|  |  |
| --- | --- |
| 1 | The Software Selection Maturity Scale is a five-level scale which measures the maturity of a given enterprise for its technology evaluation and acquisition process.  Many enterprises or parts thereof perform low down in the scale, these include…  1. Initial -> 2. Basic -> 3. Proactive & Defined -> 4. Verified & Adjusted -> 5. Optimised Tested & Approved  The SSMS describes the adoption of new technologies as it goes through the various phases of what new software adoption would entail. These include the initial being a chaotic process to the basic process being in place then a formal process defined followed and managed to functionality claims are verified and scope adjusted to match available software and finally contracts are opitimised for risk, implementation, test and post evaluation. |
| 2 | Monolithic software tends to be a small number of a large application providing many diverse functions on a wide variety of a data sources. A problem with monolithic software for the business is that the cost of maintaining such a large and complex systems and lack of flexibility to be able to provide new or bespoke solutions in timely manner.  On the other hand Service-Oriented Architectures is designed to see the construction of software solutions from a set of technology-independent components which can be composed together over a network using some well-defined network protocol. This contrasts with monolithic application in the sense that the solutions to the problems not originally addressed by the monolith can be created by end-users more flexibly from the components.  Using SOA the application is split into a number of technology-independent components or services, which allows for high coupling and cohesion throughout the application. This makes it ideal for application construction. |
| 3 | Why have enterprises moved towards web technologies for service software construction? What are the principal benefits? |
| 4 | Advantages:   * RESTful web services work seamlessly with HTTP * Easy to identify each request and each one is independent * Simplifies the server design because there is no dynamically allocated storage * If a client dies in mid-transaction, no part of the system needs to be responsible for cleaning up the present state of the server   Disadvantages:   * Data or information sent via HTTP is not encrypted * Web services need to get extra information in each request * The way it handles sessions with HTTP is adding more information every time to the request like a cookie |

|  |  |
| --- | --- |
| 5 | The Achitectural constraints of REST can be divided into its Client-Server, Stateless, Cacheable, Layered and Uniform Contract architectural constraints.  Further explained:  Client-Sever:  Establishes the separation of concerns between the service provider and the service consumer. The provider offers one or more capabilities and listens for requests for those capabilities. The consumer is responsible for presenting the responses to the user and taking any corrective actions on foot of errors  Stateless:  The principle that no server-side state exists between any two REST requests from a consumer, i.e. requests are self-contained and standalone. Contrast this with a transactional style, for example  Cacheable:  Service responses can be explicitly labeled as cacheable or non-cacheable. This allows intermediating caching nodes to reuse previous responses to requests without having to go all the way back to the service. This is a key idea in making RESTful services scalable  Layered:  An arbitrary number of nodes can be placed between the ultimate service and service consumer. Their existence must be fully transparent so that they can be added and removed at will. This allows for the distribution and scalability of RESTful service solutions in practice |
| 6 | Explain the relationship between resources, models and views? What is meant by view aggregation? |
| 7 | Describe the five RESTful operations, giving examples using HTTP. What is meant by idempotence? Mention which of the REST operations are idempotent and why.  Indempotence is the property of an operation such that operation can be applied multiple times to some value without changing the outcome beyond its application. |
| 8 | Explain the problem of failure propagation in SOA systems. What are the desirable characteristics of an API versioning system? What are the two kinds of API compatibility? |
| 9 | Describe the major elements of the logical data model. Describe how it abstracts the details of database access in the application tier. |
| 10 | Describe in detail the pathology of a SQL injection exploit. What should the application developer to avoid this kind avoid this kind of vulnerability. |