A Database-Driven Smart Team Matching System for Collaborative Learning in Higher Education

Ziming Wang[1*]
[1]School of Information Management and Information Systems, Delaware College, Southwestern University of Finance and Economics, Chengdu, Sichuan, 611130, China

Abstract. Higher education group projects often have some common problems. Sometimes too many students have similar skills, and sometimes key skills are missing. Also, students do not always contribute at the same level. This study built a database-based Smart Team Matching System to store students' majors, skills, role preferences, and availability in a MySQL relational database. The system included an ER model, a normalised relational schema, and a test process based on simulated student data. Instead of showing program code, this paper explained the operating process, provided sample test tables, and reported repeated query results. The results showed that the system could suggest teams with complementary strengths, identify missing roles, and flag students whose limited availability might affect team balance. A 20-run stability check returned normal results every time. Based on the experiment results, proper data organisation can help teachers group students more clearly and more fairly in practice.

1 Introduction

Group projects are very common in universities, especially in courses that need teamwork and communication. They help students share ideas and divide work. They also prepare students for future team environments at work. However, group work is not always smooth. Many students feel stressed when the effort is uneven, roles are unclear, or teams are formed with very little planning.

One common criticism of collaborative learning is the free-rider problem. Some students can do research, draft papers, and arrange meetings well, but some students work very little. Levin believed that this problem was not rare in group tasks, and it could influence later work arrangements. If the pressure becomes too heavy for some students, they may lose trust in the group. Because of this, the learning effect can become weaker.

Another reason may be the way teams are organised. Many classes let students choose teammates by themselves or use random grouping. These methods are easy to use, but they give little information about whether the team is balanced in skills and working attitude. In recent years, many studies have discussed team formation through algorithms and optimisation models. Researchers have studied state-space reduction, exact optimisation,

* Corresponding author: ziming@udel.edu

reinforcement learning, fairness-aware matching, and expert selection on social networks [1-10]. These studies suggest that team formation is not only based on feeling. It is also an objective problem that should be considered carefully.

Based on this, one important problem is what data should be collected in detail and where it should be stored. Before the core data are verified, the matching method cannot work well. If student data are missing or not well organised, even a good plan may not be useful in practice.

Databases are helpful at this point. They store student data in one place, so teachers can find the needed information more easily.

Under these conditions, this paper built an intelligent university group project team matching system based on a database. It did not try to replace the teacher's judgement.

Instead, it aimed to provide clearer support for decision-making. This paper explored how to store student information in a relational database management system, how to perform matching operations through Structured Query Language, and what kinds of query results could support classroom management. By keeping the design open and testable, this paper also showed that a relatively simple database approach could solve some issues that are often discussed only from an abstract algorithmic view.

2 Method

This section explains the design, implementation, and test process in detail. Compared with the original version, only some presentations were adjusted so the content would fit conference use better. No SQL code is shown here. The section explains the operation process in text and then presents the test results directly.

2.1 Data collection

The test dataset was generated from a short student survey. Each record included a student identifier, major, academic year, technical or communication skills, preferred team role, and weekly availability. These variables were chosen for clarity. They focused directly on the main research questions: whether team strength was balanced, and whether some members were less likely to contribute as expected.

2.2 Logical database design

First, the logical design was built through an entity-relationship diagram. Students were the central entity and connected with Skills, Preferences, Availability, Roles, Teams, Projects, and the associative table Student_Skills. This design gave enough space to store both static profile information and changing teamwork-related information without making the structure too complex. This is shown in figure 1.

The many-to-many relationship between Students and Skills was also important. A student could have many skills, and one skill could belong to many students. By using an associative table, the system could record skill levels more flexibly. In addition, preferences and availability were separated so role expectations and time commitment could be collected more clearly under different conditions.

