1. **Explain the purpose of the algorithms in your app. Clarify whether the chosen algorithms are the only existing algorithms that can perform the intended task. List alternative algorithms and explain your choice.**

We are using a Genetic Algorithm (GA) in our app. From our dataset of participants, the algorithm will create the most optimal groups of 4 based on each participant's preferences. This is done through a predefined number of generations to determine the most 'elite' groupings and enhance the 'eliteness' of the remaining groupings. Our aim is to help NUS students make new friends, and forming groups with similar interests will lead to a higher chance of them befriending each other. GAs are not the only matching algorithm, but it is the most suited for our intended function.

An alternative matching algorithm is the Gale-Shapley algorithm, but that algorithm is targeted for 1-to-1 matching, which makes it unsuitable as we are matching respondents into groups of 4.

Another alternative algorithm is a clustering algorithm via centroid clustering. However, it has limitations which make it unsuitable for our purpose as well. For instance, outliers may be neglected from clusters or strongly skew cluster boundaries. Hence, if a respondent has very different interests from the rest, they might not be grouped or might affect the groupings of everyone else. However, in GAs, outliers will definitely be considered in groupings without necessarily affecting others, as GAs work on pairwise similarity scores for all users to generate groups. GAs may also be better for future scalability if we eventually add more complex categories of interests as it is more efficient than clustering in dealing with high-dimension data.

2. **Explain how a particular algorithm works - what are the inputs and outputs, what is the sequence of steps performed on this input and leading to the output?**

Step 1: Generate Pairwise Similarity Scores
Input: 24 users' responses to sorting survey
Output: 276 pairwise similarity scores

1. Compare two students A and B
2. Compare if their responses to each question match
   a. On preference questions, for every match in conditions → +5
   b. On shared interests questions, for every indicated overlap in interests → +1
3. Calculate the pairwise similarity score for this pair (A + B)

Example of calculating a pairwise similarity score between A and B:

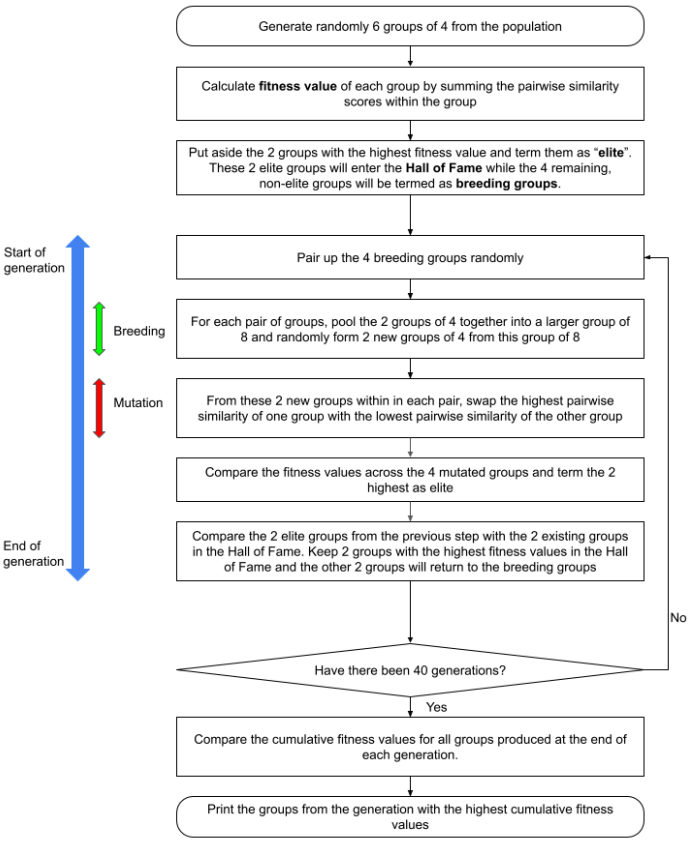
	Same major?	Same batch?	Arts interests	Sports interests	...	
Student A (PPE Y2)	Yes	Yes	Painting, Dance	Hiking, Swimming	...	
Student B (PPE Y2)	No	Yes	Painting, Dance, Theatre	Gymming	...	
Score	+0	+5	+2	+0	...	TOTAL: 8

