
- **Team-Based Project Execution**:
  - Working collaboratively in teams of five to build a website based on a provided example.
  - Project planning using backlogs, sprints, and agile boards.
  - Conducting sprint planning, daily stand-ups, retrospectives, and sprint reviews.
  - Applying project management concepts such as scope management, stakeholder engagement, and resource allocation within agile teams.
- **Collaborative Techniques and Tools**:
  - Effective communication in synchronous and asynchronous environments.
  - Resolving conflicts and adapting to team dynamics.
  - Tools for agile teamwork: Jira, Trello, or similar.
  - Understanding project scheduling techniques, risk assessment, and cost estimation in software projects.
- **Feedback and Reflection**:

- Providing and receiving constructive peer feedback.
- Individual and team-based retrospectives to evaluate performance and collaboration.
- Analyzing real-world case studies of agile and traditional project management successes and failures.

- **Intercultural Awareness and Collaboration**:
  - Addressing cultural differences in team settings.
  - Strategies for fostering inclusion and understanding in global software development contexts.
- **AI in Agile Software Development**
  - AI-powered project management tools (e.g., Jira AI, ClickUp AI, Notion AI).
  - Automating workflows: AI-driven backlog prioritization, sprint planning, and risk assessment.
  - Enhancing team collaboration with AI-assisted documentation, chatbots, and decision support systems.

Learning goals and qualifications:

Upon successful completion of this module, students will:

1. Understand and apply agile principles, including Scrum and Kanban methodologies. *(2, 3)*
2. Collaborate effectively in cross-functional teams to deliver a functional website. *(5, 6)*
3. Manage project tasks using agile boards, retrospectives, and sprints. *(3, 5)*
4. Implement effective communication strategies in both synchronous and asynchronous settings. *(3, 5)*
5. Resolve conflicts and adapt to team dynamics and intercultural challenges. *(4, 6)*
6. Provide and integrate constructive peer feedback to improve individual and team performance. *(5, 6)*
7. Critically evaluate and compare agile and traditional project management methodologies, applying project management principles to software development. (4, 5, 6)
8. Analyze project risk, scheduling, and cost estimation techniques within agile and traditional frameworks. (4, 5, 6)
9. Apply leadership and decision-making models in agile project settings. (5, 6)
10. Utilize AI-powered project management tools to improve sprint planning, backlog prioritization, and risk assessment. (3, 5, 6)
11. Automate workflow processes using AI for task allocation, dependency management, and reporting. (4, 5, 6)
12. Enhance team collaboration through AI-driven documentation, chatbots, and decision-support systems. (4, 5, 6)

(Numbers reflect Bloom's cognitive skill classification.)

Classification of cognitive skills following Bloom (1956):

1 = *Knowledge*: recalling facts, terms, basic concepts and answers; 2 = *Comprehension*: understanding something; 3 = *Application*: using a general concept to solve problems in a particular situation; 4 = *Analysis*: breaking something down into its parts; 5 = *Synthesis*: creating something new by putting parts of different ideas together to make a whole; 6 = *Evaluation*: judging the value of material or methods.

Core readings:

- Schwaber, K., & Sutherland, J. (2020). The Scrum guide: The definitive guide to Scrum: The rules of the game. Scrum.org.
- Rubin, K. S. (2012). Essential Scrum: A practical guide to the most popular agile process. Addison-Wesley.
- Pinto, J. K. (2015). Project Management: Achieving Competitive Advantage. Pearson.
- Kerzner, H. (2017). Project Management: A Systems Approach to Planning, Scheduling, and Controlling. Wiley.

Module ID ASEF9	Module name Professional Communication and Ethics in Software Engineering	
Type of course Core Module	Semester / Rotation Winter Term	Student capacity: 45
Teaching methods: Lecture, seminar, course	Prerequisites for attendance	Language English
Type of examination (Final Grade Composition) Oral exam (100%), 20 min		**ECTS (+Workload in hrs)** 10 (300 hours, of this 225 self-study hours + 75 instruction hours)
Module coordinator Dr. Lenz Belzner		Weekly workload in hours: 10
Additional instructor(s) involved: Angel Gwiza, BSc		

Syllabus

This module runs in parallel to the other modules of the first semester and provides students with the non-technical foundations needed for their work as Software Engineers.

