Module	Software Engineering and Design Patterns
Code	
Degree Program	Master of Science in Life Sciences (MSLS)
ECTS Credits	3
Workload	90h: 30h Lecture (2 Lessons/W), 15h Exercise (1 Lessons/W), 45h Self-study/project (2 Lessons/W)
Module Coordinators	Dr. Ahmad Aghaebrahimian **Email** agha@zhaw.ch **Address** ZHAW, LSFM, Schloss 1, 8820 Wädenswil
Lecturers	Ahmad Aghaebrahimian
Entry Requirements	● **Required:** Programming, Data Structures, and Algorithms module or equivalent ● Any Machine Learning, Neural Network, or Deep learning module is a nice-to-have background, although none is mandatory.
Learning Outcomes and Competences	● Proficiency in software engineering principles including software development life cycle models, planning, and requirements engineering. ● Understanding Object-Oriented Analysis and Design (OOAD) and some of its principles such as system design using UML, encapsulation, and inheritance ● Basic knowledge of design patterns including creational, structural, and behavioral. ● Understanding various software architectural designs (e.g., layered architecture, monolithic, microservices). ● Basic understanding of GUI and web applications
Module Content	The course encompasses four chapters in software engineering, including basic principles of software development (requirement engineering, development models, testing), system design (OO design), design patterns (patterns and architecture), and best practices (machine learning as software, web applications, cloud computing). **In the first chapter**, students acquire a basic understanding of software development life cycles, and models including Waterfall, Spiral, Agile, and Scrum methodologies. A more comprehensive introduction to Agile and Scrume methodology will be presented. Meanwhile, they will define a mini-project on which they incrementally apply their learning in each

	session, and present it in the last session. They will then study system requirement analysis by analyzing functional and non-functional requirements. **In the second chapter**, before designing a system in UML, they will be introduced to the basics of object-oriented designs, reviewing abstraction, classes, objects, attributes, methods, Inheritance, and encapsulation. **In the third chapter**, students will be introduced to common design patterns, their importance, and their function in designing modular and scalable software solutions. They will study creational, structural, and behavioral patterns. They will be also introduced to several system design decisions such as monolithic, SOA, Microservice, and Serverless, and evaluate their consistency, availability, and tolerance. Every architecture will exemplified by describing a prominent real-life example. This chapter concludes with a description of the layered software architecture (Persistence, Business, Presentation). **The fourth chapter** introduces several basic practices such as Graphic User Interface, web application design, test-driven development, system maintainability, and machine learning as software. Each session consists of a high-level presentation of all concepts and one core concept which shall be presented in detail with hands-on exercises.
Teaching / Learning Methods	The sessions are in the form of flipped classrooms, meaning the students are required to study the material of each session before the lectures. The lecturer then covers the basics of the material, making sure the general concepts are grasped and initiating the discussion about all elements associated with the architectural and design characteristics of large software systems. When applicable, he ground the content to popular software products such as Stackoerflow, Netflix, Google search engine, etc. In each session, students incorporate what they learned into their own continuously maturing software project. At the end of the semester, they are expected to present their system to other students.
Assessment of Learning Outcome	Written exam 50% (if more than 10 students, otherwise oral) Project 50%
Bibliography	Selected parts (as listed in the syllabus below for each session) of the following resources will be used as study material. All resources are in the open domain.

	A: Python 3 Patterns, Recipes, and Idioms (mirror) **B**: Handbook of Software Engineering Methods **C**: Think Python, 2nd edition **D**: Slides (will be provided)
Language	English