Node.js Backend Practical Exam

Version 1.0
Last update on: 14/Feb/2016

Requirements

We need to create the backend for a web application that can perform the following functions via REST API s:

1. Users can login and logout
2. There is a special user named “Admin” that has access to the backend reporting system
3. Users can book any number of meetings
   Note: In order to book a meeting, the system uses a 3rd party REST API that might take some time to respond
4. Users can cancel any meeting that was previously scheduled identifying the meeting by the meetingID field.
   Note: In order to cancel a meeting, the system uses a 3rd party REST API that might take some time to respond
5. Admin users can clear all the system counters from the reporting subsystem
6. Admin users can request to see all the counters managed by the reporting subsystem
7. Admin users can request a list of users that are currently logged in and for each user, the time he has been logged in is also reported
8. The frontend application for the Admin users has a special Real time report where every time a user logs in or logs out, a notification from the back end system is generated and delivered to the front-end application automatically with minimal delay.
9. The Admin user can start/stop the real time reports when he wants via an API call.
10. The reporting system should also keep a list of users that are currently logged in. An API available for Admin users is used to retrieve the list of logged in users, including Amin.

The following counters are managed by the reporting subsystem:

1. Pending requests: Meetings or login/logout requests that are currently in process of been served
2. Average request time for all requests since the counters were cleared
3. Number of meetings scheduled since the counters were cleared
4. Number of meetings cancelled since the counter were cleared
5. Total number of requests processed (Login/Logout, Set/Cancel meetings) since the counters were cleared

The calendar 3rd party API has the following definition:

a. Set a meeting
   POST http://,meetingserver.com/calendarId
   JSON data: {"CalendarId": calendarId, "StartTime": time, "Duration": durationInMinutes, "StartDate": date, "subject": subject_description}
   Response: MeetingId or fail

b. Cancel a meeting
   DELETE http://,meetingserver.com/calendarId/meetingId
   Response: Ok or fail

Design constrains

- Minimum coupling between the reports and the rest of the system
- In the future, Meetings, login/out and reports might be separated into different node.js servers
- Real time reports should be sent when the relevant action occurs with minimal delay
- If you need to persist data, a local file can be used
- The Code should be clean, clear, modular and well structured.
- The frontend application will not be ready soon, therefore it can't be used to test the backend application
- If you want to use any external node.js modules, explain why you selected a specific module and what it will do for you.
- Take into consideration that the call to the 3rd party API might time-out without a response.

Your task:

1) Create a Node.js project for the back-end system defined
2) Define the API that the backend system and the frontend application will use to communicate between them
3) Implement the backend server giving emphasis to the reporting subsystem. Assume the meeting and login logic is very simple and straightforward, meetings should call the 3rd party API and Login/logout is locally authenticated.
4) Make sure you can test the application, including the calls for the 3rd party meeting API
5) Unit tests are very desirable.
6) The real-time reports should be the last thing you implement.