It covers the ethical competencies needed for students to be able to use their skills responsibly by addressing ethical and societal dimensions of software development. Students will examine the implications of their technical decisions and develop a principled approach to responsible innovation. The module also covers principles of academic integrity. A theoretical foundation anchors the practical work: students are introduced to major ethical theories and professional codes of conduct.

The module further develops the communication and intercultural collaboration skills required in the global software industry. Students will refine their ability to express themselves clearly and professionally in both written and spoken formats and learn to collaborate across cultural boundaries within international teams. Students are also introduced to essential communication and intercultural models as underlying theoretical foundations. These frameworks inform the students' reflection on professional behavior and support their ability to recognize the abstract concepts explaining situations they encounter in practice.

Drawing on real-world case studies and practical exercises, this module prepares students to engage constructively and ethically in professional environments.

Key topics covered:

- **Theoretical Foundations - Ethics:**
  - Definition of Ethics
  - Ethics vs Morality
  - Ethical theories: Consequentialism/Utilitarianism, Deontology, Virtue Ethics
  - Stakeholder Theory
  - Professional codes of conduct: ACM/IEEE Code of Ethics
- **Ethics and Responsibility in Software Development**
  - Application of professional standards and codes of conduct
  - Ethical decision-making in software design, data usage, and deployment
  - Societal and global impacts of software technologies
  - Inclusivity, accessibility, and sustainability in development practices
- **Theoretical Foundations - Communication and Intercultural Collaboration:**
  - Communication models: Shannon-Weaver Model, Johari Window
  - Intercultural communication frameworks: Hofstede's Cultural Dimensions
- **Professional Communication Practices and Intercultural Collaboration**
  - Structured communication in digital workspaces (stand-ups, documentation, messaging, and

reporting)
    - Public speaking and technical presentation skills
    - Asynchronous and cross-cultural communication in distributed teams
    - Writing effective emails, messages, summaries, and professional reports
- **Interpersonal Skills for Collaboration**
    - Giving and receiving feedback
    - Conflict resolution and intercultural sensitivity
    - Reflective practices for personal and team improvement
    - Improvisational techniques for spontaneity, active listening, and empathy
    - Building trust and rapport in virtual and in-person teams
- **Habits for Professional Growth**
    - Active learning techniques and time management strategies
    - Precision in understanding and responding to briefs
    - Peer-learning formats such as mental model sessions and mob programming

Learning goals and qualifications:

Upon successful completion of this module, students will be able to:

1. Communicate effectively and professionally within international software development teams, adapting to both synchronous and asynchronous formats. *(3, 5)*
2. Demonstrate intercultural sensitivity and collaboration skills needed for globalized team environments. *(3, 4, 6)*
3. Apply ethical principles and professional standards (including academic honesty and rigor) to assess and guide their work in software development. *(3, 4, 6)*
4. Analyze the societal impact of software systems and evaluate the ethical implications of design and implementation choices. *(4, 6)*
5. Understand the ACM Code of Ethics and Professional Conduct and/or IEEE Code of Ethics and how it relates to theoretical ethical principles and professional and social responsibilities. *(2)*
6. Engage in peer-learning formats that promote self-reflection, teamwork, and mutual improvement. *(5, 6)*
7. Develop structured thinking and active learning strategies for ongoing professional development. *(3, 5)*

(Numbers reflect Bloom's cognitive skill classification.)

Classification of cognitive skills following Bloom (1956):

1 = *Knowledge*: recalling facts, terms, basic concepts and answers; 2 = *Comprehension*: understanding something; 3 = *Application*: using a general concept to solve problems in a particular situation; 4 = *Analysis*: breaking something down into its parts; 5 = *Synthesis*: creating something new by putting parts of different ideas together to make a whole; 6 = *Evaluation*: judging the value of material or methods.

Core readings:

- Floridi, L. (2013). The Ethics of Information. Oxford University Press.
- Hofstede, G., Hofstede, G. J., & Minkov, M. (2010). Cultures and Organizations: Software of the Mind (3rd ed.). McGraw-Hill.
- Adler, R. B., Rosenfeld, L. B., & Proctor, R. F. (2018). Interplay: The Process of Interpersonal Communication (14th ed.). Oxford University Press.

2nd Semester

Summer Term 2025

<table>
<tr><td>Module ID
ASEF6</td><td colspan="2">Module name
Back-end Web Technologies</td></tr>
<tr><td colspan="2">Type of course
Core module</td><td>Semester / Rotation
Summer Term</td><td>Student capacity:
45</td></tr>
<tr><td colspan="2">Teaching methods:
Lecture, seminar, course</td><td>Prerequisites for attendance
Introduction to Software Development</td><td>Language
English</td></tr>
<tr><td colspan="3">Type of examination (Final Grade Composition)
Oral exam (100%), 20 minutes</td><td>ECTS (+Workload in hrs)
6 (180 hours, of this 135 self-study hours + 45 instruction hours)</td></tr>
<tr><td colspan="3">Module coordinator
Prof. Dr. Adam Roe</td><td>Weekly workload in hours:
40</td></tr>
<tr><td colspan="4">Additional instructor(s) involved:
Yannick Musafiri, BSc</td></tr>
</table>

Syllabus

This module delves into back-end development for web applications, teaching students to design and implement robust, secure, and scalable server-side systems. Students will learn how to create APIs, manage databases, and apply security best practices. By the end of the module, they will be able to develop complete back-end solutions using Node.js and Express, integrating them seamlessly with front-end systems.

Key topics covered:

- **Back-end Architectures**: Monolithic and RESTful architectures.
- **Node.js & Express**:
  - Setting up and configuring back-end environments.
  - Middleware, routing, and API creation.
- **Database Integration**:
  - SQL (PostgreSQL) and NoSQL (MongoDB): Queries, indexing, and optimization.
  - Database schema design and implementation.
- **Security Best Practices**:
  - Authentication (OAuth, JWT) and authorization.
  - Protecting APIs: Input validation, encryption, and securing endpoints.
- **Deployment and Scaling**:
  - Basic cloud deployment using platforms like AWS or Heroku.
  - CI/CD pipelines for back-end systems.

Learning goals and qualifications:

Upon successful completion of this module, students will be able to:

8. Design and implement secure APIs using Node.js and Express. *(4)*
9. Integrate SQL and NoSQL databases into back-end systems. *(4)*
10. Apply security best practices to protect back-end systems. *(3)*
11. Deploy and manage scalable back-end solutions in production environments. *(3)*
12. Collaborate effectively on back-end projects using version control and CI/CD pipelines. *(4)*

(Numbers reflect Bloom's cognitive skill classification.)

Classification of cognitive skills following Bloom (1956):

1 = *Knowledge*: recalling facts, terms, basic concepts and answers; 2 = *Comprehension*: understanding something; 3 = *Application*: using a general concept to solve problems in a particular situation; 4 = *Analysis*: breaking something down into its parts; 5 = *Synthesis*: creating something new by putting parts of different ideas together to make a whole; 6 = *Evaluation*: judging the value of material or methods.

Core readings:

- Sunyaev, A. (2020). Internet computing: Principles of distributed systems and emerging internet-based technologies. Springer.
- Casciaro, M., & Mammino, L. (2020). Node.js design patterns (3rd ed.). Packt Publishing.
- Ackermann, P. (2023). Full Stack Web Development: The Comprehensive Guide (Rheinwerk Computing). Rheinwerk Computing.

Module ID	Module name	
ASEF7	Project Study 1: Front-End Development of a Web-Based Application	
Type of course Project Study Module	Semester / Rotation Summer Term	Student capacity: 45
Teaching methods: Lecture, seminar, course, project work	Prerequisites for attendance None	Language English
Type of examination (Final Grade Composition) Oral exam (30%), 20 minutes Project Work (40%) Presentation (30%)		**ECTS (+Workload in hrs)** 12 (360 hours, of this 300 group-work hours + 60 instruction hours)
Module coordinator Dr. Lenz Belzner		Weekly workload in hours: 40
Additional instructor(s) involved: Matthias Moosburger, BSc, M.A. Eileen Fürstenau, MSc Yannick Musafiri, BSc		

Syllabus

This project-based module immerses students in the full development lifecycle of creating a dynamic and responsive front-end for a web-based application. This capstone module replicates the workflow and collaboration of a professional software development environment, preparing students for industry roles by requiring them to apply their technical and collaborative skills.

