Web Applications – Specialization EASV, DMU Location Sønderborg.

Timing: 4th semester
Scope: 7.5 ECTS

Content:
The purpose of the course is to enable the student to design and develop full-stack web applications using a modern three-tier architecture. The student will gain hands-on experience with Node.js (with Express) and Typescript for the business logic layer, Redis for persistence and in-memory caching, and Flutter for the presentation layer. The course also covers role-based authentication/authorization architecture, state management, clean architecture principles, and selected design patterns. Tools such as Postman and Python scripts will be used for testing and simulating client requests. The course culminates in a multidisciplinary project in collaboration with other study programmes currently IOT and mobile development.

Learning objectives:

Knowledge
The student must have knowledge of:

- Three-tier web application architecture and the role of the business, persistence, and presentation tiers
- Node.js with Express and TypeScript for backend development
- Redis as an in-memory dictionary database and caching solution.
- Redis in an event based micro service architecture (Pub/Sub pattern)
- Redis streams for managing continuous streams of data (e.g. live data read from an IOT device)
- Flutter and Dart for cross-platform frontend development.
- RESTful API design, including CRUD operations and endpoint implementation
- Authentication methods, including sessions, JWT tokens, two-factor authentication, and OAuth2-based social logins
- State management patterns, in particular the State pattern.
- Clean Architecture and SOLID principles in backend development
- Common design patterns such as Chain of Responsibility and Decorator
- Integration of backend and frontend components.
- Role-based authentication with guards and access control.
- Declarative programming in Flutter and widget-based UI design
- State management in Flutter (stateless vs. stateful widgets)
- API consumption in frontend applications.

Skills
The student can:

- Set up and configure Node.js + Express projects using TypeScript
- Implement RESTful CRUD operations with in-memory caching using Redis
- Create secure authentication flows with sessions, JWT tokens, and OAuth2 integrations
- Apply micro service architecture principles in backend systems

- Implement the State pattern, in web applications
- Apply Clean Architecture and SOLID principles in backend codebases
- Implement role-based access control in backend services
- Set up and configure Redis in local or containerized environments
- Implement Redis features such as collections, timeouts, Pub/Sub, and streams.
- Build cross-platform frontends using Flutter and Dart
- Develop UI components using stateless and stateful widgets
- Integrate frontend applications with backend APIs and display retrieved data
- Use Postman and Python scripts for API testing and client emulation

Competences
The student can:

- Design, implement, and integrate all three tiers of a web application using a modern tech stack
- Select and apply suitable design patterns for backend and frontend problems
- Design secure and scalable architectures for web applications
- Collaborate across disciplines to deliver a functional web application for real-world use cases
- Evaluate trade-offs between architectural choices, authentication methods, and persistence solutions

The examination:
Internal oral exam of 30 minutes duration based on a project and questions that can be prepared in advance.

Assessment:
7-point grading scale. Grading is based on an overall assessment of the oral presentation and examination in all the involved study programmes, but not on the submitted project.

Teacher: Tommy Haugaard

IoT and Embedded software

Timing: 2. year of study **Scope: 7,5** ECTS

Content: The purpose of the course is to introduce the student to the importance of IoT in society and the current components of typical IoT devices. They will gain knowledge and understanding of the fundamental characteristics of IoT and embedded systems. Furthermore, students will learn about the Python programming language, embedded software, electronic components, and sensors. They will also acquire practical experience in IoT system design and architecture, as well as the integration of IoT in a distributed system.

Learning objectives:

Knowledge
The student must have knowledge of:

- Software development for IoT devices, sensors, and the challenges involved in developing such solutions
- The language Python
- IoT network solutions and standards for short- and long-range communication
- Basic electronics
- Existing development tools and libraries used to implement IoT solutions
- Protocols and techniques that support IoT development

Skills
The student can:

- Design and implement solutions where IoT devices are used to feed data into a system
- Develop solution that interface with sensors and peripheral devices
- Apply IoT network protocols to transmit data from IoT devices to central servers
- Use communication protocols for peripheral devices

Competencies
The student is:

- Able to develop a proof-of-concept IoT solution that incorporates aspects of IoT programming and IoT network protocols

The examination:
Internal oral exam of 30 minutes duration based on a project (sammen med Web og App valgfaget)

Assessment:
7-point grading scale. Grading is based on an overall assessment of the oral presentation and examination, but not on the submitted project.