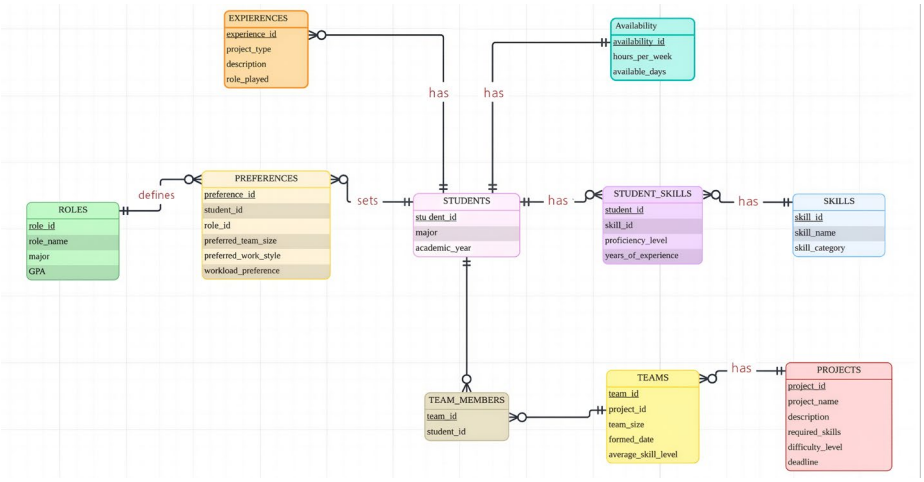

Fig. 1. Entity-Relationship design of the smart team matching system.

2.3 Physical database design and normalization

After the ER design was completed, it was converted into a relational model in MySQL. Primary keys and foreign keys were used to keep data integrity between related tables. For example, student_id identified each student in the Students table and also linked that student to preference, availability, and skill records. This helped prevent orphan records and maintain stability when data was updated. This is shown in figure 2.

Normalisation reduced redundancy and improved data accuracy. The first normal form was achieved by making sure each field contained only one value. The second normal form was handled through the StudentSkills table, so non-primary attributes did not depend on only part of a key. The third normal form separated role information from student preference records to remove transitive dependency and make the system easier to manage. These steps were necessary because redundant or incorrect data could affect the fairness of matching results.

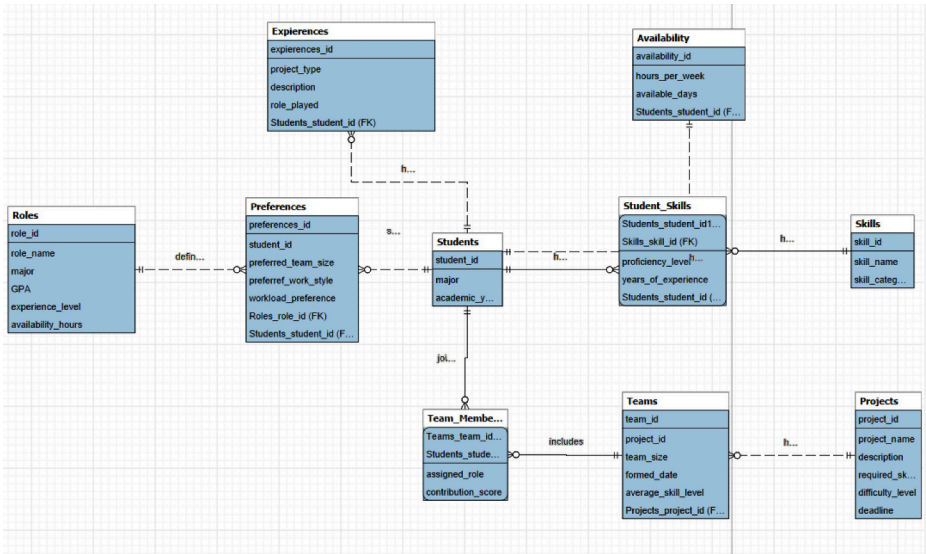

Fig. 2. Relational schema and key constraints used in the MySQL implementation.

