PG6100 Theory Mock Exam

The 10 questions in this mock exam are all worth the same. The questions are written in English. To answer these questions, it is preferred that you do it in English. However, any other language officially recognized by Westerdals (e.g., Norwegian) is obviously acceptable. Note: each question can be answered with just 1-2 paragraphs.

1) Why has JSON replaced XML as the most common data transfer type used in Web Services? Give at least two good reasons.

2) In HTTP, why a GET request must not have side-effects (e.g., creating or deleting resources)? Explain in *details* (with some concrete examples) what could go wrong otherwise.

3) In a successful HTTP PUT request, you might get back either a 201 or 204 as status code. What would they mean here? Explain in *details*.

4) Assume a PATCH endpoint based on JSON Merge Patch format (RFC-7396). The endpoint could declare to expect payload with type either "*application/json*" or "*application/merge-patch+json*", and both will work. But why should the latter be preferred? And, concretely, what is the difference between the two? Explain in *details*, especially the role of the symbol "*+*".

5) In HTTP redirection 3xx, how can the server tell the client where to redirect to?

6) In REST APIs, what does it mean to use "*Wrapped Responses*"? Why do we need to use them? Explain in *details*.

7) What is the need for using a library like WireMock? How does it work (at high level)?

8) You make a HTTP GET request, and in the answer form the server there is the header "*Cache-control: max-age=0, s-max-age=600*". What does it mean? What would be reason for a server to choose such values instead of for example "*Cache-control: max-age=600, s-max-age=600*"? Explain in *details*.

9) In the context of micro-services, how does "*Client-Side Load Balancing*" (e.g., using Eureka/Ribbon) work? Why is it needed? Explain in *details*.

10) In the context of micro-services, explain the high level steps and components involved to achieve user authentication based on *distributed* sessions (as seen in class).