The project is based on a real-world case study, complete with user personas, user story mapping, and a product backlog. Students will work in teams of four to implement the front-end during three agile sprints. Guided by senior tech leads and industry professionals, students will receive continuous feedback through code reviews and sprint reviews, simulating the iterative development process found in professional settings. The project incorporates a wide range of tools and techniques, allowing students to apply skills they developed in prior modules, including React, JavaScript, TypeScript, and CSS. By the end of the course, students will have developed a production-grade, responsive front-end that integrates seamlessly with back-end APIs.

This module not only focuses on technical implementation but also emphasizes agile workflows, collaboration, communication, and code quality, ensuring students are prepared for real-world software development projects. Importantly, this course lays the foundation for the subsequent Project Study 2: Back-End Development of a Web-Based Application, together forming a comprehensive full-stack development experience.

Key topics covered:

- **Project Background and Setup**
  - Introduction to the project, personas, and user story mapping.
  - Setting up the product backlog and prioritizing tasks.
  - Tools setup: Git repositories, collaboration tools, and CI/CD pipelines.
- **Agile Workflow**
  - Execution of three sprints following Scrum principles.
  - Sprint planning, daily stand-ups, sprint reviews, and retrospectives.
  - Managing dependencies and collaborating effectively within teams.
- **Technical Development**
  - **React Framework**: Advanced use of React for component design, state management, and performance optimization.
  - **JavaScript & TypeScript**: Leveraging prior knowledge to enhance interactivity, maintainability, and type safety.
  - **CSS**: Ensuring responsive, visually appealing, and accessible designs.

- Integration of APIs to connect the front-end to back-end systems.

- **Code Quality and Feedback**
  - Implementation of advanced code quality practices, including code reviews.
  - Participation in peer reviews and receiving feedback from industry professionals.
  - Addressing accessibility, performance, and maintainability concerns.
- **Presentation and Delivery**
  - Preparing and presenting sprint results to stakeholders.
  - Incorporating feedback into continuous improvement.

Learning goals and qualifications in this module students learn to:

Upon successful completion of this module, students will:

Technical Skills:

1. Design and implement dynamic, responsive front-end interfaces using React, JavaScript, TypeScript, and CSS. *(3)*
2. Integrate front-end components with APIs for seamless data flow and functionality. *(3)*
3. Apply advanced code quality practices to produce maintainable, scalable, and efficient code. *(5)*

Professional Competence:

1. Execute agile workflows, including sprint planning, retrospectives, and iterative delivery. *(3)*
2. Collaborate effectively within a team to deliver high-quality outputs on time. *(5)*
3. Receive and incorporate feedback from industry professionals, simulating real-world development. *(6)*

Communication and Reflection:

1. Present project progress and solutions to both technical and non-technical audiences. *(4)*
2. Reflect on team and individual performance, identifying areas for growth. *(6)*

(Numbers reflect Bloom's cognitive skill classification.)

Classification of cognitive skills following Bloom (1956):

1 = *Knowledge*: recalling facts, terms, basic concepts and answers; 2 = *Comprehension*: understanding something; 3 = *Application*: using a general concept to solve problems in a particular situation; 4 = *Analysis*: breaking something down into its parts; 5 = *Synthesis*: creating something new by putting parts of different ideas together to make a whole; 6 = *Evaluation*: judging the value of material or methods.

Core readings:

- Martin, R. C. (2011). The clean coder: a code of conduct for professional programmers. Pearson Education.
- Ackermann, P. (2023). Full Stack Web Development: The Comprehensive Guide (Rheinwerk Computing). Rheinwerk Computing.