4. Move on to the next pair, A and C
5. Repeat until all pairs are scored

Example dataset of pairwise similarity scores

	Student A	Student B	Student C	Student D	Student E
Student A	-	-	-	-	-
Student B	8	-	-	-	-
Student C	10	12	-	-	-
Student D	5	3	16	-	-
Student E	2	13	9	5	-

Step 2: Generate optimal groups via Genetic Algorithm
Input: 24 users + 276 pairwise similarity scores
Output: 6 best-matched groups of 4



3. **What is (are) the limitation(s) of this algorithm(s)?**

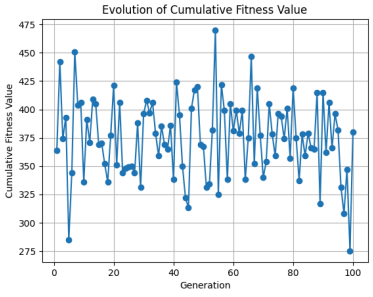
GAs face several limitations. First, its relatively high complexity implies a certain level of domain knowledge is required to apply such algorithms effectively, and hence, there is high error potential. Next, GAs perform via fundamentally random groupings. Thus, there is no guarantee that the final grouping solutions attained are truly the most optimal solutions. Moreover, GAs are quite parameter sensitive and hence, function more efficiently with consistent or standard parameters. By extension, determining a proper balance between exploration (diversification) and exploitation (intensification) can be challenging for GAs. Without adequate exploration, the algorithm may miss potentially better solutions, while excessive exploitation can lead to premature convergence on suboptimal solutions.

4. **Provide an executable code that demonstrates the work of algorithm(s) and tests which you run. You are expected to delve into the technical details of how the algorithm works using what you have learned in class. The level of technical depth should be similar to that at which PageRank was covered in class.**

The executable code for our project uses the Genetic Algorithm. The code was run on Google Colab and successfully produced the output of 6 optimal groups of 4 from a population of 24. The executable code (Appendix) can also be found on the team's GitHub.

Example and Explanation of Executable Code:

Step 0: Generate a dataset containing the pairwise similarity scores (assumed range of 0-20) for 24 individuals.
Step 1: Generate randomly 6 groups of 4 from an initial population of 24 individuals. Calculate the fitness value of each group from the cumulative pairwise similarity scores and define the 2 groups with the highest fitness value as the Hall of Fame groups.
Step 2: Define the 4 remaining groups as breeding groups and split them into 2 breeding pools comprising 2 groups each. Within each pool, mix the 2 groups randomly to form 2 new groups.
Step 3: Swap randomly 1 individual across the 2 new groups within each breeding pool to mutate the groups – this produces 4 mutated groups, each comprising 4 individuals. Define the 2 mutated groups with the highest fitness value as the Elite groups.
Step 4: Compare the fitness values of the Elite groups (Step 3) with the Hall of Fame groups (Step 1). Keep the 2 highest fitness values groups as the Hall of Fame groups and the other 2 will join the non-elite mutated groups (Step 3) for the next generation's breeding and mutation process.
Step 5: Repeat Steps 2 - 4 (which forms 1 generation) until there are 100 generations.
Step 6: Compare the cumulative groups' fitness value score for each generation and print the groups for the generation with the highest cumulative fitness value score. This is the optimal grouping.



For the above code, we can modify it to be more suitable for our project. In Step 3, the mutation process involves randomly swapping 1 individual across 2 groups. However, due to the randomness of such mutation, the cumulative fitness value of each generation (when plotted on a graph) does not stabilise even at 100 generations.

As per our flowchart, we propose that mutation should instead involve swapping a pair from each group with the highest and lowest pairwise similarity scores to make it less random and allow the fitness value to stabilise earlier without as many generations. Hence, our code would adapt Step 3's mutation process and Step 5, limiting it to 40 generations.