Assessment model (Rubrics) Project Databases: Assignment 3

Conditional requirements	**The group has submitted <u>one</u> ZIP file containing the Visual Studio solution and a text file with the division of tasks (who did which part of the assignment).**

Variant A: Manage students

Individual contribution (80%)		**Criterion**	**Points**
The student should have developed part of the application for managing students.	Unsatisfactory	There is no list of students (from the database).	0
	Weak	All students are displayed in a structured way (with headings for all fields except ids), ordered by last name.	40
	Satisfactory	All students are displayed in a structured way (with headings for all fields except ids), ordered by last name. Users have the option to add and remove students[1,2,3].	60
	Good	All students are displayed in a structured way (with headings for all fields except ids), ordered by last name. Users have the option to add and remove students. Users have the option to change students[1,2,3].	80
	Excellent	All students are displayed in a structured way (with headings for all fields except ids), ordered by last name. Users have the option to add and remove students. Users have the option to change students. The program conducts validation, which makes it impossible to add students that have already been added (with the same student number).	100

[1] *after adding, removing and changing students, the list of students is refreshed (reflecting the changes made)*

[2] *the user does not have to use a database id for adding, removing and changing students*

[3] *the user has access to adding, removing and changing students by using links/buttons (user does not have to type in URLs manually)*

Variant B: Managing lecturers

Individual contribution (80%)		**Criterion**	**Points**
The student should have developed part of the application for managing lecturers.	Unsatisfactory	There is no list of lecturers (from the database).	0
	Weak	All lecturers are displayed in a structured way (with headings for all fields except ids), ordered by last name.	40
	Satisfactory	All lecturers are displayed in a structured way (with headings for all fields except ids), ordered by last name. Users have the option to add and remove lecturers[1,2,3].	60
	Good	All lecturers are displayed in a structured way (with headings for all fields except ids), ordered by last name. Users have the option to add and remove lecturers. Users have the option to change lecturers[1,2,3].	80
	Excellent	All lecturers are displayed in a structured way (with headings for all fields except ids), ordered by last name. Users have the option to add and remove lecturers. Users have the option to change lecturers. The program conducts validation, which makes it impossible to add lecturers that have already been added (with the same name).	100

[1] *after adding, removing and changing lecturers, the list of lecturers is refreshed (reflecting the changes made)*

[2] *the user does not have to use a database id for adding, removing and changing lecturers*

[3] *the user has access to adding, removing and changing lecturers by using links/buttons (user does not have to type in URLs manually)*

Variant C: Managing rooms

Individual contribution (80%)		**Criterion**	**Points**
The student should have developed part of the application for managing rooms.	Unsatisfactory	There is no list of rooms (from the database).	0
	Weak	All rooms are displayed in a structured way (with headings for all fields except ids), ordered by room number.	40
	Satisfactory	All rooms are displayed in a structured way (with headings for all fields except ids), ordered by room number. Users have the option to add and remove rooms[1,2,3].	60
	Good	All rooms are displayed in a structured way (with headings for all fields except ids), ordered by room number. Users have the option to add and remove rooms. Users have the option to change rooms[1,2,3].	80
	Excellent	All rooms are displayed in a structured way (with headings for all fields except ids), ordered by room number. Users have the option to add and remove rooms. Users have the option to change rooms. The program conducts validation, which makes it impossible to add rooms that have already been added (with the same number).	100

[1] *after adding, removing and changing rooms, the list of rooms is refreshed (reflecting the changes made)*

[2] *the user does not have to use a database id for adding, removing and changing rooms*

[3] *the user has access to adding, removing and changing rooms by using links/buttons (user does not have to type in URLs manually)*

Variant D: Managing activities

Individual contribution (80%)		**Criterion**	**Points**
The student should have developed part of the application for managing activities.	Unsatisfactory	There is no list of activities (from the database).	0
	Weak	All activities are displayed in a structured way (with headings for all fields except ids), ordered by date/time.	40
	Satisfactory	All activities are displayed in a structured way (with headings for all fields except ids), ordered by date/time. Users have the option to add and remove activities[1,2,3].	60
	Good	All activities are displayed in a structured way (with headings for all fields except ids), ordered by date/time. Users have the option to add and remove activities. Users have the option to change activities[1,2,3].	80
	Excellent	All activities are displayed in a structured way (with headings for all fields except ids), ordered by date/time. Users have the option to add and remove activities. Users have the option to change activities. The program conducts validation, which makes it impossible to add activities that have already been added (with the same name).	100

[1] *after adding, removing and changing activities, the list of activities is refreshed (reflecting the changes made)*

[2] *the user does not have to use a database id for adding, removing and changing activities*

[3] *the user has access to adding, removing and changing activities by using links/buttons (user does not have to type in URLs manually)*

Group contribution (20%)	*20 points for every criterion met*

Criterion	**Points**
The standard C# coding convention is used. *(lowerCamelCase for local variables and method parameters, UpperCamelCase for classes, methods, properties and constants)* Classes/methods/variables/constants have meaningful names. Enumerations are used for limited option values. Constants are used instead of hardcoded values.	20
Methods are no longer than 10 lines of code. Method return values and method parameters are used (instead of using globals).	20
Object Orientation is correctly applied (good OO structures, using interfaces, passing objects instead of separate parameters). Encapsulation is applied correctly (public vs private).	20
SQL code is only used in repositories. SQL injection is prevented. Database primary keys are auto-generated when inserting records.	20
All exceptions are handled (preventing the application to crash). Error/exception messages are clearly displayed to the user.	20