Module ID	Module name		
ASEF8	Project Study 2: Back-End Development of a Web-Based Application		
Type of course Project Study Module		Semester / Rotation Summer Term	Student capacity: 45
Teaching methods: Lecture, seminar, course, project work		Prerequisites for attendance None	Language English
Type of examination (Final Grade Composition) Oral exam (30%), 20 min Project Work (40%) Presentation (30%)			**ECTS (+Workload in hrs)** 12 (360 hours, of this 300 group-work hours + 60 instruction hours)
Module coordinator Dr. Lenz Belzner			Weekly workload in hours: 40
Additional instructor(s) involved: Matthias Moosburger, BSc, M.A. Eileen Fürstenau, MSc Yannick Musafiri, BSc			

Syllabus

This project-based module builds directly upon the work completed in the Front-End Development Capstone, focusing on developing the back-end architecture for the same web-based application. This module challenges students to design and implement scalable, secure, and maintainable back-end systems, integrating seamlessly with the front-end they previously developed.

Students will work in the same teams of four, executing three additional sprints to complete the application's back-end. With a strong emphasis on real-world practices, students will develop RESTful APIs, implement authentication and authorization, and manage databases. The project replicates a professional software development environment, featuring code reviews, sprint reviews, and feedback sessions conducted by senior tech leads and software developers from partner companies. This level of professional interaction ensures that students gain industry-ready experience.

Together with the Front-End Development Capstone, this module provides students with a comprehensive full-stack development experience, preparing them for the technical and collaborative challenges of software engineering roles in the industry.

Key topics covered:

- **Project Continuation**
  - Review of the front-end progress and defining back-end requirements.
  - Refinement of user stories for back-end functionality.
- **Agile Workflow**
  - Execution of three sprints focusing on back-end development.
  - Collaboration with front-end teams to ensure seamless integration.
- **Back-End Development**
  - **RESTful APIs**: Designing and developing APIs for data handling and integration.
  - **Authentication and Authorization**: Implementing secure user authentication and role-based access control.
  - **Database Management**: Designing, implementing, and optimizing relational databases.
  - **Scalability and Performance**: Ensuring the back-end can handle growth and high demand.
- **Code Quality and Feedback**
  - Adopting advanced code quality standards and receiving peer and professional reviews.
  - Applying feedback from code reviews to refine solutions.

- **Testing and Deployment**
    - Developing test cases to ensure back-end functionality and reliability.
    - Deploying the full-stack application to a production environment.
- **Presentation and Delivery**
    - Preparing a final presentation showcasing the integrated full-stack application.
    - Reflecting on the end-to-end development process.

Learning goals and qualifications in this module students learn to:

Upon successful completion of this module, students will:

Technical Skills:

1. Design and implement scalable and secure back-end systems, including RESTful APIs and databases. (3)
2. Integrate back-end components with front-end systems to create a cohesive full-stack application. (3)
3. Apply code quality and testing practices to ensure reliability and maintainability. (5)

Professional Competence:

4. Execute agile workflows during back-end development sprints. (3)
5. Collaborate effectively with team members and industry professionals, simulating real-world workflows. (5)
6. Address feedback from code reviews and stakeholders to improve solutions. (6)

Communication and Reflection:

7. Present the final integrated application and reflect on the development process. (4)
8. Demonstrate an understanding of end-to-end software development processes, from planning to deployment. (6)

(Numbers reflect Bloom's cognitive skill classification.)

Classification of cognitive skills following Bloom (1956):

1 = *Knowledge*: recalling facts, terms, basic concepts and answers; 2 = *Comprehension*: understanding something; 3 = *Application*: using a general concept to solve problems in a particular situation; 4 = *Analysis*: breaking something down into its parts; 5 = *Synthesis*: creating something new by putting parts of different ideas together to make a whole; 6 = *Evaluation*: judging the value of material or methods.

Core readings:

- Casciaro, M., & Mammino, L. (2020). Node.js design patterns (3rd ed.). Packt Publishing.
- Ackermann, P. (2023). Full Stack Web Development: The Comprehensive Guide (Rheinwerk Computing). Rheinwerk Computing.