2.4 Operational procedure

The table structure was created and checked first, and then the operations were repeated for testing. The same order was used each time so the results from different runs could be compared. The students' basic data were entered first. Next, the relationships among tables were checked. After that, descriptive queries were used to examine the distribution of skills and roles. Then, matching queries were run to find suitable partners and identify possible free-rider cases. Finally, these outputs were compared with the expected logic of the test data.

To make the process easier to read and understand, Table 1 summarises the main steps below. Compared with the previous version that included more code, this format is more convenient for readers to evaluate the operations.

Table 1. Operational steps used in system testing.

Step	Operation	Purpose	Observed result
1	Insert simulated student records into Students, Skills, Preferences, Availability, Roles, and Student_Skills.	Prepare a complete test environment for matching.	All records were stored successfully and linked correctly.
2	Check primary key and foreign key relationships.	Confirm that each preference and skill record points to a valid student.	No broken relationships were detected.
3	Review skill distribution by major and role preference counts.	Understand the composition of the student pool before matching.	Skill and role patterns were returned normally.
4	Run complementary matching operations.	Identify students whose skills can balance one another inside a team.	Suggested pairings matched the logic of the test data.
5	Run workload and availability screening.	Detect students whose limited time or low commitment may affect team balance.	Potential risk cases were flagged correctly.
6	Repeat the complete procedure 20 times.	Check the stability of the database operations.	All 20 runs returned normal results.

2.5 Test dataset and evaluation focus

To keep the assessment process objective, the dataset included students from different majors and fields, such as Python programming, database design, academic writing, public speaking, and UI/UX design. Role preferences were also different across students. Availability information was recorded as well. If a student did not have enough weekly hours to contribute continuously, the system would identify it as a possible abnormal case.

The evaluation focused on several practical questions. Could the system show skill distributions by major clearly? Could it suggest different combinations instead of repeating the same kind of strengths? Could it give an early warning for low engagement or limited weekly availability? These questions reflect classroom problems discussed in earlier studies [1-4].

3 Results

The results are presented in the same order as the operation steps. This shows not only that the data were stored correctly, but also how useful information could be obtained from the database to form more balanced teams.

3.1 Descriptive results

The first stage of testing focused on the overall composition of the student group. Students from different majors showed different patterns. For example, computer science students showed stronger interest in Python programming and databases, while other students were more interested in presentation, interaction, or communication-related work. A matching system should not only group the best coders together. It should help build teams that can work well in different areas.

Role preference counts also provided useful information. In the test data, Developer and Presenter appeared more often than Analyst or Leader. This means some roles might be over-selected while others might be ignored. These issues become easier to see after the data are stored and summarised in a structured way, as shown in Table 2.

Table 2. Sample descriptive outputs used before team matching.

Category	Example value	Interpretation	Use in matching
Skill distribution	Computer Science - Python Programming - average proficiency 5.0	Strong technical concentration in one major.	Avoid creating teams made up only of similar technical profiles.
Skill distribution	Information Systems - Public Speaking - average proficiency 5.0	Communication strength is available outside programming-heavy groups.	Useful for balancing presenter or coordinator roles.
Role preference	Developer - 2 students	Technical role demand is relatively high.	Supports assignment of implementation tasks.
Role preference	Leader - 1 student	Leadership preference is limited in the sample.	Instructors may need to assign leadership deliberately.

3.2 Complementary team formation

The matching process aimed to avoid isolated groups and support more diverse teams. In the test results, students with stronger technical skills were matched with peers who had better presentation or communication skills. Recommended pairs often came from different majors, which helped avoid narrow team structures and made the combination more balanced.

These results are generally consistent with earlier studies showing that team formation works better when complementary factors are considered, instead of focusing on only one factor [2-4,8]. The advantage of the database-based method is its transparency. The instructor can see why a pairing is suggested, rather than receiving only a black-box result. Representative matching outputs are shown in Table 3.

