Online Judge LLD DESIGN

‣ Requirement Gathering

- Website should show the problems list.
- Users should be able to view a particular problem.
- Users should be able to submit a solution to the problem.
- The solution is evaluated against the problem's test cases.
- Leaderboard is given to the user which contains Verdict for the last 10 submissions.

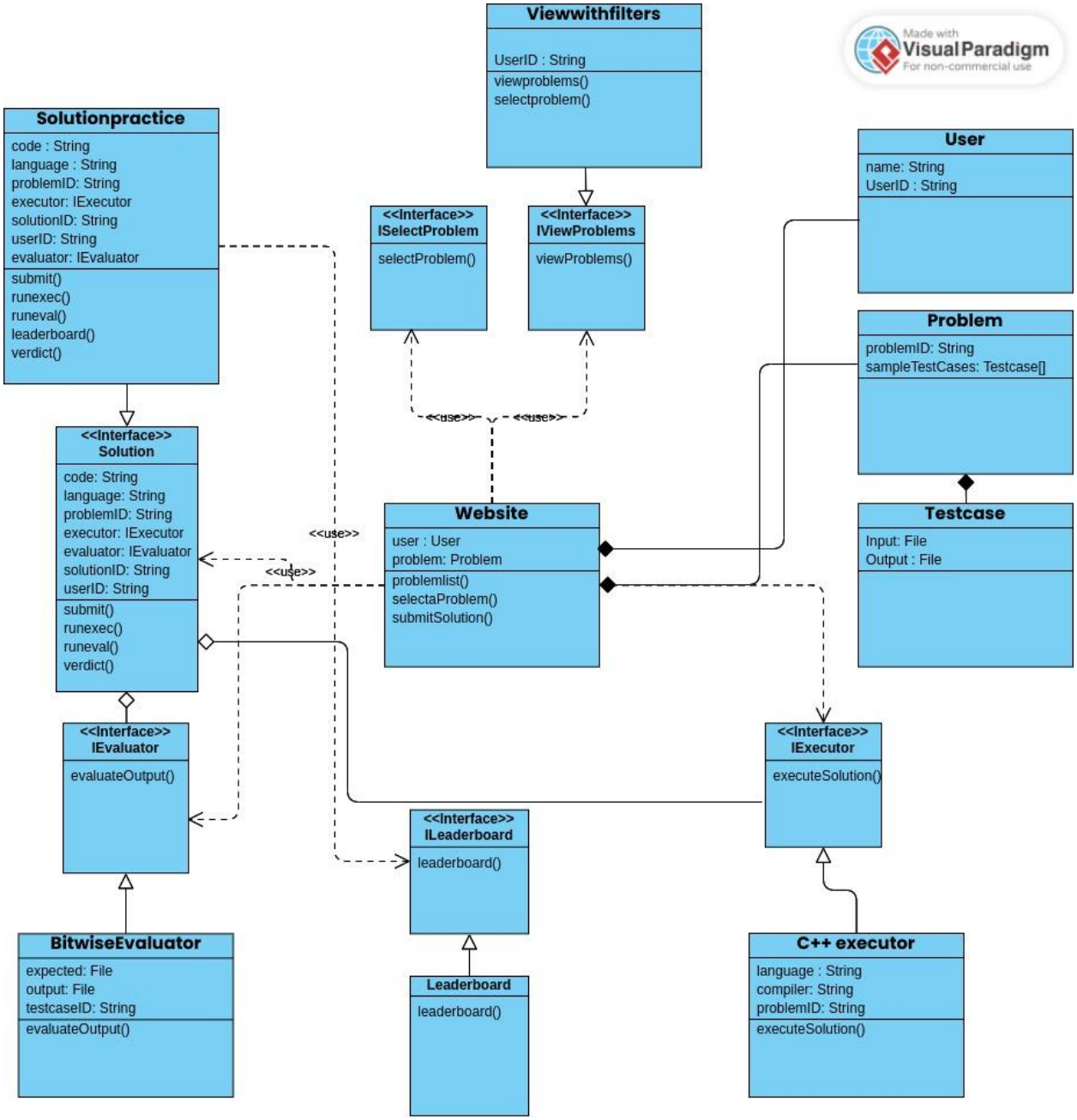
‣Objects formed

Objects	**Methods**
Website	showProblemList
User	viewProblem
Solution	submit
Problem	evaluate
Testcase	showleaderboard
Leaderboard	
verdict	

‣Class Diagram

- The **Website** class is treated as a visible Front end API handler/ DAO.
- The **Solution** class is treated as background DAO.
- IViewProblems, ISolution, IExecutor, IEvaluator, ILeaderboard are created as interfaces to provide extensibility.
  ◦ **IViewProblems** - inducing abstraction helps in having view problems with filters, view problems from a different source at the time of competition to avoid latency.
  ◦ **ISolution** - Inducing abstraction helps in having different implementations for evaluations like serial, parallel evaluations, batch evaluations.
  ◦ **IExecutor** - Inducing abstraction helps in having executors for each language and compiler selection which avoids modification, going forward we can have time and memory constraints as well in executor.
  ◦ **IEvaluator** - Inducing abstraction helps in having different evaluators to compare output files bitwise or soft evaluations which overlook spaces or consider only values returned by a function for output but not stdout.

◦ **ILeaderboard** - Inducing abstraction helps in having different metrics for solutions along with verdict.



‣ NOTES:

- Is the above design open for extension, closed for modification?
- Is the above code Scalable?
  ◦ Will treation the Executor and evaluator as microservices increases the scalability. If so, Does the above design facilitate it?
  ◦ The Solution is created as an interface to allow different run techniques with varying test case batch sizes evaluated each time.
  ◦ The current planned implementation involves executor sending the generated output files back to solution class which sends them to the evaluator to check for AC or Failed, how effective is this? Does sharing files between microservices affect responsiveness?