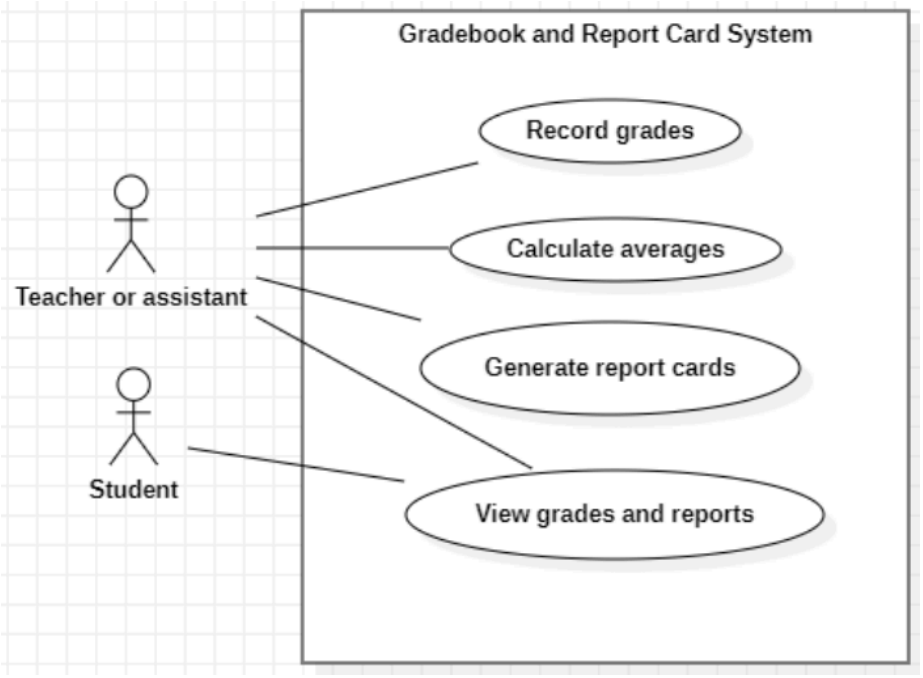
1. Project Overview

- Provide a brief description of the project you will be working on. This should include:
    - **Project Title**: Teacher's Gradebook and Report Card System
    - **Problem Summary**: We intend to make a lightweight, intuitive and simple to use gradebook and student tracking system with report card functionality
    - **Intended Audience/Users**: Teachers and teaching assistants, possibly students in expanded functionality
    - **Main Features/Components**:
    - Ability to make a profile for a teacher
    - Add courses and student profiles to those courses
    - Add tasks, homework etc. to courses
    - Customized grading
    - Record and calculate grades for students.
    - Generate report cards based on student performance.
    - Option to calculate weighted averages for assignments, exams, and projects.
    - View and compare student grades
    - Export grades



Basic use case diagram of the product.

2. Project Objectives

- Core functionality: possible to grade courses and calculate grades for students
- No-hassle database usage, ie. end user doesn't have to configure database

- Easy-to-use graphical user interface
- Database integration for permanence

3. Scope and Deliverables

- Gradebook program
- GitHub including readme
- Documentation including database setup

4. Project Timeline

Sprint 1 15.1. - 29.1.

Project management setup, requirement gathering, beginning of design

Sprint 1: Goal: Clear plan of action for 4 sprints, ready technology stack and product requirements clearly defined

Sprint 2 29.1. - 12.2.

Main design and beginning of development with mockups and simple versions

Sprint 2 Goal: Design of final software finalized, including which components will be needed and what the user interface will look like

Sprint 3 12.2. - 5.3.

Main development and iterative testing while developing features

Sprint 3 Goal: Working software with most features working

Sprint 4 5.3. - 19.3.

Testing and quality control phase, adding vanity features if there is extra time

Sprint 4 Goal: Finalized software including database integration for permanence. Documentation ready.

5. Resource Allocation

- Identify the resources required to complete the project. This could include:
    - **Team Members and Roles**:
        - Jan Falck: Planning, visual design, coding
        - Zachris Zweygberg: planning, coding, testing
        - Sampo Klaavo: coding, testing
        - Artem Danska: planning, other

- **Software, Hardware, and Tools**: Trello and GitHub for project management and version control, Figma for visual design, Java with Maven for main program management and unit testing, JavaFX for graphical user interface, MariaDB and JPA+Hibernate for permanence
- **External Resources or Support**: Likely not needed for this project

6. Risk Management

- Risk 1: Bugs
  - **Description:** Bugs can appear during the development, as with every project.
  - **Likelihood:** High
  - **Impact:** Medium
  - **Mitigation Strategies:** Use testing throughout the development and fix bugs as early as possible.

- Risk 2: First time use of Docker & Jenkins
  - **Description:** Our group is using Docker & Jenkins for the first time, which can lead to unexpected errors.
  - **Likelihood:** Medium
  - **Impact:** Medium
  - **Mitigation Strategies:** Learn about Docker & Jenkins as early as possible through official instructions, in class and tutorials.

- Risk 3: Time and procrastination
  - **Description:** The project and sprints may take longer than expected and will have to be rushed on the last day, which may result in unfinished or buggy features.
  - **Likelihood:** Low
  - **Impact:** High
  - **Mitigation Strategies:** Communication is a big priority. Regular meetings would help a lot to decide who does what and when, etc.

7. Testing and Quality Assurance

- Types of testing:
  - **Unit Testing;** Used for calculations and averages.
  - **Integration Testing:** Ensures that different components like GUI and DB work together correctly.
  - **User Acceptance Testing:** Basic usability testing to ensure that the application is easy to use and all the features are easily found.

- Criteria for Success:
  - The application runs correctly without any bugs

    - Grades are calculated correctly
    - User interface is clear and easy to use
    - Data in the database is stored correctly, retrieved easily and secured.

- Tools and Frameworks:
    - JUnit

8. Documentation and Reporting

- Documentation will include user manual, software and database design notes with UML visualization and guide on database integration(MariaDB)
- Team will weekly discuss progress and plan next steps with themselves, sprintly meetings with product owner