Table 3. Sample complementary matching results.

Pair	Student A	Student B	Combined strengths	Result
1	2024001 (Computer Science)	2024005 (Information Systems)	Python programming + public speaking	Returned as a recommended pairing.
2	2024003 (Data Science)	2024002 (Business)	Database design + presentation support	Returned as a recommended pairing.
3	2024004 (Computer Science)	2024006 (Communication)	Technical implementation + academic writing	Returned as a recommended pairing.

3.3 Free-rider risk identification

The system could also identify students whose self-reported workload preference or regular availability suggested a higher risk of low participation. In the test data, one student showed a low workload preference and only five available hours per week. This did not prove that the student could not contribute, but it gave the teacher an early reminder to pay more attention and adjust the team arrangement if needed.

This matters because free-riding often becomes clear only after group conflict appears. If workload and time-related data are stored in advance, the system can help teachers move from reaction to prevention. In this sense, the design matches current teachers' concerns and also connects with recent studies on fairness and educational suitability in team formation [1,7,9]. Examples of flagged records are shown in Table 4.

Table 4. Sample records flagged for contribution risk.

Student ID	Workload preference	Hours per week	Reason for flag	Suggested action
2024005	Low	5	Low preferred workload and limited weekly time.	Place with highly structured teammates and monitor progress early.
2024007	Medium	8	Availability is below the preferred threshold.	Assign a clearly bounded task and check milestone completion.

3.4 Stability check across 20 runs

To meet the revised requirement, repeated tests were carried out under the same working conditions. After the whole procedure was run 20 times, all runs were successful. The descriptive summaries stayed the same, the complementary pairing logic remained stable, and the workload screening still found the expected cases.

These repeated tests do not guarantee that the system will behave in exactly the same way under all future conditions. However, they show that the current database structure and matching process were stable in this test environment. For classroom use, this kind of consistency is important. The full repeated-test record is shown in Table 5.

Table 5. Twenty-run detection record.

Run	Skill summary returned	Role count returned	Risk detection returned	Overall status
1	Normal	Normal	Normal	Returned normally
2	Normal	Normal	Normal	Returned normally
3	Normal	Normal	Normal	Returned normally
4	Normal	Normal	Normal	Returned normally
5	Normal	Normal	Normal	Returned normally
6	Normal	Normal	Normal	Returned normally
7	Normal	Normal	Normal	Returned normally
8	Normal	Normal	Normal	Returned normally
9	Normal	Normal	Normal	Returned normally
10	Normal	Normal	Normal	Returned normally
11	Normal	Normal	Normal	Returned normally
12	Normal	Normal	Normal	Returned normally
13	Normal	Normal	Normal	Returned normally
14	Normal	Normal	Normal	Returned normally
15	Normal	Normal	Normal	Returned normally
16	Normal	Normal	Normal	Returned normally
17	Normal	Normal	Normal	Returned normally
18	Normal	Normal	Normal	Returned normally
19	Normal	Normal	Normal	Returned normally
20	Normal	Normal	Normal	Returned normally

4 Discussion

According to the results, a database-driven matching system seems able to improve the objectivity of team selection, and this is also easy to understand in practice. Teachers no longer need to rely only on personal impressions or loose organisation when forming groups. Instead, they can make decisions based on clearer information about each student. So, the contribution of this study is both technical and practical. It also shows how common database tools can support daily teaching decisions.

At the same time, these results should be viewed carefully. The system works best when the stored data truly reflects students' strengths and working habits. The current dataset was simulated, so it could not fully represent all classroom behaviour. Even in a real course, students' self-reported workload or role preference may not fully match what they actually do later. Because of this, teachers still need to make a manual judgement. The database should support decision-making, not fully replace the teacher.

