Pleae note: The purpose of the project assignment is to apply the competencies taught in the course and to grow as a team. It is not mainly about having the most complete or „greatest“ solution, but demonstrating your succesful learning progress as a team, demonstrating all team members are equally able to understand and apply the things taught. Your individual performance is evaluated in the written exam. This assignment is about team performance. And only your own work counts. Any kind of plagiarism or using code or documents from the internet or other sources as part of your submitted solution are strongly forbidden and will lead to not passing the exam.

Project case:

A German university wants to set up an AI-based online tutoring platform to support online lewcturers and their students. It should allow to manage courses with the attributes *name*, *description*, *focus areas*, (formerand present) *lecturers* and *information about the students*, who have taken or are taking each course. Lecturers should be able to link related course materials (e.g. slides) via a URL (e.g. from Moodle etc.).

For these courses, the students should be able to ask individual content-related questions via an online chat, which are automatically answered by a generative AI (robot tutor) based on the uploaded data stored for the courses (e.g. using retrieval augmented generation=RAG). For that purpose, in the background a Large Language Model (LLM) needs to be used, either deployed locally using Ollama or using a cloud service.

The tutoring platform needs to be written in Java, using Spring Boot, Spring Data/JPA, JDBC and DerbyDB as the database. The frontend should be a „server-less“ Single Page Application (SPA) written in JavaScript or TypeScript (when choosing Angular), calling backend web services written in Java using Spring. These technologies are define by the customer and not negotiable. The application needs to be delivered in containerized form.

For the assignment, the implementation shall be developed and submitted using the ready-made Docker-dev environment for VS Code and using Maven, as provided by the lecturer.

For the assignment, the concepts and documents (functional specification, architectural design, etc.) you submit need to cover the full functional scope as described, the working implementation shall at minimum implement the use cases that a lecturer can create/update/delete courses.

Project setup:

- Use an agile (Scrum-like) approach with 3 sprints of 3 weeks duration each:

11.11.2024 – 02.12.2024
02.12.2024 – 23.12.2024
23.12.2024 – 24.01.2025 (includes holiday break)

After the first sprint, the working increment of the product might be a paper prototype or click dummy, but after sprint 2 and 3 it must be working code including backend functionality.

Before starting with the first sprint, fill up your product backlog with some initial set of user stories derived from the above case description. User stories may be added later as well by the product owner.

The „Definition of Done“ (DoD) for each sprint requires that everything related to the user stories you worked on during that sprint is properly reflected in the requirements specifictaion and

architectual document in terms of UML use cases, descriptions and class diagrams including explanations.

The lecturer serves as the product owner. Ask him for clarifying requirements and also let him test the result of each sprint.

After each sprint, the following items need to be submitted in Moodle:

- The overall product backlog
- The sprint backlog you worked on during this sprint including your story point estimates for each story
- The tasks in which you broke down the user stories during the sprint backlog
- The team members assigmed to each task and the hours each team member worked on the respective task
– The outcome of your retrospective session for that sprint, documented in a Powerpoint slide

Each team should agree at least on a weekly scrum meeting (maybe more often), scheduled at always the same time, and denote one team member as the scrum master.

It is mandatory to invite the product owner (lecturer) to the scrum meetings (even though he might not attend always)! Ensure that he can participate online via Zoom.

At the end of the assgnment, the following artefacts need to be submitted for grading (in addition to the sprint docs mentioned above):

- A single document containing the functional specification and technical design of the application. This should have the following structure/table of contents:
  1. Introduction/Motivation
  2. Requirements specifictaion
     1. Non-functional requirements (documented e.g. using Planguage)
     2. Boundary conditions
     3. UML Use Case Diagram
     4. Use Case descriptions for all use cases in the diagram
     5. Conceptional data model described as UML class diagrams
     6. Explanation of the data model in writen text
  3. Technical Design
     1. Software architecture
        1. Functional view of the software architecture (diagram + explanation)
        2. Technical UML class diagram
        3. Explanation of all design decisions made and their derivation from the actual requirements from section 2
        4. Explanation of the naming conventions used in the code
     2. Deployement setup (Project structure, Docker container structure etc.) explained

  Please note this a written text document having some red line in, and not just a collection of items!

- The complete source code of the implemented part of the application, including Docker files etc. to make it run. Only one single source base per team is accepted. Submission shall be performed in asingle zip archive. In each source file, the responsible authors from the team with their respective estimated percentage of contributed code must be listed in a comment.

- The artifcats requied by Prof. Kohlhase (if any).