There are two parts to this exercise, only Part 1 is required - Part 2 is just to score some extra points :).
In all cases, the goal is to deliver a minimum viable product.

The goal of this exercise is to write the simplest view which enables providing “useful rated feedback” about a web application:

- A feedback is defined as a rated statement about the application which can be very general or very specific. Here are some examples of feedback entries:
    - Bad - This application is crap
    - Good - (but) I would like to have a reduced loading time
    - Bad - The application freeze all the time.
    - Very Good - (but) I miss feature XYZ
    - Good - (but) The new version XYZ introduces bug ABC
- A useful feedback is a feedback which is discussed by the whole user community and is mitigated by comments and notation (+1, 0, -1).
    - The comments are used in order to discuss and ideally scope the statement. A user can give several comments on a feedback.
    - The notation enable to mitigate the importance of the feedback. A user can only give 1 notation to a feedback.

Part 1: User Feedback list (required)

- A user of a web based application has a view which enables him to post a feedback about the application with as parameters at least:
    - a note (optional description of the issue)
    - a rating (1=bad, 5=very good).
- For each feedback entry, the application has another view which enables to browse the feedback comments and that perform the following actions:
    - add a comment to the feedback
    - give a (+1, 0, -1) notation to other user’s comments

Part 2: Product Manager view (optional)

- The product manager should be able to put the user feedback in categories
- The product manager should be able to give a status to the feedback:
    - open
    - closed (backlog)
    - closed (solved)
    - closed (rejected)
- The product manager should be able to merge similar feedbacks.

- Implementing some unit-tests is a plus.

Any additional features are of course welcome…

The front end parts must be written in HTML/JS, using the angular framework. It is required to write a backend (in Node JS) for the application. If you don't, you should at least create the necessary angular services/factories abstractions for any data that would in normal cases require a backend. I advise that you wrap any calls to the service in promises so that it matches the API of an HTTP call. You might also want to use local storage.

You will need to send me an email that includes links to the following:

- The source code itself - hosted on Github.
- A hosted version of the application for us to test. I recommend using Github Pages for this.

When reviewing the application, we will be focusing on :

- Your understanding of the javascript programming paradigms (asynchronicity, prototyping, etc.)
- Robustness of the backend (less important for the frontend position
- Code quality and best practices
- Code style and attention to detail
- Layout and design of the frontend app (less important for backend engineer)