Assessment model (Rubrics) Project Databases: Assignment 4

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: Process drink orders

Individual contribution (80%)		**Criterion**	**Points**
The student should have developed part of the application for processing drink orders.	Unsatisfactory	There is no list of students or no list of drinks (from the database).	0
	Weak	All students and drinks are displayed in a structured way (with headings for all fields except ids).	40
	Satisfactory	All students and drinks are displayed in a structured way (with headings for all fields except ids). Users have the option to add a drink order[1,2] (storing the selected student, the selected drink, and a count to indicate how many drinks).	60
	Good	All students and drinks are displayed in a structured way (with headings for all fields except ids). Users have the option to add a drink order[1,2] (storing the selected student, the selected drink, and a count to indicate how many drinks). The count for the selected drink is decreased correctly.	80
	Excellent	All students and drinks are displayed in a structured way (with headings for all fields except ids). Users have the option to add a drink order[1,2] (storing the selected student, the selected drink, and a count to indicate how many drinks). The count for the selected drink is decreased correctly. A confirmation of the processed order is displayed (showing how many drinks where sold to which student).	100

[1] *after adding a drink order, the user can immediately add another drink order*

[2] *the user does not have to use a database id for adding a drink order*

Variant B: Manage activity supervisors

Individual contribution (80%)		**Criterion**	**Points**
The student should have developed part of the application for managing activity supervisors.	Unsatisfactory	There is no list of activities or no list of supervisors (from the database).	0
	Weak	All activities are displayed in a structured way (with headings for all fields except ids). After selecting an activity all supervisors for that activity are displayed in a structured way.	40
	Satisfactory	All activities are displayed in a structured way (with headings for all fields except ids). After selecting an activity all supervisors for that activity are displayed in a structured way. Users have the option to remove supervisors from the selected activity[1,2].	60
	Good	All activities are displayed in a structured way (with headings for all fields except ids). After selecting an activity all supervisors for that activity are displayed in a structured way. Users have the option to remove supervisors from the selected activity[1,2]. Users have the option to add supervisors to the selected activity[1,2].	80
	Excellent	All activities are displayed in a structured way (with headings for all fields except ids). After selecting an activity all supervisors for that activity are displayed in a structured way. Users have the option to remove supervisors from the selected activity. Users have the option to add supervisors to the selected activity[1,2]. A confirmation is displayed (showing which lecturer was added or removed as supervisor).	100

[1] *after adding and removing supervisors for an activity, the list of supervisors is refreshed (reflecting the changes made)*

[2] *the user does not have to use a database id for adding and removing activity supervisors*

Variant C: Manage activity participants

Individual contribution (80%)		**Criterion**	**Points**
The student should have developed part of the application for managing activity participants.	Unsatisfactory	There is no list of activities or no list of participants (from the database).	0
	Weak	All activities are displayed in a structured way (with headings for all fields except ids). After selecting an activity all participants for that activity are displayed in a structured way.	40
	Satisfactory	All activities are displayed in a structured way (with headings for all fields except ids). After selecting an activity all participants for that activity are displayed in a structured way. Users have the option to remove participants from the selected activity[1,2].	60
	Good	All activities are displayed in a structured way (with headings for all fields except ids). After selecting an activity all participants for that activity are displayed in a structured way. Users have the option to remove participants from the selected activity[1,2]. Users have the option to add participants to the selected activity[1,2].	80
	Excellent	All activities are displayed in a structured way (with headings for all fields except ids). After selecting an activity all participants for that activity are displayed in a structured way. Users have the option to remove participants from the selected activity[1,2]. Users have the option to add participants to the selected activity[1,2]. A confirmation is displayed (showing which student was added or removed as participant).	100

[1] *after adding and removing participants for an activity, the list of participants is refreshed (reflecting the changes made)*

[2] *the user does not have to use a database id for adding and removing activity participants*

Variant D: Managing dormitory students

Individual contribution (80%)		**Criterion**	**Points**
The student should have developed part of the application for managing dormitory students.	Unsatisfactory	There is no list of rooms or no list of dormitory students (from the database).	0
	Weak	All rooms are displayed in a structured way (with headings for all fields except ids). After selecting a dormitory all students linked to that room are displayed in a structured way.	40
	Satisfactory	All rooms are displayed in a structured way (with headings for all fields except ids). After selecting a dormitory all students linked to that dormitory are displayed in a structured way. Users have the option to remove students from the selected dormitory[1,2].	60
	Good	All rooms are displayed in a structured way (with headings for all fields except ids). After selecting a dormitory all students linked to that dormitory are displayed in a structured way. Users have the option to remove students from the selected dormitory[1,2]. Users have the option to add students to the selected dormitory[1,2].	80
	Excellent	All rooms are displayed in a structured way (with headings for all fields except ids). After selecting a dormitory all students linked to that dormitory are displayed in a structured way. Users have the option to remove students from the selected dormitory[1,2]. Users have the option to add students to the selected dormitory[1,2]. A confirmation is displayed (showing which student was added or removed from/to the dormitory).	100

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

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

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