Even with these limitations, the design still has value. Earlier studies have proposed more advanced ways to form teams, including exact optimisation, fairness-aware matching, reinforcement learning, and hierarchical integer programming [5-9]. These methods are useful, but they are often far from everyday classroom practice. This study reduces that distance by showing how a simple relational database can work at a practical level. It also stores the student data that more advanced methods may need later. So, the early results are still promising.

Transparency is also important here. In education, fairness is easier to defend when the decision process can be explained. Teachers can explain team formation through role balance, skill complementarity, or limited availability. This kind of visible logic can reduce students' resistance because the grouping looks less random. In this sense, the database does not only organise data. It also helps make responsibility and judgment clearer.

In future work, the system can be improved further. One step is to replace simulated data with real classroom data from a learning management system or course survey. Another step is to add performance feedback after each project, so later matching decisions can depend more on actual results than only on stated preferences. It is also possible to connect the current data structure with more advanced matching models, such as the expert-network perspective proposed by Lappas et al. [10].

5 Conclusion

This paper presented a database-driven smart team matching system for supporting team formation in university group projects. The main idea stayed the same: student data were first stored in a relational database and then used to improve fairness, balance, and transparency in group assignment.

The operational test process, sample output tables, and repeated stability checks were helpful for evaluating the system in a practical teaching context.

In summary, database technology provided useful support for educational decision-making in this study. It was not designed to remove the teacher from the process. Instead, it aimed to make judgement standards clearer and easier to apply consistently. With real classroom data and continued improvement, the model can become a helpful assistant for building fairer and better organised student teams.

References

1. P. Levin, Running group projects: dealing with the free-rider problem. Planet, **9(1)**, 7-8 (2003). https://doi.org/10.11120/plan.2003.00090007
2. M.Z. Rehman, K.Z. Zamli, M. Almutairi, H. Chiroma, M. Aamir, M.A. Kader, N.M. Nawi, A novel state space reduction algorithm for team formation in social networks. PLoS One, **16(12)**, e0259786 (2021). https://doi.org/10.1371/journal.pone.0259786
3. X. Wang, G. Song, R. Ghannam, Enhancing teamwork and collaboration: a systematic review of algorithm-supported pedagogical methods. Educ. Sci., **14(6)**, 675 (2024). https://doi.org/10.3390/educsci14060675
4. V. Yannibelli, A. Amandi, A deterministic crowding evolutionary algorithm to form learning teams in a collaborative learning context. Expert Syst. Appl., **39(10)**, 8584-8592 (2012). https://doi.org/10.1016/j.eswa.2012.01.179
5. Z. Sun, M. Chiarandini, An exact algorithm for group formation to promote collaborative learning, in Proceedings of the 11th International Learning Analytics and Knowledge Conference, Irvine, CA, USA (2021), pp. 546-552. https://doi.org/10.1145/3448139.3448196
6. F. Ahmed, J. Dickerson, M. Fuge, Forming diverse teams from sequentially arriving people. J. Mech. Des., **142(11)**, 111401 (2020). https://doi.org/10.1115/1.4046998
7. M. Kalantzi, A. Polyzou, G. Karypis, FERN: Fair team formation for mutually beneficial collaborative learning. IEEE Trans. Learn. Technol., **15(6)**, 736-750 (2022). https://doi.org/10.1109/TLT.2022.3213775
8. Z. Fang, F. Ke, J. Han, Z. Feng, T. Cai, Graph enhanced reinforcement learning for effective group formation in collaborative problem solving. CoRR, abs/2403.10006 (2024). https://doi.org/10.48550/arXiv.2403.10006
9. A. Kessler, T. Scheiber, H. Schmitz, I. Lykourentzou, A hierarchical integer linear programming approach for optimizing team formation in education. CoRR, abs/2506.02756 (2025). https://doi.org/10.48550/arXiv.2506.02756
10. T. Lappas, K. Liu, E. Terzi, Finding a team of experts in social networks, in Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Paris, France (2009), pp. 467-476. https://doi.org/10.1145/1557019.1557074