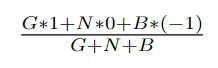
Our project is a web application “Opinion Mining for Social Networking Site”. In this application a user will post his views related to some subject, and other users will view this post and will comment on this post. The System takes comments of various users, based on the opinion, the system will specify whether the posted topic is good, bad, or worst. We use a database of sentiment based keywords along with positivity or negativity weight in the database and then based on these sentiment keywords mined in user comment is ranked.

When we read the description of our project we understood that the mediator design pattern is ideal for the implementation of this project. Indeed, our project contains various classes that constantly need to exchange information with each other. Using a mediator design pattern makes the process of interaction between classes much easier and clearer.

When we decided which pattern we would use, it was time to make a UML that would reflect our intent. We have read the description again and decided on the classes we need:

* **Client** (mediator)
* **User**
* **Post**
* **Database**
* **Word**
* **Stage**
* **Analyzer**

Class Analyzer computes the rating of the word in range [-1; 1]. Using this formula



where G - number of good words, N - number of neutral words, B - number of bad words.

* **OutputFormatter**

Class with contains complete and easy components for console standard I/O

From the moment we made the UML diagram, all we had to do was to implement UML in the form of code. All the necessary comments are written in code. This mediator design pattern has quite a wide application. It is used in systems where the interaction between modules can be very complex, but, at the same time, well defined.