Using Genetic Algorithm with our system:

- We defined our gene as follows:

  gene = [easy_reminding, easy_understanding, easy_creativity,

  difficult_reminding, difficult_understanding, difficutl_creativity]
  ex. gene = [2,1,0,1,2,2] (a random number from 0 to 2)

  Then we generate the starting population

  We chose the population size to be 10 and randomly generated the population.

- To use the genetic algorithm, you need a function to optimize for, in our case it's the accuracy of each chapter of the exam generated, calculated using the following formula:

  Accuracy of each chapter (Fitness Function) = (accuracy of easy questions + accuracy of difficult questions + accuracy of reminding questions + accuracy of understanding questions + accuracy of creativity questions + 3 * accuracy of no_of_questions)/ 8

  The 3 here is the weight of the accuracy of the number of questions to guarantee that every time the number of questions required is returned

- We chose the two parents using Tournament Selection method
- We then generate the two offsprings through crossover and mutation process on the two parents to double our population
- The most fit half of the population is kept and the rest is eliminated
- We keep doing the process above until we reach a good enough solution (or hopefully the right solution if available)

How to run the application (Linux cli)

- Create a virtual environment (Command: pipenv shell)
- Install all requirements (Command: pip install -r requirements.txt)
- Make database migrations (Command: python3 manage.py makemigrations)
- Migrate (Command: python3 manage.py migrate)
- Create a superuser that is going to be the user for the teacher in our demonstration (Command: python3 manage.py)
- Populate the database with some Chapters, Courses, Questions, and a Teacher (Command: python3 manage.py loaddata teachers.json chapters.json courses.json questions.json)
- Run the Django server (Command: python3 manage.py runserver)

ERD: