# Step1: identifying system operations

* Authentication
  + logIn() : logging in with netID and password
  + getStudentInfo():
  + getTeacherInfo():
* User
  + getNetID()
  + getRole()
* Student
  + enroll() : enrolling in a course
  + getGrade() : get grade for a specific assessment (attempt)
  + takeExam() : Create a new StudentExam
  + getExam() : Read questions and answers of an already taken StudentExam
  + getExtraTime() : Return the extra time a student is eligible for
  + selectAnswer(ExamQuestionID, AnswerID) :
  + deselectAnswer(ExamQuestionID, AnswerID) :
* Teacher
  + createCourse() : Create a new course
  + deleteCourse(): delete a course
  + createTopic(): Create a new topic
  + deleteTopic(): Delete a topic
  + createExam() : Schedule a new (general) Exam (constraints: >= 1 topic, >= 10 questions in all topics combined)
  + deleteExam(): delete an existing exam
  + createQuestion(course, topic) : Add a new question to the pool (constraints: 4 questions, >= 1 correct question)
  + deleteQuestion() : Delete a question from the pool (constraint: checks if there are still at least 10 questions in all topics of the course combined)
  + updateQuestion() : changes a question in the pool
  + getAverageGrade(): get average grade for a specific exam (average over 3 or less attempts)
  + getWorstMadeQuestions(): get the 2 worst made questions per exam.
* Course
  + getID()
  + getTopics() :
  + getTeacher() :
  + getYear() :
  + getName() :
  + getStudents() :
* Topic
  + getID()
  + getName() :
  + getAllQuestions() :
  + getRandomQuestion() :
* Questions
  + getID() :
  + getTitle() :
  + getDescription() :
  + getAnswers() :
  + equals() :
* ExamQuestions
  + getID() :
  + getStudentExam() :
  + getQuestion() :
  + isCorrect() :
* Answers
  + getID():
  + getQuestion()
  + getOrder():
  + getDescription():
  + isCorrect(): true iff answer is correct
* StudentAnswers
  + getExamQuestion() :
  + getAnswer() : Return answer ID
  + getDescription() :
  + getOrder() :
  + isSelected() : True iff selected by student
  + select() :
  + deselect()
  + isCorrect() :
* StudentExam
  + getID():
  + getExam(): return the exam
  + getUser(): return the netID of the student who took the exam
  + getCorrectQuestions(): return the amount of correct questions
  + getGrade(): initialized to 0, iff 0 calculate the grade with getCorrectQuestions() / 10 \* 9 + 1 else return grade.
  + isExtraTime(): true iff student is eligible for extra time.
* Exam
  + getID():
  + getCourse():

# Step 2: identify services

* Authentication service: service that handles the logging in and authenticating users
  + Storage:
    - netID
    - pwd
  + Calls to other microservices:
    - Authorization Service if netID and pwd are valid
* Authorization: checks if a microservice is accessible to the client and forwards it.
  + Storage:
    - NetID
    - Role
  + Calls to other microservices:
    - Student service, if role is Student (int-representation: 0)
    - Teacher service, if role is Teacher (int-representation: 1)
    - Exam service
    - Course service
* Student service: service that allows students to enroll in courses, see their grades, retrieve exams, get extra time if possible, select answers and deselect answers.
  + Storage
    - netID
    - extraTime
  + Calls to other microservices:
    - examService: taking, seeing answers to already made exams, getting grades.
    - courseService: to enroll in a certain course.
* Teacher service: allows teachers to create and delete courses, topics, exams and questions. Also to update questions. Furthermore to get the average grade and get the two worst made questions.
  + Storage
    - netID
  + Calls to other microservices:
    - examService: creating exams and seeing the grades of students.
    - courseService: crud operations on courses and topics.
* Course service: provides the course-ID, topics, teacher year, name and students.
  + Storage
    - Course (entity)
    - Topic (entity)
* Exam service that allows crud operations on exams, associated questions and answers by a teacher. It also handles the whole process of taking an exam, like getting randomized questions, the student selecting the answers and calculating the grade.
* Storage:
  + Exam (entity)
  + Questions (entity)
  + Answers (entity)
  + StudentExam (entity)
  + StudentAnswers (entity)
  + ExamQuestions (entity)
* Calls to other microservices:
  + CourseService when creating a StudentExam and Exam.

# Step 3: define service APIs and collaborations

