1. Explain some additional concerns related to using REST apis.

- The REST(Representational State Transfer) is an Application Programming Interface(api) that makes calls from the client to the server and gets back data over the HTTP protocol.
- There are only a few HTTP requests like GET,POST etc and some may not be supported by the client or server side machines.
- The failures in the code are many a times very difficult to point out. The output maybe a simple error message. But why it went wrong and what exactly went wrong is many a times very challenging to find.
- The message content is HTTP and hence is not independent of the channel. This may cause problems and bottlenecks thus slowing down the service.
- The code must be such that it is platform independent and can be run on all computers. Computers that do not support the API and its platform cannot run the code.
- Since it is a client server based architecture, all the concerns of the client server protocol apply here as well.
- The names given to functions and items must be consistent and their category must be predictable. Inconsistent naming may be problematic for users to interpret and this may lead them to not wanting to use the api at all
- The architecture assumes that the middleware implements REST correctly. This may not be always true and hence must be verified.

(Sources: https://mmikowski.github.io/the_lie/

https://news.ycombinator.com/item?id=13759520)

2. Compare and contrast the benefits and disadvantages of using a RESTful architecture vs. a graph query language.

- A graph query language passes queries that could be converted to JSON. But in RESTful architecture the queries have to be in JSON.
- The GraphQL syntax is very intuitive and easy to write. Also mistakes are easy to recognize in GraphQL. In RESTful architecture the errors and problems are difficult to debug.
- GraphQL is much more clearly defined than RESTful architecture.
- GraphQL prefers larger individual requests and hence reducing the overall overhead from all individual requests.
- GraphQL allows us to create APIs without versions and thus giving a single evolving version as compared to RESTful architecture which needs to have versions.
- GraghQL generates a uniform API throughout the application hence making it much more user friendly.
- RESTful APIs are not guaranteed to be mutually compatible. That is why a specific flavour of REST is needed for a specific application which is tightly coupled to the application.
- GraphQL server libraries include many other important languages other than JavaScript. Languages like Python, Java, C#, etc are supported.s

(Sources: http://graphql.org/

https://symfony.fi/entry/what-is-graphql-and-how-does-it-differ-from-rest-apis)